KERNEL
SYSTEMS MANAGEMENT GUIDE

Version 8.0

July 1995

Revised June 2008

Department of Veterans Affairs (VA)
Office of Information & Technology (OI&T)
Common Services (CS)

Revision History

Documentation Revisions

The following table displays the revision history for this document. Revisions to the documentation are based on patches and new versions released to the field.

Table i. Documentation revision history

Date	Revision	Description	Author
07/95	1.0	Initial Kernel V. 8.0 software and documentation release **Software Version: 8.0**	San Francisco, CA Office of Information Field Office (OIFO): • Project Manager – Hans Von Blanckensee • Developers – Kernel Development Team • Technical Writer – Kyle Clarke
12/09/04	2.0	Kernel V. 8.0 documentation reformatting/revision. This is the initial complete reformatting of the *Kernel Systems Management Guide* since its original release in July 1995. The largest change with the *Kernel Systems Management Guide* is that all developer-specific content has been extracted and placed into a new *Kernel Developer's Guide*. Also, at this point in time, only minimal content updates have been made based on select released Kernel patches. Due to time constraints, not all released Kernel patches with content changes have been added at this time. We wanted to get a new baseline document published so that in the future we can more easily update the *Kernel Systems Management Guide*. As time allows, we will be updating this reformatted manual with all released patch information that affects its content. Because of the chapter-numbering scheme, future additions can be made with minimal disruption to the entire	Technical Writer – Thom Blom, Oakland OIFO

Date	Revision	Description	Author
		manual page flow. Thanks for your patience! **Software Version: 8.0**	
12/20/04	2.2	Reviewed document and edited for the "Data Scrubbing" and the "PDF 508 Compliance" projects. **Data Scrubbing—**Changed all patient/user TEST data to conform to OI&T standards and conventions as indicated below: • The first three digits (prefix) of any Social Security Numbers (SSN) start with "000" or "666." • Patient or user names are formatted as follows: XUPATIENT,[N] or XUUSER,[N] respectively, where the N is a number written out and incremented with each new entry (e.g., XUPATIENT, ONE, XUPATIENT, TWO, etc.). • Other personal demographic-related data (e.g., addresses, phones, IP addresses, etc.) were also changed to be generic. **PDF 508 Compliance—**The final PDF document was recreated and now supports the minimum requirements to be 508 compliant (i.e., accessibility tags, language selection, alternate text for all images/icons, fully functional Web links, successfully passed Adobe Acrobat Quick Check). **Software Version: 8.0**	Technical Writer – Thom Blom, Oakland OIFO
02/03/06	3.0	Updates: • Reformatted document to the latest SOP and Style Guidelines. • Updated files, routines, options, APIs, security keys, etc. **Software Version: 8.0**	Oakland, CA OIFO ISS Maintenance Team
07/13/06	4.0	Updates: • Made minor formatting updates throughout. • Changed the original "Other Tools" section to become the new	Oakland, CA OIFO: • Maintenance Project Manager – Jack Schram • Developers – Alan

Date	Revision	Description	Author
		"Toolkit" section, see note below. • Added "Multi-Term Look-Up (MTLU)" and "Tools" chapters from the original *Toolkit User Manual* (V. 7.3), see note below. • Removed the "Response Time Measures" topic from the original "Capacity Management" chapter in the *Toolkit User Manual* (V. 7.3), see note below. Kernel Toolkit Patch XT*7.3*102 removed all Response Time Log Option menu options [XURTL*]. **NOTE:** All Kernel Toolkit content currently in the *Kernel Toolkit User Manua*l and *Kernel Toolkit Technical Manual* is being absorbed by the *Kernel Systems Management Guide*, *Kernel Developer's Guide*, and *Kernel Technical Manual*. Other Toolkit content has been replaced by other manual sets, including: • Duplicate Record Merge: Patient Merge • Resource Usage Monitor (RUM) • Statistical Analysis of Global Growth (SAGG) • Capacity Management (CM) Tools **Software Version: 8.0**	Chan, Wally Fort, and Joel Ivey. • Technical Writer: – Thom Blom
02/08/07	5.0	Security & Other Common Services (S&OCS) is combining the Kernel Toolkit documentation set with the Kernel documentation set. All Kernel Toolkit content will be moved to the appropriate Kernel manual, section, and chapter. In the *Kernel Systems Management Guide*, the following Kernel Toolkit chapters have been added to the new "Toolkit" Section VI: • Multi-Term Look-Up (MTLU) • Routine Tools • Verification Tools	Oakland, CA OIFO: • Maintenance Project Manager – Jack Schram • Developers – Alan Chan, Wally Fort, Jose Garcia, Joel Ivey, Mike Meighan, Raul Mendoza, Roger Metcalf, Skip Ormsby and Ba Tran. • Technical Writer – Thom Blom

Date	Revision	Description	Author
		Also Changed Kernel document title references to: • *Kernel Developer's Guide* (previously known as the *Kernel Programmer Manual*). • *Kernel Systems Management Guide* (previously known as the *Kernel Systems Manual*). **Software Version: 8.0**	
06/10/08	5.1	Updates: • Updated the "Defining a Strong Verify Code" topic. • Updated Chapter 5, "File Access Security," based on the newly created *VA FileMan (Version 22) and Kernel (Version 8.0) File Access Security* supplemental document on the VDL. • Deleted "Default Task Priority" topic from this manual. • Moved the "Error Screens" topic from Chapter 22, "TaskMan: System Management—Operation" to Chapter 13, "Error Processing." • Updated the "Alpha/Beta Tracking" topic in Chapter 23. Merged information from the *Kernel Systems Management Guide* (this manual) into the *Kernel Developer's Guide* in order to avoid duplication and confusion with instructions/procedures. • Updated VA OI&T organization changes and the document properties (e.g., Title, Author, Creation Dates, Keywords, etc.). • Updated references to the VDL. • Removed all references to HSD&D. • Updated Alert options in Figure 10-3 and added the missing descriptions for those Alert-related options. • Completed updates to remove obsolete references to MSM, PDP, 486, VAX Alpha, etc. and updated	Oakland, CA OIFO: • Maintenance Project Manager – Jack Schram • Developers—Gary Beuschel, Alan Chan, Wally Fort, Jose Garcia, Joel Ivey, Raul Mendoza, Roger Metcalf, Skip Ormsby, and Ba Tran. • Technical Writer – Thom Blom

Date	Revision	Description	Author
		references to DSM for OpenVMS to Caché where appropriate. • Updated content references to checksum compares based on Kernel Patch XU*8.0*393. • Changed references from "%INDEX" to "XINDEX" where appropriate. • Updated Section III, Device Handler chapters. • Deleted "Kermit" section. • Updated "SPECIAL QUEUEING" topic in Chapter 22. Added Table 22-1. • Updated "Security Forms" section in Chapter 3, "Signon/Security: System Management." **Software Version: 8.0**	

Patch Revisions

For the current patch history related to this software, please refer to the Patch Module on FORUM.

Contents

Figures and Tables

Figures

Tables

Orientation

How to Use this Manual

Throughout this manual, advice and instruction are offered about the numerous tools and functionality that Kernel V. 8.0 provides for the Veterans Health Information Systems and Technology Architecture (VistA) management and end-users. For example, site parameters are discussed in various topics throughout this manual.

CAUTION: To protect the security of VistA systems, distribution of this software for use on any other computer system by VistA sites is prohibited. All requests for copies of Kernel for non-VistA use should be referred to the VistA site's local Office of Information Field Office (OIFO).

Otherwise, there are no special legal requirements involved in the use of Kernel.

The Kernel Systems Management Guide is divided into six major sections, based on the following functional divisions within Kernel:

I Signon/Security (e.g., techniques for granting user access and monitoring computing activity)

II Menu Manager (e.g., techniques for managing menus)

III Device Handler

IV TaskMan

V Kernel Installation and Distribution System (KIDS)

VI Toolkit

REF: For information on developer tools (e.g., Direct Mode Utilities and Application Program Interfaces [APIs]), please refer to the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

Information on recommended system configuration and setting Kernel's site parameters, as well as lists of files, routines, options, and other components are documented in the *Kernel Technical Manual*.

Information about managing computer security, which includes a detailed description of techniques that can be used to monitor and audit computing activity, is presented in the *Kernel Security Tools Manual*.

Instructions for installing Kernel are provided in the *Kernel Installation Guide*. This guide also includes information about software application management (e.g., recommended settings for site parameters and scheduling time frames for tasked options).

This manual is further organized within each section of Kernel in the following order:

1. User Interface—Information of relevance to the general end-user.
2. System Management—Information of relevance to system managers.

When a subject is large enough (e.g., Signon/Security), separate chapters are devoted to the "User Interface" and "System Management" topics. In other cases, where the subject matter is smaller (e.g., the discussion of the Browser device), the two divisions of audience are contained entirely within a chapter or sub-chapter.

This manual uses several methods to highlight different aspects of the material:

- Various symbols are used throughout the documentation to alert the reader to special information. The following table gives a description of each of these symbols:

Table ii. Documentation symbol descriptions

Symbol	Description
	**NOTE/REF:** Used to inform the reader of general information including references to additional reading material.
	**CAUTION/DISCLAIMER:** Used to caution the reader to take special notice of critical information.

- Descriptive text is presented in a proportional font (as represented by this font).
- Conventions for displaying TEST data in this document are as follows:
  - The first three digits (prefix) of any Social Security Numbers (SSN) will begin with either "000" or "666".
  - Patient and user names will be formatted as follows: [Application Name]PATIENT,[N] and [Application Name]USER,[N] respectively, where "Application Name" is defined in the Approved Application Abbreviations document and "N" represents the first name as a number spelled out and incremented with each new entry. For example, in Kernel (KRN) test patient and user names would be documented as follows: KRNPATIENT,ONE; KRNPATIENT,TWO; KRNPATIENT,THREE; etc.
- Sample HL7 messages, "snapshots" of computer online displays (i.e., character-based screen captures/dialogues) and computer source code are shown in a *non*-proportional font and enclosed within a box. Also included are Graphical User Interface (GUI) Microsoft Windows images (i.e., dialogues or forms).
  - User's responses to online prompts will be boldface.
  - References to "**<Enter>**" within these snapshots indicate that the user should press the **Enter** key on the keyboard. Other special keys are represented within **< >** angle brackets. For example, pressing the **PF1** key can be represented as pressing **<PF1>**.
  - Author's comments are displayed in italics or as "callout" boxes.

    NOTE: Callout boxes refer to labels or descriptions usually enclosed within a box, which point to specific areas of a displayed image.
- This manual refers in many places to the M programming language. Under the 1995 American National Standards Institute (ANSI) standard, M is the primary name of the MUMPS

programming language, and MUMPS will be considered an alternate name. This manual uses the name M.

- Descriptions of direct mode utilities are prefaced with the standard M ">" prompt to emphasize that the call is to be used *only in direct mode*. They also include the M command used to invoke the utility. The following is an example:

```
>D ^XUP
```

- All uppercase is reserved for the representation of M code, variable names, or the formal name of options, field/file names, and security keys (e.g., the XUPROGMODE security key).

NOTE: Other software code (e.g., Delphi/Pascal and Java) variable names and file/folder names can be written in lower or mixed case.

How to Obtain Technical Information Online

Exported VistA M Server-based software file, routine, and global documentation can be generated through the use of Kernel, MailMan, and VA FileMan utilities.

NOTE: Methods of obtaining specific technical information online will be indicated where applicable under the appropriate topic.

REF: Please refer to the *Kernel Technical Manual* for further information.

Help at Prompts

VistA M Server-based software provides online help and commonly used system default prompts. Users are encouraged to enter question marks at any response prompt. At the end of the help display, you are immediately returned to the point from which you started. This is an easy way to learn about any aspect of VistA M Server-based software.

Obtaining Data Dictionary Listings

Technical information about VistA M Server-based files and the fields in files is stored in data dictionaries (DD). You can use the List File Attributes option on the Data Dictionary Utilities submenu in VA FileMan to print formatted data dictionaries.

REF: For details about obtaining data dictionaries and about the formats available, please refer to the "List File Attributes" chapter in the "File Management" section of the *VA FileMan Advanced User Manual*.

Assumptions About the Reader

This manual is written with the assumption that the reader is familiar with the following:

- VistA computing environment:
  - Kernel—VistA M Server software
  - VA FileMan data structures and terminology—VistA M Server software
- Microsoft Windows environment
- M programming language

This manual provides an overall explanation of Kernel and the functionality contained in Kernel V. 8.0. However, no attempt is made to explain how the overall VistA programming system is integrated and maintained. Such methods and procedures are documented elsewhere. We suggest you look at the various VA Internet and Intranet Web pages for a general orientation to VistA. For example, go to the Veterans Health Administration (VHA) Office of Information and Technology (OI&T) Veteran Health Information Technology (VHIT) Portfolio – VistA Development Home Page at the following Intranet Web address:

http://vista.med.va.gov/

Reference Materials

Readers who wish to learn more about Kernel should consult the following:

- *Kernel Release Notes*
- *Kernel Installation Guide*
- *Kernel Systems Management Guide* (this manual)
- *Kernel Developer's Guide*
- *Kernel Technical Manual*
- *Kernel Security Tools Manual*
- Kernel Home Page at the following Web address:

  http://vista.med.va.gov/kernel/index.asp

  This site contains other information and provides links to additional documentation.

If the reader is not already familiar with VA FileMan or MailMan, the respective user, developer, and technical manuals for each should be obtained and reviewed. Other source documents describing overall VistA policy are:

- *VA Programming Standards and Conventions (SAC)*
- *MIRMO/OIFO Operations Document*

VistA documentation is made available online in Microsoft Word format and in Adobe Acrobat Portable Document Format (PDF). The PDF documents *must* be read using the Adobe Acrobat Reader (i.e., ACROREAD.EXE), which is freely distributed by Adobe Systems Incorporated at the following Web address:

http://www.adobe.com/

VistA documentation can be downloaded from the VHA Software Document Library (VDL) Web site:

http://www.va.gov/vdl/

VistA documentation and software can also be downloaded from the Enterprise Product Support (EPS) anonymous directories:

- Preferred Method download.vista.med.va.gov

  This method transmits the files from the first available FTP server.
- Albany OIFO ftp.fo-albany.med.va.gov
- Hines OIFO ftp.fo-hines.med.va.gov
- Salt Lake City OIFO ftp.fo-slc.med.va.gov

DISCLAIMER: The appearance of external hyperlink references in this manual does *not* constitute endorsement by the Department of Veterans Affairs (VA) of this Web site or the information, products, or services contained therein. The VA does *not* exercise any editorial control over the information you may find at these locations. Such links are provided and are consistent with the stated purpose of this VA Intranet Service.

1 Introduction

This manual provides descriptive information about Kernel for use by Information Resource Management (IRM) staff, end users, Automated Data Processing Application Coordinators (ADPACs), and application developers.

This manual assumes that the reader is familiar with the computing environment of the VA's Veterans Health Information Systems and Technology Architecture (VistA), and understands VA FileMan data structures and terminology. Some understanding of the M programming language is helpful for some parts of the manual. No attempt is made to explain how the overall VistA programming system is integrated and maintained; such methods and procedures are documented elsewhere. This manual does, however, provide an explanation of Kernel utilities, describing how they can be used to establish a standard user interface, monitor and manage the computer system, customize the environment according to local site needs, and define new areas of computing activities for users.

Kernel is a vendor-independent applications development environment, as well as a run-time environment providing standard vendor-independent services to applications software. It is not an operating system, but a set of utilities and associated files that are executed in an M environment. Kernel is central to VA VistA software strategy, in that it permits any VistA software application to run without modification on any hardware/software platform that supports American National Standards Institute (ANSI) Standard M. All operating system-specific, M implementation-specific, or hardware-specific code is isolated to Kernel. Therefore, porting VistA to a new environment requires modification only to a handful of Kernel routines.

As a whole, Kernel provides a computing environment that permits controlled user access, presents menus for choosing from various computing activities, allows device selection for output, enables the tasking of background processes, and offers numerous tools for system management and application programming. Kernel also provides tools for software distribution and installation.

VistA users see the same user interface, regardless of the underlying system architecture, because VistA applications are built using Kernel facilities for signon, database access, option selection, and device selection. As a result, user interaction with the system is constant across VistA applications.

User Introduction

Kernel provides the doorway into the VistA computer system, the menus that tie together the options and utilities to enhance those options.

For the doorway, Kernel provides the Access and Verify code system that you use to establish your identity to the VistA computer system.

Once you have signed on, Kernel provides your menus. Each user on the computer system, as identified by their Access code, has their own individual set of menus and options.

The person or department managing the computer system organizes each user's menus. From your menu, you can run any application the computer system managers have made available to you. Kernel's menu system is what is used to make VistA applications (e.g., Scheduling, Nursing, and Personnel) available to users.

To produce output from VistA applications (e.g., to printers or to the terminal screen), Kernel provides a common device interface called the Device Handler. To queue a job rather than run it directly, the Device Handler links to a common queuing system called TaskMan.

This manual contains information about these and other parts of Kernel. The intent of this manual is to help you learn to use Kernel and take fullest advantage of the facilities it provides. This manual also includes information for system managers and developers; to find the information of interest to you, the general user, look for chapters and sub-chapters containing the phrase "User Interface" in their titles.

ADP Application Coordinators (ADPACs) may want to skim through the *Kernel Systems Management Guide* and concentrate on the user interface chapters and sub-chapters, particularly issues concerning every Kernel user (e.g., signon process and menu navigation).

System Manager Introduction

Kernel provides the backbone of an M computing platform, providing a mechanism to organize M programs as options, and a way to organize those options into a menu system for users. Kernel provides the following major system management components:

- Alerts provide an integrated notification system.
- Device Handler provides a common device interface.
- Electronic Signature Codes provide a secure electronic approval system.
- File Access Security system manages access to VA FileMan files.
- Kernel Installation and Distribution System (KIDS) provides an application distribution and installation system.
- Menu Manager provides a common menu management system.
- Signon/Security organizes users and allows secure logons.
- TaskMan provides a common job queuing system.

Kernel provides the system manager the means to manage a secure, multi-user M-based computer system. Some typical daily tasks performed by system managers using Kernel system management tools include:

- Setting up accounts for new users and terminating accounts for expired users.
- Adding and subtracting options from users' menus.
- Controlling file access for users.
- Monitoring TaskMan task queues.
- Terminating unwanted tasks.
- Monitoring devices.
- Creating and modifying links to output devices in the DEVICE file (#3.5).
- Installing software applications.

Within chapters and sub-chapters of this manual you can find general user information in the "User Interface" topic and system manager information in the "System Management" topic.

REF: For information on developer tools (e.g., Direct Mode Utilities and Application Program Interfaces [APIs]), please refer to the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

Information on recommended system configuration and setting Kernel's site parameters, as well as lists of files, routines, options, and other components are documented in the *Kernel Technical Manual*.

Information about managing computer security, which includes a detailed description of techniques that can be used to monitor and audit computing activity, is presented in the *Kernel Security Tools Manual*.

Instructions for installing Kernel are provided in the *Kernel Installation Guide*. This guide also includes information about software application management (e.g., recommended settings for site parameters and scheduling time frames for tasked options).

I Signon/Security

2 Signon/Security: User Interface

The first step you take each time you access the computer system is called signing on. When you sign on to the VistA computer system, you are required to enter an Access and Verify code. These codes identify you to the computer system, and, as these codes are private to you, serve to prevent unauthorized access to your account.

You are shielded from most steps in the signon process. In the background, Kernel's Signon/Security establishes the proper environment, records and monitors the signon event, and takes you to Menu Manager. Menu Manager presents a list of menu options that let you interact with other parts of Kernel and software applications. When you complete a session on the computer system, you sign out to exit.

Signing On

To authenticate yourself to VistA (Kernel's "front door"), you need to sign onto the system by entering your Access and Verify codes. The user signon (authentication) interface varies based on the type of Vista application software being run:

- Character User Interface (CHUI)-based applications—This includes M-based roll-and-scroll applications used to access Kernel on the VistA M Server (e.g., Laboratory, Pharmacy). With this type of authentication interface, users are first prompted with an "ACCESS CODE:" prompt. Entering an Access code and pressing the **<Enter>** key brings up the "VERIFY CODE:" prompt.

  REF: For a sample of the roll-and-scroll signon prompts, please see Figure 2-1 below.

- Graphical User GUI client/server applications—This includes rich client and/or client/server applications used to access Kernel on the VistA M Server via RPC Broker (Delphi/Pascal)- or VistALink (Java)-based components (e.g., Computerized Patient Record System [CPRS] or Care Management). With this type of authentication interface, users are presented with a GUI signon dialogue box. Users can click in or tab to the Access and Verify code entry fields and press an "**OK**" button.

  REF: For a sample of the RPC Broker signon dialogue box and more information on RPC Broker, please refer to the RPC Broker documentation located on the VHA Software Document Library (VDL) at the following Web address:

  http://www.va.gov/vdl/application.asp?appid=23

- Web-based applications—This includes Web-based applications that use a client Web browser and Kernel Authentication and Authorization Java (2) Enterprise Edition (KAAJEE) to access Kernel on the VistA M Server (e.g., Blind Rehab). With this type of authentication interface, users are presented with a GUI signon dialogue Web page. Users can click in or tab to the Access and Verify code entry fields and press an "**Login**" button.

  REF: For a sample of the KAAJEE signon dialogue Web page and more information on KAAJEE, please refer to the KAAJEE documentation located on the VHA Software Document Library (VDL) at the following Web address:

  http://www.va.gov/vdl/application.asp?appid=151

Your Access code establishes your unique identity to Kernel. Your matching Verify code corroborates your identity completing the VistA Kernel authentication process. Asterisks only are displayed when you enter your Access and Verify codes so that the actual characters are *not* displayed (echoed back) on the screen. Codes are encrypted after they are entered and compared with the encrypted stored values for a match.

REF: For a description of valid and strong Access and Verify codes, please refer to the "Defining a Strong Verify Code" topic that follows in this chapter.

Figure 2-1. Signing on to VistA—Sample roll-and-scroll user authentication dialogue

```
ACCESS CODE: ******** <Enter>
VERIFY CODE: ******** <Enter>
Device: _TNA8628:
Not a valid ACCESS CODE/VERIFY CODE pair.

ACCESS CODES: ******** <Enter>
VERIFY CODES: ******** <Enter>
Good evening FRIEND     You last signed on Apr 21,1992 at 07:57

There was 1 unsuccessful attempt since you last signed on:

You were last executing the 'MailMan Menu' menu option.
Do you wish to resume? YES//
```

An invalid Access and Verify code pair produces an error.

Entering a valid Access and Verify code combination completes the signon authentication process and takes you beyond Signon/Security into Kernel's Menu Manager (or other security role-based access keys) used to authorize your appropriate level of access to data and/or application functionality.

If you have not been assigned a primary menu, Kernel displays a message indicating that access is not allowed, and signs you out from the computer system. Similarly, if your primary menu has been marked as "out-of-order" (an option attribute), Kernel also denies you access (see Figure 2-2).

REF: For more information on primary menus, please refer to the "Menu Manager" section in this manual.

Figure 2-2. Access denied due to no primary menu or menu "out of order" message

```
ACCESS CODES: ******** <Enter>
VERIFY CODES: ******** <Enter>
Device: _TNA8628: <Enter>
No access allowed for this user.
```

Defining a Strong Verify Code

While Access codes are a unique identifier (i.e., username) for your user record in Kernel's NEW PERSON file (#200), Verify codes are secret passwords assuring that the person signing on is the one for whom the user record was established. You rarely need to be issued a new Access code, but you *must* change your Verify code if you suspect that someone else has used it to gain access to the system or when your Verify code has expired (i.e., every 90 days or less). You can change your Verify code with the Edit User Characteristics option is available from the Common menu User's Toolbox menu.

As of Kernel Patch XU*8.0*180, Strong Access and Verify codes *must* adhere to the following criteria:

- Access and Verify codes *cannot* be identical.
- Verify codes (i.e., passwords) *must* be at least 8 characters in length.
- Strong passwords in general contain at least three of the following four character types:
  - Uppercase letters
  - Lowercase letters
  - Numbers
  - Special characters/symbols that are neither letters nor numbers (e.g., -, _, #, **&**, $, *, @)

    NOTE: Some *non*-VistA-based systems restrict certain special characters/symbols used as part of a username and/or password.

  Because VistA is case-insensitive, VistA only has three sets of characters from which to build a strong Verify code (i.e., password):

  - Letters (of any case)
  - Numbers
  - Special characters/symbols that are neither letters nor numbers (e.g., -, _, #, **&**, $, *, @)

    NOTE: Some *non*-VistA-based systems restrict certain special characters/symbols used as part of a username and/or password.

- Verify codes *must* be changed at least every 90 days (or less). You *must* change your Verify code at periodic intervals as specified by IRM. Information systems shall not permit re-assignment of the last three passwords used. When required, you will be prompted during signon to pick a new Verify code.
- Accounts that have been inactive for 90 days shall be disabled.

- To preclude password guessing, an intruder lockout feature shall suspend accounts after five invalid attempts to log on. Where around-the-clock system administration service is available, system administrator intervention shall be required to clear a locked account. Where around-the-clock system administration service is *not* available, accounts shall remained locked out for at least ten minutes.

NOTE: These rules are taken from the *VA Account and Password Management Interim Policy* document.

All of these restrictions are enforced whenever Access or Verify codes are created or changed.

These changes were made to meet VHA DIRECTIVE 6210 available at the following Web address:

http://vaww1.va.gov/vhapublications/ViewPublication.asp?pub_ID=54

REF: For more tips and general advice regarding Access and Verify codes and security in general, please refer to the *Kernel Security Tools Manual*.

Why Longer Passwords?

Passwords used to access VA systems *must* be at least 8 characters long because longer passwords are stronger, and thus, harder to guess than shorter ones.

The more tries it takes a hacker or a program to guess a password, the more secure the system is. Adding just one character to the length of a password greatly increases the difficulty of guessing the password.

For an 8-character password made up of letters and numbers (assuming you can repeat characters and that there are no restrictions, such as requiring the first character to be a letter), there are 36 possibilities for the first position, 36 possibilities for the second position, 36 possibilities for the third position, and so on. Thus, there are 36 x 36 x 36 x 36 x 36 x 36 x 36 x 36 = 2,821,109,907,456 possibilities for an 8-character password.

If you have forgotten your Verify code, the site's Information Security Officer (ISO) should delete the existing code, and then instruct you to sign on again. At the "Verify code" prompt simply press the **<Enter>** key without making any other entries. You will be prompted to enter a new Verify code and then re-prompted to enter the same Verify code again as confirmation. If you do not want to bother inventing a Verify code, entering a question mark ("**?**") at the Verify code prompt displays a possible although cryptic choice (e.g., DKMl&493). Entering a question mark a second time displays another choice. When you log off, you're reminded to remember the new Verify code for use at your next signon.

LOGIN Menu Template

You can execute a script of options on your first signon of the day by having a MENU template called LOGIN.

REF: For more information, please refer to the "Menu Manager: User Interface" chapter in this manual.

Signon Shortcuts

In roll-and-scroll VistA, to reach the primary menu in one step at the "ACCESS CODES:" prompt, you can enter the Access and Verify code as one string separated by a semicolon:

Figure 2-3. Entering the Access and Verify codes at the same time

```
ACCESS CODES: accesscode;verifycode
Good afternoon.     You last signed on today at 12:00
```

To "jump start" directly to a particular option, you can specify the name of an option after another semicolon:

Figure 2-4. Entering the Access and Verify codes at the same time and jumping directly to a specified option

```
ACCESS CODES: accesscode;verifycode;Intro
Good afternoon.     You last signed on today at 12:00
INTROductory text edit
```

To force the Kernel query of the terminal type identity, you can include a colon anywhere in the string.

REF: If you want to avoid the terminal type query, please refer to "Terminal Type Prompt" topic that follows.

Normal Signoff

When you complete a session on the computer system, you should sign off the system so that no one can come along and use the computer system under your identity. There are several ways you can sign off of the system.

Figure 2-5. System commands: Menu options for signoff

```
SYSTEM COMMAND OPTIONS                                          [XUCOMMAND]
  Halt                                                             [XUHALT]
  Continue                                                      [XUCONTINUE]
  Restart Session                                                 [XURELOG]
```

One way to sign off is to enter "halt" at any menu prompt. When you sign off using "halt," at next signon, after entering Access and Verify codes, your normal primary menu will be your first menu.

Or, to sign off, you can enter "continue." At your next signon, after entering Access and Verify codes, your last-used menu when you signed off will be your first menu for that session.

If remotely connected via modem or other network device, you can enter "restart" to sign out of Kernel without dropping the communication line.

Finally, you can sign off without using any of these shortcuts simply by pressing **<Enter>** at each menu prompt to step back up the menu pathway and finally exit.

REF: For more information on menus and menu prompts, please refer to the "Menu Management" section in this manual.

Abnormal Signoff and Error Handling

If you encounter an error while using the VistA computer system, Kernel will trap it, issue the message "Sorry 'bout that", and attempt to return you to your primary menu. Kernel can recover from most error conditions and, given a suitable environment, will permit you to continue. Some error conditions, however, cause an abnormal exit such that you are immediately logged off the computer system. When this happens, you can sign on again if you still need to use the computer system.

Terminal Type Prompt

When signing on, you may be prompted to enter a terminal type. You should not see this prompt very often, however, since Kernel usually can identify your terminal type without needing to prompt you to enter one. If you are prompted, you should enter the name of the actual terminal type to use (e.g., C-VT220). The entered terminal type tells Kernel how to support screen-oriented and other enhanced displays. If unusual circumstances arise and the wrong terminal type is in effect, you can redefine it by using the Edit User Characteristics option (available through the User's Toolbox menu, discussed below).

The Edit User Characteristics option lets you edit a setting (ASK DEVICE TYPE AT SIGN-ON) that allows you to decide whether to bypass the usual terminal type query. If you always work at the same terminal and want to save a small amount of time during the signon process, you can set ASK DEVICE TYPE AT SIGN-ON to DON'T ASK. Kernel then assumes that your last terminal type should be used as the default.

If you have ASK DEVICE TYPE AT SIGN-ON set to DON'T ASK, and sign on using a terminal whose terminal type is different from the one normally used, you should signon by including a colon ("**:**") after your Access code. This forces Kernel to query the terminal for its identity. Alternatively, once signed on, you could invoke the Edit User Characteristics option to change your terminal type to the one currently in use. Or, you could use this option to reset the ASK DEVICE TYPE AT SIGN-ON question to ASK, log off and sign back on (whereby Signon/Security will obtain the correct terminal type identification).

Escaping from a Jumbled Screen

One consequence of your signon terminal type not matching the actual one being used is that full-screen display could appear jumbled. To escape from a ScreenMan form (e.g., Edit User Characteristics), all you need to do is enter two carets ("^"), each followed by the **<Enter>** key. To escape from VA FileMan's Screen Editor, you should press **<PF1>E** to exit.

Alerts

After signing on, you could be presented with an alert notice just before the menu prompt. If so, you need to pick the View Alerts "VA" option [XQALERT] for viewing alerts to take care of urgent, pending matters.

REF: For more information about alerts, please refer to the "Alerts" chapter, in the "Menu Manager" section in this manual.

Figure 2-6. System commands: View Alerts "VA" option

```
SYSTEM COMMAND OPTIONS ...                                          [XUCOMMAND]
  View Alerts "VA"                                                    [XQALERT]
```

User's Toolbox Menu

The User's Toolbox menu [XUSERTOOLS] is available from any menu prompt, by entering the toolbox synonym (e.g., "TBOX") or "User's Toolbox." It makes available, from one menu, some of the most frequently used Kernel options.

Figure 2-7. User's Toolbox menu options

```
Select User's Toolbox Option:

          Change my Division                                    [XUSER DIV CHG]
          Display User Characteristics                             [XUUSERDISP]
          Edit User Characteristics                             [XUSEREDITSELF]
          Electronic Signature code Edit                              [XUSESIG]
          Menu Templates ...                                          [XQTUSER]
          Spooler Menu ...                                        [XU-SPL-MENU]
             **> Locked with XUMGR
          Switch UCI                                            [XU SWITCH UCI]
          TaskMan User                                              [XUTM USER]
          User Help                                                [XUUSERHELP]
```

The following table lists the options contained in the User's Toolbox menu and the chapters in this manual where each option is described:

Option Text	Chapter Described
Change my Division [XUSER DIV CHG]	Signon/Security: User Interface
Display User Characteristics [XUUSERDISP]	Signon/Security: User Interface
Edit User Characteristics [XUSEREDITSELF]	Signon/Security: User Interface
Electronic Signature code Edit [XUSESIG]	Electronic Signatures
Menu Templates [XU-SPL-MENU]	Menu Manager: User Interface
Spooler Menu [XU-SPL-MENU] (locked with XUMGR security key)	Spooling
Switch UCI [XU SWITCH UCI]	Signon/Security: User Interface
TaskMan User [XUTM USER]	TaskMan: User Interface
User Help [XUUSERHELP]	(accesses online help)

Table 2-1. User's Toolbox menu options and documentation references

Change my Division Option

The Change my Division option [XUSER DIV CHG] allows users to select from a list of divisions, if any, stored for that user in the NEW PERSON file (#200).

Edit User Characteristics Option

The Edit User Characteristics option [XUSEREDITSELF] is one of the options available from the User's Toolbox menu. It allows you define some characteristics of your online environment via ScreenMan, as shown below:

Figure 2-8. Edit User Characteristics ScreenMan form

```
                          EDIT USER CHARACTERISTICS

 NAME:  XUUSER,ONE                                              PAGE 1 OF 1

------------------------------------------------------------------------------

                        INITIAL: OX              PHONE: (555) 555-5555
                      NICK NAME: ONE      OFFICE PHONE:
                                           VOICE PAGER:
                                         DIGITAL PAGER:


     ASK DEVICE TYPE AT SIGN-ON: DON'T ASK
                     AUTO MENU: YES, MENUS GENERATED
                   TYPE-AHEAD: ALLOWED
               TEXT TERMINATOR:
             PREFERRED EDITOR: SCREEN EDITOR – VA FILEMAN

  Want to edit VERIFY CODE (Y/N):

------------------------------------------------------------------------------
Exit     Save     Refresh

Enter a command or '^' followed by a caption to jump to a specific field.


COMMAND:                                        Press <PF1>H for help  Insert
```

There are a number of NEW PERSON file (#200) field values that you can edit with the Edit User Characteristics option:

- **INITIAL:** You can enter your initials, which can serve as an alternate way for users to specify your account (e.g., when sending mail to you).
- **NICK NAME:** You can enter a nick name, which can serve as an alternate way for users to specify your account (e.g., when sending mail to you).
- **Telephone Contact Information:** You can enter phone numbers in the following fields:
  - **PHONE (HOME)**
  - **OFFICE PHONE**
  - **VOICE PAGER**
  - **DIGITAL PAGER**
- **ASK DEVICE TYPE AT SIGN-ON:** This field controls whether Kernel should determine what kind of terminal you are using when you sign on. If this is set to DON'T ASK, Kernel assumes you are using the same kind of terminal you used the last time you signed on. This can cause

problems if you are using a different kind of terminal (screen displays may not work properly), so this should normally be set to ASK.

- **AUTO MENU:** The setting of AUTO MENU determines whether, in the menu system, a list of items on the current menu is displayed with the menu prompt. Beginning users should usually set AUTO MENU to YES so that they can see menu items for each menu. Experienced users who are familiar with their menus may prefer to set this field to NO, which makes menu displays speedier since individual items on each menu are not displayed.
- **TYPE-AHEAD:** This setting controls whether characters you type faster than the system can process end up being processed or not. Normally you should set TYPE-AHEAD to YES, so that keystrokes you enter are not lost due to system slowness.
- **TEXT TERMINATOR:** The TEXT TERMINATOR is a setting used by VA FileMan's Line editor. When you are using the Line editor and are importing text from an external source, you may not want a blank line to indicate the end-of-file, which could prematurely terminate the text transfer. By default, the TEXT TERMINATOR in the VA FileMan line editor is the carriage return character (**<Enter>**). Setting this to another character string, like ZZ (something that will not be encountered in the target text) can permit downloading without interruption. If you change the setting of the TEXT TERMINATOR from the default of the carriage return character, you will need to remember your TEXT TERMINATOR when using the Line editor; otherwise, you will be unable to exit the Line editor.

  REF: For more information on the TEXT TERMINATOR, please refer to the *VA FileMan Getting Started Manual.*

- **PREFERRED EDITOR:** You can choose which editor Kernel uses when you edit word-processing fields on the system. You can choose any editor defined on your system.
- **VERIFY CODE:** You can change your VERIFY CODE by answering YES to this field. First enter your current VERIFY CODE; then, enter a new VERIFY CODE. You will be asked to confirm the new VERIFY CODE by entering it a second time; if you confirm it, the new VERIFY CODE will take effect immediately.

Display User Characteristics Option

The Display User Characteristics option [XUUSERDISP], like Edit User Characteristics, is an option in the User's Toolbox menu. It prints out a description of many of the characteristics of your current computing environment, including some of the characteristics that can be set through the Edit User Characteristics option.

Figure 2-9. Display User Characteristics option—Sample output and user dialogue

```
KRNUSER,TWO (#9999)  DEVICE: DEVICE: TELNET  ($I: TNA730:)      JOB: 541754169

ENVIRONMENT                              ATTRIBUTES
-----------                              -----------
   Site ........ OAKLAND                 Type-ahead ....... Y
   UCI ......... KRN,KDE                 Time-out ......... 300
   Signed on ... 08:48                   Fileman code(s) .. #
   Terminal type C-VT100

Person Class: Physicians (M.D. and D.O.)
                Physician/Osteopath
                  Pathology, Anatomic

KEYS HELD
---------
XMMGR              XUPROG             XUPROGMODE

MENU PATH
---------
  SYSTEM COMMAND OPTIONS (XUCOMMAND)
    User's Toolbox (XUSERTOOLS)
      Display User Characteristics (XUUSERDISP)

'^' to escape, <CR> to view Mailman user info: <Enter>


Current Banner: S&OCS Technical Writer
Last used MailMan: 07/12/06@15:09
NEW messages: 274 (274 in the IN basket)

Office phone:  (555) 555-5555
Fax:           (555) 555-5555
Add'l phone:   (555) 555-5555
Add'l phone:   (555) 555-5555


Introduction:
  My name is One Xmuser and I am one of the Technical Writers for the
  Security and Other Common Services (S&OCS) products/projects (e.g., Broker,
  Components, Kernel, VA FileMan, MailMan, Toolkit).

Mail Groups:
  ISC STAFF                                      (Public)
  KERNEL PROGRAMMERS                              (Public)
```

Switch UCI Option

The Switch UCI option [XU SWITCH UCI] allows users to select from a list of UCIs, if any, stored for that user in the NEW PERSON file (#200).

Summary

VistA's Kernel's Signon/System Security module provides the means for signing into Kernel with a unique identity. Once you complete the signon process, you are sent to Kernel's menu system, where you can run any option your system manager has placed in your menus. When you finish a computer session, always be sure to sign off; this protects your account from misuse by someone else.

3 Signon/Security: System Management

This chapter describes the system management tools for Kernel's Signon/Security module.

Signon Process

If signons are enabled, as shown in the Signon Flow Chart that follows, the signon process begins with a gathering of information from the KERNEL SYSTEM PARAMETERS file (#8989.3) and then from the DEVICE file (#3.5) to determine whether to allow signon for this session and, if so, how to create an appropriate environment. If, for example, the MAX SIGNON ALLOWED limit has been reached, the signon attempt will fail. If the current device is tied to a routine (as specified in the TIED ROUTINE field of the DEVICE file [#3.5]), that routine is executed and the session is halted. If not, the user is prompted for Access and Verify codes. After a successful signon, attributes for that user are then retrieved from the NEW PERSON file (#200). Signon/Security then sends the user to Menu Manager. If a primary menu is associated with the device (PRIMARY MENU OPTION field in the DEVICE file [#3.5]), that menu is presented. Otherwise, the user's primary menu is presented. If the user does not have a primary menu (the PRIMARY MENU OPTION field in the NEW PERSON file [#200] is NULL), the session is halted.

The signon flow chart in this topic (see Figure 3-4) illustrates the procedural steps taken by Kernel's Signon/Security system to determine whether to permit signons and, if so, how to create an appropriate computing environment. Typically, after site parameters and device characteristics are checked, the user is prompted for Access and Verify codes, user attributes are collected, and a primary menu prompt is presented.

Introductory Text

Before gathering system parameters or prompting for Access and Verify codes, Signon/Security displays contents of the INTRO TEXT field in the KERNEL SYSTEM PARAMETERS file (#8989.3). The text can be edited with the Enter/Edit Kernel Site Parameters option or with the Introductory text edit option [XUSERINT], an option specially designed for this purpose).

Figure 3-1. Introductory text edit option

```
SYSTEMS MANAGER MENU ...                                          [EVE]
Operations Management ...                                     [XUSITEMGR]
   Introductory text edit                                      [XUSERINT]
```

Parameters Checked during Signon

Various parameters are checked as an initial step in the signon process. The KERNEL SYSTEM PARAMETERS file (#8989.3) stores the default values for most of the parameters. Values for critical fields should be defined by IRM when Kernel is installed. The values in the KERNEL SYSTEM PARAMETERS file (#8989.3) can be edited any time, though, with the Enter/Edit Kernel Site Parameters option [XUSITEPARM].

Figure 3-2. Enter/Edit Kernel Site Parameters option

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Operations Management ...                                                [XUSITEMGR]
   Kernel Management Menu ...                                             [XUKERNEL]
      Enter/Edit Kernel Site Parameters                                  [XUSITEPARM]
```

Figure 3-3. Enter/Edit Kernel Site Parameters option ScreenMan form

```
                        Kernel Site Parameter edit
             DOMAIN:FO-OAKLAND.MED.VA.GOV

       DEFAULT # OF ATTEMPTS: 5                     AGENCY CODE: VA
       DEFAULT LOCK-OUT TIME: 600
    DEFAULT MULTIPLE SIGN-ON: Only one   MULTIPLE SIGN-ON LIMIT: 5
           DEFAULT AUTO-MENU: YES          DEFAULT AUTO SIGN-ON: Yes
            DEFAULT LANGUAGE:
          DEFAULT TYPE-AHEAD: YES
DEFAULT TIMED-READ (SECONDS): 999                BROKER TIMEOUT: 180

      BYPASS DEVICE LOCK-OUT: YES       ASK DEVICE TYPE AT SIGN-ON: NO
     LIFETIME OF VERIFY CODE: 90
         DEFAULT INSTITUTION: ISC SAN FRANCISCO
  AUTO-GENERATE ACCESS CODES: NO
         LOG RESOURCE USAGE?: YES

_____________________________________________________________________________
Exit     Save     Next Page     Refresh

Enter a command or '^' followed by a caption to jump to a specific field.


COMMAND:                                        Press <PF1>H for help    Insert
```

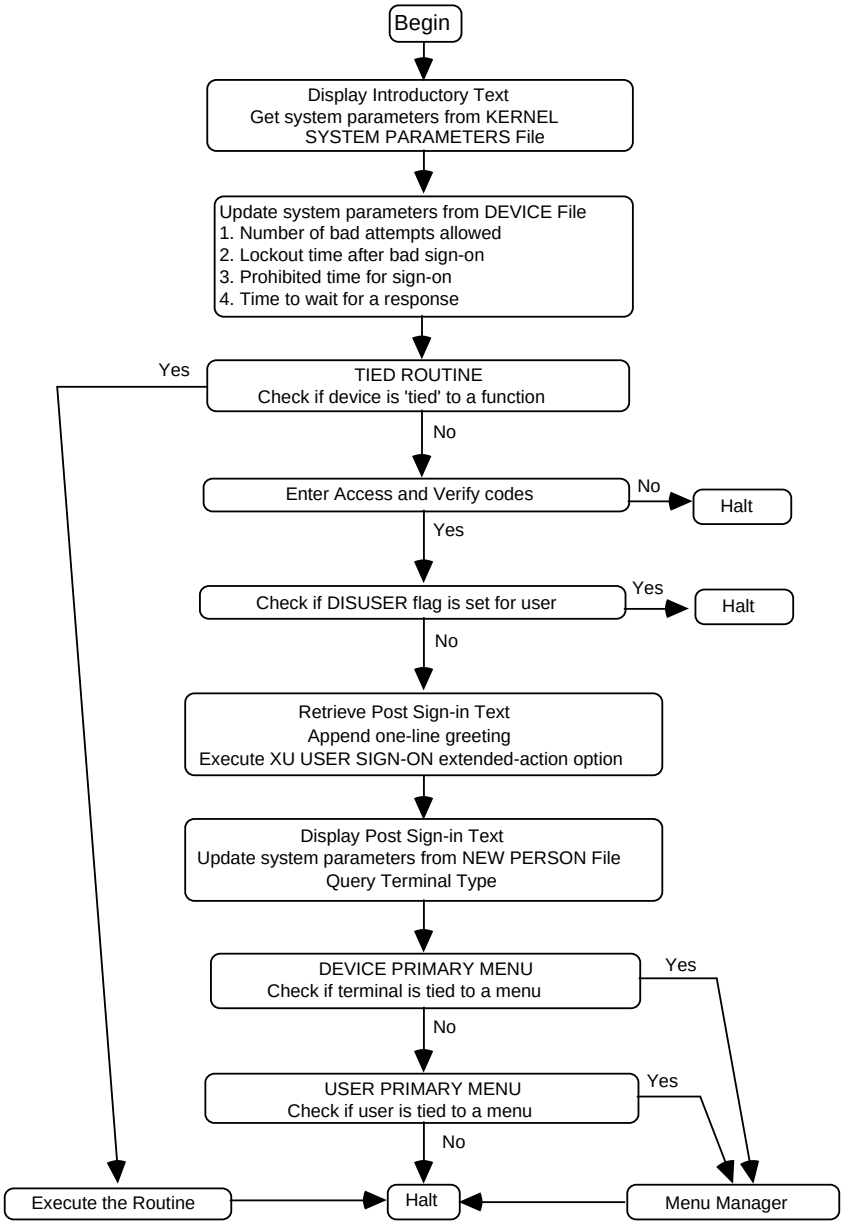
Signon Attempts and Device Lock-out Times: The DEFAULT # OF ATTEMPTS field in the KERNEL SYSTEM PARAMETERS file (#8989.3) holds the default limit of the number of times a user can try to enter a valid Access and Verify code pair. When the limit is reached, Signon/Security is unresponsive for the duration specified by the DEFAULT LOCK-OUT TIME field. The values for number of attempts and lock-out time are overridden by any values for the current device specified by comparable fields in the DEVICE file (#3.5). Device values are ignored, however, if the BYPASS DEVICE LOCK-OUT site parameter in the KERNEL SYSTEM PARAMETERS file (#8989.3) is set to YES. In particular, the fields that are bypassed are OUT-OF-SERVICE DATE, SECURITY, and PROHIBITED TIMES FOR SIGN-ON. Device values are put back into effect for the current device if the DEVICE file's PERFORM DEVICE CHECKING field is set to YES.

MAX SIGNON ALLOWED: One Kernel site parameter used in the initial signon screening is MAX SIGNON ALLOWED. It is a field within the VOLUME SET field (Multiple) in the KERNEL SYSTEM PARAMETERS file (#8989.3). Its value sets an upper limit for number of M processes (interactive, background, and system) that can run concurrently on the specified Volume Set or CPU. The TASKMAN JOB LIMIT, a field in the TASKMAN SITE PARAMETERS file (#14.7), should be set to a number slightly lower than MAX SIGNON ALLOWED to leave room for a few interactive logons when TaskMan is busiest.

For OpenVMS sites: the OpenVMS interactive logins parameter (set by the DCL command SET LOGINS/INTERACTIVE) should be set to a number less than the Kernel MAX SIGNON ALLOWED to conserve system resources. If the OpenVMS limit is set too high in relation to the Kernel limit, users will try to access Kernel only to be rejected when reaching Signon/Security. That means that they would waste system resources by creating a new OpenVMS process and activating a Caché image, all to no avail.

PROHIBITED TIMES FOR SIGN-ON: Time periods can be specified, during which interval signons can be barred by device or by user. This is controlled by the PROHIBITED TIMES FOR SIGN-ON field in the DEVICE file (#3.5) and a comparable field in the NEW PERSON file (#200).

Figure 3-4. Signon flow chart



Multiple Sign-On Restriction: The DEFAULT MULTIPLE SIGN-ON field in the KERNEL SYSTEM PARAMETERS file (#8989.3) controls whether users can create two or more simultaneous sessions by signing on to more than one device. The setting is overridden by comparable fields in the DEVICE (#3.5) and NEW PERSON (#200) files, respectively. The value is checked at signon to prevent unauthorized multiple sessions.

If multiple signons are prohibited, problems can occur if users experience an abnormal exit such that the signon record cannot be cleared. To clear an individual user, the Release User option can be used (described later in this chapter). To make sure all users are clear when the system is brought up after a crash, IRM can use the Clear all users at startup option.

INTERACTIVE USER'S PRIORITY: The INTERACTIVE USER'S PRIORITY parameter in the KERNEL SYSTEM PARAMETERS file (#8989.3) should usually be left NULL. A setting here affects the job priority of interactive users and could result in poor response time.

ASK DEVICE TYPE AT SIGN-ON: The ASK DEVICE TYPE AT SIGN-ON parameter controls whether the user's current device at signon is queried for its display attributes (DA). Thus, the correct terminal type can be identified without prompting the user.

It is recommended that ASK DEVICE TYPE AT SIGN-ON be set to ASK so that Signon/Security performs the DA query and allows the Device Handler to set up the correct terminal type attributes. This has become more important with the advent of screen control. VA FileMan's Screen Editor and Screen Manager, for example, will not function properly if the terminal type recorded by Kernel fails to match the actual terminal type being used.

As with other parameters, the site default (ASK DEVICE TYPE AT SIGN-ON field in the KERNEL SYSTEM PARAMETERS file [#8989.3]) is overridden by a DON'T ASK setting for the device (like-named field in the DEVICE file [#3.5]), which would similarly be overridden by a DON'T ASK setting for the user (like-named field in the NEW PERSON file [#200]). A NULL value functions as ASK. The user override can be set by any user via the Edit User Characteristics option.

If the parameter is set to DON'T ASK, Signon/Security does not perform the DA query and assumes the user's last terminal type is still appropriate. Although the difference in resource consumption is negligible, the user can appreciate a split second's savings in time. Thus, bypassing the DA query can be acceptable, if the same terminal type is always being used. But if the user should sign onto another terminal type, problems can occur with the presentation of screen-oriented displays unless the user knows how to change the terminal type to match the actual current one.

If the device is non-ANSI-standard, Signon/Security may not find a DA but will continue to determine the terminal's identity by querying its answerback message. All known non-ANSI devices (e.g., Qume 102 terminal) should have their answerback messages programmed. This is accomplished by using the terminal type setup mechanism and entering C-QUME as the Qume 102's answerback message. The name *must* match an entry in Kernel's TERMINAL TYPE file (#3.2) to take effect. If the answerback message contains additional characters (e.g., a serial number), the message will *not* match an entry in the TERMINAL TYPE file (#3.2) and will be useless for signon purposes.

If the terminal's DA return code does not match an entry in the DA RETURN CODES file (#3.22), or if the terminal is non-ANSI and cannot be programmed with an appropriate answerback message, Signon/Security prompts the user to identify the terminal type if the user's ASK DEVICE TYPE AT SIGN-ON setting is set to ASK. This is the only case in which the terminal type prompt is asked during signon. The last terminal type used will be presented as the default (it is stored in the NEW PERSON file [#200]). If ASK DEVICE TYPE AT SIGN-ON is set to DON'T ASK, Signon/Security assumes that the last terminal type is appropriate and does not prompt the user for validation.

Display Attributes (DA) Return Codes: The DA RETURN CODES file (#3.22) is used to equate DA return codes to entries in the TERMINAL TYPE file (#3.2). You can use the DA Return Code Edit option to automate the population of the DA RETURN CODES file (#3.22).

REF: For more information, please refer to the "Display Attributes (DA) Return Codes" topic in the "Device Handler: System Management" chapter in this manual.

SELECTABLE AT SIGNON: IRM can also control which devices can be selected at signon with a field in the TERMINAL TYPE file (#3.2). The SELECTABLE AT SIGN-ON flag should be set to YES for all devices commonly used for sign on. Ordinarily, it should not be set for printers (e.g., P- terminal types P-DEC or P-OTHER). To allow the loading of ScreenMan forms and proper functioning of other screen-oriented displays, the flag should also not be set for PK- types, that is, printers with keyboards. This is not an actual restriction, however, but a recommendation.

LIFETIME OF VERIFY CODE: To insure that users change their Verify codes at periodic intervals, IRM should set the LIFETIME OF VERIFY CODE parameter in the KERNEL SYSTEM PARAMETERS file (#8989.3) to a certain number of days. The maximum number is 90 days and the minimum number is 1 day. Thus, sites can choose any number from 1-90 days before requiring users to change their Verify code. At the end of that period (e.g., every 90 days), users *must* then change their Verify codes. Signon/Security checks whether the Verify code needs to be changed, and if so, prompts the user at signon to enter a new Verify code.

AUTO-GENERATE ACCESS CODES: When assigning Access codes, the security officer or IRM staff can invent an alphanumeric string or can ask Kernel to generate one. If the AUTO-GENERATE ACCESS CODES site parameter in the KERNEL SYSTEM PARAMETERS file (#8989.3) is set to YES, only generated, cryptic codes can be assigned. It is not necessary to pick the first one presented; others can be generated for selection.

DEFAULT INSTITUTION and AGENCY: The institution running Kernel software is defined during the Kernel installation when prompted for the DEFAULT INSTITUTION in the KERNEL SYSTEM PARAMETERS file (#8989.3). This field is a pointer to the INSTITUTION file (#4). One or more institutional affiliations can also be associated with a user (e.g., a VA Outpatient Clinic and an Army Medical Center). This data is stored in the DIVISION field (Multiple) in the NEW PERSON file (#200). If a user is associated with more than one Institution, the user will be prompted at signon to select a division. In this way, the local variable DUZ(2) can be set to the appropriate value. If the user's DIVISION field (Multiple) is blank, the DEFAULT INSTITUTION field (File #8989.3) is used to define DUZ(2). Since the INSTITUTION file (#4) contains a pointer to the AGENCY file (#4.11), the signed-on user's agency affiliation can also be determined.

The KERNEL SYSTEM PARAMETERS file (#8989.3) also contains a field named AGENCY. This field is not a pointer but is instead a SET OF CODES (e.g., "N" for Navy). This field is presented for editing during Kernel installation. Its value is used at signon to set the DUZ("AG") local variable. Thus, the agency associated with the overall Kernel system can be determined.

AUTO MENU: The AUTO MENU flag, stored in the local variable DUZ("AUTO"), is used by Menu Manager to control whether all items on a menu are presented automatically after each cycle through the menu system. If the items are not displayed, the user can always invoke the display by entering a question mark ("**?**"). New users often like to see all the menu choices. Experienced users probably do not need to see the choices and the display can be suppressed to save system resources. The user setting for AUTO

MENU (in the NEW PERSON file [#200]) will override any comparable device setting (File #3.5), which will, in turn, override the site parameter default (File #8989.3). Users can edit the setting with the Edit User Characteristics option.

TYPE-AHEAD: If TYPE-AHEAD is disabled, any keystrokes that the user enters while computer system processes previously issued instructions will not register. If TYPE-AHEAD is enabled, keystrokes entered in advance of processing will be stored in the TYPE-AHEAD buffer and will be interpreted when the earlier process is finished. New users may experience unwanted results if TYPE-AHEAD is enabled and they had not anticipated the effect. Experienced users may prefer TYPE-AHEAD for efficiency. The user setting overrides the device setting, which, in turn, overrides the site parameter setting. Users can edit the setting with the Edit User Characteristics option.

TIMED READ: The value for the TIMED READ parameter is stored in the local variable DTIME and is used to calculate how long Kernel should wait before terminating a READ. If, for example, a user does not respond to a menu prompt in the number of seconds defined by the TIMED READ, Kernel will take steps towards signoff and, without subsequent user response, will halt the user session. The user setting overrides the device setting, which, as usual, will override the site default.

POST SIGN-IN MESSAGE: The POST SIGN-IN MESSAGE is similar to introductory text (i.e., INTRO TEXT field in File #8989.3), except that Kernel displays it only after a successful signon. Like the introductory text, you can edit the message text using the Enter/Edit Kernel Site Parameters option; alternately, you can use the Post sign-in Text Edit option [XUSERPOST], which is specially designed for this purpose:

Figure 3-5. Post sign-in Text Edit option

```
SYSTEMS MANAGER MENU ...                                          [EVE]
Operations Management ...                                    [XUSITEMGR]
   Post sign-in Text Edit                                    [XUSERPOST]
```

Applications can append information to the POST SIGN-IN MESSAGE (on a per-user, per signon basis only) by attaching to the new XU USER SIGN-ON option.

REF: For more information on the XU USER SIGN-ON option, please refer to the "Signon/Security: Developer Tools" chapter in the *Kernel Developer's Guide*.

XU USER SIGN-ON Option

The XU USER SIGN-ON option can attach action-type options to this extended-action-type option, so that software-specific actions can be performed at signon.

REF: For more information, please refer to the "Signon/Security: Developer Tools" chapter in the *Kernel Developer's Guide.*

Clear all users at startup Option

Figure 3-6. Clear all users at startup option

```
PARENT OF QUEUABLE OPTIONS ...                              [ZTMQUEUABLE OPTIONS]
  Clear all users at startup                                   [XUSER-CLEAR-ALL]
```

If multiple signons are prohibited, users may be prevented from signing on to the system when it is brought up after a crash (which can cause numerous abnormal exits). To prevent this problem from occurring, IRM can use the Clear all users at startup option [XUSER-CLEAR-ALL]. Kernel recommends this option be scheduled to run at system startup. Although this option can be invoked interactively without ill effects, it was designed as a background process, thus, it is placed along with other tasked options on the PARENT OF QUEUABLE OPTIONS menu.

REF: For information on how to release a single user, please refer to the "Release User option" in the "User Management Options" in the "Operations Menu" topic in this chapter.

Enabling and Disabling Logons

IRM has full control over whether logons are enabled. Access to a particular Volume Set can be disabled by setting the INHIBIT LOGONS? flag in the VOLUME SET file (#14.5). Setting the flag to YES sets the ^%ZIS("14.5","LOGON","*volume set*") node, whose presence disallows user logons. That is, logons through Signon/Security, invoking the ^ZU routine, will fail (terminals for user access are usually linked to ZU within the operating system setup. Some special terminals, like the console, are untied.) The ^%ZIS("14.5","LOGON","*volume set*") node is also checked after each cycle through the menu system; signed-on users will be logged off as soon as they return to a menu prompt.

Adding New Users

Creating a new user account involves adding a record to the NEW PERSON file (#200), assigning an Access code, and assigning a primary menu. You need the XUMGR security key to assign primary menu options. Even the at-sign ("@"; programmer access) is insufficient, as checked by the PRIMARY MENU OPTION field's input transform.

Figure 3-7. User Management menu options: Associated menu options when adding a new user

```
SYSTEMS MANAGER MENU ...                                              [EVE]
User Management ...                                                  [XUSER]
   Add a New User to the System                                   [XUSERNEW]
   Grant Access by Profile <locked: XUMGR>                        [XUSERBLK]
   User Inquiry                                                   [XUSERINQ]
```

Add a New User to the System Option

You can use the Add a New User to the System option [XUSERNEW] to set up user accounts one-by-one. The option presents a standard scrolling-mode editing sequence for user attributes.

When using this option, entry of a social security number in the SSN field (#9) is usually required. While SSN is not required in the NEW PERSON file (#200) data dictionary, it is a required field when using this option. If the option is used by someone who holds the XUSPF200 security key, however, entry of an SSN is *not* required.

You can also print security forms for the new user with this option.

When signing on for the first time, the new user should simply press **<Enter>** at the "Verify code" prompt, which then lets them enter their own secret Verify code.

NEW PERSON File (#200) Required Fields

When adding new users, a default set of fields is required, at a minimum. This set is defined by the NEW PERSON IDENTIFIERS field in the KERNEL SYSTEM PARAMETERS file (#8989.3). If it is NULL, the default set of required fields for the NEW PERSON file (#200) entries is:

- INITIAL (#1)
- SEX (#4)
- SSN (#9)

If, given local site policy, a different set should be used, IRM can use this field to specify other identifiers.

NOTE: SSN is *not* required if the person entering accounts holds the XUSPF200 security key.

Grant Access by Profile Option

The Grant Access by Profile option [XUSERBLK] includes features unavailable in the Add a New User to the System option [XUSERNEW]. With the Grant Access by Profile option you can grant access to one or more people based on a typical user profile. All characteristics of the typical user, including menus, keys, and service/section, are copied to the new user or replace the characteristics of an existing user. For new users, access security forms are generated as part of the process. These forms can be delivered to the service/section coordinator by inter-office mail and can be distributed to the new users.

The Grant Access by Profile option is locked with the XUMGR security key and is strictly limited for use by IRM. It *must* be restricted, because any user profile, even that of a developer, can be copied to another user. As with the Add a New User option, the SSN field (#9) is required when adding new records except by holders of the XUSPF200 security key or if another default set of New Person Identifiers has been defined.

Access is assigned according to an existing user profile. Characteristics of the new user are cloned from the existing one. Rather than copying the characteristics from an actual user, creating several dummy users with profiles of typical positions can be worthwhile. A user (e.g., PHARMACY,TECH or RESIDENT,SURGERY) could be created with the appropriate user attributes, including menu options, keys, and service/section codes.

Several steps are involved in copying access to new or existing users. First you enter the name of the user account to clone from. Then, optionally, you can specify a TERMINATION DATE. Next, you enter the names of the new users to create. The system will pause for each new user as it verifies identifiers, checks for duplicates, and updates the NEW PERSON file (#200). You *must* enter a device upon which to print the computer account notification letters. You can either run the access assignment immediately or queue it for a later time.

Security Forms

Figure 3-8. Reprint Access agreement letter option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
User Management ...                                                  [XUSER]
   Reprint Access agreement letter                               [XUSERREPRINT]
```

Two security forms are printed for each new user:

1. **The Computer Account Notification—**Includes the user's auto-generated Access code and the name of the service/section coordinator who can answer questions.
2. **The Computer Access Policy—**A contract to which users *must* adhere. It states the terms of granting access to sensitive information; the user *must* accept these terms as a condition of being given system access.

These security forms are stored in the XUSER COMPUTER ACCOUNT help frame and should be edited for local use as follows:

1. Copy the XUSER COMPUTER ACCOUNT help frame into a new site help frame (e.g., SFO COMPUTER ACCOUNT).
2. Edit the security forms for local use. Replace the "placeholder" text with the actual name and address of the facility.
3. Repoint the Kernel Parameter to the new site XUSER COMPUTER ACCOUNT help frame using VA FileMan.

For example:

Figure 3-9. Security Forms—Sample user entries (1 of 4)

```
>D ^XUP

Setting up programmer environment
This is a TEST account.

Terminal Type set to: C-VT320

You have 13 new messages.
Select OPTION NAME: SYSTEMS MANAGER MENU

          Device Management ...
          Programmer Options ...
          Operations Management ...
          Spool Management ...
          Information Security Officer Menu ...
          Taskman Management ...
          User Management ...
          Application Utilities ...
          Capacity Management ...
          Manage Mailman ...
          Menu Management ...
          VA FileMan ...
          Verifier Tools Menu ...

Select Systems Manager Menu Option: VA FileMan

          VA FileMan Version 22.0

          Enter or Edit File Entries
          Print File Entries
          Search File Entries
          Modify File Attributes
          Inquire to File Entries
          Utility Functions ...
          Data Dictionary Utilities ...
          Transfer Entries
          Other Options ...

Select VA FileMan Option: TRANSFER ENTRIES

Select TRANSFER OPTION: TRANSFER FILE ENTRIES

INPUT TO WHAT FILE: HELP FRAME// HELP FRAME <Enter>    (562 entries)
TRANSFER FROM FILE: HELP FRAME// <Enter>
TRANSFER DATA INTO WHICH HELP FRAME: ISC COMMPUTER ACCESS
Not a known package or a local namespace.
  Are you adding 'ISC COMMPUTER ACCESS' as a new HELP FRAME (the 563RD)?
No// Y  (Yes)
   HELP FRAME NUMBER: 742// <Enter>
   HELP FRAME HEADER: Computer Access
TRANSFER FROM HELP FRAME: XUSER COMPUTER ACCOUNT <Enter>  Batch user
access document
```

```
   WANT TO DELETE THIS ENTRY AFTER IT'S TRANSFERRED? No// <Enter>  (No)
...SORRY, LET ME THINK ABOUT THAT A MOMENT...
SINCE THE TRANSFERRED ENTRY MAY HAVE BEEN 'POINTED TO'
BY ENTRIES IN THE 'HELP FRAME' FILE, ETC.,
DO YOU WANT THOSE POINTERS UPDATED (WHICH COULD TAKE QUITE A WHILE)? No//
<Enter>  (No)

          Enter or Edit File Entries
          Print File Entries
          Search File Entries
          Modify File Attributes
          Inquire to File Entries
          Utility Functions ...
          Data Dictionary Utilities ...
          Transfer Entries
          Other Options ...

Select VA FileMan Option: ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: HELP FRAME// <Enter>
EDIT WHICH FIELD: ALL// TEXT <Enter>   (word-processing)

Select HELP FRAME NAME: ISC COMMPUTER ACCESS <Enter>     Computer Access
NAME: ISC COMMPUTER ACCESS// <Enter>
HEADER: Computer Access// <Enter>
TEXT:  . . .
       . . .
suspension/termination of access privileges.

I affirm with my signature that I have read, understand, and agree to
fulfill the provisions of this User Access notice.

|INDENT(5)||WIDTH(75)||NOWRAP|
Signature:_____________________________
          |#20.2| |#29|
RETURN THIS FORM TO: IRMS - NEW ACCTS (xxx/xxx)

  Edit? NO// YES
```

Figure 3-10. Security Forms—Sample user entries (2 of 4)

```
==[ WRAP ]==[ INSERT ]=============< TEXT >============[ <PF1>H=Help ]====
|INDENT(5)| |WIDTH(70)|
|NOWRAP|
|CENTER("USER ACCOUNT NOTIFICATION")|

|CENTER("Department of Veterans Affairs")|
|CENTER("SuperStar VAMC")|
|CENTER("123 Any Street")|
|CENTER("Any Town, ST., 99999")|
|XUVT(12)|
|#20.2|
|#29|   ( |#29:#1.5| )
|XUVT(19)|
---
 |WRAP|


A user account has been created in your name to enable you to access
on-line clinical and/or administrative data required to perform your
duties as an employee of the Department of Veterans Affairs.  Please read
<=======T=======T=======T=======T=======T=======T=======T=======T=======T>

Select RELATED FRAME KEYWORD: <Enter>
Want to LOAD KEYWORDS (Y/N)?: N
Select INVOKED BY ROUTINE: <Enter>
Select EDITOR: <Enter>
Select OBJECT: <Enter>
ENTRY EXECUTE STATEMENT: <Enter>
EXIT EXECUTE STATEMENT: <Enter>
Select HELP FRAME NAME: <Enter>


          Enter or Edit File Entries
          Print File Entries
          Search File Entries
          Modify File Attributes
          Inquire to File Entries
          Utility Functions ...
          Data Dictionary Utilities ...
          Transfer Entries
          Other Options ...

Select VA FileMan Option: ENTER OR EDIT FILE ENTRIES


INPUT TO WHAT FILE: HELP FRAME// 8989.2 <Enter>  KERNEL PARAMETERS    (6
entries)
EDIT WHICH FIELD: ALL// <Enter>


Select KERNEL PARAMETERS NAME: XUSER COMPUTER ACCOUNT
NAME: XUSER COMPUTER ACCOUNT  Replace <Enter>
TYPE: <Enter>
DEFAULT: <Enter>
REPLACEMENT: ISC COMPUTER ACCESS
```

Read through and edit entries specific to your site information and save your changes.

```
Select KERNEL PARAMETERS NAME: <Enter>


          Enter or Edit File Entries
          Print File Entries
          Search File Entries
          Modify File Attributes
          Inquire to File Entries
          Utility Functions ...
          Data Dictionary Utilities ...
          Transfer Entries
          Other Options ...

Select VA FileMan Option: <Enter>


   FM     VA FileMan ...
          Core Applications ...
          Device Management ...
          Information Security Officer Menu ...
          Manage Mailman ...
          Menu Management ...
          Operations Management ...
          Programmer Options ...
          Spool Management ...
          Taskman Management ...
          User Management ...

Select Systems Manager Menu Option: USER MANAGEMENT

          Add a New User to the System
          Grant Access by Profile
          Edit an Existing User
          Deactivate a User
          Reactivate a User
          List users
          User Inquiry
          Switch Identities
          File Access Security ...
          Clear Electronic signature code
          Electronic Signature Block Edit
          Manage User File ...
          OAA Trainee Registration Menu ...
          Person Class Edit
          Reprint Access agreement letter

Select User Management Option: REPRINT ACCESS AGREEMENT LETTER
Select NEW PERSON NAME: REQUEST,ACCESS <Enter>     AR     COMPUTER
SPECIALIST

 Is REQUEST,ACCESS the one you want? YES// <Enter>
DEVICE: 0;80;60 <Enter>  Telnet Terminal
```

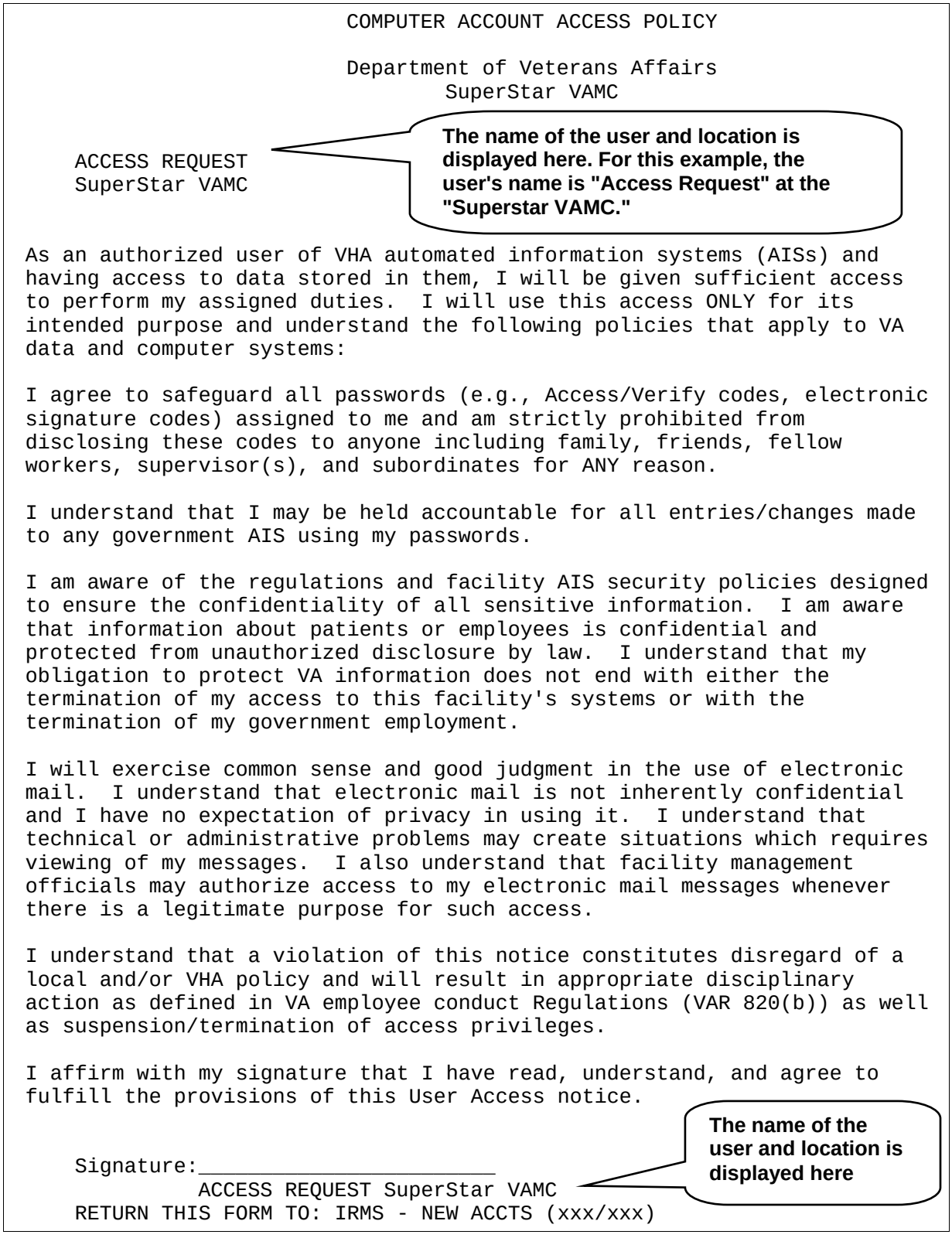
Figure 3-11. Security Forms—Sample User Account Notification form (3 of 4)

```
                        USER ACCOUNT NOTIFICATION

                      Department of Veterans Affairs
                           Superstar VAMC
                              123 Any Street
                            Any Town, ST. 99999


ACCESS REQUEST
Superstar VAMC


---


A user account has been created in your name to enable you to
access on-line clinical and/or administrative data required to
perform your duties as an employee of the Department of Veterans
Affairs.  Please read the enclosed NEW USER INFORMATION before
you attempt your first log-on to the system.  Questions about
access should be referred to the AIS Application Coordinator in
your service, your facility Information Security Officer (ISO),
or your IRM Service.


Your Computer Access Coordinator is:
                                XUUSER,ONE
                                123X
                                510-555-9999
Your Facility Information Security Officer:
                                Two Xuser
Your Alternate Information Security Officer:
                                Three Xuser

---


                                  NT Domain: _____________
                                NT Username: VHA________
                                NT Password: _____________

                                VistA Access Code: _________
                                VistA Verify Code: _________
```

The name of the user and location is displayed here. For this example, the user's name is "Access Request" at the "Superstar VAMC."

The names and contact information specific to your site will be displayed here.

Figure 3-12. Security Forms—Sample Computer Account Access Policy form (4 of 4)



VA FileMan word-processing "windows" are used to retrieve the user's name, service/section, and service/section coordinator's name. To be effective, the SERVICE/SECTION field in the NEW PERSON file (#200) *must* be filled in for the new user. The COORDINATOR (IRM) field, a field in the SERVICE/SECTION file (#49), *must* also be filled in and updated when necessary. Word-processing "windows" are also used for formatting, like |TOP|, to separate the two forms. When using the File Access Security system, READ access to the SERVICE/SECTION file (#49) is needed to retrieve the Coordinator's name within the window command.

REF: For more information on using word-processing "windows," the File Access Security system, and navigation, please refer to the *VA FileMan Getting Started Manual.*

The Reprint Access Agreement Letter option allows you to reprint the computer access agreement letter in case there was a problem printing the first form (e.g., the first form is jammed in the printer). It does not reprint the Access code on the letter, however.

Editing an Existing User Option

```
SYSTEMS MANAGER MENU ...                                          [EVE]
User Management ...                                              [XUSER]
   Edit an Existing User                                      [XUSEREDIT]
```

The attributes of an existing user can be edited with the Edit an Existing User option [XUSEREDIT]. This option invokes a screen-oriented display, using ScreenMan.

It is impossible to exit the form and save changes unless all required fields (e.g., the SERVICE/SECTION field in the NEW PERSON file [#200]) are filled in.

Listed below are descriptions of each of the user attributes you can edit with the Edit an Existing User option:

NAME (required): The user's name should be entered in capital letters. The syntax should be "LAST,FIRST MI." with only a comma (no spaces) between the last and first name. A middle initial can follow, separated with a space and followed with a period. It is not appropriate to add credentials (e.g., M.D.), since there are other ways to specify such additional information (by the Title and the Signature Block Printed Name). Furthermore, the parsing algorithms commonly used in software applications only recognize two pieces, before and after the comma, rearranging them and using uppercase/lowercase to generate "First MI. Last".

INITIAL: The user's initials can be entered, usually two or three capital letters with no spaces. The NEW PERSON file (#200) contains a lookup-type cross-reference by INITIAL (C), so if the INITIAL field is filled in, the user can be found in the NEW PERSON file (#200) by entering the initials. For example, just the initials can be used at the "Select NEW PERSON Name:" prompt, or when addressing mail messages, or for other lookup purposes. Users can edit their initials at any time since this field is included in the common option Edit User Characteristics.

TITLE: The TITLE field points to the TITLE file (#3.1), a file exported with Kernel but without data (records). The User Management options to add or edit a user's record allow LAYGO into the TITLE file (#3.1), so titles can be added via the NEW PERSON file (#200). Although not required, it may be wise to assign appropriate titles to users, so this field can be referenced by other software applications. MailMan, for example, displays titles in message headers if the user who is reading mail has so indicated with a flag in MailMan's Edit User Options called Show Titles.

NICK NAME: Like INITIAL, NICK NAME has a lookup type cross-reference (D) in the NEW PERSON file (#200) so that lookups will succeed simply by using the NICK NAME. This field is also included in Edit User Characteristics.

Figure 3-13. Screen 1 of Edit an Existing User option

```
                          EDIT AN EXISTING USER
NAME: USER8,ANTONY                                          PAGE 1 OF 3


     NAME: KRNUSER,EIGHT                          INITIAL: EK
    TITLE: HISTORIAN                              NICK NAME: EIG
      SSN: 000888888                              MAIL CODE:

           PRIMARY MENU OPTION: XMUSER
  Select SECONDARY MENU OPTIONS: ZZ MAIN

 Want to edit ACCESS CODE (Y/N):     FILE MANAGER ACCESS CODE:
 Want to edit VERIFY CODE (Y/N):

              PREFERRED EDITOR:
               Select DIVISION: GREAT BRITAIN
               SERVICE/SECTION: INFORMATION RESOURCES MGMT

 Exit     Save    Next Page    Refresh

 Enter a command or '^' followed by a caption to jump to a specific field.

 COMMAND:                          Press <PF1>H for help   INSERT
```

SSN: The SSN field (#9) is not a required field in the data dictionary for the NEW PERSON file (#200). SSN is required when using the User Management options to add a new user unless the XUSPF200 security key is held by the person using the option. It is highly recommended that each new user have the SSN field (#9) filled in to minimize the problem of subsequent duplicate entries. Since many existing users do not have an SSN entered, however, the Edit an Existing User option does not require that one be entered.

MAIL CODE: The user's MAIL CODE can be entered for purposes of interoffice routing of manually delivered mail.

PRIMARY MENU OPTION (required for functional access): Users *must* be assigned a PRIMARY MENU OPTION in order to reach Menu Manager after successfully entering Access and Verify codes.

The PRIMARY MENU OPTION should provide a route to all the computing functions the user can be expected to need. The XUMGR security key *must* be held by the person assigning the menu (unless delegated options are available for use with the Secure Menu Delegation system).

REF: Building and rearranging menus is discussed in the "Menu Manager: System Management" chapter in this manual.

SECONDARY MENU OPTIONS: The SECONDARY MENU OPTIONS can be used to assign particular options to individual users to customize their menu choices. While a user may have a standard primary menu to carry out the usual functions of a department or service, additional special functions just for this user can be assigned as secondary options. This is a multiple field, unlike the PRIMARY MENU OPTION, so additional items can easily be added.

ACCESS CODE/VERIFY CODE Edit: These fields can be used to edit a user's Access or Verify Code as needed. If a user has forgotten the Verify code, or needs a new one, IRM/ISO should delete the existing code so that when the user logs on and presses the **<Enter>** key at the "VERIFY CODE" prompt, a new (secret) password (VERIFY CODE) can be entered. To accomplish this, "**Y**" should be entered at the "Want to edit VERIFY CODE (Y/N) :" prompt. An at-sign ("**@**") should then be entered to delete the existing code. The change will be filed immediately, unlike other changes that will be processed as part of the overall transaction when leaving the ScreenMan form.

Users can edit their Verify code at any time via the Edit User Characteristics option on the Common menu. If this option uses a local template, the ability to edit the VERIFY CODE field should probably remain, as a security measure. IRM can choose to add the ability to edit the ACCESS CODE field as well.

FILE MANAGER ACCESS CODE: The FILE MANAGER ACCESS CODE (i.e., FILE MANAGER ACCESS CODE field [#3] in the NEW PERSON file [#200]) is stored in the local variable DUZ(0). If DUZ(0)=**@**, the user is a developer with the highest level of programmer access authority. Other non-reserved symbols can be assigned for File Access Security, depending on the user's needs. Software applications will indicate which symbols are needed for site-specific File Access Security.

NOTE: In previous documentation and data dictionaries, it has been *implied* that the pound sign ("#") symbol/character was reserved for File Access Security for IRM, however, this is *not* true. It has merely been used as a *convention.*

If the File Access Security conversion has been run, the FILE MANAGER ACCESS CODE is *not* used to control file-level access security as it was *before* the conversion. The File Access Security system (formerly known as Part 3 of the Kernel installation) permits the association of a user with a file whereby explicit access can be granted. While the conversion process is somewhat involved, the benefits resulting from implementing the File Access Security system are worthwhile.

Even after running the file access conversion, the FILE MANAGER ACCESS CODE field continues to serve several functions:

- If a user has been granted full file access privileges for a particular file, a further restriction can be placed at the file or field level to prohibit modification of the definition or entry of data. Files have top-level restrictions of READ, WRITE, or DELETE access as do fields and templates.
- If the file, field, or template is protected with the at-sign ("@"; programmer access), the user *must* also have the at-sign in the FILE MANAGER ACCESS CODE field (#3) in the NEW PERSON file (#200).

The Device Handler also checks the FILE MANAGER ACCESS CODE of the user if the SECURITY field in the DEVICE file (#3.5) has been defined with a character string. The user would not be able to select the device unless at least one of the characters in the user's code matched at least one character in the device code.

The most important FILE MANAGER ACCESS CODE field (#3) character is the at-sign ("@"; programmer access). It has special meaning and overrides other file access restrictions or other FILE MANAGER ACCESS CODE field (#3) characters. It is *not* recommended that the at-sign be allocated unless absolutely needed. Allocation is, in part, restricted by the fact that only those few users who have developer access to the system can give other users the at-sign.

NOTE: A SET statement from programmer mode can be used to temporarily assign DUZ(0)="@" *without* storing the code in the NEW PERSON file (#200), which would give permanent programmer access.

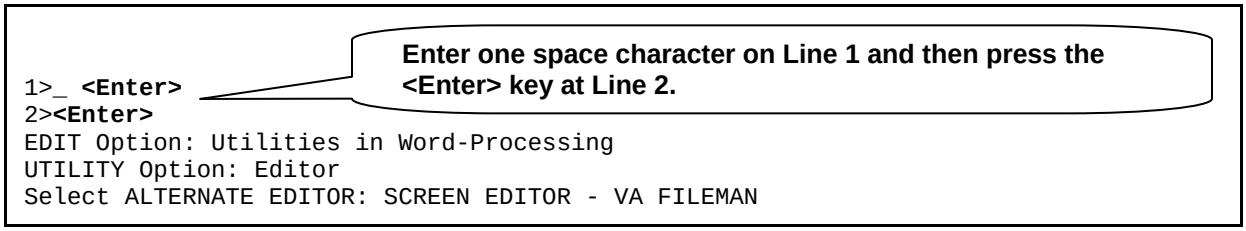
Use of the at-sign ("@"; programmer access) is less common now than in the past since alternative security measures have been developed. It is still required for several critically sensitive checks, however, such as entering M code into VA FileMan files (e.g., OPTION [#19] and FUNCTION [#.5] files).

REF: For more information on File Access Security, please refer to Chapter 5, "File Access Security," in this manual and the *VA FileMan (Version 22.0) and Kernel (Version 8.0) File Access Security* supplemental documentation located on the VHA Software Document Library (VDL) at the following Web address:

http://www.va.gov/vdl/application.asp?appid=5

PREFERRED EDITOR: If a user's PREFERRED EDITOR field is NULL, Kernel uses VA FileMan's Line Editor to edit word-processing fields. If the PREFERRED EDITOR is set to another entry in the ALTERNATE EDITOR file (#1.2), like VA FileMan's Screen Editor, Kernel uses that editor when the user edits word-processing fields. As described in VA FileMan's documentation, users can switch from the Line Editor to another editor by using the Utility sub-option on the Edit option menu.

Figure 3-14. VA FileMan Line Editor—Sample user dialogue



If the PREFERRED EDITOR is the Screen Editor, it is also possible to switch to another editor, like the Line Editor, to take advantage of line editor features such as File Transfer from Foreign CPU.

NOTE: Other editors (e.g., WordMan or VA LetterMan) do *not* support switching to the Line Editor, which may be a limitation in some circumstances.

This field is also included in Edit User Characteristics and MailMan's Edit User Options so that all users can define a PREFERRED EDITOR if they so choose.

DIVISION: The DIVISION field (Multiple) has a corresponding site parameter, the Default Institution, that sets users' DUZ(2) if this field is not filled in. A user setting, however, takes precedence over the site parameter. This is a multiple field and if the user is associated with more than one institution, the user is prompted at signon to pick the one corresponding to the computing activities to be carried out in that session.

SERVICE/SECTION (required): This field points to the SERVICE/SECTION file (#49) distributed with Kernel's virgin installation. No data is included. It is a required field since applications have begun to use it in various utilities. Kernel's CPU/Service/User/Device Stats option [XUSTAT], for example, can summarize signon information for all users in the same Service/Section. The Grant Access by Profile option also makes use of this field to specify the Service/Section Coordinator to whom the access forms of the new users should be delivered.

TIMED READ: As discussed with other site parameters earlier in this chapter, TIMED READ defines the length of time Kernel should wait for a user response to a READ. A setting for the user attribute overrides the site default. It is used to define the local variable DTIME.

MULTIPLE SIGN-ON: As discussed with other site parameters, MULTIPLE SIGN-ON controls whether the user will be permitted to have two or more concurrent signon sessions. The user setting takes precedence.

AUTO MENU: As discussed with other site parameters, AUTO MENU controls whether the entire list of menu options is automatically presented or whether the user needs to enter a question mark ("**?**") to invoke the display. The user setting takes precedence.

ASK DEVICE TYPE AT SIGN-ON: As discussed with other site parameters, the ASK DEVICE TYPE AT SIGN-ON field controls whether the device being used at signon will be queried for its terminal type. The user setting takes precedence.

TYPE-AHEAD: This field controls whether the user can enter text faster than the computer can read it. If set to YES, the computer buffers input from the user. If set to NO, keystrokes from the user are lost if they are typed faster than the computer can process them.

ALLOWED TO USE SPOOLER: This field controls whether a user can pick the spool device at the device prompt to send output to the spooler.

PAC (Programmer Access Code): For users who have been granted the Programmer Mode option along with the XUPROG and XUPROGMODE security keys, a Programmer Access Code can be assigned as additional security. If a PAC is defined, Kernel prompts for the PAC just before allowing a user to enter programmer mode. If this field is NULL, a PAC is not asked.

CAN MAKE INTO A MAIL MESSAGE: This field controls whether a spooled document can be transformed into a regular mail message for use within MailMan.

DISUSER: If set to YES, disables access to the system for this user (without terminating the user's account).

FILE RANGE: Users who have VA FileMan privileges to create files can be given a numeric range of numbers to use as file numbers. Assigning number ranges acts as a safeguard to keep users from picking a number within a range that is nationally reserved for VistA software applications. It can also serve local database administration needs of segmenting local development by number ranges.

TERMINATION DATE: As mentioned in the topic about terminating users, the TERMINATION DATE indicates when a user's access privileges should be revoked.

ALWAYS SHOW SECONDARIES: If set to YES, contents of a user's SECONDARY MENU OPTIONS are shown when the user enters one question mark ("**?**") at a menu prompt. Otherwise, the user *must* enter two question marks ("**??**") to see their secondary menu.

PROHIBITED TIMES FOR SIGN-ON: As discussed with other signon parameters, PROHIBITED TIMES FOR SIGN-ON can be used to regulate when the user can sign on. The user setting takes precedence over any corresponding device setting.

PHONE (HOME), OFFICE PHONE, PHONE #3, PHONE #4, COMMERCIAL PHONE, FAX NUMBER: Set up phone numbers for the user in these fields.

VOICE PAGER, DIGITAL PAGER: Set up pager numbers for the user in these fields.

LANGUAGE: Overrides the setting of the DEFAULT LANGUAGE field in the KERNEL SYSTEM PARAMETERS file (#8989.3). Both of these are used to set the DUZ("LANG") flag for each user. VA FileMan uses this setting to enable the display of language-specific dates and times, numeric formats, and dialogues.

Additional Attributes Editable by Users

Some but not all of the user attribute fields can be edited by users using the Edit User Characteristics option. The only field the user can edit that is not part of the system manager's Edit an Existing User form is the TEXT TERMINATOR field.

REF: For a description of the fields users can edit (using the default Edit User Characteristics form and template), please refer to the "Signon/Security: User Interface" chapter in this manual.

Edit User Characteristics Form and Template

Kernel exports a ScreenMan form and a template to be used in the Edit User Characteristics option. Both are called XUEDIT CHARACTERISTICS. The INPUT template by the same name is invoked if the ScreenMan form cannot be loaded on the current terminal type.

IRM can substitute a locally-developed template by entering its name in the USER CHARACTERISTICS TEMPLATE field in the KERNEL PARAMETERS file (#8989.2). IRM can also design a customized form with the same name as the local INPUT template that will be displayed instead, terminal type setup permitting. In other words, to invoke a locally modified display, an INPUT template *must* exist. If a ScreenMan form by the same name also exists, an attempt will be made to display the form before defaulting to the INPUT template.

REF: For more information on creating a local Edit User Characteristics form and template, please refer to the *Kernel Installation Guide*.

Deactivating and Reactivating Users

Kernel provides options to deactivate and reactivate users on the User Management menu [XUSER]. When users no longer need access privileges, IRM can partially or entirely close access to their account.

Figure 3-15. User Management menu options

```
SYSTEMS MANAGER MENU ...                                          [EVE]
User Management ...                                             [XUSER]
   Deactivate a User                                        [XUSERDEACT]
   Purge Inactive Users' Attributes                      [XUSERPURGEATT]
   Reactivate a User                                        [XUSERREACT]
```

Deactivating Users

The Deactivate a User option [XUSERDEACT] lets you temporarily or permanently disable access for users. You can schedule termination of a user for a future date. The Deactivate a User option loads a ScreenMan form with four fields, described below.

DISABLE USER: Setting the DISABLE USER field to YES prevents a user from signing on, but leaves all of their menus, keys, and other attributes (essentially the user's entire account) still enabled. It sets a field in the user's NEW PERSON file (#200) entry, called DISUSER.

You might want to use the Disable User feature to prevent access to your system by an external support person, except during pre-approved times (where you may want to monitor their actions). Setting DISUSER to YES prevents them from logging on to the system until you clear the field.

If you set Disable User to YES, *do not set any other fields* in the Deactivate a User form (they only apply to terminating a user). Then, to re-enable access, use the Reactivate a User option.

REF: For a description of the Reactivate a User option, please refer to the "Reactivating Users" topic that follows.

TERMINATION DATE: Terminating a user is the way to formally deactivate a user (as opposed to temporarily disabling their account). Setting a TERMINATION DATE effectively terminates that user's account, effective from the TERMINATION DATE forward.

The Deactivate a User option automatically performs the following steps when you deactivate a user:

- Revokes the user's status as an authorized sender of any mail groups.
- Revokes the user's status as a surrogate.
- Revokes the user's status as a Secure Menu Delegation delegate.
- Deletes the user's Access code, Verify code, Electronic Signature code, VA FileMan Access code (i.e., FILE MANAGER ACCESS CODE field [#3]), and Programmer Access code.
- Deletes the user's menu templates.

- Deletes the user's delegated options.
- Purges the ^DISV global on that CPU for that user.

You can also decide whether all mail messages and all security keys for the account will be deleted on the TERMINATION DATE with the final two fields in the Deactivate a User option [XUSERDEACT] (DELETE ALL MAIL ACCESS and DELETE KEYS AT TERMINATION). If the user is expected to return to the facility and will need to have the user account reopened, security keys and mail could be retained.

REF: For more information on cleaning up user access and privileges at termination, please refer to the "XU USER TERMINATE Option" topic in the "Signon/Security: Developer Tools" chapter in the *Kernel Developer's Guide*.

DELETE ALL MAIL ACCESS: Setting the DELETE ALL MAIL ACCESS field causes all mail messages for the user to be deleted when their account is terminated on the TERMINATION DATE.

DELETE KEYS AT TERMINATION: Setting the DELETE KEYS AT TERMINATION field causes all security keys for the user to be deleted at termination (except security keys marked "KEEP AT TERMINATE").

As discussed in the "Security Keys" chapter in this manual, the application developer can export a security key with the KEEP AT TERMINATE field set to YES in such a situation. The Provider security key, included with Kernel, has the flag set to YES for this purpose. Although a user may have been deactivated, it could be important to continue a processing activity that the user had authorized, based on privileges associated with a security key. A medical order could continue to hold an approved status, for example, even though the authorizing provider had been deactivated.

Automatically Deactivating Users

Kernel's XUAUTODEACTIVATE option finds all users in the NEW PERSON file (#200) with a TERMINATION DATE in the past, but who still have an Access code. Any such users are users that had been scheduled for termination but who were not terminated (usually because the task that should have terminated them did not run). XUAUTODEACTIVATE terminates any users it finds in this category. It acts as a safety net to ensure that all users who were scheduled for termination are, in fact, terminated. It should be regularly scheduled.

REF: For recommended frequency of scheduling, please refer to the *Kernel Installation Guide*.

Because this option is not intended for interactive use it is placed on the ZTMQUEUABLE OPTIONS menu.

Purging Mail and Security Keys for Inactive Users

You can use the Purge Inactive Users' Attributes option [XUSERPURGEATT] to clean up files. It removes all mailboxes, messages, mail groups, and security keys for users who have been terminated. If any of these users still retain Access codes, they are deleted.

This is particularly significant with mail. A mail message cannot be completely removed from a system until all recipients have deleted it from their mail baskets. If a user is no longer active, then it becomes unlikely that the message will ever get purged.

There are two modes of running this option. You can VERIFY the process for each user that the computer selects as eligible. If you choose not to verify the process for each user, then for every user with a non-future TERMINATION DATE, their set of security keys, mail groups, messages, and mail baskets will be deleted.

Reactivating Users

You can use the Reactivate a User option [XUSERREACT] to re-enable access for a user who has either been terminated, or whose access has been temporarily disabled. To re-enable access for someone whose account is merely disabled (with the DISUSER field set to YES), use this option to simply clear the DISUSER field. Otherwise, using this option, you can fill in all the fields needed for an active account (i.e., FILE MANAGER ACCESS CODE field [#3], PRIMARY MENU OPTION, etc.).

When you reactivate a user, you are asked whether to deny access to old mail messages. If the reactivated user account is a less privileged account than previously, it may be appropriate to deny the user access to messages that were received in the user's prior capacity. Even if that user's mailbox was deleted at termination, once the user is reactivated, an old message would be delivered if responded to by another recipient.

User Management Options on the Operations Menu

Kernel provides a set of options for IRM to monitor and support users logged onto the system. These options are on the User Management Menu [XUOPTUSER], under the Operations Management menu [XUSITEMGR], as shown below:

Figure 3-16. User Management Menu options

```
SYSTEMS MANAGER MENU ...                                          [EVE]
Operations Management ...                                      [XUSITEMGR]
   User Management Menu ...                                    [XUOPTUSER]
      Find a user                                            [XU FINDUSER]
      List users                                               [XUSERLIST]
      Print Sign-on Log                                        [XUSC LIST]
      Release user                                              [XUSERREL]
      User Inquiry                                              [XUSERINQ]
      User Status Report                                     [XUUSERSTATUS]
```

Find a User Option

You can use the Find a User option [XU FINDUSER] to find a user who is currently signed on to the system in this UCI group. If you are on the same CPU as the user, this option will also show the menu path of the user. The option finds users based on the "CUR" cross-reference of the SIGN-ON LOG file (#3.081).

List Users Option

The List Users option [XUSERLIST] lists all users known to the system.

Print Sign-on Log Option

The Print Sign-on Log option [XUSC LIST] prints out the SIGN-ON LOG file (#3.081).

Figure 3-17. Sample Kernel Sign-On Log report

```
USERS WHO HAVE SIGNED ONTO THE COMPUTER        MAY  3,2006  15:20    PAGE 1
                          ELAPSED
                             TIME
Sign-on time            (MINUTES)  USER               $I         NODE NAME
  IP ADDRESS
------------------------------------------------------------------------------

MAY  2,2006  10:28          -628   XUUSER,ONE         TNA35:     ISC6A2
  10.6.15.11
MAY  2,2006  10:28             0   XUUSER,TWO         TNA35:     ISC6A2
  10.6.17.191
MAY  2,2006  12:51             0   XUUSER,TWO         TNA35:     ISC6A2
  10.6.17.191
MAY  2,2006  14:00             3   XUUSER,TWO         TNA35:     ISC6A2
  10.6.17.191
MAY  2,2006  14:14             3   XUUSER,TWO         TNA35:     ISC6A2
  10.6.17.191
MAY  2,2006  14:45             6   XUUSER,TWO         TNA35:     ISC6A2
  10.6.17.191
```

Release user Option

If multiple signons are prohibited, problems can occur if users experience an abnormal exit such that the signon record cannot be cleared. IRM can use the Release user option [XUSERREL] to remedy the problem for individual users. To clear all users on startup, schedule the Clear all users at startup option.

User Inquiry Option

The User Inquiry Option option [XUSERINQ] displays various attributes of a specified user. If the user is currently signed on, it displays the job and device numbers, the signon time, and what option is being executed. Otherwise, it displays the last signon time. It also displays which security keys are held by the user.

User Status Report Option

The User Status Report option [XUUSERSTATUS] produces a report of the users currently signed on to this CPU and UCI. It shows the option each user is running and when they signed on, as well as their device and job numbers.

Signon Audits

Signon events are recorded in the SIGN-ON LOG file (#3.081). Statistics, such as the time of access and the user's identity, are stored for audit purposes. If the user exits normally (is not "bumped" off the system), the signon record will include the time of exit. If the user exits abnormally with an error or enters programmer mode, the signon record cannot include a time of exit.

Information about signon activity can be reviewed with options on the Operations and System Security menus.

The SIGN-ON LOG file (#3.081) is purged with the XUSCZONK option that should be tasked to run on a regular schedule (e.g., every night). This option cannot be reached from Menu Manager; like other options that should only be queued, it is on the PARENT OF QUEUABLE OPTIONS menu.

Signon Statistics

Statistics about active sessions can be obtained with the CPU/Service/User/Device Stats option [XUSTAT]. This option permits sorting by CPU, by the user's Service/Section (e.g., MAS) by individual users, or by particular devices.

Figure 3-18. CPU/Service/User/Device Stats option

```
SYSTEMS MANAGER MENU ...                                          [EVE]
Operations Management ...                                    [XUSITEMGR]
   CPU/Service/User/Device Stats                                [XUSTAT]
```

Failed Access Attempts Audit

When a user enters invalid Access and Verify code pairs, the number of attempts is recorded and the device appears to lock after the site parameter limit of failed access attempts is reached. After this point, Signon/Security continues to record what the user types (but only to create a record in the FAILED ACCESS ATTEMPTS LOG file [#3.05]). If a valid Access code is entered, Signon/Security can link the attempt with a known user and will record that user's name in the log. Since it is a valid code, its text is not recorded in the log. The text of subsequently entered invalid Verify codes can, however, be recorded as clues to the source of the access attempt. If the Access code is not valid, a user's name cannot be associated but the text of the attempt can be recorded. The log also records the time of day, device used, and CPU/UCI location.

Kernel Signon Auditing Files

Table 3-1. Kernel Signon Auditing Files

File	Global Location	Set Parameters	Display Parameters	Initiate/ Terminate	Print Reports	Purge Logs
SIGN-ON LOG (#3.081)	^XUSEC(0,	Predefined	N/A	Always done	Print Sign-on Log [XUSC LIST]	Purge Sign-on Log [XUSCZONK]
FAILED ACCESS ATTEMPTS LOG (#3.05)	^%ZUA(3.05,	Establish System Audit Parameters [XUAUDIT]	Display the Kernel Audit Parameters [XU-SPY-SHOW]	On/Off switch	Devices: Device Failed Access Attempts [XUFDEV] Users: User Failed Access Attempts [XUFDISP]	Failed Access Attempts Log Purge [XUFPURGE]
OLD ACCESS AND VERIFY CODES (#200 XREF)	^VA(200,	Predefined	N/A	Always done	N/A	Purge Log of Old Access and Verify Codes [XUSERAOLD]

Purge Old Access and Verify Codes

Figure 3-19. Purge Log of Old Access and Verify Codes option

```
SYSTEMS MANAGER MENU ...                                          [EVE]
User Management ...                                              [XUSER]
   Purge Log of Old Access and Verify Codes                   [XUSERAOLD]
```

The Purge Log of Old Access and Verify Codes option [XUSERAOLD] purges all inactive Access and Verify codes, which allows for the recycling of codes. Old Access and Verify codes are stored so that users cannot pick a previously used code when required to choose a new code. If old codes are stored indefinitely, though, it may become difficult for users to invent new codes. When you use this option interactively, you can purge codes older than a retention period you specify, from 7 to 90 days. When scheduled, the retention period defaults to 90 days, but can be changed to anything from 30 to 90 days by putting the number of days in the TASK PARAMETERS field.

The log of Access codes is stored in the whole-file AOLD cross-reference of the NEW PERSON file (#200). The log of Verify codes is stored per user in the VOLD cross-reference of the NEW PERSON file

(#200, not a whole-file cross-reference). Thus, Verify codes are *not* necessarily unique between users, while Access codes are.

4 Electronic Signatures

User Interface

An electronic signature is a security tool that software applications can use as an additional identification check. Software can, for example, require that an electronic signature be applied to a particular form or document before subsequent processing can continue.

If you need create an electronic signature for yourself, you can choose the Electronic Signature code Edit option [XUSESIG], available from the User's Toolbox menu.

You can enter a new electronic signature code, or change an existing code. The length of the code *must* be between 6 and 20 uppercase characters. Requiring all uppercase allows the code to be verified with either uppercase or lowercase input (since lowercase will be converted to uppercase in the matching process). You should choose a code that other users are not likely to guess, as this code verifies that it is actually you who are signing off on some important action.

Figure 4-1. Electronic Signature code Edit option

```
SYSTEM COMMAND OPTIONS ...                                          [XUCOMMAND]
  User's Toolbox ...  "TBOX"                                       [XUSERTOOLS]
    Electronic Signature code Edit                                    [XUSESIG]
```

The Electronic Signature code Edit option [XUSESIG] also allows you to edit the following fields:

- INITIAL
- SIGNATURE BLOCK PRINTED NAME
- SIGNATURE BLOCK TITLE
- OFFICE PHONE
- VOICE PAGER
- DIGITAL PAGER

Applications can print some or all of these fields when printing an electronically signed document. You should therefore ensure that the values entered in these fields are accurate.

System Management

Figure 4-2. User Edit menu options

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
User Edit ...                                                             [XUSER]
    Electronic Signature Block Edit                                 [XUSESIG BLOCK]
    Clear Electronic signature code  <locked:  XUMGR>                [XUSESIG CLEAR]
```

Electronic signature codes are stored in the NEW PERSON (#200) file.

The Electronic Signature Block Edit option [XUSESIG BLOCK] lets you edit the electronic signature code for any user on the system. When you create an electronic signature code for a user, the SIGNATURE BLOCK PRINTED NAME field is initially filled in by a cross-reference on the NAME field (#.01) (and will be overwritten if the NAME field [#.01] is changed). Credentials (e.g., "**M.D.**") can be added to customize the printed name. As a security feature, an input transform requires that the user's last name (first comma piece of the NAME field (#.01)) be included in the printed name. (This field *cannot* be edited through VA FileMan since it is WRITE-protected with a caret ["^"].)

The Clear Electronic signature code option [XUSESIG CLEAR] is another option available to IRM that allows the clearing (deleting) of an electronic signature code. This option is locked with the XUMGR security key. This option can be used to clear a user's electronic signature code if the user has forgotten the code. The user can then enter a new code with the Electronic Signature code Edit option [XUSESIG] in the User's Toolbox menu.

5 File Access Security

The File Access Security system is an optional Kernel module. It provides an enhanced security mechanism for controlling user access to VA FileMan files.

REF: For more information on File Access Security, please refer to the *VA FileMan (Version 22.0) and Kernel (Version 8.0) File Access Security* supplemental documentation located on the VHA Software Document Library (VDL) at the following Web address:

http://www.va.gov/vdl/application.asp?appid=5

User Interface

As a user, you typically access VistA data by use of application options. You enter data into files and retrieve information from files through the menu options within the software. Except under a few unusual circumstances, your use of the system will *not* be affected by the File Access Security system. If you need to work directly with files by using VA FileMan options, however, you will be affected.

VA FileMan options provide direct access to data files. Figure 5-1 lists some sample VA FileMan options:

Figure 5-1. Sample VA FileMan menu options

```
Select VA FileMan Option: ?

          Enter or Edit File Entries                                    [DIEDIT]
          Print File Entries                                           [DIPRINT]
          Search File Entries                                         [DISEARCH]
          Inquire to File Entries                                    [DIINQUIRE]
```

If the File Access Security system is implemented, the only files you can access directly through VA FileMan options are those listed in your ACCESSIBLE FILE Multiple field (#32) in the NEW PERSON file (#200). IRM grants file access by using a sub-menu on the User Management menu [XUSER].

There are six File Access Security level properties (listed alphabetically):

- AUDIT
- DATA DICTIONARY ("DD")
- DELETE ("DEL")
- LAYGO
- READ ("RD")
- WRITE ("WR")

REF: These File Access Security level properties are described in Table 5-1.

Each level of access is granted as **YES** or **NO**. If the File Access Security system is implemented, file access is controlled by these YES/NO flags, not by the matching of your FILE MANAGER ACCESS CODE field (#3) string in the NEW PERSON file (#200) with security placed on the file.

If you have *not* been granted any security access to VA FileMan files, entering two question marks ("**??**") when prompted for a file name/number shows no files to access:

Figure 5-2. User has *not* been granted security access to any VA FileMan files—Sample user dialogue

```
Select VA FileMan Option: Enter or Edit File Entries
INPUT TO WHAT FILE: ??


INPUT TO WHAT FILE:
```

No files displayed here indicates that the user has *not* been granted any security access to VA FileMan files.

In this case, you need to contact IRM and be granted access to the VA FileMan files you need.

File Access Security is also invoked when an option uses the VA FileMan line editor. In particular, the Transfer Lines from Another Document option on the line editor's Edit menu does not permit access to other word-processing documents in the current file or other files unless READ access to that file has been explicitly granted. If you need to transfer text from other files using the line editor, contact IRM to request access to those files.

System Management

Prior to introduction of the File Access Security system, user access to VA FileMan files through VA FileMan options was controlled by matching a character in a user's FILE MANAGER ACCESS CODE field (#3) (the DUZ(0) string) in the NEW PERSON file (#200) with a character in the file's top level file security fields.

Kernel's optional File Access Security system uses a different method. It allows you to control access to files for any user using VA FileMan options directly. Access is granted (or denied) by adding (or removing) a file from a user's ACCESSIBLE FILE Multiple field (#32) in their NEW PERSON file (#200) entry.

The File Access Security system does *not* affect access to files through *non*-VA FileMan options; security in this case is managed by controlling the availability of the option.

REF: For exceptions, please refer to the "When is File Access Security Checked?" topic in this chapter.

If a user's DUZ(0) is set to the at-sign ("@"; programmer access), VA FileMan options allow complete file access. If it is set to anything else (except the caret ["^"]), VA FileMan options use the ACCESSIBLE FILE Multiple field (#32) specifications in the NEW PERSON file (#200) to grant varying levels of file access.

NOTE: The caret ("^") overrides the at-sign ("@"; programmer access).

This higher degree of control over a user's file access comes at a price, because it requires more management on IRM's part to provide each user access to the files to which they need access. However, the payoff in using the File Access Security system is in enhanced control and security for VA FileMan files.

When is File Access Security Checked?

When using VA FileMan options, access to files through the File Access Security system is checked.

When initially accessing data in a file through software options (e.g., options using VA FileMan Application Program Interfaces [APIs]), File Access Security is *not* checked. File Access Security is checked, however, when calling the following APIs:

- ^DIC calls – Adding an entry to the top level of a file (i.e., LAYGO access)
- ^DIE calls – Deleting an entry at the top level of a file (i.e., DELETE access).

Developers can bypass these LAYGO and DELETE access checks using the following variables:

- DLAYGO
- DIDEL, respectively.

When accessing data through software options, File Access Security is also checked when a file is navigated to from another file (i.e., READ, WRITE, DELETE, and LAYGO access). Currently, there is no way for developers to override access checks when navigating to a file from another file, so explicit access to files navigated to/from an application option *must* be granted by IRM.

What in VA FileMan is Still Protected by the File Manager Access Code?

When the File Access Security system is enabled, access to templates (e.g., INPUT, PRINT, SORT, etc.) is denied when using VA FileMan options, if the user's DUZ(0) string does *not* contain a matching character. Similarly, when editing fields via VA FileMan's Enter or Edit File Entries option [DIEDIT], the DUZ(0) matching process is invoked to permit or deny editing for protected fields. The DUZ(0) value is also checked by some non-VA FileMan applications. Finally, if a user's DUZ(0) is "@", they are allowed complete access to all files.

Purpose for Granting File Access

IRM is responsible for granting file access. The needs of each user *must* be determined and an appropriate degree of access authority assigned. Too much access may risk the security of your system, while too little may inhibit productive activity.

What is the purpose of File Access Security? Why bother specifying who has access to which files? The answer is threefold:

- To monitor the use of VA FileMan.
- To regulate the extent of VA FileMan access from among six levels of security that allow AUDIT,, DATA DICTIONARY ("DD"), DELETE ("DEL"), LAYGO, READ ("RD"), or WRITE ("WR") access.

  REF: These File Access Security level properties are described in Table 5-1.

- To reserve DUZ(0), the FILE MANAGER ACCESS CODE field (#3), as a security measure to protect just templates and fields, *not* files, from VA FileMan options.

With file access security, it is possible to know who has access to which files and what kind of access they have. This information can also be retrieved by user or by file. In addition, privileges can also be entirely restricted for an individual user or for a single file that may contain sensitive information.

Who Needs File Access?

You need to grant File Access Security in the following cases:

- A user needs to access files directly through VA FileMan options.
- Within an application option, VA FileMan is used to navigate from one file to another.
- Within an application option that calls the ^DIE API to edit a file entry; a user is unable to add or delete entries in a pointed-to file.

- Within an application option that calls the ^DIE or ^DIC APIs to edit a file entry; a user is unable to add or delete entries in the primary file (because the application did not set the DLAYGO or DIDEL variables).
- A user needs to use the VA FileMan Line Editor's Transfer Lines from Another Document option.

Application developers can document which files need to be granted to whom, or can modify their code or data dictionary (DD) specifications to allow access.

Levels of File Access Security

There are six file access security properties involved with File Access Security. If a file access security property is *not* defined (i.e., the value is null), the VA FileMan exported menu option(s) for that property are *not* open to full access for users.

REF: The following table is taken from the *VA FileMan (Version 22.0) and Kernel (Version 8.0) File Access Security* supplemental documentation located on the VHA Software Document Library (VDL) at the following Web address:

http://www.va.gov/vdl/application.asp?appid=5

Table 5-1. File Access Security level properties

Access	Security Property Description	Property Location (Classic VA FileMan)
AUDIT	The AUDIT security property controls the setting of auditing characteristics and the deletion of audit trails. This property only deals with the auditing of data and *not* the auditing of data dictionary (DD) changes. To audit DD changes, users would enter "**YES**" at the "DD AUDIT? NO// " prompt when modifying a file's File Security Access. Examples of the VA FileMan options that this property controls are as follows: • Fields Being Audited [DIAUDITED FIELDS] • Data Dictionaries Being Audited [DIAUDIT DD] • Purge Data Audits [DIAUDIT PURGE DATA] • Purge DD Audits [DIAUDIT PURGE DD] • Turn Data Audit On/Off [DIAUDIT TURN ON/OFF]	^DIC(<file number>,0,"AUDIT")=<value>
DATA DICTIONARY ("DD")	The DATA DICTIONARY security property controls who has access to modify the data dictionary. Examples of the VA FileMan	^DIC(<file number>,0,"DD")=<value>

Access	Security Property Description	Property Location (Classic VA FileMan)
	options that this property controls are as follows: • Modify File Attributes [DIMODIFY] • Utility Functions [DIUTILITY] • Data Dictionary Utilities [DI DDU] For example, to use the Map Pointer Relations option, DD access is needed to the PACKAGE file (#9.4) and to the files one selects for mapping.	
DELETE ("DEL")	The DELETE security property controls who can delete an existing record that is contained within the file. It does *not* permit deletion of the file or any of its attribute fields. Examples of the VA FileMan options that this property controls are as follows: • Enter or Edit File Entries [DIEDIT] • Transfer Entries [DITRANSFER]	^DIC(<file number>,0,"DEL")=<value>
LAYGO	The LAYGO (Learn As You Go) security property controls who can add a new record to the file. Examples of the VA FileMan options that this property controls are as follows: • Enter or Edit File Entries [DIEDIT] **NOTE:** You *must* have LAYGO and WRITE access to a file to add new entries. In addition, you must have WRITE access at the field level for all required identifier fields.	^DIC(<file number>,0,"LAYGO")=<value>
READ ("RD")	The READ security property controls who has access to read data contained within a file. Examples of the VA FileMan options that this property controls are as follows: • Print File Entries [DIPRINT] • Search File Entries [DISEARCH] • Inquire to File Entries [DIINQUIRE] • Statistics [DISTATISTICS] • List File Attributes [DILIST] • Transfer Entries [DITRANSFER] To transfer text, the user needs READ access to the file from which	^DIC(<file number>,0,"RD")=<value>

Access	Security Property Description	Property Location (Classic VA FileMan)
	text is being transferred. Similarly, WRITE access is needed for the file to which entries are being transferred with this option. – Transfer File Entries (transfer-to file) **NOTE:** READ access is also required to use some of the Filegram and Audit options.	
WRITE ("WR")	The WRITE security property controls who can alter data in an existing record that is contained within the file. It will *not* permit the adding of new entries to the file. Examples of the VA FileMan options that this property controls are as follows: • Enter or Edit File Entries [DIEDIT] • Transfer Entries [DITRANSFER] To transfer text, the user needs READ access to the file from which text is being transferred. Similarly, WRITE access is needed for the file to which entries are being transferred with this option. – Transfer File Entries (transfer-to file) – Compare/Merge File Entries	^DIC(<file number>,0,"WR")=<value>

These are the six levels of access. Any or all can be enabled for each of the user's accessible files. This is done by changing the field value from NULL to YES. This flag is overridden for developers whose DUZ(0)=@.

Granting the READ, WRITE, DELETE, and LAYGO levels of access permits adding and deleting file entries as well as editing their attribute field data values. This is true unless the attribute field has been protected. If so (i.e., if there is READ, WRITE, or DELETE protection within the data dictionary [DD] for a given field), the user's FILE MANAGER ACCESS CODE field (#3), DUZ(0), is checked. Access is denied if the user's DUZ(0) does *not* contain a character matching the field protection. Again, DUZ(0)=@ overrides this restriction.

The DATA DICTIONARY ("DD") and AUDIT levels of access pertain to the structure of the file itself. While this provides a generous scope for VA FileMan data dictionary (DD) modification, it falls short of, for example, deleting a field protected with the at-sign ("@"; programmer access).

The same applies to templates. If the template is protected, the user who has access to the file will *not* have access to the template from VA FileMan options unless there is a match in the DUZ(0) character string.

Audit Access to Files

Audit privileges might be granted to advanced VA FileMan users who are interested in developing new audit capabilities. With AUDIT access, which *must* be accompanied by DD access, VA FileMan's Modify File Attributes option [DIMODIFY] can be used to set an audit flag for a particular field within a file. This access does *not* include setting audit conditions with M code, which is reserved for users with a FILE MANAGER ACCESS CODE field (#3) containing "**@**".

The data values for attribute fields can be recorded in the AUDIT file (#1.1) by setting an audit flag in the data dictionary (DD) for that field. For example, the SSN field in the PATIENT file (#2) could be audited. There are two choices for the audit in the AUDIT file (#1.1):

- An entry can be made when a value is entered or changed.
- An entry can be made *only* when the value is changed (i.e., edited or deleted).

The second method may be all that's needed. In the SSN example, you would monitor just the circumstances of the change, not of the initial SSN assignment.

To display the results of the audit, your DUZ(0) *must* equal the at-sign ("**@**"; programmer access). Then, you can query the AUDIT file (#1.1) in the usual way with VA FileMan's Inquire to File Entries option [DIINQUIRE].

How to Grant File Access

IRM specifies the particular files and levels of access for users. The File Access Security menu [XUFILEACCESS], on the User Management menu [XUSER], provides options to grant file access security. These options edit the ACCESSIBLE FILE Multiple field (#32) in the NEW PERSON file (#200).

The options for granting file access privileges fall into three functional categories:

- EDITING—To assign file access to an individual user or a group of users. One user's profile can also be duplicated or copied to another user or group of users. To simplify adding files, number ranges can be specified.
- LISTING—To display one user's profile, a name-sorted list of all user's profiles, or a file or range of files with associated users and the access levels of each.
- RESTRICTING—To entirely limit access by user or by file, or to delete a range of files for a user or group of users.

The options are designed to facilitate queries by user or by file. You can add or delete file access for one user or for many users. Or, you can begin with the file and list users with access or restrict access.

Using the File Access Options

Figure 5-3. File Access Security menu options

```
SYSTEMS MANAGER MENU ...                                              [EVE]
User Management ...                                                  [XUSER]
   File Access Security ...                                      [XUFILEACCESS]
      Grant Users' Access to a Set of Files                       [XUFILEGRANT]
      Copy One User's File Access to Others                       [XUFILECOPY]
      Single file add/delete for a user                     [XUFILESINGLEADD]
      Inquiry to a User's File Access                         [XUFILEINQUIRY]
      List Access to Files by File number                        [XUFILELIST]
      Print Users Files                                         [XUFILEPRINT]
      Delete Users' Access to a Set of Files                [XUFILESETDELETE]
      Remove All Access from a Single User                  [XUFILEREMOVEALL]
      Take away All access to a File                           [XUFILEDELETE]
      Assign/Delete a File Range                          [XUFILERANGEASSIGN]
```

When using options on the File Access Security menu [XUFILEACCESS], you may have the following questions:

- What is the DUZ# that appears next to the user's name?
- How is a range of file numbers specified?
- What are the queuing questions all about?

Understanding DUZ (the User Number)

When listing the file accesses by user or by file, the user's name is followed by a number in parentheses. The heading indicates that this is the "User #," which is the same as the DUZ#.

DUZ is a local variable that identifies the user who has signed onto the system. Once the user enters an Access and Verify code, Kernel's Signon/Security uses this variable to identify an entry in the NEW PERSON file (#200). It *must* be unique, so the user's name will not do. Instead, the internal entry number (IEN) is used. That is what becomes the value of DUZ.

Some users have low numbers while others have high ones. This simply indicates the order their names were entered into the NEW PERSON file (#200). Users with low numbers are often people who began using the system some years ago, while users with high numbers tend to be recent entries in the file.

Using Ranges of File Numbers

Can files be specified by number ranges? Yes; it is useful to do this when granting several files at once. First, find out the number of the files. Typing a question mark ("**?**") at the "to Files:" prompt will display the number and name of the files. Note the numbers and then put them together on one line. You can use dashes to indicate a consecutive range and commas to separate the single numbers and dashed groups as follows:

2,3,4,6,7,8,125,236,799

OR

2-4,6-8,125,236,799

File numbers are also used when printing a group of consecutive files. The prompt asks for a place to start with a default file name presented. To print just this one file, respond to the next prompt by simply pressing the **<Enter>** key, thereby accepting the default of ending after printing that one file.

To print a consecutive range of files, the lowest number is entered as the starting point and the highest number as the ending point. All files that fall in this range will be printed.

Queuing File Access Specifications

Most of the options provide the opportunity to queue, after specifying who is to be granted which files. Queuing sends the specifications to TaskMan to assign to users at a later time. TaskMan can work at an off-peak time (e.g., midnight) to avoid consuming system resources during the daytime. If the system is not busy, queuing is still a good idea since your terminal will otherwise be tied up while the report is being printed.

Running the File Access Security Conversion

Advantages

To implement File Access Security you need to run a conversion. Some advantages of implementing File Access Security include:

- **Easier to identify levels of access –** Running the conversion makes it possible to identify the levels of access each individual user has to each file.
- **Enhanced system performance –** Checking file access by user is slightly faster in terms of global accesses and CPU time.

Advance Preparation for the Conversion

The File Access Security conversion is designed to allocate access privileges to all of your users according to their current FILE MANAGER ACCESS CODE field (#3) value in the NEW PERSON file (#200), DUZ(0), combined with information about their file access through options stored in the ^DISV global. After the conversion you should get only a few user requests for file access. The File Access Security menu [XUFILEACCESS], an option on the User Management menu [XUSER], should then be used to add a file to a user's ACCESSIBLE FILE Multiple field (#32) in the NEW PERSON file (#200).

The conversion uses the FILE MANAGER ACCESS CODE field (#3) (DUZ[0] string) to assign file access according to the characters in the string. If a file is protected with a particular character that matches one in the user's code, that file is entered into the user's ACCESSIBLE FILE Multiple field (#32). Levels of access are granted according to the file's original security (field-level security continues to function the same, by checking the FILE MANAGER ACCESS CODE field (#3)).

NOTE: Users with programmer-level access (FILE MANAGER ACCESS CODE field [#3] = @) will *not* need to have any files in their ACCESSIBLE FILE Multiple field (#32), since they will be able to access *all* files *without* restriction.

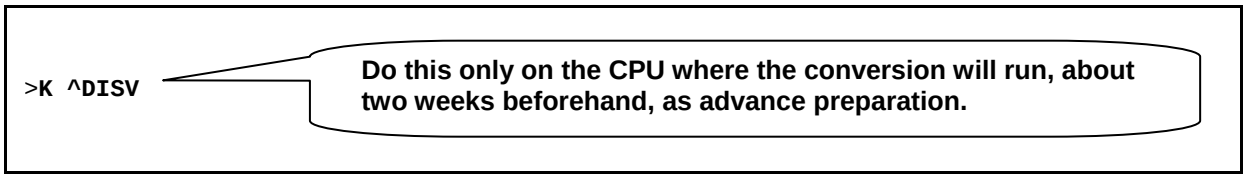
^DISV Global

The File Access Security conversion process makes use of the ^DISV global to identify which files have recently been accessed by which users. The conversion adds all files that the user has been able to access (select from) to the user's ACCESSIBLE FILE Multiple field (#32) list. It grants READ access to these files.

Using the ^DISV global to grant file access has the benefit of permitting option usage "as usual" the day after the conversion is run. KILLing the ^DISV global just *before* the conversion is *not* advised, since many users will suffer inappropriate access restrictions and will need special attention by IRM just after the conversion. KILLing the ^DISV global a week or two before the conversion, however, may be worthwhile as a way of purging obsolete user data. In multi-CPU environments, where each CPU has its own copy of the ^DISV global, you should choose the busiest user node upon which to run the conversion (in order to pick up the most comprehensive information from that node's ^DISV). Caché sites should run the conversion from their busiest user node.

It is assumed that ^DISV is not translated, so K ^DISV on the CPU where the conversion will be run. Do this about two weeks before you perform the conversion, as advance preparation. ^DISV will be reset as soon as a user responds to a "Select:" prompt.

Figure 5-4. KILLing ^DISV—Sample code



Adding Explicit File Access for IRM

If there are any files that are neither protected nor accessed by users (e.g., the DOMAIN file [#4.2]) the conversion will *not* list them in any user's ACCESSIBLE FILE Multiple field (#32). Before the conversion, these types of files are accessible to everyone, while after the conversion these files will only be accessible to users with programmer-level access. Therefore, before the conversion, assign a unique symbol/character to otherwise unprotected files. This will ensure that at least those users with that unique symbol (e.g., IRM staff) will be granted access. VA FileMan's Edit File option [DIEDFILE] can be used to edit the codes.

NOTE: In previous documentation and data dictionaries, it has been *implied* that the pound sign ("#") symbol/character was reserved for File Access Security for IRM, however, this is *not* true. It has merely been used as a *convention.*

Figure 5-5. Updating file access settings (*before* conversion)

```
Select OPTION: UTILITY FUNCTIONS
Select UTILITY OPTION: EDIT FILE

MODIFY WHAT FILE: USER// DOMAIN <Enter>            (227 entries)
Do you want to use the screen-mode version? YES// n <Enter>  NO
NAME: DOMAIN// <Enter>
DESCRIPTION:
  No existing text
  Edit? NO// <Enter>
Select APPLICATION GROUP: <Enter>
DEVELOPER: <Enter>
DATA DICTIONARY ACCESS: <Enter>
READ ACCESS: <Enter>
WRITE ACCESS: <Enter>
DELETE ACCESS: <Enter>
LAYGO ACCESS: <Enter>
AUDIT ACCESS: <Enter>
```

Enter a unique symbol/character for each level of access so that those unprotected files will be assigned to IRM staff.

Summary of How the File Access Security Conversion Works

The File Access Security conversion prepares the NEW PERSON file (#200) for VA FileMan's method of file access (lookup into a user's record for file access). VA FileMan's ability to protect data within files on fields and templates, remains the same. The steps that occur when the conversion is run are outlined below:

1. The structure for implementing the file access method is set up via the following:
   a. Placing the data dictionary (DD) for the ACCESSIBLE FILE Multiple field (#32) in the NEW PERSON file (#200). (This multiple will be permanently put in place by running the File Access Security conversion.)
   b. Installing menu options, help frames, and templates used for maintaining the user file access method (i.e., entries with the XUFI namespace).
2. Each user's FILE MANAGER ACCESS CODE field (#3) is used to add entries in the ACCESSIBLE FILE Multiple field (#32) by:
   a. Creating a list of files to be processed by examining each file's protection codes. Files that meet *both* of the following requirements are temporarily stored in the ^UTILITY($J global:
      - Files that have protection defined.
      - Files with protection *not* equal to "@".

NOTE: Files that lack any protection will be bypassed. Such unprotected files will *not* later be listed in anyone's ACCESSIBLE FILE Multiple field (#32). Protection should therefore be applied *before* running the conversion so that at least some users (e.g., IRM Staff) will be granted access.

b. Examining each user in the NEW PERSON file (#200). Each user meeting *all* of the following requirements is selected for further processing:

- Users *not* terminated.
- Users with an Access code.
- Users with a VA FileMan Access code (i.e., FILE MANAGER ACCESS CODE field [#3] in the NEW PERSON file [#200]).
- Users with a FILE MANAGER ACCESS CODE field (#3) in the NEW PERSON file [#200] *not* equal to "@".

The user's FILE MANAGER ACCESS CODE field (#3) in the NEW PERSON file [#200] is parsed. Each symbol/character is compared with the list of files in the ^UTILITY($J global. All files that have a protection code matching this symbol/character are added to the user's ACCESSIBLE FILE Multiple field (#32) in the NEW PERSON file [#200]. If the symbol/character is used as the file's DATA DICTIONARY ("DD") file security, the user is granted DD access; if it is used as LAYGO, the user is granted LAYGO access, and so on.

3. Files accessed by the user through options since the last time the ^DISV global was KILLed are added to the user's ACCESSIBLE FILE Multiple field (#32) by the processing of the ^DISV global. Entries in ^DISV that meet *both* of the following requirements will be added to the ACCESSIBLE FILE Multiple field (#32), with READ access:

- The file *must* not be in VA FileMan's file number range (i.e., file number *must* be equal to or greater than 2).
- The user does *not* already have access to this file.

File Access Security Conversion Instructions

1. Identify unprotected files and assign protection codes as desired (as described in the "Advance Preparation for the Conversion" topic described earlier). For example, the DOMAIN file (#4.2) may need to be protected so that it will be granted to users having a FILE MANAGER ACCESS CODE field (#3) containing the assigned symbol/character.

NOTE: In previous documentation and data dictionaries, it has been *implied* that the pound sign ("#") symbol/character was reserved for File Access Security for IRM, however, this is *not* true. It has merely been used as a *convention.*

2. Review the FILE MANAGER ACCESS CODE fields (#3) of VA FileMan users. The codes should contain symbols/characters matching those used to protect the files that these individuals use. Since the conversion will automatically grant files to users according to previous privileges as indicated by the FILE MANAGER ACCESS CODE field (#3), add any additional symbols/characters to their FILE MANAGER ACCESS CODE fields (#3) to take advantage of the conversion's automated file assignment according to levels of access.
3. Be ready to use the File Access Security menu [XUFILEACCESS], Figure 5-3, to review and grant file access privileges *after* the conversion.

4. In the production account, enable File Access Security system features and options with ENABLE^XUFILE3, as illustrated below:

Figure 5-6. Enabling File Access Security—Sample user dialogue

```
In VAH:

>D ENABLE^XUFILE3

>
```

5. In the production account, begin the conversion with ^XUINCON:

Figure 5-7. ^XUINCON conversion routine—Sample user dialogue

```
In VAH:

>D ^XUINCON

Version 7 of the Kernel defined a new multiple-valued field
in the New Person File called Accessible File.  This conversion
will store file access in this multiple in the following manner:

Those Users who have a FileMan Access Code (DUZ(0)) which
is not null, i.e., contains some character string,
will have their access string matched to the protection
currently on your files.  For each match between the file
and the user, the file will be listed in the user's
Accessible File multiple as will the type of access
(dictionary, delete, laygo, read, write, audit).

NOTE: Files with no protection will NOT be assigned to any user.

Would you like to run the conversion now? NO//
```

6. If you are ready to run the conversion, answer YES:

Figure 5-8. Running a conversion—Sample user dialogue

```
Would you like to run the conversion now? NO// YES
56237,36565
Build Table.
Convert Users.
Give access from DISV file.
X-ref.
Done56237,36565.
>
```

7. Review the newly assigned access settings. Use the File Access Security menu [XUFILEACCESS], Figure 5-3, located on the User Management Menu [XUSER], to display file access by user and by file.

After the Conversion

After the conversion, users may complain about not being able to add entries to files as they previously could. This typically results from use of an option that navigates from one file to another. To be able to add entries to the navigated-to file, the user needs LAYGO access to that file. IRM can solve the problem by granting LAYGO access using the File Access Security menu options [XUFILEACCESS], Figure 5-3.

If this form of security is implemented, IRM will find that it provides a more accurate and precise knowledge of who has what level of access to which files. When the conversion is run, privileges are granted to existing users by making use of information stored in the VA FileMan record of file manipulation activity, the ^DISV global. The file access conversion grants each user READ access to files that the user had recently accessed as indicated in the ^DISV global. IRM can grant file access privileges to new users by copying the profile of an existing user with similar duties (e.g., a laboratory application coordinator or admissions clerk).

To be sure that appropriate levels of access have been allocated, IRM staff should determine who has what level of access to which files. Access to sensitive files (e.g., the NEW PERSON file [#200]) should be reviewed and readjusted for individual users as appropriate. All files on a system should be reviewed before and after running the File Access Security conversion.

Figure 5-9 shows how to create a PRINT template to display a report on the current file access security:

Figure 5-9. Creating a PRINT template to display file access security – Sample user dialogue

```
Select OPTION: PRINT FILE ENTRIES

OUTPUT FROM WHAT FILE: FILE
SORT BY: NAME// @NUMBER
START WITH NUMBER: FIRST// 3
GO TO NUMBER: LAST// 4
  WITHIN NUMBER, SORT BY: <Enter>
FIRST PRINT ATTRIBUTE: NUMBER;L8;S;""
FIRST PRINT ATTRIBUTE: NAME;L25;""
THEN PRINT ATTRIBUTE: DD ACCESS;R6
THEN PRINT ATTRIBUTE: RD ACCESS;R6
THEN PRINT ATTRIBUTE: WR ACCESS;R6
THEN PRINT ATTRIBUTE: DEL ACCESS;R6
THEN PRINT ATTRIBUTE: LAYGO ACCESS;R6
THEN PRINT ATTRIBUTE: AUDIT ACCESS;R6
THEN PRINT ATTRIBUTE: <Enter>
HEADING: FILE LIST// FILE SECURITY
STORE PRINT LOGIC IN TEMPLATE: ZZFILE SECURITY
```

Enter the starting and ending file numbers.

Store in a local template for later use (e.g., ZZFILE SECURITY).

Once the conversion has been run, you can use the File Access Security menu [XUFILEACCESS], Figure 5-3, to print the accessible files for individual users. Thus, you can establish profiles that would be typical of groups of users (e.g., Nursing, Pharmacy, or other services). Then, when establishing an account for a new user or reactivating the access of a previously terminated user, the profile will be available for copying to the new user.

II Menu Manager

6 Menu Manager: User Interface

Kernel's menu system presents menu options within VistA software in a standard fashion. Once you become familiar with using the menu system in one application, using other applications will be easier since the same rules apply.

Navigating Kernel's Menus

When you successfully sign into the computer system, Menu Manager presents your primary menu options. Your primary menu is the top-level menu assigned to you by IRM. Most options that are available to you are available from your primary menu, or from a sub-menu attached to your primary menu.

The menu system prompts you with a "Select (menu name) Option:" prompt. For example, in a menu named Billing, Menu Manager would prompt you with "Select Billing Option:". You can navigate through the menu system by responding to this prompt in different ways, which are described in this chapter.

You can enter question marks to see option choices and obtain online help. You can enter an option's synonym or the first few letters of its menu text, using upper or lowercase, to select the option. Or you can enter a caret ("^") along with the option specification (option menu text or synonym) to jump to the destination option rather then traversing the menu pathways step-by-step.

Choosing Options

You can choose an option from your current menu at the select prompt. Choosing the option launches the computer application associated with the option. To choose an option, type in the first few letters of the option as it is displayed and press the **<Enter>** key. If the option is another menu, indicated by trailing ellipses (**...**), it will become the current menu, and so on down the menu pathway.

To come back up the menu pathway, press **<Enter>** at the select prompt. Each time you press **<Enter>**, Menu Manager will return you to the next higher menu level, until you reach your highest menu, the primary menu. If you press **<Enter>** at the primary menu, Menu Manager asks if you want to halt. If you answer YES, your Kernel session will be ended.

Entering One Question Mark to List Options

When you enter a menu, the items may or may not be displayed automatically, based on whether you have AUTO MENU turned on. The AUTO MENU feature, as described in the "Signon/Security: User Interface" chapter in this manual, is a flag that controls the menu display. If you do not have a setting specified for AUTO MENU, the site parameter default will be used. Often, to save system resources, the site parameter can be set to disable automatic display. In this case, to display your menu's items, simply enter a question mark ("**?**"), as shown below:

Figure 6-1. One question mark (?) help—Sample user dialogue

```
Select Any Level Menu Option: ?

          First Item
          Second Item
          Third Item of Menu Choices ...
          Fourth Item

Enter ?? for more options, ??? for brief descriptions, ?OPTION for help text.

Select Any Level Menu Option:
```

Entering ?Option to Display Option Help

To obtain a lengthier description of an individual option, enter a single question mark ("**?**"), and the first few letters of the option name. If there is an extended description of the option, or a help frame describing the option, they are displayed.

Figure 6-2. Using ?Option to get help on a named option—Sample user dialogue

```
Select User's Toolbox Option: ?

          Display User Characteristics
          Edit User Characteristics
          Electronic Signature Code Edit
          Menu Templates...
          Spooler Menu...
          TaskMan User
          User Help

Select User's Toolbox Option: ?DISPLAY



'Display User Characteristics'     Option name: XUUSERDISP
    Display the user's name, location, and characteristics

 **> Press 'RETURN' to continue, '^' to stop: <Enter>

Select User's Toolbox Option:
```

Entering Two Question Marks to List Secondary and Common Options

At any select prompt you can enter two question marks ("**??**") to see options on the Secondary and Common menus, as well as options available on the current branch of your menu tree.

The Secondary menu and the Common menu contain options that you can select at any location in the menu system. Options on the Secondary menu are typically created by your system manager. Options on the Common menu are standard Kernel options available from anywhere in the menu system. Options on the current menu, on the other hand, can only be directly selected while that menu is the current menu.

The two-question-mark display shows the option's synonym (a short abbreviation), if one exists . You can select an option by its synonym as well as by its full name. On the same line, it lists the option's full name followed by the formal option name in capital letters enclosed in square brackets. (The name is the .01 field of the OPTION file [#19].) It also shows any option restrictions such as:

- Out-of-Order
- Locked
- Prohibited times.

Figure 6-3. Two question marks (??) help—Listing Primary, Secondary, and Common menu options

```
Select Systems Manager Menu Option: ??

   FM     VA FileMan ...                                              [DIUSER]
          Core Applications ...                                       [XUCORE]
          Device Management ...                                        [XUTIO]
             **> Locked with XUPROG
          Information Security Officer Menu ...                       [XUSPY]
          Manage Mailman ...                                          [XMMGR]
          Menu Management ...                                       [XUMAINT]
          Operations Management ...                                [XUSITEMGR]
          Programmer Options ...                                      [XUPROG]
             **> Locked with XUPROG
          Spool Management ...                                     [XU-SPL-MGR]
          Taskman Management ...                                    [XUTM MGR]
          User Management ...                                         [XUSER]

You can also select a secondary option:

   OUT    Equipment Checked Out to Myself                        [A6A EQUIP USER]
   PAID   SIGN INTO MARTINEZ VIA TELNET, TYPE DUSER                [A6A USE PAID]
   RUM    Capacity Planning ...                                     [XTCM MAIN]
          ISC OFFICE MENU OPTIONS ...                               [ISCSTAFF]

Or a Common Option:

   KNF    Kernel New Features Help                             [XUVERSIONEW-HELP]
          Halt                                                        [XUHALT]
          Continue                                                 [XUCONTINUE]
          Restart Session                                             [XURELOG]
   MM     MailMan Menu ...                                            [XMUSER]
   NPI    PROVIDER NPI SELF ENTRY                  [XUS NPI PROVIDER SELF ENTRY]
   TBOX   User's Toolbox ...                                       [XUSERTOOLS]
   VA     View Alerts                                                 [XQALERT]
          Time                                                        [XUTIME]
          Where am I?                                              [XUSERWHERE]
```

Entering Three Question Marks to Display Option Descriptions

Entering three question marks ("**???**") at any select prompt displays option descriptions (from a word-processing-type field in the OPTION file [#19]). If entered at the select prompt for a menu within the primary tree, the top-level options are described; then you are prompted whether you want to see descriptions for secondary or Common options.

Figure 6-4. Three question marks (???) help—Sample user dialogue

```
Select Spooler Menu Option: ???

'Allow other users access to spool documents'     Option name: XU-SPL-ALLOW
     This option edits the 'OTHER AUTHORIZED USERS' field of the SPOOL
     DOCUMENT file to allow other users access to a spool document.

'Delete A Spool Document'     Option name: XU-SPL-DELETE
  **> Extended help available.  Type "?Delete" to see it.
     Delete a spool document from the spool document file and delete the
     associated message if they are still linked.

'List Spool Documents'     Option name: XU-SPL-LIST
  **> Extended help available.  Type "?List" to see it.
     This option lists entries in the spool document file.

'Make spool document into a mail message'     Option name: XU-SPL-MAIL
  **> Extended help available.  Type "?Make" to see it.
     This option will take a spool document and post it as a mailman
     message to the user's IN basket.  This doesn't move the data at all
     but does decrease the number of lines charged to the user.

  **> Press 'RETURN' to continue, '^' to stop, or '?[option text]' for more
                          help: <Enter>

'Print A Spool Document'     Option name: XU-SPL-PRINT
  **> Extended help available.  Type "?Print" to see it.
     This allows the printing of a document that has been spooled.


  Shall I show you your secondary menus too? No// <Enter>
  Would you like to see the Common Options? No// <Enter>


Select Spooler Menu Option:
```

You should be ready to use three question marks ("**???**") to learn more about unfamiliar options (e.g., options distributed in a new software release).

Jumping to Options ("Up-arrow Jump")

The pathways of the primary, secondary, and Common menus have tree-like structures. You can step up or down the pathways to reach your destination, or invoke the menu system's "Up-arrow Jump" feature as a shortcut. To jump to an option, enter a caret ("^") before the option specification (the option's menu text or synonym in upper- or lowercase letters). You only need to enter the first few characters needed to

uniquely identify the option. You can use the option's synonym to limit ambiguity, especially if the synonym is distinct from other synonyms or menu texts.

Figure 6-5. Using the "Up-arrow Jump"—Sample user dialogue

```
Select Systems Manager Menu Option: ^INTRO <Enter> ductory text edit
```

The menu system carries out the necessary footwork to reach the desired option. If, along the way, there are pathway restrictions (e.g., locks or prohibited times), access to the option will be denied, just as when stepping to an option. If a match is found within the primary or secondary menus, that option is executed (the menu system will not search the Common menu if it can find a match in the primary or secondary menus).

If the menu system finds *more than one* matching option on *either* the primary, secondary, or Common menu tree, the menu system presents a list of matching choices. Entering a caret ("^") followed by a question mark ("**?**") will display all of the options available to you.

Figure 6-6. List of choices—Sample user dialogue

```
Select Systems Manager Menu Option: ^List Names

    1    List Namespaces  [XUZ NAMESPACES]
    2    List Namespaces  [ZZ NAMESPACE LIST]

Type '^' to stop, or choose a number from 1 to 2 :
```

IRM should assign "shallow" secondary menus to facilitate menu jumping. When a jump is requested, the menu system searches all the way through the primary as well as the secondary, looking for a match. Users will be inconvenienced and system resources will be consumed if secondary menus are "deep" in terms of their hierarchical tree-like structure.

You may occasionally find jumping disabled; when you try to jump, you may get a message that quick access is temporarily disabled. Jumping will stay disabled until the needed menu trees are rebuilt.

"Rubber-band Jump"

The menu system's jump feature includes the ability to jump out to a destination option and then back again, something like the motion of a rubber band. The syntax for the "Rubber-band Jump" request is the use of a double caret ("^^") followed by the usual option specification. For example:

Figure 6-7. "Rubber-band jump"—Sample user dialogue

```
Select Systems Manager Menu Option: ^^TASKMAN USER
```

As with the single "Up-arrow Jump" ("^"), restrictions along the menu pathways are checked.

If you enter two carets ("^^") without a following option specification/name, you are returned to the primary menu. This technique is a quick way for you to "go home" to the menu that is displayed at signon, and is called the "Go-home Jump."

CAUTION: It is important to note that when you invoke the "Rubber-band Jump," there is no attempt to protect variables that can be SET or KILLed, via Entry or Exit Actions, as you jump through the menu tree. Thus, the "Rubber-band Jump" can be inappropriate under certain circumstances, since it could cause significant alteration of your environment.

Common Menu

The Common menu is designed as a collection of options that are available to all users. The standard Common menu items are:

- **User's Toolbox:** As described in the "User's Toolbox Menu" topic in the "Signon/Security: User Interface" chapter, the User's Toolbox is a menu containing options that allow users to control some aspects of their computing environment.
- **Halt, Continue, Restart Session:** As described in the "Signon/Security: User Interface" chapter in this manual, these options are three different ways to log out of the system.
- **View Alerts:** As described in the "Alerts" and "Signon/Security: User Interface" chapters in this manual, View Alerts is an option that lets you process Alerts.
- **Time:** The Time option simply displays the date and time.
- **Where am I?:** This option lists information identifying what computer system you are signed into (e.g., UCI, Volume Set, Node, and Device).

Selecting Common Options with the Double Quote

Since Common options are intended to be readily accessible, there is a shortcut method to reach them. While you could use an "Up-arrow Jump," it is quicker to enter a quotation mark followed by the option specification/name. Figure 6-8 selects the User's Toolbox menu from the Common menu via its synonym, TBOX:

Figure 6-8. Selecting common options via the double quote—User's Toolbox menu option

```
Select Sample Menu Option: "TBOX

          Display User Characteristics
          Edit User Characteristics
          Electronic Signature code Edit
          Menu Templates ...
          Spooler Menu ...
          TaskMan User
          User Help

Select User's Toolbox Option:
```

Menu Templates Option

Menu templates are like scripts. You can use them to execute a fixed series of options, in sequence. Tools for creating, deleting, listing, and renaming templates are options on the Menu Templates menu, part of the User's Toolbox (TBOX) menu:

Figure 6-9. Menu Templates Option

```
Select Menu Templates Option: ?

          Create a new menu template
          Delete a Menu Template
          List all Menu Templates
          Rename a menu template
          Show all options in a Menu Template

Select Menu Templates Option:
```

When you create a MENU template, you are prompted for a series of options that lead to a final non-menu (i.e., executable) destination option. Once you choose one non-menu option to be executed, you can navigate to other options and choose them to be executed as well, if you wish. When you have selected each executable option to be part of the template, enter a plus sign ("+") to store the sequence of options. You will be asked to confirm the sequence of options in the template, and then to give the template a name.

To invoke the template, simply enter a left square bracket followed by the template name:

Figure 6-10. Invoking a template—Sample user dialogue

```
Select Option: [mytemplate

Loading MYTEMPLATE...
```

The template will then execute each option that is part of the template, in the same order as the options were selected for the template.

MENU templates are stored in the MENU TEMPLATE field (Multiple) of the NEW PERSON file (#200), so you can use any name for MENU templates. If your MENU template points to options that are subsequently removed from the OPTION file (#19), you receive a message that the MENU template will no longer function properly and needs to be deleted and/or rebuilt.

Use menu jumping (i.e., the "Up-arrow Jump") when you want to jump immediately to an option. Use MENU templates when you have a series of options that you will need run in the same order repeatedly, over a period of time.

LOGIN Menu Template

Beginning with Kernel V. 8.0, you can have a MENU template execute automatically, on your first signon of the day. If you have a MENU template named LOGIN (all uppercase), the MENU template will be executed on your first signon of the day. So if you have a series of options you execute on your first signon every day, an easy way to execute them is to create a MENU template; store the series of options in the template; and name the template LOGIN.

Summary

Once you learn how to navigate Kernel's menu tree, you can use some of Menu Manager's additional features to help increase your productivity in the VistA computer system. These features include the "Up-arrow Jump," the "Rubber-band Jump," using three question marks ("**???**") to obtain online option help, and using MENU templates as scripts.

7 Menu Manager: System Management

Menu Manager is built around options, which are entries in the OPTION file (#19). There are several types of options. One type of option, Menus, has subentries in the MENU (item) field (Multiple), a multiple that points back to the OPTION file (#19) itself. Other types of options are designed as items that plug into the MENU (item) field (Multiple) of a menu-type option.

Kernel provides a number of tools to create and manage menus and options.

Kernel Menus

Most of the options exported with software are tied to a parent option, or master menu, as a collection point. Kernel exports three menu tree "roots." The Systems Manager menu [EVE] is the master menu for IRM. The Common menu [XUCOMMAND] is linked through the display function of the menu system rather than the OPTION file (#19). Kernel also exports a menu that is a miscellaneous collection of options that should not normally be invoked by the interactive user; it is the PARENT OF QUEUABLE OPTIONS menu [ZTMQUEUABLE OPTIONS]. Most of the options on this menu should be scheduled to run as TaskMan jobs.

Figure 7-1. Kernel menu tree roots

```
SYSTEMS MANAGER MENU ...                                          [EVE]
SYSTEM COMMAND OPTIONS ...                                   [XUCOMMAND]
PARENT OF QUEUABLE OPTIONS ...                    [ZTMQUEUABLE OPTIONS]
```

Creating Menus and Options

Figure 7-2. Edit options option

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
  Menu Management ...                                                    [XUMAINT]
    Edit options                                                       [XUEDITOPT]
```

One task IRM performs frequently is defining local primary menus that are appropriate for their users. This task of menu creation is accomplished by grouping exported menus from various software application together on a new master menu. You can use Edit options, on the Menu Management menu, to define a new menu if READ, WRITE, and LAYGO access to the OPTION file (#19) has been granted (either through the FILE MANAGER ACCESS CODE field [#3] or through the File Access Security system if that is enabled). Only a few fields need to be defined, as shown below. The new menu can then be assigned to a user, as described in the "Signon/Security: User Interface" chapter, with one of several options on the User Edit menu.

Figure 7-3. Defining local primary menus (IRM)—Sample user dialogue

```
Select OPTION to edit: ZZSTAFF MENU
  Located in the Z (Local) namespace.
  ARE YOU ADDING 'ZZSTAFF MENU' AS A NEW OPTION (THE 721ST)? Y <Enter> (YES)
   OPTION MENU TEXT: Staff Menu
NAME: ZZSTAFF MENU// <Enter>
MENU TEXT: Staff Menu// <Enter>
PACKAGE: <Enter>
OUT OF ORDER MESSAGE: <Enter>
LOCK: <Enter>
REVERSE/NEGATIVE LOCK: <Enter>
DESCRIPTION:
  1>This is the primary menu for staff members.
  2><Enter>
EDIT Option: <Enter>
TYPE: menu
Select ITEM: XUCORE <Enter>    Core Applications
  ARE YOU ADDING 'XUCORE' AS A NEW MENU (THE 1ST FOR THIS OPTION)? Y <Enter> (YES)
   MENU SYNONYM: <Enter>
  SYNONYM: <Enter>
  DISPLAY ORDER: 10
Select ITEM: XUSPY <Enter>   System Security
  ARE YOU ADDING 'XUSPY' AS A NEW MENU (THE 2ND FOR THIS OPTION)? Y <Enter> (YES)
   MENU SYNONYM: <Enter>
  SYNONYM: <Enter>
  DISPLAY ORDER: 20
Select ITEM: XT-KERMIT MENU <Enter>    Kermit menu
  ARE YOU ADDING 'XT-KERMIT MENU' AS A NEW MENU (THE 3RD FOR THIS OPTION)?
YES <Enter> (YES)
   MENU SYNONYM: <Enter>
  SYNONYM: <Enter>
  DISPLAY ORDER: 30
Select ITEM: <Enter>
CREATOR: SITE,MANAGER// <Enter>
HELP FRAME: <Enter>
PRIORITY: <Enter>
Select TIMES PROHIBITED: <Enter>
```

```
Select TIME PERIOD: <Enter>
RESTRICT DEVICES?: <Enter>
Select PERMITTED DEVICE: <Enter>
```

Option Name and Menu Text: By convention, the formal option name is usually entered in all capital letters. According to namespacing conventions, it *must* begin with a namespace that identifies the associated software. It is the NAME field (#.01) of the OPTION file (#19). The menu text is what is displayed to the user at the select prompt. Like the words of a heading or title, initial capitalization is used for all words except prepositions and articles, all of which are presented in lowercase. To minimize the number of keystrokes needed to select an option, different first letters should be used for the text of each menu item. Menus should be limited to about seven items so they will all appear together on one screen. The most frequently used items should be presented first.

Synonyms and Display Order: By default, the items on the menu are displayed in alphabetical order by menu text. If any of the items is assigned a synonym, those items will be displayed before others lacking synonyms. To facilitate menu jumping, synonyms should ideally be unique; numbers are not good choices for synonyms.

To customize the order of the display, each item on the menu can be assigned a Display Order. This field is an option attribute that will be presented when using Edit options. When first assigning a number for the display order, you may want to use 10, 20, and 30 rather than 1, 2, and 3 to permit easier modification in the future if another item needs to be inserted.

PRIORITY: You can set an option's PRIORITY field to set a run priority for an option. Experimentation will be needed to determine the effect of priority settings.

HELP FRAME: You can specify a help frame for an option. The help frame is displayed if, at the "Select..." menu prompt, the user enters ?OPTION (where OPTION is the name of an option).

DISPLAY OPTION: If AUTO MENU is in effect for a user, the items on that user's current menu are always displayed. A problem can arise when, if an option displays output and then quits, AUTO MENU's automatic display of menu options scrolls the output off the screen. Since the AUTO MENU display usually scrolls the option's output off the screen faster than the user can read the output, it can effectively render the option unusable. You can avoid this problem by setting the option's DISPLAY OPTION field to YES. If set to YES and the user has AUTO MENU turned on, Menu Manager will ask "Press RETURN to continue..." after the option completes, but before displaying the list of menu options. The user will then have a chance to review the output before returning to their menu.

REF: For information on other fields in the OPTION file (#19), including how to create options of a type other than Menu, please refer to the "Menu Manager: Developer Tools" chapter in the *Kernel Developer's Guide*.

If the Option Invokes Non-VistA Applications

If you create an option that invokes non-VistA applications (e.g., WordMan or CalcMan) include a call to the Device Handler with the code D HOME^%ZIS in the EXIT ACTION field of the OPTION file (#19) so that the required IO variables will be present when leaving these options. Do the same for any other utility that is known to KILL IO variables upon exit.

If the Option Should Be Regularly Scheduled

If an option should be regularly scheduled to run through TaskMan, you *must* set its SCHEDULING RECOMMENDED field (#209) in the OPTION file (#19) to **YES**. You will not be able to use Schedule/Unschedule Options to schedule an option unless this field is set to **YES** for the option.

Auditing Option Use

Figure 7-4. Auditing menu options

```
SYSTEM MANAGER MENU...                                                  [EVE]
  System Security...                                                   [XUSPY]
    Audit Features ...                                             [XUAUDIT MENU]
      Maintain System Audit Options ...                            [XUAUDIT MAINT]
        Establish System Audit Parameters                              [XUAUDIT]
        Audited Options Purge                                        [XUOPTPURGE]
      Audit Display ...                                                [XUADISP]
        Option Audit Display                                         [XUOPTDISP]
```

You can establish an audit on options to record every time an option is used. You can do this with the Establish System Audit Parameters option [XUAUDIT], which is in the Audit Features [XUAUDIT MENU] menu tree. Simply enter a time to initiate audit and a time to terminate audit. Then enter the specific options you want to audit (you can also choose all options).

Each time a user uses an audited option, an entry is made in the AUDIT LOG FOR OPTIONS file (#19.081). You can display these entries using the Option Audit Display option [XUOPTDISP]. You can purge the AUDIT LOG FOR OPTIONS file (#19.081) with the Audited Options Purge option [XUOPTPURGE].

If Kernel Toolkit is installed at your site, you can also use its Alpha/Beta Test Option Usage menu to count the number of times an option is invoked.

REF: For more information, please refer to the Kernel Toolkit documentation and the *Kernel Security Tools Manual*.

Display Menus and Options Menu

Figure 7-5. Display Menus and Options menu

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
  Menu Management ...                                                    [XUMAINT]
    List Options by Parents and Use                                       [XUXREF]
    Display Menus and Options                                  [XQDISPLAY OPTIONS]
      Abbreviated Menu Diagrams                                        [XUUSERACC2]
      Diagram Menus                                                     [XUUSERACC]
      Inquire                                                           [XUINQUIRE]
      Menu Diagrams (with Entry/Exit Actions)                          [XUUSERACC1]
      Print Option File                                                    [XUPRINT]
```

Kernel provides a number of options to display and diagram menus and options on the Display Menus and Options menu [XQDISPLAY OPTIONS].

Diagramming Options

To discover the menu tree roots of other software applications and how options and suboptions are related, you can use the following menu diagramming options:

Table 7-1. Menu diagramming options to discover tree roots and relationships between options/suboptions

Menu	Description
Abbreviated Menu Diagrams	Outlines the menu tree.
Diagram Menus	Outlines the menu tree, and shows option attributes (e.g., locks and prohibited times).
Menu Diagrams (with Entry/Exit Actions)	Outlines the menu tree, shows option attributes, and shows entry/exit and header actions as well.

Also, the List Options by Parents and Use option [XUXREF] identifies which options have "no parents," and thus, are standalone roots. It also indicates whether options are used as primary menus, secondary menus, and/or as regularly scheduled tasks.

Option Descriptions

To learn more about the options included in a software application, you can use the Print Option File option [XUPRINT] (from the Display Menus and Options menu [XQDISPLAY OPTIONS]) to print the option description, type, and other information. This listing can be sorted by namespace. For example, to print all the VA FileMan options, you can sort from DD to DI.

Displaying Options

To display an option, use the Inquire option:

Figure 7-6. Inquire option—Sample user dialogue

```
Select Display Menus and Options Option:  Inquire

Which OPTIONS item to display: XT-KERMIT MENU <Enter>     Kermit menu

NAME: XT-KERMIT MENU                    MENU TEXT: Kermit menu
  TYPE: menu                            CREATOR: POSTMASTER
  PACKAGE: KERNEL                       E ACTION PRESENT: YES
  X ACTION PRESENT: YES
 DESCRIPTION:   This is the top level menu for kermit functions.  It gives access
to the send, receive, and edit options.
ITEM: XT-KERMIT RECEIVE                 SYNONYM: R
ITEM: XT-KERMIT SEND                    SYNONYM: S
ITEM: XT-KERMIT EDIT                    SYNONYM: E
  EXIT ACTION: D CLEAN^XTKERM4          ENTRY ACTION: D INIT^XTKERM4
  UPPERCASE MENU TEXT: KERMIT MENU
```

Option Access by User Option

Figure 7-7. Option Access by User option

```
Menu Management ...                                                    [XUMAINT]
  Show Users with Selected Primary Menu                               [XUXREF-2]
  Option Access By User                                               [XUOPTWHO]
```

Use the Show Users with Selected Primary Menu option [XUXREF-2] to show which users have been assigned a particular option as a primary or secondary menu. The Option Access by User option [XUOPTWHO] is another cross-referencing tool.

Managing Menus and Options

Managing Primary Menus

When IRM receives new software applications, existing primary menus should be modified to include the new menus. It is not wise to create a new primary menu for every new or unusual circumstance. This would lead to a tremendous variety of menus that would be difficult to sort out and use in the future. Primary menus can be customized with security keys.

REF: For more information on security keys, please refer to the "Security Keys" chapter in this manual.

If there are a few menu options that require special privilege, they can be locked and the security keys assigned to the appropriate users. In this way, a smaller number of primary menus can serve the needs of a larger number of users.

Also, while putting new master menus onto users' secondary menus can be a quick fix, it is not a good idea to do this. Too many options on a user's secondary menu can be cumbersome for the user. In addition, in the long run, it is easier for IRM to manage access to a menu reached from a few well-defined primary menus than to manage access to a menu reached from a large number of users' secondary menus.

Assigning Secondary Menus

An easy way to allocate menu options is to assign them to users individually as SECONDARY MENU OPTIONS. Secondary options are unique for each user and are stored in a multiple in the user's NEW PERSON file (#200) entry. Assignment of SECONDARY MENU OPTIONS should be limited to the essential few, and should not involve deep structures with multiple levels. Instead, new primary menus should be built or existing ones modified. During menu jumping, all branches of both the primary and secondary menu trees are searched each time a jump request is received by the menu system. Greater efficiency and user convenience will result if the depth of the secondary menu trees is confined.

ALWAYS SHOW SECONDARIES Field

You can set the ALWAYS SHOW SECONDARIES field in a user's NEW PERSON file (#200) entry. If set to YES for a user, that user will always have their secondary and common options listed when options on their primary menu are listed (which occurs either by the user entering two question marks ["**??**"] at the "Select..." menu prompt, or when AUTO MENU is turned on).

Redefining the Common Menu

All users automatically have access to the options on the Common menu [XUCOMMAND]) by virtue of the menu system's design. As described earlier, entering two question marks ("**??**") at any select prompt will display the Common menu. The only way to deny access to a particular user is to lock the Common menu option with a reverse key and then allocate the security key to the same user.

REF: For more information on security keys, please refer to the "Security Keys" chapter in this manual.

The items on the Common menu can be left as they are distributed by Kernel, or modified locally as desired. For example, an item can be added to display online help about local computer access policies. This is accomplished by using Edit options to edit the XUCOMMAND menu option. The Item multiple lists the existing menu choices; other locally namespaced options can be added.

If options are locally added to the standard XUCOMMAND menu set, new installations of Kernel will not overwrite the changes. During installation, items on the local XUCOMMAND menu are compared with the exported items. Any previously exported items that were removed by the site will not be added back. Brand new items, however, will be added and any matching items will be updated. Other items that the site may have added will be left in place.

Altering Exported Menus

Generally speaking, exported menu structures should stay intact. If local modifications to exported menus are made, great care *must* be taken to preserve any logic that may exist in the exported structure. For example, the entry action of one option can set up key variables that are then assumed to exist when another option, one further down on the menu tree, is invoked. Although each one of a software's options should be able to be invoked independently once the steps described in the *Kernel Technical Manual* for creating and KILLing software-wide variables have been taken (according to the Programming Standards and Conventions [SAC]), this is not always the case and cannot be assumed.

If an option cannot be invoked independently, the developer can set that option's INDEPENDENTLY INVOCABLE field to NO, as an alert that some other option or action *must* be done before the option can be called.

To give users the options associated with new software applications, IRM should try to allocate the menus as whole entities. If dissection appears necessary, the "Internal Relations" section of the software documentation should be consulted before rearranging any of the items.

Delete Unreferenced Options Option

Figure 7-8. Delete Unreferenced Options option

```
Programmer Options ...  <locked:  XUPROG>                              [XUPROG]
  Delete Unreferenced Options                                 [XQ UNREF'D OPTIONS]
```

All options for interactive use (not designed exclusively as queuable tasks) should normally be tied to a menu that is used as a primary menu or at least as a secondary menu. Standalone options that have no parents and are not menu-type options should be reviewed. They may be obsolete software options or local test options and could be candidates for deletion. Use the Delete Unreferenced Options option [XQ UNREF'D OPTIONS] to delete unreferenced options. It can be used to cycle through the entire OPTION file (#19) and delete non-menu options that are not referenced by other options. Deletion should obviously be done with care. Use of this option is limited to those who hold the XUPROG security key.

Fix Option File Pointers Option

Figure 7-9. Fix Option File Pointers option

```
Menu Management ...                                                    [XUMAINT]
  Fix Option File Pointers                                            [XQOPTFIX]
```

After performing maintenance work on the OPTION file (#19) (e.g., deleting obsolete options that may have been items on a menu), you can use the Fix Option File Pointers option [XQOPTFIX] (see Figure 7-10) to remove any dangling pointers that may have been left in the Item multiple. Running this option is an alternative to having VA FileMan update the pointers each time an individual option is deleted.

Figure 7-10. Fix Option File Pointers option—Sample user dialogue

```
Select OPTION NAME: ZZTEST3 <Enter>          Test Option
NAME: ZZTEST3// @
   SURE YOU WANT TO DELETE THE ENTIRE 'ZZTEST3' OPTION? Y <Enter> (YES)
SINCE THE DELETED ENTRY MAY HAVE BEEN 'POINTED TO'
BY ENTRIES IN THE 'USER' FILE, ETC.,
DO YOU WANT THOSE POINTERS UPDATED (WHICH COULD TAKE QUITE A WHILE)? NO// <Enter>
```

Testing a User's Menus

Figure 7-11. Switch Identities option

```
User Management...                                                      [XUSER]
  Switch Identities                                                [XUTESTUSER]
```

You can test a user's menus using the Switch Identities option [XUTESTUSER]. It lets you test the user's menus and security keys. It does not allow you to execute any bottom-level menu options, however; it only lets you navigate menu trees. You are reminded at each prompt whose menu it is that you are testing. To exit this mode and return to your own menus, simply enter an asterisk ("*****").

Managing Out-Of-Order Option Sets

Figure 7-12. Out-Of-Order Set Management menu options

```
Menu Management ...                                                  [XUMAINT]
  Out-Of-Order Set Management...                                     [XQOOMAIN]
    Create a Set of Options To Mark Out-Of-Order                     [XQOOMAKE]
    List Defined Option Sets                                         [XQOOSHOW]
    Mark Option Set Out-Of-Order                                       [XQOOFF]
    Options in the Option File that are Out-of-Order                [XQOOSHOFIL]
    Protocols Marked Out-of-Order in Protocol File                  [XQOOSHOPRO]
    Recover Deleted Option Set                                       [XQOOREDO]
    Remove Out-Of-Order Messages from a Set of Options                  [XQOON]
    Toggle options/protocols on and off                              [XQOOTOG]
```

Menu Manager, starting with Kernel V. 8.0, provides a mechanism for defining sets of options and protocols, and a way to disable and enable access for these pre-defined option and protocol sets via options on the Out-Of-Order Set Management menu [XQOOMAIN]. This can be handy when you need to repeatedly disable and enable sets of options and protocols.

Use the Create a Set of Options to Mark Out-Of-Order option [XQOOMAKE] to define a set of options. You are prompted first to select options, and then to select protocols.
For both options and protocols, you can use the following to:

- Add a group of options to the set—Use the wildcard asterisk ("*") with or without a namespace.
- Add a range of options to a set—Use **NAM1-NAM2** to add a range of options from NAM1 to NAM2 to the set, where "NAM" represents a namespace.
- Subtract/Remove a group of options from a set—Use the minus sign (i.e., hyphen, "-") followed by a namespace.

Use the Mark Option Set Out-Of-Order option [XQOOFF] to disable access to a set of options. You are asked to enter the message used to place all options in the set out-of-order. The option then places the message in each option's OUT OF ORDER MESSAGE field (#2).

Use the Remove Out-Of-Order Messages from a Set of Options option [XQOON] to enable access to an option set.

To toggle the status of an individual option only, use the Toggle Options/Protocols On and Off option [XQOOTOG].

Out-of-Order Option sets are stored in the ^XTMP global, with a purge date set for seven days in the future. If you place a set of options out of order, but the option set is purged from ^XTMP before you enable access to it, you can rebuild the out-of-order option set using the Recover Deleted Option Set option [XQOOREDO]. It asks you to specify the exact text of the message used to place the set of options out of order; it then recreates an out-of-order option set containing all options currently placed out of order with the specified message

NOTE: Make sure the message you specify is unique to the set of options you are re-enabling.

You can then enable access to the rebuilt option set with the Remove Out-Of-Order Messages from a Set of Options option [XQOON].

To see what sets of options have been grouped in sets on the system, use the List the Defined Options Sets option [XQOOSHOW]. To show all options and protocols currently marked out of order, use the Options in the Option File that are Out-of-Order option [XQOOSHOFIL] and the Protocols Marked Out-of-Order in Protocol File option [XQOOSHOPRO].

Restricting Option Usage

Figure 7-13. Restrict Availability of Options option

```
Menu Management ...                                                [XUMAINT]
  Restrict Availability of Options                               [XQRESTRICT]
```

Options can be restricted in terms of when users can select them and when devices can be used to invoke them. Many of the option restrictions are included in the Restrict Availability of Options option [XQRESTRICT].

OUT OF ORDER MESSAGE: To completely restrict access, you can mark an option to be out-of-order. Do this by entering text in an option's OUT OF ORDER MESSAGE field. If a user attempts to invoke the option, the Out of Order Message will be displayed.

Locks: Both the normal lock, and also the Reverse/Negative lock can be associated with options (as described in the "Security Keys" chapter). Also, M code can be entered in the HEADER, ENTRY ACTION, or EXIT ACTION fields to restrict the use of an option given certain conditions.

Prohibited Times: You can prohibit the use of an option at certain times during the day by assigning a set of prohibited time periods at the "Select TIMES PROHIBITED" prompt.

Permitted Devices: If the RESTRICT DEVICES flag is set to YES, the option can only be invoked on one of the devices listed in the PERMITTED DEVICES field (Multiple). Thus, the running of an option can be restricted. This flag does not affect the choice of devices used for the output from options. It instead controls the processing involved in the use of the option itself.

QUEUING REQUIRED Flag: Using the option Edit options, you can allow users to invoke an option, but force any output to be queued outside of certain times of day, by editing the option's QUEUING REQUIRED field (Multiple). In this multiple's TIME PERIOD (#.01) and DAY(S) FOR TIME PERIOD (#.02) fields enter the time periods and days in which you do not want the option's output to be produced. During these time periods, the output of the options can only be queued. When a user requests a time for queuing, the menu system will determine the next permissible day and time for output. Thus, users can

invoke the option and use it to define the parameters for the subsequent processing, but the actual work will be done during a later time period, presumably when the system is less busy.

Menu Manager Options that Should Be Scheduled

This topic describes the two Menu Manager options that should be regularly scheduled.

Kernel exports a number of other options that should be scheduled to run at regular intervals. Most of these are located on the PARENT OF QUEUABLE OPTIONS menu.

REF: For a complete list, along with suggested scheduling frequencies, please refer to the *Kernel Installation Guide.*

Clean Old Job Nodes in XUTL Option

The Clean old Job Nodes in the XUTL option [XQ XUTL $J NODES] is Kernel's purge option for Kernel globals. This option purges the following globals:

- ^XUTL
- ^UTILITY
- ^TMP
- ^XTMP
- ^XUSEC

Figure 7-14. Clean old Job Nodes in XUTL

```
Operations Management ...                                    [XUSITEMGR]
    Clean old Job Nodes in XUTL                          [XQ XUTL $J NODES]
```

User stacks for each user's job are stored in the ^XUTL global.

REF: For more information, please refer to the "^XUTL Global: Structure and Function" topic in this chapter.

This is also called the compiled menu system. If a job ends abnormally (e.g., upon error, UCI switching, or developer exits that bypass ^XUS), the entries remain in the global (this explains why developers are advised to halt out of programmer mode with D ^XUSCLEAN rather than simply halting.)

The purge routine sets a purge date of seven days in the past. Any user stack in ^XUTL older than seven days is purged. Any entries with a matching $J at the top level of ^UTILITY and ^TMP are also KILLed.

Next, after cleaning out the user stacks in ^XUTL, the purge routine checks ^UTILITY and ^TMP. Any entry at subscript ($J) or (namespace, $J) that does not have a matching entry in the user stacks in ^XUTL is KILLed.

Next, the purge routine checks ^XTMP. Any entry in ^XTMP at subscript (namespace) lacking a header node at (namespace,0), or with a purge date in the header node less than the purge date determined by the purge routine is KILLed.

Finally, the purge routine goes through the signon nodes stored at ^XUSEC(0,"CUR",DUZ,DATE). Any nodes older than the purge date are KILLed.

The XQ XUTL $J NODES option should be queued to run on a regular basis. If separate copies of ^XUTL are maintained on different CPUs, separate entries should be made in the OPTION SCHEDULING file for each CPU so that a separate job will purge each CPU's XUTL global. Because this option deletes any user stacks that are time-stamped with a date earlier than the purge date determined by this option (seven days) you need to take care how frequently you schedule it (in the unusual event of a seven-day long job, this option should obviously not be run).

Rebuilding Primary Menu Trees

Figure 7-15. Building primary menu trees options

```
PARENT OF QUEUABLE OPTIONS                                  [ZTMQUEUABLE OPTIONS]
  Non-interactive Build Primary Menu Trees                       [XQBUILDTREEQUE]
Menu Management ...                                                  [XUMAINT]
  Build Primary Menu Trees                                       [XQBUILDTREE]
```

The menu system uses local menu trees to process requests. When changes are made to the menu structure, the local menu trees are rebuilt (a process also known as microsurgery). If a user attempts an "Up-arrow Jump" when the local trees need to be rebuilt or are being rebuilt, a message is issued about quick access being temporarily disabled; the user will not be able to jump to reach the option. Microsurgery is triggered in the following situations:

- The option Edit options is used.
- An Out-of-Order option set is enabled or disabled.
- A sufficiently large number of changes have been made to a menu tree.

It is also recommended to rebuild all primary menu trees every other day during non-peak hours, using the XQBUILDTREEQUE option. If separate copies of ^XUTL are maintained on different CPUs, separate entries should be made in the OPTION SCHEDULING file (#19.2) for each CPU so that a separate job will rebuild each CPU's ^XUTL global.

Primary menu trees can also be built/repaired immediately using the Build Primary Menu Trees option. In particular, if menu jumping has stopped working and microsurgery is not fixing the menus, use the Build Primary Menu Trees option to force a menu rebuild to fix the problem.

Error Messages During Menu Jumping

There are some conditions under which a menu jump may not be completed. In these cases the user will see one of the following error messages:

Figure 7-16. Menu jump error message (1 of 6)

```
I NEED TO REBUILD MENUS .... QUICK ACCESS IS TEMPORARILY DISABLED  Please proceed
to {target option's menu text}
```

This means that the time stamps on the OPTION file (#19) and the ^XUTL global indicate that the OPTION file (#19) has been modified since the menus were compiled in ^XUTL and the global is therefore locked until XQ8 can recompile the modified menus. This error message can be generated by both user-generated jumps and phantom jumps.

Figure 7-17. Menu jump error message (2 of 6)

```
*** WARNING ***
Illegal jump requested to option '{option's menu text}'  Jump pathway locked at
option '{locked option's menu text}'
```

This indicates that a locked option for which the user does not possess the security key has been encountered in the tree between the option where the jump was requested and the target option to which the jump was requested. This error message can be generated by both user-generated jumps and phantom jumps.

Figure 7-18. Menu jump error message (3 of 6)

```
*** WARNING ***
Illegal jump was requested to option '{option menu text}'  Jump path out of order
from '{option's menu text}'  with message '{out of order message}'
```

This means that an option on the tree between the option where the phantom jump was requested and the target option has been marked as out of order (OUT OF ORDER MESSAGE Field [#2] of the OPTION file [#19]). This error message can be generated by both user-generated jumps and phantom jumps.

Figure 7-19. Menu jump error message (4 of 6)

```
*** WARNING ***
Illegal jump was requested to option '{option menu text}'  Variable XQUIT
encountered at option '{option name}'
```

This means that the jump logic has encountered the variable XQUIT (detected with a $DATA statement). This variable is usually set by an Entry Action (Field #20 of the OPTION file [#19]) and causes the menu system to refuse to run or jump past that option. This error message can be generated by both user-generated jumps and phantom jumps.

Figure 7-20. Menu jump error message (5 of 6)

```
*** WARNING ***
Background jump requested to option '{value in XQMM("J")}'  but this option does
not exist on your system.
```

A VA FileMan lookup was attempted for the option set in the variable XQMM("J") but no such option was found in the OPTION file (#19). This error message can only be generated from a phantom jump.

Figure 7-21. Menu jump error message (6 of 6)

```
*** WARNING ***
Background jump requested to option '{option's menu text}' but  you do not have
access to this option.  See your computer  representative.
```

This means that the target option requested by XQMM("J") is not in the tree of options to which this user has access (that is, the target option was neither in the user's primary menu tree nor specifically listed as a secondary menu for that user). This error message can only be generated from a phantom jump.

REF: For more information on phantom jumps, please refer to the "Menu Manger: Developer Tools" chapter in the *Kernel Developer's Guide*.

^XUTL Global: Structure and Function

The ^XUTL global is an account-specific global. It should exist in each production account on your system. This global is created primarily from information in the OPTION file [^DIC(19)] and is therefore sometimes referred to as "the compiled menu system."

^XUTL is divided into three main sections:

- **User Stacks**

  ```
  ^XUTL("XQ",$J)
  ^XUTL("XQT",$J)
  ```
  (MENU templates only)

- **Display Nodes**

  ```
  ^XUTL("XQO",ien)
  ```

- **Jump Nodes**

  ```
  ^XUTL("XQO","P"_ien)
  ```

User Stacks

User stacks are stored in nodes in ^XUTL("XQ",$J) and ^XUTL("XQT",$J).

The example illustrated in Figure 7-22 shows a typical user stack. In this case the $J is 541065826.

The "XQ" nodes can be divided into meaningful sets according to what is contained in the third subscript. The numeric third subscripts begin with the zero node which is set to the date and time in VA FileMan format by the program ^XUS1 when the user logs on or ^%XUCI when the user is changing UCIs.

The other numeric, third subscripts (in this case the numbers 1 to 3) reflect the user's progression through the menu system. Each time a new option is invoked, a new node is created which contains the option number, concatenated with a "P", the number of the option whose compiled menu tree contains the current option, a caret ("^"), and the zero-node of the OPTION file (#19) for that option. A different format is used for options in a user's secondary menu tree.

A pointer in the node ^XUTL("XQ", $J, "T") indicates which option in this list of numbered nodes the menu driver is currently using. This pointer is set and reset by the menu driver as the user moves up and down the menu tree. In the example, XUPROGMODE is the option that the menu driver is currently using.

Other "XQ" nodes of the global that have a non-numeric third subscript are used to store various pieces of Kernel information that are set up at signon. ^XUTL("XQ",$J,"XQM") points to the user's primary menu.

In the following example (Figure 7-22), the user's primary menu is OPTION file (#19) entry #29.

Figure 7-22. User Stack example

```
^XUTL("XQ",541065826,0) = 2920113.081624
^XUTL("XQ",541065826,1) = 29P29^EVE^Systems Manager
                          Menu^^M^.5^^192^^^^^^n^1^^^
^XUTL("XQ",541065826,2) = 31P29^XUPROG^Programmer Options^^M^^
                          XUPROG^^^^^^^n^^
^XUTL("XQ",541065826,3) = 49P29^XUPROGMODE^Programmer mode^^R
                          ^^XUPROGMODE^^^^^^^ n^^
^XUTL("XQ",541065826,"DUZ") = 63
^XUTL("XQ",541065826,"DUZ(0)") = LlPp
^XUTL("XQ",541065826,"DUZ(2)") = 16000
^XUTL("XQ",541065826,"IO") = _TNA5103:
^XUTL("XQ",541065826,"IOBS") = $C(8)
^XUTL("XQ",541065826,"IOF") = #,$C(27,91,50,74,27,91,72)
^XUTL("XQ",541065826,"IOM") = 79
^XUTL("XQ",541065826,"ION") = LAT DEVICE
^XUTL("XQ",541065826,"IOS") = 158
^XUTL("XQ",541065826,"IOSL") = 24
^XUTL("XQ",541065826,"IOST") = C-VT100HIGH
^XUTL("XQ",541065826,"IOST(0)") = 149
^XUTL("XQ",541065826,"IOT") = VTRM
^XUTL("XQ",541065826,"IOXY") = W $C(27,91)_((DY+1))_$C(59)_((DX+1))_$C(72)
^XUTL("XQ",541065826,"T") = 3
^XUTL("XQ",541065826,"XQM") = 29
```

XQT Nodes (MENU Templates)

The "XQT" nodes are used to create a stack of options similar to the "XQ" stack when a MENU template is invoked. These nodes are translated from the ^VA(200,DUZ,19.8) Multiple when a user precedes an option selection with a left square bracket character, "**[**", much like a PRINT template is invoked in VA FileMan. For example, if the user has defined a MENU template named "DOIT" using the Menu Template options of the User's Tool Box, typing "**[DOIT**" will load that sequence of options into the "XQT" nodes and begin executing them. When a MENU template is requested by the user, the option tree of that template is loaded into the "XQT" nodes and remains loaded as long as the user is logged on. Further requests for "[DOIT" will use that same stack.

Display Nodes

Display nodes are stored in ^XUTL("XQO", internal number).

The first example below (Figure 7-23) shows the display nodes for EVE, the System Manager's Menu. The internal number of EVE in this particular OPTION file (#19) is 29. In the first part of the example the option names and menu texts, along with a limited number of fields for that option compiled from the OPTION file (#19), are concatenated together. It is from this part that XQ2 (the menu display program) gets the information it needs.

In the second part, all the menu texts and synonyms are listed in order in uppercase. It is here that XQ tries to match what the user entered at the terminal with the correct option. The third part of the example, the 0th node of the options, is listed by number and provides the remaining information that the Menu

System may need to make the option work. To understand what the various "^" pieces mean, look at a VA FileMan global format data dictionary listing of the OPTION file (#19).

Illustrated in the second example (Figure 7-24) is the display node for the SECONDARY MENU OPTIONS of a user whose DUZ is equal to 66. Here, the user has only a single secondary menu called "Secondary Menu" (with an internal number of 580 in the OPTION file [#19]). The various parts of this example are identical to those of the Display Nodes for the EVE menu example above.

NOTE: The second subscript, instead of pointing to a menu in the OPTION file (#19), is a "U" concatenated with the user's DUZ which points to the NEW PERSON file (#200) entry. This is because secondary menu options are stored in the SECONDARY MENU OPTIONS field in the NEW PERSON file (#200) entry for each user.

Figure 7-23. Display Nodes for EVE example

```
^XUTL("XQO",29,0) = 2^55048,38923
^XUTL("XQO",29,0,1) = ^XUCORE^Core Applications ...^NOT
                      AVAILABLE^^^^^^^XUTIO^Device Handler
                      ...^^^^n^^FM^DIUSER^VA FileMan ...^^^^n^^^XMMGR^
                      Manage Mailman ...^^^^^^^XUMAINT^Menu Management
                      ...^^^^n^^^XUPROG^Programmer Options ...^^XUPROG^^^
                        ...^
^XUTL("XQO",29,0,2) = ^XUSITEMGR^Operations Management ...^^^^^^^XU-SPL-MGR
                        ^Spool Management ...^^^^^^^XUSPY^System Security
                      ...^^^^^^^ZTMMGR^Task Manager ...^^^^n^^^XUSER^User
                      Edit ...^^^^^^
^XUTL("XQO",29,"CORE APPLICATIONS") = 40^1
^XUTL("XQO",29,"DEVICE HANDLER") = 32^1
^XUTL("XQO",29,"FM") = 19^0
^XUTL("XQO",29,"MANAGE MAILMAN") = 30^1
^XUTL("XQO",29,"MENU MANAGEMENT") = 9^1
^XUTL("XQO",29,"OPERATIONS MANAGEMENT") = 174^1
^XUTL("XQO",29,"PROGRAMMER OPTIONS") = 31^1
^XUTL("XQO",29,"SPOOL MANAGEMENT") = 415^1
^XUTL("XQO",29,"SYSTEM SECURITY") = 226^1
^XUTL("XQO",29,"TASK MANAGER") = 83^1
^XUTL("XQO",29,"USER EDIT") = 39^1
^XUTL("XQO",29,"VA FILEMAN") = 19^1
^XUTL("XQO",29,"^",9) = ^XUMAINT^Menu Management^^M^^^105^^^n^n^^n^^^^
^XUTL("XQO",29,"^",19) = FM^DIUSER^VA FileMan^^M^^^^^^n^^^n^1^^
^XUTL("XQO",29,"^",30) = ^XMMGR^Manage Mailman^^M^^^299^^^^^54^^1^1^^^
^XUTL("XQO",29,"^",31) = ^XUPROG^Programmer Options^^M^^XUPROG^^^^^^^n^^
^XUTL("XQO",29,"^",32) = ^XUTIO^Device Handler^^M^^^413^^^n^^20^n^^
^XUTL("XQO",29,"^",39) = ^XUSER^User Edit^^M^^^153^^^^^^n^^
^XUTL("XQO",29,"^",40) = ^XUCORE^Core Applications^1^M^^^^^^^^^n^^
^XUTL("XQO",29,"^",83) = ^ZTMMGR^Task Manager^^M^^^^^^n^^50^^1^^
^XUTL("XQO",29,"^",174) = ^XUSITEMGR^Operations Management^^M^^^^^^^y^^n^^
^XUTL("XQO",29,"^",226) = ^XUSPY^System Security^^M^^^^^^^^119^n^^
^XUTL("XQO",29,"^",415) = ^XU-SPL-MGR^Spool Management^^M^^^419^^^^^20^^
```

Figure 7-24. Display Nodes for a secondary menu

```
^XUTL("XQO","U66",0) = 1^54927,30758
^XUTL("XQO","U66",0,1) = ^ZZTSTSM^Secondary Menu ...^^^^n^^
^XUTL("XQO","U66","SECONDARY  MENU") = 580^1
^XUTL("XQO","U66","^",580) = ^ZZTSTSM^Secondary Menu^^M^^^^^^n^^^^1^1^^1
```

Jump Nodes

Jump nodes are stored in ^XUTL("XQO","P"_internal number), where there is one "P_..." entry in ^XUTL("XQO") for each primary menu that exists. The jump nodes, for each primary menu, store the pathways to all options that can be jumped to.

The jump nodes are created in the XQ8* series of programs. They are very similar to display nodes, except that:

1. They have a "P" concatenated on the front of the primary option's number in the second subscript.
2. These nodes describe the entire primary menu tree rather than just the single level tree.

Examples of the jump nodes for a single primary menu are shown in Figure 7-25 and Figure 7-26. Since these nodes can be very extensive in number, some nodes have been removed from the examples to save space.

In the first example (Figure 7-25) are the "lookup" nodes, where the jump software tries to match a menu text or synonym with what the user has entered at the terminal. Each node is set to its internal number in the OPTION file (#19) and, in the second "^" piece, a 0 if it is a synonym or a 1 if it is menu text.

In the second example (Figure 7-26), the "menu pathway" entries below the "P580" node show all of the options that can be jumped to from the primary menu whose internal entry number (IEN) is 580. Each entry contains list(s) of the series of options that *must* be navigated through in a jump from the primary menu. In the case of the option DILIST (# 17), the list of options that will have to be processed is 520,519,518,411,17. If, as in the case of ZZTEST4 (# 318), there is more than one possible pathway, then each is listed along with various other necessary pieces of information (e.g., locks, time restraint, etc.).

Figure 7-25. Jump Nodes example—Lookup nodes

```
^XUTL("XQO","P580",0) = 55165,28536
^XUTL("XQO","P580","19^") = 394^0
^XUTL("XQO","P580","2ND SECOND LEVEL MENU TEST^") = 575^1
^XUTL("XQO","P580","3^") = 518^0
^XUTL("XQO","P580","ACTN^") = 391^0
^XUTL("XQO","P580","ALL^") = 420^0
```

Figure 7-26. Jump Nodes example—Menu Pathways

```
^XUTL("XQO","P580","LIST FILE ATTRIBUTES^") = 17^1
^XUTL("XQO","P580","TEST 4^") = 318^1
...
^XUTL("XQO","P580","TOOL^") = 581^0
^XUTL("XQO","P580","X-TYPE OPTION TEST^") = 576^1
^XUTL("XQO","P580","X^") = 576^0
^XUTL("XQO","P580","ZDAVE^") = 411^1
^XUTL("XQO","P580","^",5) = ^XUEDITOPT^Edit
                            options^^E^581,5,^^106^^^^^20^n^^^^
^XUTL("XQO","P580","^",17) = ^DILIST^List File Attributes^^A^
                            520,519,518,411,17,^^^^^n,^y^^n^1^^^
...
^XUTL("XQO","P580","^",318) = ^ZZTEST4^Test
                            4^^O^520,575,397,318,^^^^^n,^^^^^^
^XUTL("XQO","P580","^",318,0) = 2
^XUTL("XQO","P580","^",318,0,1) = 520,575,578,397,318,^^^n,^
^XUTL("XQO","P580","^",318,0,2) = 520,575,578,318,^^^n,^
...
^XUTL("XQO","P580","^",579) = ^ZZLEVEL3B^Phantom
                            Mother^^M^520,575,579,^^^^^n,^^^^1^1^^1
^XUTL("XQO","P580","^",580) = ^ZZTSTPM^Primary Menu^^M^^^^^^n^^^^1^1^^1
^XUTL("XQO","P580","^",581) = ^ZZLUKTOOLS^Luke's
                            Tools^^M^581,^^^^^^^^^1^1^^1
```

Menu Manager Variables (Troubleshooting)

There is a group of Menu Manager variables that is always defined. It may be useful for IRM staff to know what these variables signify when investigating errors. If an error is reported in VA FileMan's DIP routine, for example, knowing the value of XQY at the time of the error indicates which option was invoking the DIP routine. The option can then be reviewed to discover the name of the routine that was calling DIP.

Table 7-2. Menu Manger variables (always defined)

Variable	Description
XQABTST	Flag that signals whether alpha-beta testing is in effect.
XQDIC	Internal entry number (IEN) of the option's parent (which *must* be a menu) in the OPTION file, if an option is executing. If the user is in a menu, XQDIC is set to the IEN of the current menu's parent (unless they are in their primary menu, in which case XQDIC is set to the IEN of the primary menu). The value of XQDIC also corresponds to the second subscript in the display nodes portion of the ^XUTL global, ^XUTL("XQO",) for the menu in question.
XQPSM	Like XQDIC, a lookup value into the second subscript of ^XUTL, the compiled menu global. XQPSM points to the tree of the target option in the jump. It resulted from the ability to jump to any option, not just ones on the primary menu tree. It can help identify jumps from a primary, secondary, or Common option.
XQT	Current option's type (e.g., M for menu, A for action).
XQUR	User's response to the menu prompt (replaces A).

Variable	Description
XQUSER	User's name in the form SEVEN A. KRNUSER.
XQY	Internal entry number (IEN) of the current option or menu (replaces Y).
XQY0	First node (subscript of zero) of the current option (replaces Y(0)).
XQXFLG	Contains several flags, including whether capacity management testing is active.

8 Security Keys

User Interface

Security keys are primarily used to allow access to specially protected options. If a software application exports a menu that has one or two options that require a secured level of access, they can use security keys to lock those special options. When an option is locked, you can only use the locked option if you hold the security key matching the key with which the option was locked.

Entering two question marks ("**??**") at the menu system's select prompt displays the current options. If any of the options are locked, that fact is listed also, along with the names of any associated security keys. In the following example, the option Programmer Options is locked with a security key named XUPROG:

Figure 8-1. Sample locked menu options showing required security key—Entering two question marks (??)

```
Select Systems Manager Menu Option: ??
          Device Handler ...                                          [XUTIO]
          Menu Management ...                                       [XUMAINT]
          Programmer Options ...                                     [XUPROG]
             **> Locked with XUPROG
```

You can list which security keys you currently hold by using the Display User Characteristics option on the Common menu. It displays a list of all security keys you hold, similar to the following:

Figure 8-2. Display User Characteristics option—Sample output

```
KEYS HELD
---------
   XUPROG          XUMGR          XUPROGMODE      XUAUTHOR        ZTMQ
```

The security keys you need to carry out computing activities should be assigned by IRM when your computer account is first added to the system. Other keys can be allocated at a later time by IRM or by an IRM designee (e.g., an application coordinator) with the use of the Secure Menu Delegation menu utilities.

System Management

Identifying Locked Options

IRM can list which security keys lock what options by using Menu Management's Diagram Menus option. The following example (Figure 8-3) shows that the Programmer Options menu is locked with the XUPROG security key. It also shows that one of its options, Programmer mode, is locked with the XUPROGMODE security key:

Figure 8-3. Diagram Menus option—Sample user dialogue

```
Select Menu Management Option:  Diagram Menus
Select USER (U.xxxxx) or OPTION (O.xxxxx) name: O.XUPROG
Programmer Options (XUPROG)
**LOCKED: XUPROG**
-------------------------PG Programmer mode
                  [XUPROGMODE]
                    **LOCKED: XUPROGMODE**
```

Security keys are stored in the SECURITY KEY file (#19.1). Security keys given to users are stored in the users' NEW PERSON file (#200) entries, in the KEYS field (Multiple).

Options are locked by a given security key when the name of that key is entered into the LOCK field (#3) of the OPTION file (#19). If an option is locked, users need to be given the security key in order to invoke the option.

Key Management

Keys are defined and allocated to users with options on the Key Management menu.

Figure 8-4. Key Management menu options

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
  Menu Management ...                                                     [XUMAINT]
     Key Management ...                                                  [XUKEYMGMT]
       Allocation of Security Keys                                        [XUKEYALL]
       De-allocation of Security Keys                                   [XUKEYDEALL]
       Enter/Edit of Security Keys                                        [XUKEYEDIT]
       All keys a user needs                                               [XQLOCK1]
       Change user's allocated keys to delegated keys                 [XQKEYALTODEL]
       Keys for a given menu tree                                          [XQLOCK2]
       Delegate keys                                                      [XQKEYDEL]
       List users holding a certain key                                   [XQSHOKEY]
       Remove delegated keys                                              [XQKEYRDEL]
       Show the keys of a particular user                                 [XQLISTKEY]
```

Allocating and De-allocating Security Keys

The main option to assign security keys to a user or users is the Allocation of Security Keys option [XUKEYALL]. Allocating a security key to a user lets them invoke options that are locked with the key. For options with reverse locks, allocating the security key locks the user out from the option. In either case, allocating the key to a user does *not* allow the user to give the key to anyone else.

To remove a security key from a user, use the De-allocation of Security Keys option [XUKEYDEALL].

Unless you have been delegated a security key, the only way you can allocate or de-allocate keys is if you hold the XUMGR security key or have a FILE MANAGER ACCESS CODE field (#3) of "**@**".

REF: For more information on delegating security keys, please refer to the "Delegating Security Keys" topic that follows.

All of the security keys that a new user needs to use their assigned options can be determined by using the All Keys a User Needs option on the Key Management menu [XQLOCK1]. This produces a list of the primary and secondary menus for that user, and compiles a list of the keys for that menu tree. This list can then be assigned or delegated. It can also be edited before the keys are given to the user. Similarly, the Keys For a Given Menu Tree option [XQLOCK2] examines a menu and lists all of the security keys associated with all sibling options.

Delegating Security Keys

Delegating keys allows you to give a user the ability to assign specific security keys to other users, as opposed to the XUMGR security key and "**@**" VA FileMan Access code (i.e., FILE MANAGER ACCESS CODE field [#3]), which allow all keys to be assigned.

One way to delegate security keys is to use the Change user's allocated keys to delegated keys option [XQKEYALTODEL]. This option delegates to a user all of the security keys that are currently allocated to that user. Any entries in their KEYS field (Multiple) are entered in the DELEGATED KEYS field (Multiple) as well. They can now use the Allocation of Security Keys option [XUKEYALL] to give the security keys to others.

Alternatively, IRM can use the Delegate keys option [XQKEYDEL] to populate the DELEGATED KEYS field (Multiple) one-by-one.

A user who has been delegated a security key can allocate that key to others in two ways:

- Through the Allocation of Security Keys option [XUKEYALL], if it is on their menu.
- By delegating an option locked by the security key in question; the key will be allocated along with the option.

The key recipients (excepting holders of the XUMGR security key or a FILE MANAGER ACCESS CODE field [#3] of "**@**") *cannot* assign the security key to others, however, even if they have access to the Allocation option, because the key does not exist in their DELEGATED KEYS field (Multiple).

One example of key delegation is an IRM designee, delegated the Provider key, who allocates that key to incoming medical residents.

For security reasons, users who have a key in their DELEGATED KEYS field (Multiple) cannot allocate that key to themselves. That key *must* be awarded by another user who has been delegated the key or by an IRM staff member who holds the XUMGR system security key.

Creating and Editing Security Keys

Keys can be created using the Enter/Edit of Security Keys option [XUKEYEDIT] on the Key Management menu. If a security key has already been defined, its name cannot be edited. It also cannot be deleted, as discussed below. Other key attributes stored in the SECURITY KEY file (#19.1) can be used for special purposes. Attributes of the Provider key are shown in the following example:

Figure 8-5. Attributes for the Provider security key—Sample user dialogue

```
Select SECURITY KEY NAME: PROVIDER
NAME: PROVIDER// <Enter>
DESCRIPTIVE NAME: Provider// <Enter>
PERSON LOOKUP: LOOKUP// <Enter>
KEEP AT TERMINATE: YES// <Enter>
DESCRIPTION:
  1>This KEY is given to all entries in the New Person file that need
  2>to be looked up as a Provider. Those entries that hold this key
  3>are considered to be providers.  It was given to all active
  4>Providers in file 6 at the time of the Kernel 7 install.
EDIT Option: <Enter>
Select SUBORDINATE KEY: <Enter>
GRANTING CONDITION: <Enter>
```

No editing

PERSON LOOKUP: As described in the "Security Keys: Developer Tools" chapter in the *Kernel Developer's Guide*, a special AK cross-reference on the NEW PERSON file (#200) is maintained automatically for anyone who is granted a security key that is flagged for Person Lookup. This cross-reference has been introduced to facilitate identification of user groups, like providers.

KEEP AT TERMINATE: As described in the "Signon/Security" section concerning user deactivation, security keys that are marked as "KEEP AT TERMINATE" will *not* be removed as a user attribute of terminated users. This allows the continued processing of activities that had been previously authorized because the user held the key.

SUBORDINATE KEY (Exploding Keys): If a security key has any associated subordinate keys (entries in this multiple), the subordinate keys are automatically assigned along with the overall key. A security key with this feature is called an exploding key since it and its subordinates are assigned all at once.

NOTE: If entries in the SUBORDINATE KEY Field (Multiple) are edited, dynamic updating of the security keys already assigned to users does *not* occur.

Exploding security keys *cannot* be exported with software, although, there may be support for this functionality in the future. They are intended to be created by IRM as a timesaving method in the key allocation process.

Deleting Security Keys

Keys should not be deleted from the SECURITY KEY file (#19.1). Kernel has made the NAME field (#.01) of the SECURITY KEY file (#19.1) uneditable to prevent deletion of security keys through VA FileMan. IRM should *not* attempt to edit the key global directly to remove a key, since associated pointing relationships will be left to cause errors. The one mechanism Kernel does provide for deletion of security keys is through the Kernel Installation and Distribution System (KIDS).

REF: For more information, please refer to the "KIDS" section in this manual and the "KIDS Developer Tools" chapter in the *Kernel Developer's Guide*.

Reindexing All Users' Security Keys Option

Figure 8-6. Reindex the users key's option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
User Management ...                                                  [XUSER]
   Manage User File ...                                        [XUSER FILE MGR]
      Reindex the users key's                                [XUSER KEY RE-INDEX]
```

You can use the Reindex the users key's option [XUSER KEY RE-INDEX] to re-index all users' security keys in the NEW PERSON file (#200). If a user has a security key, but is lacking the corresponding ^XUSEC cross-reference for the key, you can use this option to regenerate the ^XUSEC cross-reference. While the ^XUSEC cross-reference is being rebuilt, there can be an impact on all users with security key lookups failing in ^XUSEC until the index is entirely rebuilt; therefore, this option should be used with caution and is best delayed until users are not signed on.

Using Security Keys with Reverse Locks

If a security key is associated with an option via the REVERSE/NEGATIVE LOCK field, rather than the LOCK field (#3), it functions to lock out users who hold the key. The security key used for a reverse lock is just like any other key, differing only in the way it is associated with an option. Menu Management's Diagram Menus option indicates the existence of any reverse locks, such as the use of the XMNOPRIV security key to prevent access to MailMan's shared mail facility.

The typical use of a security key with the REVERSE/NEGATIVE LOCK field is to restrict access to options otherwise available to all users (e.g., MailMan User and other options on the Common menu).

Security Key Delegation Levels

Starting with Kernel V. 8.0, security keys are subject to delegation levels just as options are subject to delegation levels. A field in the NEW PERSON file (#200), DELEGATION LEVEL, stores a user's delegation level (for security keys and options). When a security key is delegated, the person to whom it is delegated is assigned a level one number lower than the delegation level of the person doing the delegating. This is to prevent the delegated-to person from removing DELEGATED KEYS from someone with a lower delegation level.

REF: For more information about delegation levels, please refer to the "Secure Menu Delegation" chapter in this manual.

9 Secure Menu Delegation

The job of allocating menu options to users can be a time-consuming activity, so site managers may want to consider delegating this responsibility to application coordinators. Application coordinators are familiar with the menus for their software and can learn how to assign these to new users in their service area.

Secure Menu Delegation allows the Site Manager to delegate the management of certain menu options to another user (e.g., an application coordinator). This user, now a delegate, can then assign these as primary or secondary options (along with their security keys) to users who fall under their administrative jurisdiction.

For example, the Site Manager might delegate the management of the Laboratory software options to the Lab Application Coordinator (LAC), and the LAC could then allocate or remove options from everybody in the Laboratory software. The system is set up in such a way that the LAC could also delegate, with the Site Manager's permission and manager's menu, the management of all the chemistry menus to the head of the Chemistry Section, and so on, creating another level of delegation.

There are two divisions in Secure Menu Delegation:

- The menu to create and manage delegates.
- The menu for the delegates themselves to assign options to end users.

User Interface: Acting as a Delegate

As a delegate, you have been delegated options (usually by IRM). If you have been delegated options, you can assign these options to computer users on the computer system.

As a delegate, you can assign the following options to your users:

- Options that have been delegated to you.
- Menus that you have created from options delegated to you.
- Options you have created from VA FileMan templates.

As a delegate, you need to understand the basic structure of the OPTION file (#19), which is a file that points back to itself. That is, a menu is an entry in the OPTION file (#19); but items on menus are themselves pointers to other entries in the OPTION file (#19). You should also understand the difference between types of options, be familiar with menu trees, and be sufficiently reluctant to assign great numbers of secondary menus.

Delegate's Menu

To delegate options to users, you need to be assigned a menu called Delegate's Menu Management [XQSMD USER MENU], which is located under the Secure Menu Management menu. The options on the Delegate's Menu Management menu are as follows:

Figure 9-1. Delegate's Menu Management options

```
Delegate's Menu Management                                    [XQSMD USER MENU]
  Build a New Menu                                          [XQSMD BUILD MENU]
  Edit a User's Options                                    [XQSMD EDIT OPTIONS]
  Copy Everything About an Option to a New Option                    [XQCOPYOP]
  Copy One Users Menus and Keys to others                  [XQSMD COPY USER]
  Limited File Manager Options (Build)                [XQSMD LIMITED FM OPTIONS]
```

Each of these options on the delegate's menu is discussed in the topics that follow.

Edit a User's Options

Using the Edit a User's Options option [XQSMD EDIT OPTIONS] allows you to edit a user's primary and secondary menus. This is the chief method you can use to add (and subtract) options on your users' menus.

Most of your work will be in adding and deleting options on your users' secondary menus. You are only able to add or delete options from a user's secondary menu if the option in question has been delegated to you. That means that you do not have access to a user's entire secondary menu; instead, only those options on the secondary menu that are also delegated to you.

If, when you edit a user's secondary menu, you choose an option that is already on a user's secondary menu, you are asked if you want to delete it from their secondary menu. Otherwise, you are asked if you want to add the option to their secondary menu.

If you are assigning an option that is locked with a security key, the delegation process checks whether you have been delegated the key as well. If you have, the key is automatically assigned to the user along with the option. If you have not been delegated the key, you get an error message saying that you have not been delegated the needed security key (the option is assigned to the user, but they will not have the key to unlock the option).

If you delete an option that is locked with a security key and that key is delegated to you (and you are at a higher key delegation level than the option holder), the key is deleted along with the option (unless the user holds another option locked by the same security key).

Example

In this example, the menu option LRZ MAIN is added to the user's secondary menu. LRZ MAIN is locked with a security key and that key is automatically assigned when the option is assigned:

Figure 9-2. Edit a User's Options—Sample user dialogue

```
Select Delegate's Menu Management Option: Edit a User's Options

Select NEW PERSON NAME: KRNUSER,FIVE
     PRIMARY MENU OPTION: XMUSER// <Enter>      MailMan Menu  .
     No keys needed to delete!.
     No keys needed to give!

     SECONDARY MENU OPTION: LRZ MAIN <Enter>    Lab User Menu  ...
     ZZLRMAIN key also given!

     SECONDARY MENU OPTION: <Enter>

Select NEW PERSON NAME:
```

Unlike secondary menus, you are only able to edit a user's PRIMARY MENU OPTION if their current primary menu is an option that has been delegated to you. Otherwise, you are not allowed to change that user's PRIMARY MENU OPTION.

NOTE: You *cannot* add or subtract options on a user's primary menu; you can only replace the user's entire PRIMARY MENU OPTION with another one.

Build a New Menu Option

Using the Build a New Menu option [XQSMD BUILD MENU], located on the Delegate's Menu Management menu [XQSMD USER MENU, you can create new menus, with menu items chosen from your delegated options.

First, you need to provide an option name for the new menu you are creating. The menu option name *must* begin with a namespace assigned to you by IRM. Once you provide a name for the menu, you are asked:

- Text for the menu.
- Description for the menu.
- Items for the menu (choose from your delegated options).

Once you have created a new menu, you can assign it to your users just as if it were an option delegated to you.

Copy Everything About an Option to a New Option Option

Using the Copy Everything About an Option to a New Option option [XQCOPYOP], you can copy any option on the computer system into a new option. First you are asked which existing option you would like to copy; then, you are asked for a name for the copied option. The option name *must* begin with a namespace assigned to you by IRM.

Copy One Users Menus and Keys to others Option

Using the Copy One Users Menus and Keys to others option [XQSMD COPY USER], you can copy the menus and security keys of one user to another user. Each menu or security key you copy, however, *must* have been delegated to you; otherwise, they are skipped in the copy process. What gets copied from one user into the other user are:

- PRIMARY MENU OPTION (and all descendant menus).
- SECONDARY MENU OPTIONS.
- KEYS.

The PRIMARY MENU OPTION of the user you're copying from *replace***s** the PRIMARY MENU OPTION of the user you are copying to. The SECONDARY MENU OPTIONS and the KEYS of the user you're copying from are *merged* into the SECONDARY MENU OPTIONS and the KEYS of the user you're copying to.

Limited File Manager Options (Build) Option

The Secure Menu Delegation system provides a way for delegates to create options out of VA FileMan templates. Delegates who have enough access to VA FileMan to create INPUT, SORT, or PRINT templates can create menu options for their users that directly call these templates.

Characteristics of Intended Users

The Limited File Manager Options (Build) option [XQSMD LIMITED FM OPTIONS] is designed for delegates, such as some application coordinators who have VA FileMan access to a set of files and can create INPUT, SORT, or PRINT templates. These delegates may have the VA FileMan options for editing or printing without the ability to modify data dictionaries. They may also have explicit file access to a specified set of files via the File Access Management system. Typically they would be working without the special FILE MANAGER ACCESS CODE field (#3), DUZ(0).

IRM Setup to Enable Building Options from Templates

To allow a user to create menu options from VA FileMan templates, IRM *must* first assign to the user:

- Delegate's Menu Management menu [XQSMD USER MENU].
- XQSMDFM Security Key.
- A namespace in which to create options. Do this with the Secure Menu Delegation option Specify Allowable New Menu Prefix [XQSMD SET PREFIX]. This forces any options created by the user to be namespaced with the assigned namespace, appended with a Z, appended with an option name.

Building Options

The tool for building options with VA FileMan templates is called the Limited File Manager Options (Build) option [XQSMD LIMITED FM OPTIONS]. It is part of the Delegate's Menu Management menu under the Secure Menu Management menu and is locked with the XQSMDFM security key.

First, you *must* have created a SORT, PRINT, or INPUT template for a VA FileMan file. Once you have created a template, you can make this template available as an option to your users by turning it into an option.

You can create three types of options:

- Edit-type option (from an EDIT template).
- Print-type option (from PRINT and SORT templates).
- Inquire-type option (from either a PRINT template or a file name).

Once you have turned the template into an option, you can assign that option to your users as you deem necessary. Then, when a user uses the option, they execute the PRINT, SORT, or INPUT template from which the option was created.

Example

Suppose you have created a PRINT template called LRZ REFERRAL PRINT for the Lab's REFERRAL file. To turn this PRINT template into an Inquire option, use the Limited File Manager Options (Build) option, as shown below:

Figure 9-3. Limited File Manager Options (Build)—Sample user dialogue

```
Select Delegate's Menu Management Option:  Limited File Manager Options (Build)
The menu options you build or edit must begin with the namespace:
          LRZ


The option types that may be built are P(rint), E(dit), and I(nquire), and
you must have a template or templates ready to be included in the option.

Or enter D(elete) to DELETE an option


Select Option Type (P/E/I/D): I
     Enter Print Template Name (Optional): LRZ REFERRAL PRINT

     Option Name: LRZ REFERRAL INQUIRE
  Located in the LR (LAB SERVICE) namespace.
  ARE YOU ADDING 'LRZ REFERRAL INQUIRE' AS A NEW OPTION (THE 996TH)? Y <Enter>
(YES)
   OPTION MENU TEXT: Display a Referral
MENU TEXT: Display a Referral  Replace <Enter>
DESCRIPTION:
  1> Display Lab Referral entries (option created by LAB ADPAC).
  2> <Enter>
EDIT Option: <Enter>

Select Delegate's Menu Management Option:
```

System Management: Managing Delegates

The options for creating and managing delegates are on the Secure Menu Delegation menu [XQSMD MGR], which is on the Menu Management menu. Typically, IRM would be the sole holder of this menu. The options on this menu are:

Table 9-1. Secure Menu Delegation menu options

Option Text	Function
Select Options to be Delegated	Delegate options
List Delegated Options and their Users	Print Report
Print All Delegates and their Options	Print Report
Remove Options Previously Delegated	Undo Delegation
Replicate or Replace a Delegate	Copy a Delegate
Show a Delegate's Options	Print Report
Delegate's Menu Management ...	Delegate's menu
Specify Allowable New Menu Prefix	Assign namespaces

The main options to create and manage delegates are:

- Select Options to be Delegated
- Replicate or Replace a Delegate

Delegating Options: Select Options to be Delegated Option

To delegate options, use the Select Options to be Delegated option from the Secure Menu Delegation menu. Using this option is a two-step process:

1. Choose the users to whom options are delegated.
2. Choose which options to delegate to that group of users.

You can choose to set up one user or many users as delegates. You can choose one option or a group of options to delegate to them.

You also need to assign (not delegate!) the Delegate's Menu Management menu [XQSMD USER MENU] to the delegate; this menu gives delegates the means to assign delegated options to users.

Figure 9-4. Delegating Options: Select Options to be Delegated option—Sample user dialogue

```
Select Secure Menu Delegation Option: Select Options to be Delegated

Enter the name(s) of your delegate(s), one at a time

 Name: KRNUSER,THREE

 Name: KRNUSER,FOUR

 Name: <Enter>

Enter options you wish to DELEGATE TO these users

Add option(s): XUINQUIRE

Add option(s): XUUSERACC

Add option(s): <Enter>

For the following user(s):

1. KRNUSER,THREE
2. KRNUSER,FOUR

You will delegate the following options:

XUINQUIRE   Inquire
XUUSERACC   Diagram Menus

 Delegated by KRNUSER,FIVE on Jul. 21, 2004  3:55 PM.


Ready to delegate these options to these people? Y// <Enter>

Request to add delegated options has been queued, task # 465,
     named: KRNUSER,FIVE adding delegated options.
```

Delegating Security Keys

If options that you intend to delegate are locked with security keys, you need to delegate the matching keys to the delegate; otherwise, the delegate will *not* be able to assign keys to unlock options they have assigned to their users.

If the option is locked with a security key that you possess, the Select Options to be Delegated option branches you to the Key Management program, and lets you allocate (if you so wish) the appropriate keys to the delegates you are creating.

However, to assign security keys to users, the delegate *must be delegated* the key. To do that, you need to use the Key Management menu option, Delegate keys option [XQKEYDEL]. This option allows you to delegate security keys to delegates by populating the DELEGATED KEYS field (Multiple) in their NEW PERSON file (#200) entry. Security keys entered in a delegate's DELEGATED KEYS Multiple allow them to allocate the entered keys to other users (but not themselves).

When a delegate assigns options to a user, they can assign the matching security keys as part of that process. However, as an enhancement to a delegate's ability to work with keys, IRM can assign the delegate the following options from the Key Management menu:

- Allocation of Security Keys
- De-allocation of Security Keys
- Show the Security Keys of a Particular User

As long as the delegate does not hold the XUMGR security key, which allows any key to be allocated, the Key Management menu options only allow delegates to allocate and de-allocate security keys they've been delegated. Kernel also follows key delegation levels with the Allocation of Security Keys and De-allocation of Security Keys options.

NOTE: Key management options *must* be separately assigned; they are not a part of the Delegate's Menu Management menu [XQSMD USER MENU].

Delegation Level (Options and Keys)

DELEGATION LEVEL is a field in the NEW PERSON file (#200) specifying the number of steps that a person is from the original delegation of options by the Site Manager (whose Delegation Level is 0). Starting with Kernel V. 8.0, the delegation level is also maintained for DELEGATED KEYS. For instance, if the Site Manager delegates all laboratory options to the Lab ADP Application Coordinator (ADPAC), then the Lab ADPAC would have a Delegation Level of 1. Should the Lab ADPAC further delegate a set of those options to the Chief of Chemistry, the Chief would have a level of 2, and so on.

The use of levels insures that supervision is not compromised such that the lower level user could alter menus or remove security keys of the higher level person. No attempt is made to determine who actually works for whom since that information is not available to the software. Delegation chains should therefore be constructed with some care.

To modify the set of options (and accompanying security keys) delegated to a particular person, you *must* have a Delegation Level equal to, or less than, the person you are trying to modify. If you create a new delegate by delegating some (or all) of the options delegated to you, that person will have a Delegation Level equal to your level +1.

It may be necessary to modify Delegation Levels using the VA FileMan as the organization's structure changes over time.

Further Delegation

The only way a delegate can delegate, rather than simply assign, options to someone else is if the delegate has access to the Select Options to be Delegated option [XQSMD ADD], or the Replicate or Replace a Delegate option [XQSMD REPLICATE]. These options should only be on the Secure Menu Delegation menu [XQSMD MGR]. You should carefully evaluate whether to give this menu to delegates (which gives them the right to further delegate).

Options Too Sensitive to Delegate

Certain options (e.g., Programmer-related options) are considered too sensitive or powerful to be delegated. They are marked as not delegable in the OPTION file (#19), and the Secure MenuMan Delegation software will *not* delegate these options. The traditional methods of assigning these menu options *must* be employed by the Site Manager.

It should be noted that a higher-level option, such as EVE, would still give the delegate access to lower level options, such as XUMAINT, even though XUMAINT is itself marked in the OPTION file (#19) as non-delegable. The Delegation software does not follow the option trees down to insure that options of options are not delegable.

CAUTION: It is highly recommended that the Site Manager, Information Security Officer (ISO), or IRM Chief review the options marked as too sensitive to be delegated and, using VA FileMan, add any locally sensitive options to this list.

It is the responsibility of each site to insure that the security of the system is not violated.

Replicate or Replace a Delegate Option

You can copy the Delegated Options of a delegate to another user. Use the Replicate or Replace a Delegate option [XQSMD REPLICATE] to do this. The options that you transfer to another user do not replace any options the user has been previously delegated. They will be added to those options, if any. Like the Select Options to be Delegated option, this option also can branch you to the security key allocation program for the new delegate.

You are also asked if the delegated options should be removed from the original delegate. If you say NO ("N"), the original delegate remains a delegate. If you say YES ("Y"), all Delegated Options are removed from the original delegate, who will no longer be an active delegate. In order to remove the options from a delegate, however, you *must* have a Delegation Level lower than they do.

Remove Options Previously Delegated Option

To simply remove an option from a delegate's list of delegable options, use the Remove Options Previously Delegated option:

1. Enter the name or names of the delegate(s) from which you want to remove options.
2. Enter the option or options you want to remove from the specified set of delegates.

You're given a chance to review the choices you made; if you say to proceed, a task is queued that removes the options you selected from the delegates you specified.

Specify Allowable New Menu Prefix Option

Use the Specify Allowable New Menu Prefix option to assign allowable menu prefixes to your delegates. Your delegates need to be given allowable new menu prefixes if they:

- Build new menus.
- Copy options.
- Create options from VA FileMan templates.

Typically, if your delegate works with one particular software application, you would assign them that software's namespace as an allowable prefix. Options that the delegate creates *must* then be prefixed with that namespace, appended with a Z.

If you do not specify an allowable prefix for a delegate, they will *not* be able to use the following options:

- Build a New Menu
- Copy Everything About an Option to a New Option
- Limited File Manager Options (Build)

You can specify multiple new menu prefixes for a given delegate.

Reports

You can use the following options to generate reports about delegates on your system:

- List Delegated Options and their Users
  (Sort by delegated option.)
- Print All Delegates and their Options
  (Sort by delegate name.)
- Show a Delegate's Options
  (Display all delegated options for one delegate.)

10 Alerts

User Interface

When you receive an alert, something on the computer system is requesting your immediate attention. A software application might issue an alert to one or more users when certain conditions are met (e.g., depleted stock levels or abnormal lab test results).

The first time you reach a menu prompt after receiving a particular alert, the alert's message is displayed to you by the menu system. The alert message is displayed along with a standard notice to select the View Alerts "VA" option on the Common menu to process the alert (see Figure 10-1).

When you receive an alert, you should find out what the alert is asking of you, and attend to it. This is called processing the alert.

Until you process all unprocessed alerts you receive, you'll be reminded that you have pending alerts each time you're at a menu prompt. You will not, however, see the alert message; you only see that the first time you receive an alert and reach the menu prompt.

Figure 10-1. Alert—Sample user message

```
Dr. You need to enter a progress note on 'KRNPATIENT,ONE'.
          Enter  "VA   VIEW ALERTS     to review alerts


Select Systems Manager Menu Option:
```

Processing Alerts

To process alerts, choose the View Alerts "VA" option from the Common menu. The View Alerts "VA" option presents a list of all pending alerts, numbered consecutively with the most recent alerts listed first.

Information-only alerts are displayed with the letter "I" in front of the alert message. When you process information-only alerts, all that happens is that they are removed from the pending alerts list. Their only purpose was to send you the one-line alert message.

When you process alerts that are *not* information-only, processing the alert may send you to a particular option or program. Afterwards, you are returned to the View Alerts screen if more alerts need processing, or back to the menu prompt if no pending alerts remain.

There are various methods for processing alerts from the View Alerts screen. You can enter any of the following alert process codes (listed alphabetically):

Table 10-1. Alert processing codes

Process Code	Description
A	Process all alerts in the order shown.
D	Delete specific alerts (some alerts can *not* be deleted). Only listed if one or more INFORMATION-ONLY alerts have been listed. If unable to delete an alert users will see: "Unable to delete alerts which require action: n,n,n, …"
F	Forward one or more specific alerts. Forwarding may be sent as an alert to a specific user(s) and/or mail group(s), a mail message, or sent to a specific printer.
I	Process all INFORMATION-ONLY alerts. Only listed if one or more INFORMATION-ONLY alerts have been listed.
M	List pending alerts in a mail message and deliver the message to your IN basket.
n	Single number to process a single alert.
n,n,n-n	Range of numbers to process a range of alerts (e.g., 1,3,5-8).
P	Print a copy of the pending alerts to a printer.
R	Redisplay available alerts.
S	Add or remove a surrogate to receive alerts for you. An optional start and end date can also be entered.
^	Exit the alert processing screen.

The Alert Handler ordinarily deletes alerts once you have processed the alert. If you have processed all pending alerts, and try to select the View Alerts "VA" option, nothing is displayed. View Alerts only offers a listing when there are pending alerts; if no alerts are pending, View Alerts simply returns you to the menu prompt.

Figure 10-2. View Alerts "VA" option—Sample user dialogue

```
ACCESS CODES: ********
VERIFY CODES: ********
Good evening  Jim  You last signed on Jan 9,2004 at 14:39

Dr. You need to enter a progress note on 'KRNPATIENT,ONE'.
          Enter  "VA   VIEW ALERTS     to review alerts

Select Clinic Manager Menu Option: "VA
 1.   Dr. You need to enter a progress note on 'KRNPATIENT,ONE'.
 2.   Alk Phos elevated, schedule fu bone scan
 3.I  For your information, meeting at 12 noon, room 223
          Select from 1 to 3
          or enter ?, A, I, F, S, P, M, R, or ^ to exit: ?

YOU MAY ENTER:
   One or more numbers in the range 1 to 3 to select specific alert(s)
      for processing.  This may be a series of numbers, e.g., 2,3,6-9
   A to process all of the pending alerts in the order shown.
   I to process all of the INFORMATION ONLY alerts, if any, without further ado.
   S to add or remove a surrogate to receive alerts for you
   F to forward one or more specific alerts.  Forwarding may be as an ALERT
to specific user(s) and/or mail group(s), or as a MAIL MESSAGE, or to a
specific PRINTER.
   D to delete specific alerts (some alerts may not be deleted)
   P to print a copy of the pending alerts on a printer
   M to receive a MailMan message containing a copy of these pending alerts
   R to Redisplay the available alerts
   ^ to exit
   or RETURN to see additional pending ALERTS


          Select from 1 to 3
          or enter ?, A, I, F, S, P, M, R, or ^ to exit
          or RETURN to continue:
```

Deleting Alerts

As of Kernel Patch XU*8.0*114, you can delete alerts by using the "D" alert processing code when viewing alerts. The user can, if desired, delete specific alerts without viewing and/or processing them. This option provides the ability to delete "INFORMATION ONLY" alerts. Alerts that require processing can *not* currently be deleted. However, if alerts requiring processing are created with the XQACNDEL variable set to 1 they too would be able to be deleted (i.e., the developer of the code that creates the alert can specify if it *must* be processed or can be deleted). Any alerts that were selected for deletion, but could not be deleted will be noted for the user.

The ability for the user to delete alerts other than INFORMATION ONLY will require that the developers within a software application decide that specific alerts, which would normally invoke processing via an option or routine, can be deleted specifically by the user *without* processing. They would then set the XQACNDEL variable to a value of 1 (one) prior to calling SET^XQALERT to set up the alert. Deletion of an alert by the user (or by IRM or ADPAC staff using the existing option) is noted within the ALERT TRACKING file (#8992.1) as deletion by a user (with the user ID) *without* processing of the alert.

Forwarding Alerts

Beginning with Kernel V. 8.0, you can forward alerts by using the "F" alert processing code when viewing alerts.. You can choose one or more alerts and forward them in the following ways:

- Forward as alert(s) to a specific user on the computer system.
- Forward as alert(s) to a mail group on the system.
- Copy alert(s) into mail message(s) and send to users and mail groups on the system.
- Print to an output device on the system (e.g., a printer).

Surrogates and Alerts

Beginning with Kernel Patch XU*8.0*114, you can designate and/or remove a surrogate for alerts by using the "S" alert processing code when viewing alerts. The user can, if desired, specify a start date/time and/or an end date/time for the surrogate to be effective. If a start date/time is not specified, the surrogate becomes active immediately. If an end date/time is specified, the surrogate will be removed automatically effective with the first alert sent to the user after the end date/time has passed. If an end date/time is *not* specified, the surrogate is active until another surrogate is specified or the user removes the surrogate.

A message is sent to the surrogate to indicate that he has been designated as a surrogate, and a message is sent when the surrogate is removed.

If the user has no alerts and selects the alert option, he will be asked if he wants to add or remove a surrogate. The XQALERT SURROGATE SET/REMOVE option is also provided. It can be used by IRM or ADPAC staff to add or remove a surrogate for a selected user. This option is located on the Alert Management menu.

System Management

An alert notifies one or more users of a matter requiring immediate attention. Thus, alerts function as brief notices that are distinct from mail messages or triggered bulletins.

Starting with Kernel V. 8.0, alerts are stored in the ALERT file (#8992, stored in ^XTV(8992,). Also the ALERT TRACKING file (#8992.1), stored in ^XTV(8992.1,) provides a means to track alerts and users' responses to alerts.

For each user to whom an alert is sent, the ALERT TRACKING file (#8992.1) stores the following data:

- Alert name
- Date created
- Software identifier of alert
- User who generated the alert
- Message text of the alert
- Action associated with the alert
- Data associated with the alert

For each recipient of the alert, the ALERT TRACKING file (#8992.1) stores the following data:

- First date and time observed (shown in menu cycle)
- First date and time selected for processing
- Date and time processing completed (if any)
- Date and time alert was deleted
- Forwarding information—If alert was forwarded, user who forwarded it, and date and time of forwarding
- Surrogate information—If a surrogate was added for alerts, user who was the surrogate, and date and time of the surrogate

The PATIENT^XQALERT and USER^XQALERT functions provide access to information in the ALERT TRACKING file (#8992.1).

REF: For a description of the XQALERT and other alert-related APIs, please refer to the "Alerts: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

Alert Management Menu

The Alert Management menu [XQALERT MGR] contains the following options, described below:

Figure 10-3. Alert Management menu options

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Operations Management ...                                                [XUSITEMGR]
  Alert Management...                                                   [XQALERT MGR]
    SURO Alerts - Set/Remove Surrogate for User      [XQALERT SURROGATE SET/REMOVE]
         Delete Old (>14 d) Alerts                          [XQALERT DELETE OLD]
         Make an Alert on the fly                                  [XQALERT MAKE]
         Purge Alerts for a User                         [XQALERT BY USER DELETE]
            **> Locked with XQAL-DELETE
         Report Menu for Alerts ...                             [XQAL REPORTS MENU]
         Set Backup Reviewer for Alerts                [XQAL SET BACKUP REVIEWER]
         Surrogate for which Users?                [XQAL SURROGATE FOR WHICH USERS]
```

Alerts - Set/Remove Surrogate for Users Option

The Alerts - Set/Remove Surrogate for User option [XQALERT SURROGATE SET/REMOVE] is provided so that IRM or ADPAC personnel can do the following:

- Set a surrogate to receive alerts for a user.
- Remove a surrogate from receiving alerts for a user.

The option asks for a user to be selected, then is ready to specify a new surrogate for the selected user, or to remove the current surrogate for that user.

This option is *not* needed by the individual users who may select to name or remove a surrogate as one of the options while processing alerts (or if no alerts are present for the user, as his/her only option on selecting alert processing).

Delete Old (>14 d) Alerts Option

The Delete Old (>14 d) Alerts option [XQALERT DELETE OLD] performs the following functions:

- Purges unprocessed alerts from the ALERT file (#8992).
- Purges alert tracking information from the ALERT TRACKING file (#8992.1).
- Forwards unprocessed alerts to supervisors and/or surrogates.

You can use the Delete Old (>14 d) Alerts option to purge all alerts that have been unprocessed for longer than a specified retention period (the default is 14 days.) It is assumed that an alert becomes obsolete within this period and can be purged by IRM staff. This option also performs additional functions, which are described below.

This option can be run either directly or as a queued job. You can specify a retention period other than the 14-day default when you queue the option only, by using the TASK PARAMETERS field of the

OPTION SCHEDULING file (#19.2). If you put a numeric value in the TASK PARAMETERS field, this value replaces the default alert retention value of 14 days.

The Delete Old (>14 d) Alerts option also purges the ALERT TRACKING file (#8992.1). It purges all entries in the ALERT TRACKING file (#8992.1) that are more than 30 days old. The only exception is if, when an alert is created, the call to create the alert specified a retention period different than 30 days; in this case, the different period is used.

Finally, this option forwards unprocessed alerts to supervisors and surrogates (if this was requested when the alert was created). However, if the period to wait before forwarding exceeds the purging retention period used by this option, the alerts will be purged rather than forwarded.

Due to the number of tasks performed by this option, it should be queued through TaskMan on a regular basis. The suggested scheduling frequency is once every day.

Make an Alert on the Fly Option

The Make an Alert on the Fly option [XQALERT MAKE] allows you to generate an alert on the fly. It interactively asks you for the alert message, recipients, and alert action, if any (you can specify an alert action type of routine or option). It then generates the alert on the fly.

This option is recommended primarily for IRM personnel and ADPACs; it may or may not be appropriate for other selected users.

NOTE: This option does *not* allow the user to set the CAN DELETE WITHOUT PROCESSING field (#.1) in the ALERT file (#8992)

Purge Alerts for a User Option

The Purge Alerts for a User option [XQALERT BY USER DELETE] allows you to delete alerts for a user. The main purpose of this option is to provide a way to delete alerts for a user who has been inactive for a period of time (e.g., on leave), and who has accumulated a number of alerts that should *not* need processing.

This option is locked with the XQAL-DELETE security key, and should only be used by IRM personnel and/or ADPACs.

Report Menu for Alerts Menu

The Report Menu for Alerts menu [XQAL REPORTS MENU] This menu provides several options for generating reports on alerts for users or patients. It consists of the following submenu items:

Figure 10-4. Report Menu for Alerts menu options

```
Select Report Menu for Alerts Option: ??

Critical Alerts Count Report                              [XQAL CRITICAL ALERT COUNT]
List Alerts for a user from a specified date              [XQAL ALERT LIST FROM DATE]
Patient Alert List for specified date                       [XQAL PATIENT ALERT LIST]
User Alerts Count Report                                     [XQAL USER ALERTS COUNT]
View data for Alert Tracking file entry                [XQAL VIEW ALERT TRACKING ENTRY]
```

Critical Alerts Count Report Option

The Critical Alerts Count Report option [XQAL CRITICAL ALERT COUNT] is used to generate a report of users who have more than a specified number of alerts containing the word "critical" or the words "abnormal imaging" between the specified start and end dates. The report is presented in descending order for the number of critical/abnormal imaging alerts present.

For each user who has the specified number of critical/abnormal imaging alerts or more, the report includes the following:

- User name
- Section/Service for the user
- Number of alerts in the ALERT file (#8992)
- Last signon date
- Number of Critical alerts or Abnormal Imaging alerts
- Date of the oldest alert

List Alerts for a user from a specified date Option

The List Alerts for a user from a specified date option [XQAL ALERT LIST FROM DATE] is used to obtain an interactive list of alerts from the ALERT TRACKING file (#8992.1) for a specified user starting from a specified date.

The listing includes the following:

- Internal Entry Number (IEN) for the alert in the ALERT TRACKING file (#8992.1)
- Date and time the alert was generated
- Message text of the alert
- Information about any option or routine to be executed for processing the alert

Patient Alert List for specified date Option

The Patient Alert List for specified date option [XQAL PATIENT ALERT LIST] is used to obtain a list of alerts for a specified patient from the ALERT TRACKING file (#8992.1) for a selected date.

A prompt is provided to obtain a quick scan listing of dates with at least some alerts for the patient on it based on OR and DVB alerts (other patient related alerts need to be identified by looking at each alert's message text and are included in the full list, but not the quick scan).

The listing includes the following:

- Internal Entry Number (IEN) for the alert in the ALERT TRACKING file (#8992.1)
- Date and time the alert was generated
- Message text of the alert
- Information about any option or routine to be executed for processing the alert

User Alerts Count Report Option

The User Alerts Count Report option [XQAL USER ALERTS COUNT] is used to generate a report on users who have more than a specified number of alerts in the ALERT file (#8992). The report covers a specified range of dates, and can be sorted by any of the following data:

- User name
- Number of alerts
- Service/Section

In addition, the report in each of these formats may be generated by Divisions if desired.

For each user who has the specified number of alerts or more, the report includes the following:

- User name
- Section/Service for the user
- Number of alerts in the ALERT file (#8992)
- Last signon date
- Number of Critical alerts or Abnormal Imaging alerts
- Date of the oldest alert

View data for Alert Tracking file entry Option

The View data for Alert Tracking file entry option [XQAL VIEW ALERT TRACKING ENTRY] can be used to view data for one or more entries in the ALERT TRACKING file (#8992.1) in captioned format. The internal entry numbers for the entries to be displayed *must* be entered individually.

Set Backup Reviewer for Alerts Option

The Set Backup Reviewer for Alerts option [XQAL SET BACKUP REVIEWER] was added with Kernel Patch XU*8.0*174.

This option provides a mechanism for a user to set entries into the PARAMETERS file (#8989.5) that will assign an individual as the "Backup Reviewer for Unprocessed Alerts," which is the DISPLAY TEXT Field (#.02) for the "XQAL BACKUP REVIEWER" entry in the NAME field (#.01) in the PARAMETER DEFINITION file (#8989.51), if there is a date specified in the DAYS FOR BACKUP REVIEWER field (#.15) in the ALERT DATE/TIME Multiple field (#.01) in the ALERT file (#8992) for that alert.

If this is the case, an alert that remains unread for the specified number of days will be forwarded to the "Backup Reviewer for Unprocessed Alerts" indicated at the lowest level found for processing for the user in the PARAMETERS file (#8989.5). The following is the processing order (listed lowest to highest level):

1. User
2. OERR Team
3. Team
4. Service
5. Division
6. System

Surrogate for which Users? Option

The Surrogate for which Users? option [XQAL SURROGATE FOR WHICH USERS] provides a view of which users have specified a selected user as surrogates for themselves.

11 Server Options

System Management

What is a Server Option?

A server option is a special type of option (stored in the OPTION file [#19]) that can be triggered by mail messages. Addressing a mail message to a server option is termed a "server request." A server request awakens the option and causes it to execute the following:

- Any M code in the server option's ENTRY ACTION field (#20).
- Any M code in the HEADER field (#26).
- The routine indicated in the ROUTINE field (#25).
- Any M code in the EXIT ACTION field (#15).

A server-type option is similar to a run routine-type option. The difference is that a server option is activated by a mail message while a run routine option is activated by a user choosing that option from a menu on a screen. Server options should only be invoked by mail messages (never directly by a user).

The form of the mail message that activates the server option is identical to any other mail message except that it is addressed to S.<option name>. The "S." (like the "G." form for sending to mail groups) routes the message to the server request software.

What Can Server Options Do?

A server request might trigger a bulletin, send a MailMan reply, and/or initiate an audit of itself. Developers and local IRM staff can also customize the bulletins or MailMan replies.

Can Server Requests Be Denied?

Only server-type options can be activated by mail messages. The following *must* be true for a server request to be processed:

- The server option *must* be set to type "**s**" in the TYPE field (#4) of the OPTION file (#19). If the type is not "**s**" and a request is received, it results in an error that, by default, is recorded in the AUDIT LOG FOR OPTIONS file (#19.081).
- The server option name *must* be complete and exact when a server request is made or the request will be denied.
- The server option *must not* be disabled (it can be disabled for all requests by setting its LOCK (#3) and/or OUT OF ORDER MESSAGE fields).

As long as the conditions listed above are satisfied, the only mechanism a site has for security for server requests is the setting of the server option's SERVER ACTION field (#221). This field has the following settings:

Table 11-1. SERVER ACTION field (#221) security values for server requests

Value	Description
R	**R**un immediately. This code causes the server request to be honored in real time as soon as it is received from MailMan (run immediately), provided it is *not* prevented by a setting in the TIMES/DAYS PROHIBITED field (#3.91).
Q	**Q**ueue server. This code causes the server request to be honored (queued) as soon as permitted by the TIMES/DAYS PROHIBITED field (#3.91).
N	**N**otify local authorities. This code causes the server request to create a TaskMan entry but does *not* schedule it to run. A local mail group is notified along with the task number so that it can be approved locally and then scheduled to run using TaskMan's Requeue Tasks option.
I	**I**gnore any server requests. This code causes the software to ignore all requests for this server option. A bulletin or MailMan message can still be sent, however.

When a server request is received, the server option itself is executed similarly to the way a normal option is executed. That is, if a server request causes a server option to be run or queued, the server option, (along with its associated entry action code, header code, routine, and exit action code), does *not* run until the option as a whole runs as scheduled by TaskMan.

How Can the Number of Instances of a Server Option Be Controlled?

To tie a server option to a device of type RESOURCES, use the SERVER DEVICE field (#227) and set the SERVER ACTION field (#221) to "Q" (Queue server) in the OPTION file (#19). This allows you to control how many instances of the server option can run at any one time. Only as many server option processes can run at any one time as are set up in the associated device's RESOURCE SLOTS field (#35) in the DEVICE file (#3.5). So if 30 mail messages come in at the same time and attempt to fire off 30 server option processes, you can control the maximum number of simultaneous processes that actually run. Additional server options will be able to run when resource slots are freed up from the resource device.

Setting Up a Server Option

A server option has many fields in common with other option types and is set up using the Menu Management option Edit options. This option calls the FileMan edit template option [XUEDITOPT], which prompts for data to be entered in the following fields (listed in field number order):

Table 11-2. OPTION file (#19) field values when setting up a server option

Field Name	Description
NAME (#.01)	This should be a namespaced set of 3 to 30 uppercase letters.
MENU TEXT (#1)	Since there is never a menu prompt for a server option, this field should instead contain an accurate description of what this server option does, as it is used by the server request in error messages, bulletins, and MailMan replies. It should be 3 to 50 characters in length.
OUT OF ORDER MESSAGE (#2)	If this field contains between 1 and 80 characters of text, the server option is placed "out of order" and will *not* be activated by a server request. The message itself is included in bulletins or MailMan replies that report the failure.
LOCK (#3)	Since server options have no online user associated with them, the existence of a lock in this field prevents the execution of a server option, much like an OUT OF ORDER MESSAGE. The user for all server options is the PostMaster. The originator of a server request is recorded, however, in the return address variable.
DESCRIPTION (#3.5)	This word-processing field should contain an extensive description of the server option intended for the local Site Manager and IRM staff. The description should include an exact description of what the server option does and the resources it requires.
PRIORITY (#3.8)	This field determines the priority at which the server option runs.
TIMES/DAYS PROHIBITED (#3.91)	This multiple allows the local IRM staff to control the days and times during which the server request is honored. If data is entered that prevents the server option from being honored immediately, the software determines the next available time slice that is not prohibited and queues the request for that time. Server options that are marked "R" for Run Immediately in the SERVER ACTION field are instead queued to run at the next non-prohibited time period.
TYPE (#4)	This field *must* always contain the code "**s**" for server-type option or the request will be denied and an error will result.
EXIT ACTION (#15)	The M code stored in this field is executed just before the server option exits.
ENTRY ACTION (#20)	The M code in this field is executed if the server request is honored. If, as with other options, the variable XQUIT exists after the Entry Action is executed, the request is terminated at that point and an error is generated.
ROUTINE (#25)	If there is a routine name in this field in the forms ROUTINE,

Field Name	Description
	^ROUTINE, or TAG^ROUTINE, the routine is run.
HEADER (#26)	This field of M code is executed, if it exists.
SERVER BULLETIN (#220)	This field is a pointer to the BULLETIN file (#3.6); it indicates the bulletin to use to notify the local mail group of a server request on their system. If there is no bulletin entered in this field, the default bulletin XQSERVER is used. Unless there are pressing reasons to do otherwise, it is recommended that the default bulletin XQSERVER be used by leaving the SERVER BULLETIN field blank. If the mail group(s) pointed to by XQSERVER (or the bulletin pointed to in this field) does not contain an active user (i.e., a user possessing a Verify code and no effective TERMINATION DATE) the software turns on auditing (i.e., SERVER AUDIT described below) and sends a MailMan message to the local PostMaster.  **CAUTION: The most common reason for server options not functioning is that there is no active user associated with the bulletin specified. For security reasons, server options will *not* run without a locally defined active user associated with the chosen bulletin.**
SERVER ACTION (#221)	This SET OF CODES field allows the local IRM staff to decide how a server request is to be treated (see Table 11-1).
SERVER MAIL GROUP (#222)	This field is a pointer to another mail group (the first is pointed to by XQSERVER and/or the bulletin in Field #220) to which server request notifications are to be sent. The software will notify all legitimate users in all mail groups pointed to. It is recommended that this field be left blank and a mail group be assigned the chosen bulletin instead.  **CAUTION: Server options will *not* work unless there is a local, active user associated with the specified mail group.**
SERVER AUDIT (#223)	This field causes the server request to be audited in the AUDIT LOG FOR OPTIONS file (#19.081). The default is YES. The information stored for an audited server option includes: • Option name • User (always PostMaster) • Device • Job number • Date/Time • CPU • Message number • Return address of sender • Subject of the message

Field Name	Description
	• Error message A server option can also be audited using the normal option auditing software. Auditing the PostMaster or the namespace "XQSRV" will capture all server requests.
SUPPRESS BULLETIN (#224)	If set to "Y" (YES), it prevents a bulletin from being sent under normal conditions. If there is an error or a possible security breach, a bulletin will still be fired. If the field is *not* filled in, it takes the default of "N," which means that the sending of bulletins is *not* suppressed.
SERVER REPLY (#225)	This SET OF CODES controls the MailMan reply to a server request. The reply is a message returned to the user who has sent the server request and should not be confused with the local user to whom the bulletin is addressed. If a reply is requested, the software uses the return address of the sender as supplied by MailMan to send a local or network reply.  **REF:** For an example of a server-type option return message, please refer to the Figure 11-2. The possible codes are: **N** No reply is sent (the default). **E** A reply is sent to the return address of the sender only in the event of an error. **R** A reply is always sent.
SERVER DEVICE (#227)	Optionally, use this field and the SERVER ACTION field (#221) set to "Q" (Queue server) to control the number of server requests for this server option that can be processed at any one time. Enter the name of a device of type RESOURCES (in the DEVICE file [#3.5]). The number of instances of this server option that can run at any one time is limited to the number of resource slots in the selected resource device (i.e., RESOURCE SLOTS field (#35) in the DEVICE file (#3.5)).

Testing if a Site is Reachable: XQSPING Server Option

You can use the XQSPING server option to invoke the Kernel XTSPING utility at a site. This utility tests to see if the domain to which a message is addressed is reachable. For example, if you want to see if the network link to the Oakland OIFO is working properly, you could address a message to:

```
S.XQSPING@ISC-SF.VA.GOV
```

If the text of the message and the subject are simply the line "Testing", you should get the following message in return:

Figure 11-1. Sample message received when "pinging" a domain address

```
MailMan message for User2,JOHN  COMPUTER SPECIALIST
Subj: PING reply to: TESTING [#999] 28 Nov 92 12:17  1 line
From: PING SERVER in 'IN' basket.
-------------------------------------------------------
Testing.
```

The XTSPING utility copies the message addressed to it and returns it to the person who sent it.

Testing a Server Option: XQSCHK

You can use the XQSCHK server option to return information about a server option on a remote system. You should list the server option you want to test in the text of the message addressed to XQSCHK. The subject of the message sent to the XQSCHK server option is not important. However, the body of the text *must* contain the name of the server option to be checked. When you specify the server option to be checked, do *not* precede the server option name with an "S.", instead, list the server option's name exactly as it appears in the OPTION file's .01 field.

The XQSCHK server option returns Fields #220 to #225 from the OPTION file (#19) to show how the option has been set up. In addition, several other things about the option are investigated and error or warning messages may be also returned.

For example, if you want diagnostic information about a server option named ZZSERVER, and the option resides on the system at the Oakland OIFO, you should create a message containing the text ZZSERVER and send it to:

```
S.XQSCHK@ISC-SF.VA.GOV
```

The XQSCHK server option at Oakland will unload the name of the server option (in this example ZZSERVER, see Figure 11-2). Assuming such a server option exists, you would expect to receive a reply in a MailMan message as shown below:

Figure 11-2. XQSCHK server option—Sample MailMan return message

```
MailMan message for KRNUSER,ONE  COMPUTER SPECIALIST
Subj: Server Request Reply from SF-ISC.VA.GOV
From: Postmaster  in 'IN' basket
-------------------------------------------------------------------------------

                Nov. 28, 1992  12:18 PM

    Sender: KRNUSER,ONE
    Option name: ZZSERVER
    Subject: TESTING XQSCHK
    Message #: 999


    This is a reply from ISC-SF.VA.GOV
    Checking Server Option ZZSERVER.

    Fields 220 to 225 in the Option File:
           220 - No bulletin selected, will use default XQSERVER.
           221 - The server action code is Run Immediately.
           222 - The mail group ZZGROUP is pointed to.
           223 - Auditing is turned off.
           224 - The server's bulletin is not suppressed.
           225 - Reply mail is sent when an error is trapped.
```

Errors and Warnings from the XQSCHK Server Option

The following table lists the errors or warnings that might be included in the return message from the XQSCHK server option, along with an explanation of each:

Table 11-3. XQSCHK server option—Error/Warning messages

Error/Warning Message	Description
Can't unload name of server from message: [message subject].	The name of the server option to be tested could *not* be unloaded from the text of the message sent to waken the XQSCHK server option. The message should contain just the name of the server option to be tested and nothing more. XQSCHK ignores blank lines (up to 4) and any lines of text that follow the line where it finds the options' name.
The option [option name] is not in the Option File.	There is no option in the remote site's OPTION file (#19) that matches the name of the server option that was unloaded from the text of the message. The string it is using to search the OPTION file (#19) is returned in [option name].
Option [option name] is not shown as a server-type option but a [type].	The option is not marked in the remote OPTION file (#19) as a server-type option, but some other kind of option returned in [type], such as a print-type option.
[Option name] is marked as Out Of Order with the message: [message].	The OUT OF ORDER MESSAGE field for that option has been filled in with the text that is returned in [message].
The expected data in ^DIC(19,[option number], 220) is missing.	There is no information for this option in fields 220 through 225. The 220 node of the OPTION file (#19) is missing or blank.
No bulletin associated with this option default XQSERVER is missing from system.	There is no bulletin pointed to by Field #220 of this option in the OPTION file (#19), and the default XQSERVER bulletin has been removed from the system. Server options will *not* run without an associated bulletin, even if it is suppressed.
Option [option name] points to a bulletin not in the bulletin file.	WARNING: there is an invalid pointer in Field #220 of the OPTION file (#19) that points to a nonexistent bulletin. The default bulletin XQSERVER will be used.
Option [option name] points to a mail group not in the Mail Group File.	WARNING: there is an invalid pointer in Field #222 of the OPTION file (#19) indicating a mail group that should receive the bulletin in addition to the mail group pointed to by the BULLETIN file.
There are no mail groups associated with the bulletin [bulletin name].	The bulletin returned in [bulletin name] does not have a mail group associated with it in the BULLETIN file (#3.6).

Error/Warning Message	Description
There is no active user associated with the bulletin [bulletin name].	When following the pointers from the bulletin to the mail group to the NEW PERSON file (#200), an active user was not found. Each server option *must* be linked to a user who has an Access and Verify code and is not terminated.
There is no routine in field 25 of the Option File for this option.	This server option has no routine associated with it in the ROUTINE field of the remote site's OPTION file (#19).
The routine [routine name] is not on the system.	The routine that is named in the ROUTINE field of the OPTION file (#19) is not found on the system. It has been removed or is in another UCI.
There is no server action code for this option.	The required server option action code in field 221 of the OPTION file (#19) is blank.

12 Help Processor

User Interface

Kernel's Help Processor is a utility for displaying help frames. A help frame is a screen of text that explains some part of a software application. Each individual help frame can have keyword links to other help frames. Using these keywords, you can navigate through a series of related help frames to learn more about each help frame topic.

Some places where you may encounter help frames are:

- When requesting help on options in the menu system.
- When requesting help on a menu in the menu system.
- As a standalone option describing some part of a software application.

Figure 12-1. Help frame example

```
              USING THE 'Help Processor' OPTION
The Help processor is a frame-oriented display system which allows
users and programmers to access and manage help text.

The system is driven off of the HELP FRAME FILE.

There are several LINKS which will cause the help text to be
displayed to the user.  The system is interactive, and the user may
select which topic he/she wishes further information on.
The Help Frame Processor Menu contains the following options:


DISPLAY/EDIT      - Displays the text of a help frame and allows for the
                    edit of the name, header, text, or related frames.


CROSS REFERENCE   - Lists all the help frames for a specified package,
                    showing parent help frames, linked to menu option,
                    and invoking routine.

LIST              - Lists the help frames in several different formats.

MORE OPTIONS

Select HELP SYSTEM action or <return>:
```

At the bottom of every displayed help frame is a "Select HELP SYSTEM action..." prompt. You have several choices at this prompt. To back your way out of the help frame system, you can simply press the **<Enter>** key. This backs you up one level, or exits you if you are at the top level of a help frame tree. If you want to exit quickly from help frames, you can enter "**^Q**" to quit immediately without having to back all of the way out.

You can list other choices at the "Select HELP SYSTEM action..." prompt by entering a question mark ("**?**"). The full list of choices is:

Table 12-1. Help system command actions

Response	Action
Keyword	Jump to help frame associated with Keyword.
<Enter>	Quit to previous help frame (exit if no previous).
^Q	Quit the help system.
^R	Refresh the current frame.
^T	Table of related frames.
^O	On/off switch for bracketing/reverse video of keywords.
^H	How you got to this frame.
^E	Edit this frame (only if authorized as editor of frame).

Keywords in a help frame are displayed by the help processor in reverse video. If you enter the first few letters of a keyword and press the **<Enter>** key, the help processor will jump to the help frame linked to the entered keyword.

Help Frames in the Menu System

If a menu option has associated help frames, you can display them by entering a question mark ("**?**") followed by an option's menu text or synonym at a menu prompt (i.e., **?**option). For example:

Figure 12-2. Display a help frame for an option—Entering one question mark (?) and option name

```
Select Office Menu Option: ?MAILMAN
```

Entering three question marks ("**???**") at the menu prompt indicates which options have associated extended help (help frames).

Figure 12-3. Display a help frame for an option—Entering three question marks (???)

```
Select Office Menu Option: ???
```

If a menu itself has an associated help frame, entering four question marks ("**????**") at the menus "Select ... action: " prompt displays the help frame associated with that menu if one exists:

Figure 12-4. Display a help frame for an option—Entering four question marks (????)

```
Select Help Processor Option: ????
```

System Management

Help frames are entries in the HELP FRAME file (#9.2). The Header and Text of help frames can be displayed to users to provide instruction about software or other topics. Help frames can be distributed with software or can be created locally to provide information about local policies and procedures.

The options used to create, edit, and link help frames are on the Help Processor menu [XQHELP-MENU], shown below:

Figure 12-5. Help Processor menu options

```
SYSTEMS MANAGER MENU ...                                          [EVE]
Menu Management ...                                             [XUMAINT]
  Help Processor ...                                          [XQHELP-MENU]
     Display/Edit Help Frames                               [XQHELP-DISPLAY]
     List Help Frames                                          [XQHELP-LIST]
     New/Revised Help Frames                                [XQHELP-UPDATE]
     Cross Reference Help Frames                               [XQHELP-XREF]
     Assign Editors                                         [XQHELP-ASSIGN]
     Unassign Editors                                     [XQHELP-DEASSIGN]
     Fix Help Frame File Pointers                               [XQHELPFIX]
```

Use of the Help Processor options is explained by help frames associated with the options.

Display/Edit Help Frames Option

The help frames can be displayed with the Display/Edit Help Frames option [XQHELP-DISPLAY] You can use the **?option** syntax at the select prompt, as follows:

Figure 12-6. Display/Edit Help Frames option—Displaying help using the ?option syntax

```
Select Help Processor Option: ?DISPLAY <Enter>/Edit Help Frames
```

List Help Frames Option

The List Help Frames option [XQHELP-LIST] can be used to print a series of frames with a table of contents and page numbering to resemble a hard copy manual.

Figure 12-7. List Help Frames option—Sample user dialogue

```
Select Help Processor Option: List Help Frames
Select primary HELP FRAME from which to list: XUDOC NEW
```

New/Revised Help Frames Option

The New/Revised Help Frames option [XQHELP-UPDATE] produces a VA FileMan-generated print of all help frames that have been updated during a specified time period.

Cross Reference Help Frames Option

The Cross Reference Help Frames option [XQHELP-XREF] lists any of the following cross-references to a specified set of help frames:

- Parents (other help frames that call the specified help frame).
- Options (options whose HELP FRAME field references the specified help frame).
- Routines (if a developer has entered the routine in the specified help frame's INVOKED BY ROUTINE field).

Fix Help Frame File Pointers Option (Deleting Help Frames)

There is no Kernel utility to delete help frames, but the menu system does *not* generate errors if a pointed-to help frame is missing. If a site chooses to delete help frames using VA FileMan, they should use the Fix Help Frame File Pointers option [XQHELPFIX] afterwards to delete dangling pointers from the OPTION file's HELP FRAME field.

Assigning/De-assigning Help Frame Editors

An existing help frame can be edited, through the Help Processor options, by the following people:

- The help frame author.
- Any holder of the XUAUTHOR security key.
- Anyone who has been assigned as an editor to that help frame.

To assign an editor to a given help frame use the Assign Editors option [XQHELP-ASSIGN] or to de-assign an editor to a given help frame, use the Unassign Editor option [XQHELP-DEASSIGN].

Disk Space Concerns

Help frames consume disk space. The amount can be considerable if numerous frames are exported with a software application. You can estimate the size of the HELP FRAME file (#9.2) by Kernel's Block Count utility.

Figure 12-8. Estimating the size of the HELP FRAME file (#9.2) using Kernel's Block Count utility

```
Select Systems Manager Menu Option:  Prog <Enter> rammer Options
Select Programmer Options Option:  Global <Enter> Block Count
Block Count for Global ^DIC(9.2)
```

Creating and Editing Help Frames

One way to edit help frames from the HELP FRAME file (#9.2) is to use the Display/Edit Help Frames option to display the help frame in question. Then, at the "Select Help System Action:" prompt, you can enter "**^E**" to edit the help frame if you have edit access to the help frame. You have edit access if:

- You are the help frame's author.
- You are assigned as an editor for the help frame.
- You are a holder of the XUAUTHOR security key.

Another handy way to edit help frames is within the help frame system as invoked from a software application. For example, if the help frames are tied to a software's options, you can use the software, invoke the help frame for each field or option, and then edit that help frame on the spot. To edit a help frame in this manner, enter "**^E**" at the help frame action prompt. To do this, however, you *must* have edit access to the help frame as described above.

Namespacing of Help Frames

Like entries in the OPTION (#19) or SECURITY KEY (#19.1) files, entries in the HELP FRAME file (#9.2) *must* be namespaced to avoid overwriting problems.

Help Frame Layout Considerations

When entering the text of help frames, you should keep each line to fewer than 80 characters for proper screen display.

NOTE: The text is displayed "as it stands" and is not processed by VA FileMan's text formatter. That is, the text is not wrapped, and word-processing "windows" are not evaluated. Frames are usually 22 lines in length although an end-of-page READ is issued to allow a pause if the frame exceeds 22 lines.

If there are only a few lines of text, the Help Processor displays a table at the bottom of the screen of all related frames (those frames that the current frame has keyword links to). The table shows the choices of other frames so the user need not enter the keywords in the text. You can force the table of related frames out of the display by entering enough blank lines so that the frame's length is 20 lines (assuming the display has a page length of 24 lines).

For the Help Processor to identify and highlight keywords, the keywords are entered in the text of the help frame enclosed in square brackets. By convention, keywords in help frames are usually in all capital letters. A square bracket character can be displayed as part of the frame's text by entering two of the characters (e.g., **[[** or **]]**).

If the frames are to be printed using the List Help Frames option, the resulting help manual will have an organized outline, if the frames are linked in a top-down tree structure without any circular connections among the branches.

Linking a Help Frame as Help for an Option or Menu

Once a help frame (or a series of help frames) has been created, you can associate it (them) with options by entering the name of the top-level help frame in the HELP FRAME field of the OPTION file (#19). You can use Menu Manager's option Edit options to do this. That way, when a user enters a single question mark ("**?**") in conjunction with the option name, Menu Manager will invoke the associated help frame.

Figure 12-9. Linking help frames to an option—Sample user dialogue

```
Select Systems Manager Menu Option:  Menu <Enter> Management
Select Menu Management Option:  Edit options
Select OPTION to edit:  XQHELP-MENU <Enter>  Help Processor
NAME: XQHELP-MENU// ^HELP FRAME
HELP FRAME: XQHELP
```

13 Error Processing

User Interface

When an option you are using encounters an error condition, you are usually returned to the menu system. A message is displayed indicating that an error has occurred. You are then presented with the last menu prompt and can continue.

There are certain error conditions, however, that may prohibit or prevent return to the menu system. In these situations, you will be halted off the system.

System Management

The Error Processing menu handles errors for Caché systems. It provides access to options pertaining to the error trap, displaying, printing, and purging errors. Like the error traps provided by the operating systems, the utility allows the investigation of program execution errors or the examination of system errors by capturing a picture of the environment for later reconstruction.

The %ZTER* routines are called from ERR^ZU to trap errors and store them in the ^%ZTER global, a Manager account global that should be translated so that all errors are included on one report. The XTER* routines are used to format the error report.

Error Screens

At times you may not want to trap a certain type of error, but merely to count them because you are already aware of the error and can do nothing to prevent it. At other times you may not even want to count the error because it is inevitable or harmless. An error screen is a string of characters that is compared with the error message of every error trapped. Any trapped error whose message contains the screen is screened out. You decide for each screen whether the error is counted or completely ignored. In either case the error is not recorded in either the Kernel ERROR LOG file (#3.075) or the Taskman Error Log. In TaskMan, if a running task encounters a screened error, the Submanager still notes the error in the record for that task.

Kernel gives you four options with which to manage your error screens:

- List Error Screens [XUTM ERROR SCREEN LIST]
- Add Error Screens [XUTM ERROR SCREEN ADD]
- Edit Error Screens [XUTM ERROR SCREEN EDIT]
- Remove Error Screens [XUTM ERROR SCREEN REMOVE]

NOTE: These error screen options are described in the topics that follow in this chapter, "Error Processing." Previously, these options were described in Chapter 22, "TaskMan: System Management—Operation." Because these error screen options apply to all errors and not just TaskMan-specific errors, they were moved to this chapter. For historical purposes, however, these four option names are still prefixed with "XUTM" and located on the Taskman Error Log menu [XUTM ERROR], located under the Taskman Management Utilities menu [XUTM UTIL], located under the Taskman Management menu [XUTM MGR], which are all located under the Systems Manager Menu [EVE].

List Error Screens

Figure 13-1. List Error Screens option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                                [XUTM UTIL]
      Taskman Error Log ...                                       [XUTM ERROR]
         List Error Screens                            [XUTM ERROR SCREEN LIST]
```

The List Error Screens option [XUTM ERROR SCREEN LIST] lists in a simple table the screens you have established and the number of errors that have been screened out by each.

Add Error Screens Option

Figure 13-2. Add Error Screens option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                                [XUTM UTIL]
      Taskman Error Log ...                                       [XUTM ERROR]
         Add Error Screens                             [XUTM ERROR SCREEN ADD]
```

With the Add Error Screens option [XUTM ERROR SCREEN ADD] you can enter a screen and specify whether the errors should be counted. If there are already similar screens in place (e.g., entering SYN when SYNTAX is already established) you will be so informed, shown the similar screens, and prompted for confirmation before being asked about the count. Entering two question marks ("**??**") at the "Enter Screen To Apply:" prompt displays the list of error screens.

Edit Error Screens Option

Figure 13-3. Edit Error Screens option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                                [XUTM UTIL]
      Taskman Error Log ...                                       [XUTM ERROR]
         Edit Error Screens                            [XUTM ERROR SCREEN EDIT]
```

Use the Edit Error Screens option [XUTM ERROR SCREEN EDIT] if you want to reset the counter on a screen or change your mind about whether or not the screen counts its errors. You *must* type in the exact screen you wish to edit. Again, entering two questions marks displays the list of error screens currently in place.

Remove Error Screens Option

Figure 13-4. Remove Error Screens option

```
SYSTEMS MANAGER MENU ...                                                [EVE]
Taskman Management ...                                               [XUTM MGR]
   Taskman Management Utilities ...                                  [XUTM UTIL]
      Taskman Error Log ...                                         [XUTM ERROR]
         Remove Error Screens                          [XUTM ERROR SCREEN REMOVE]
```

When you type in a screen at the prompt for this option, the screen is removed for you. If there are any similar screens, the Remove Error Screens option [XUTM ERROR SCREEN REMOVE] asks whether you wish to remove them also. Again, entering two question marks ("**??**") displays the list of error screens.

Enhanced Error Processing

Enhanced error processing for Caché sites is supported. Kernel's error trap captures variables in their state at the time errors occur, regardless of how variables may have been NEWed beforehand. Stack levels for the routine call stack are recorded in the error trap in the $STACK variable.

The descriptions of the Error Processing menu option topics that follow are arranged in the same order as the options appear on the Error Processing menu [XUERRS].

Figure 13-5. Error processing options

```
SYSTEMS MANAGER MENU ...                                                [EVE]
  Programmer Options ...                                              [XUPROG]
    Error Processing ...                                              [XUERRS]
      P1 Print 1 occurrence of each error for T-1 (QUEUE) [XUERTRP PRINT T-1 1 ERR]
      P2 Print 2 occurrences of errors on T-1 (QUEUED)    [XUERTRP PRINT T-1 2 ERR]
      Clean Error Trap                                              [XUERTRP CLEAN]
      Error Trap Display                                                [XUERTRAP]
      Interactive Print of Error Messages                     [XUERTRP PRINT ERRS]
```

Print 1 Occurrence of Each Error for T-1 (QUEUE) Option

The Print 1 occurrence of each error for T-1 (QUEUE) option [XUERTRP PRINT T-1 1 ERR] lists the first occurrence of each error recorded on the previous day. T-1 represents "Today-1 = Yesterday". You can queue it to run shortly after midnight. If a device is specified, the output is sent to the specified device. If a device is *not* specified, the output is placed in a mail message and sent to the individual who queued the option to run. It should be set to automatically requeue at a 1 day ("D") interval.

Print 2 Occurrences of Errors on T-1 (QUEUED) Option

The Print 2 occurrences of errors on T-1 (QUEUED) option [XUERTRP PRINT T-1 2 ERR] lists the first two occurrences of each error recorded on the previous day. T-1 represents "Today-1 = Yesterday". It can be queued to run shortly after midnight. If a device is specified, the output is sent to the specified device. If a device is *not* specified, the output is placed in a mail message and sent to the individual who queued the option to run. It should be set to automatically requeue at a 1 day ("D") interval.

Clean Error Trap Option

You can use the Clean Error Trap option [XUERTRP CLEAN] to purge the error log. It is locked with the XUPROGMODE security key. You can use the corresponding direct mode utility, ^XTERPUR, in programmer mode. There is also a queuable version, Error Trap Auto Clean option [XUERTRP AUTO CLEAN].

Purging is a partial clearing of the ERROR LOG file (#3.075) stored in the ^%ZTER(1, global. This global node should not be deleted directly since potentially important recent errors would be purged. Deletion of the entire ^%ZTER global would be a greater mistake since the standard reference data contained in the ERROR MESSAGES file (#3.076) stored in ^%ZTER(2, would be lost.

You are first prompted for the number of days to leave in the error trap. If you enter a number of days to retain errors, all errors older than the specified number of days are immediately purged:

Figure 13-6. Choosing the number of days to leave errors in the error trap

```
To Remove ALL entries except the last N days, simply enter the number N at the
prompt.  OTHERWISE, enter return at the first prompt, and a DATE at the second
prompt.  If no ending date is entered at the third prompt, then only the date
specified will be deleted.  If an ending date is entered that range of dates
INCLUSIVE will be deleted from the error log.

Number of days to leave in error trap: 50

          DONE
```

If you just press **<Enter>** instead of entering a number of days to retain, you are then prompted for a start date and end date between which to remove errors. Errors in the period you specify will then be purged immediately:

Figure 13-7. Choosing a start and end date range to delete errors from the error trap

```
Starting Date to DELETE ERRORS from: 1/1 <Enter> (JAN 01, 2004)
Ending Date to DELETE ERRORS from: 1/31 <Enter> (JAN 31, 2004)
```

The queuable version of this option, Error Trap Auto Clean, can be scheduled to run in the background. By default, it cleans up errors recorded more than 7 days in the past. You can specify a different interval

by placing a numeric value (representing the number of days beyond which to purge) in this option's TASK PARAMETERS field.

Error Trap Display Option

The Error Trap Display option [XUERTRAP] displays errors that have been trapped on the system. The messages for these errors are operating-system dependent. You can use the corresponding direct mode utility, ^XTER, from programmer mode.

The error trap tries to capture a description of the error, the local symbol table, the last global reference, and other signon statistics. For Caché, $ZC calls are used to record IO counts, CPU time, and page faults.

Figure 13-8. Error trap display option—Sample user dialogue

```
In response to the DATE prompt you can enter:
     'S' to specify text to be matched in error or routine name.

Which date? > T-1
1 error logged on 2/9/95
  1)  <ECODETRAP>PRGMODE+5^%ZOSV:2    07:41:52  KDE,KDE  20801D46
     _TNA4523:

No disconnect error

Which error? >  1

Process ID:  2020107A  (538972282)          JAN 18, 1992 17:19:21

Username: EXAMPLE            Process Name: VISTA User

UCI/VOL: [NXT~NXT~ISC6A2~NXT:KDAISC6A2]

$ZA:   0                                $ZB:  \013

Current $IO: _TNA4523:       Current $ZIO: LTA_00129420196A

CPU time:   3.17              Page Faults:       1204

Direct I/O:  81              Buffered I/O:         96

$ZE= <ECODETRAP>PRGMODE+5^%ZOSV:2

 D @XQZ G OUT"

Last Global Ref: ^XUSEC(0,"CUR",24,2950209.074142)

Which symbol? >
```

Errors can be reported by searching for a date range or character string. Question marks show a count of errors for the selected range. Two question marks ("**??**") exclude disconnects and three ("**???**") include disconnects. A string search could be used to find XQ in all routines or an UNDEF in the definition of all errors. Once an error is identified, the report generator shows the job number, username, IO value, date/time, UCI/Volume Set, error type, last global reference, and the line of code that caused the error. It then prompts for a listing of variables, enter "**^L**" to list all or a letter, such as X, to list those starting with

X. The listing can be printed to the screen or to an output device. You can page through the screen listing, one screen at a time, and enter "**^Q**" to quit or enter "^" to exit at the end of each screen.

A restore feature can be invoked by entering "**^R**" provided that the user is working in programmer mode. Programmer mode is required as a protection against restoration of variables from within the menu system. To the extent possible, the environment at the time of the error is restored with the routine and local symbol table intact.

Figure 13-9. Local symbol table help

```
Which symbol? > ?

Enter:
     ^Q to EXIT
     '^' to return to the last question
     Leading character(s) of symbol(s) you wish to examine
     $ to get a display of the $ system variables
     ^L to obtain a list of all symbols
     ^R to restore the symbol table and ... and enter direct mode
```

After reviewing the error log, you are given the opportunity to examine the operating system's error log. Now that most VistA applications record their errors in Kernel's error log, however, there is less need to track VistA errors in the operating system error log.

Figure 13-10. Choosing to examine the operating system's error log—Sample user dialogue

```
Do you want to check the OPERATING SYSTEM ERROR TRAP too? NO//
```

Interactive Print of Error Messages Option

The Interactive Print of Error Messages option [XUERTRP PRINT ERRS] provides for an interactive print of the first "**n**" of occurrences of an error (where "**n**" is user selectable) over a specified date range.

III Device Handler

14 Device Handler: User Interface

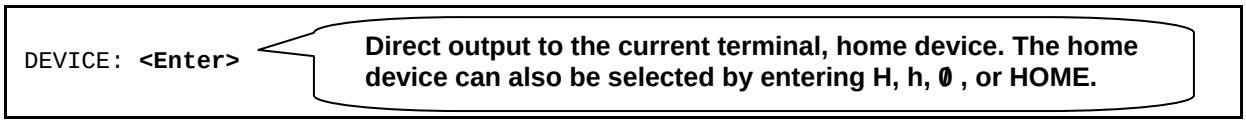
Applications that are designed for the Kernel environment perform output in a consistent manner, using Kernel's Device Handler. This ensures consistency, both for how you are asked to select devices for output, and also for how output is actually performed.

When you respond to the "DEVICE: " prompt, you are using the Device Handler.

Printing to Devices

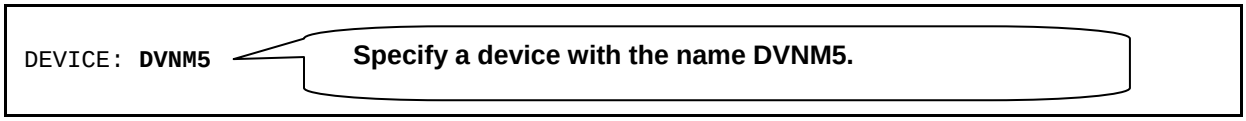
At the "DEVICE: " prompt, to send output to your terminal, you can simply press the **<Enter>** key. This tells the Device Handler to display the report on the home device (that is, on your terminal), as shown below:

Figure 14-1. Choosing the home device



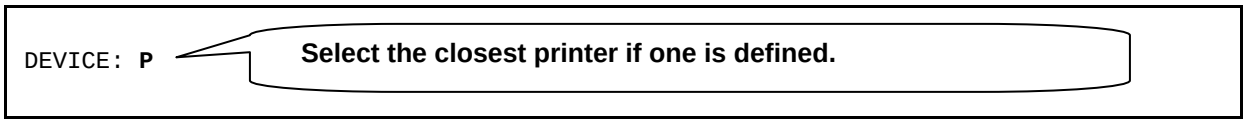
To send output to a printer, enter the name of the printer at the "DEVICE:" prompt, as shown below:

Figure 14-2. Choosing a printer device



To select the closest printer, if one is defined (unlikely), you can simply enter **P** and press **<Enter>**, as shown below:

Figure 14-3. Choosing the closest printer device



You can enter a question mark ("**?**") to display help about the syntax of the response.

Figure 14-4. Device syntax help—One question mark (?)

```
DEVICE: ?
Specify a device with optional parameters in the format
          Device Name;Right Margin;Page Length
                           or
     Device Name;Subtype;Right Margin;Page Length
```

You can enter two question marks ("**??**") to display available printers and other devices connected to the current Volume Set or "reachable from" the current Volume Set. You can also ask for a series of help frames under extended help:

Figure 14-5. Displaying devices help—Two question marks (??)

```
DEVICE: ??
The following information is available:
                    All Printers
                    Printers only on 'ROU'
                    Complete Device Listing
                    Devices only on 'ROU'
                    Extended Help

                Select one (A,P,C,D, or E):
```

You can list all devices. In addition to printers, this list shows other types of devices you can use to handle output. An example of a partial printer listing is shown below:

Figure 14-6. Sample printer listing

```
                Select one (A,P,C,D, or E): P

GENICOM10P 6th Floor 301               GENICOM16P 6th Floor 301
HP LASER DEV-10P                        HP LASER DEV-12P
```

REF: Unusual device types (e.g., Hunt Groups and Resource devices) are discussed in the "Special Device Issues" chapter in this manual.

Specifying Right Margin and Page Length

Ordinarily, when choosing an output device, you only need to specify the device name. There can be times, however, when you may find it useful to specify the right margin and/or the page length for your output. The syntax to specify margin and page length uses semicolon delimiters. The format is:

```
DEVICE: Device Name ; Right Margin ; Page Length
```

The following examples show how to use the additional semicolon-delimited pieces at the "DEVICE: " prompt:

Table 14-1. Sample semicolon-delimited pieces at the "DEVICE: " prompt

Semicolon-delimited Piece	Description
DEVICE: **DVNM5;80;66**	Use the DVNM5 device with a right margin of 80 columns and page length of 66 lines.
DEVICE: **;132**	Use the home device, right margin of 132.
DEVICE: **;;66**	Use the home device and format the output with page breaks at 66 lines.
DEVICE: **;;9999**	Scroll output on the home device without needing to press the **<Enter>** key at page breaks.

Queuing

At the "DEVICE: " prompt, if you enter a device's name, the output goes directly to that device. If the output you're sending is, for example, a long report, this ties your terminal up until the report finishes printing to that device.

You can print output and yet keep your terminal free for other processing by queuing your jobs rather than running them directly. As described in the "TaskMan: User Interface" chapter, you can queue output by entering **Q** at the "Device: " prompt. The device prompt is then presented a second time so that you can specify the output device.

Figure 14-7. Specifying a device and queuing a print job—Sample user dialogue (1 of 2)

```
DEVICE: Q
DEVICE: DVNM5
REQUESTED TIME TO PRINT: NOW// <Enter>
REQUEST QUEUED!
Task number: 856103
```

Alternatively, you can still specify the device first. The Device Handler checks to see if the device is available and, if so, asks you if you want to queue your output. If the device cannot be reached at the current time, Device Handler indicates that the device is busy or unavailable. You can avoid the preliminary availability check by entering **Q** at the first prompt (see Figure 14-7).

Figure 14-8. Specifying a device and queuing a print job—Sample user dialogue (2 of 2)

```
DEVICE: DVNM5
DO YOU WANT YOUR OUTPUT QUEUED? NO// YES

REQUESTED TIME TO PRINT: NOW// T@18:00 <Enter>  (JUL 11, 2004@18:00)
REQUEST QUEUED!
Task number: 856109
```

Whether you request queuing before or after naming a device, Device Handler then asks you to specify a time for the queued job to run. You can accept the default (NOW) or indicate a later time in the usual format. Queuing sends output to TaskMan for scheduling. Meanwhile, you can continue working on the computer system without a delay.

Figure 14-9. Queuing a print job—Sample user dialogue

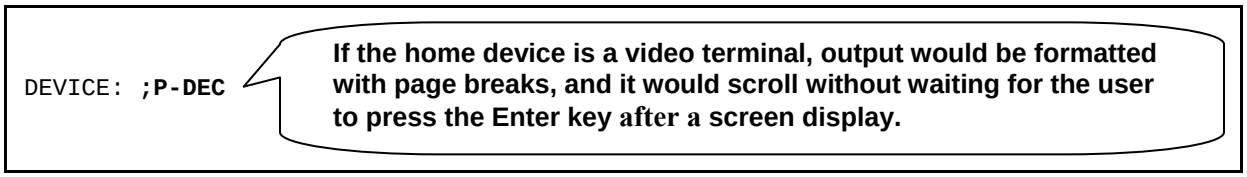
```
REQUESTED TIME TO PRINT: NOW// T@18:00 <Enter>  (JUL 11, 2004@18:00)
REQUEST QUEUED!
Task number: 856109
```

REF: For more information about queuing output, please refer to the "TaskMan: User Interface" chapter in this manual.

Specifying a Special Subtype

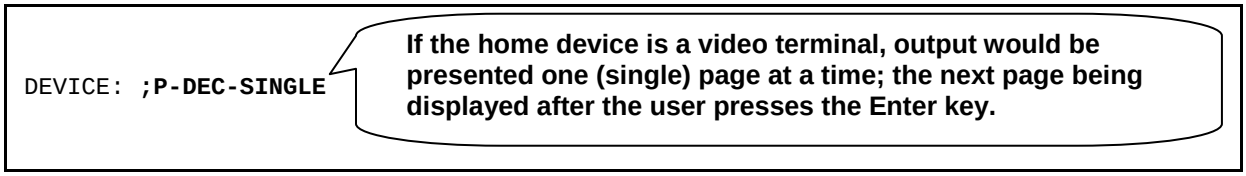
There is an exception to using numbers in the second semicolon piece to indicate a right margin setting. If, instead of a number, you use a letter and then a hyphen in a device specification (e.g., P-DEC), the second semicolon piece specifies a terminal type entry from the TERMINAL TYPE file (#3.2) to use for the output. A terminal type entry specifies information about what commands to use with specific printers (e.g., escape codes).

Figure 14-10. Terminal-type device entry— *Without* pauses



One form of the subtype request made possible by **VA FileMan's print routines** is the use of the word SINGLE along with **P-** or **PK-**. Appending "-SINGLE" indicates that a pause should occur after the display of each page. If using a slaved device to print the screen display, for example, the next page is displayed only after the user has pressed **<Enter>**:

Figure 14-11. Terminal-type device entry— *With* pauses



If you're not sure which subtype to use, you can enter a partial specification of the subtype in the second piece, and the Device Handler will let you choose from all matching subtypes. For example, if a dozen subtypes begin with "P-LASER...", you can list them by entering only the beginning of the subtype name (e.g., P-LASER):

Figure 14-12. Partial Device specification—Unknown subtype

```
DEVICE:  LASER;P-LASER
```

All subtypes beginning with P-LASER are listed; you can then choose a subtype from this list.

When using a subtype as the second semicolon piece of a device specification, you can still specify a right margin and page length to use, but you then do so with the 3rd and 4th semicolon pieces:

Figure 14-13. Device specification—Four semicolon piece: Sample

```
DEVICE:  LASER;P-LASER-NEW;132;100
```

The syntax for the four semicolon piece form of the device specification is:

Figure 14-14. Device specification—Four semicolon piece: Syntax

```
DEVICE:  Device Name ; Subtype ; Right Margin ; Page Length
```

Spool Document Names—An Exception

When you request the spool device at the device prompt, you can use the following formats to specify the spool document name:

Figure 14-15. Device syntax—Specifying a spool document name: Sample formats (1 of 2)

```
DEVICE:  Spooler ; Spool Document Name ; Right Margin ; Page Length
```

Figure 14-16. Device syntax—Specifying a spool document name: Sample formats (2 of 2)

```
DEVICE:  Spooler ; Subtype ; Spool Document Name
```

Although neither right margin nor page length can be specified when including a subtype as the second piece and spool document name as the third, no functionality is lost. The explanation is simple; the spooler only responds to these two terminal type specifications. In other words, identifying a subtype for the spooler does no more than define a margin and page length.

Spool document entries in the SPOOL DOCUMENT file (#3.51) cannot have names beginning with: **P-**, **PK-**, **C-**, etc. (i.e., one or two letters followed by a hyphen). Because this syntax is the required naming convention for subtypes, you are allowed to specify the document name and the subtype in any order.

REF: For more information about Spool Devices, please refer to the "Spooling" chapter in this manual.

Alternate Syntax for Device Specification

An alternate syntax is available for specifying right margin and page length when responding to the device prompt. Using the alternate format, you can specify pitch, intensity, and quality. The success of specifying these additional attributes, however, depends on whether the corresponding fields have been defined by IRM at your site.

The syntax requires the use of a slash ("/") after the last semicolon (see Figure 14-17).

You can use the following codes to specify special device attributes (in any order), without separating punctuation to delimit the pieces:

Table 14-2. Alternate device attribute codes

Code	Description
B	Boldface
L	Page length
M	Margin
P	Pitch
Q	Quality (can be Q, Q1, or Q2)

For example, you could specify:

Figure 14-17. Specifying a device—Using alternate syntax

```
DEVICE:  LASER;P-LASER-LANDSCAPE;/M132L100P16BQ2
```

In this example (Figure 14-17), the margin ("**M**") is set to 132 (i.e., M132), the page length ("**L**") is set to 100 lines (i.e., L100), the pitch ("**P**") is set to 16 (P16), the intensity to boldface ("**B**"), and the quality ("**Q**") set to letter quality (i.e., Q2). An absence of the **B** would indicate normal intensity. The quality settings are: Q, Q1, and Q2.

Your IRM needs to confirm that the appropriate code to set the specified printer attributes is set up for the device that you are using. Then, when the Device Handler closes the device, IRM needs to be sure that appropriate reset code is in the CLOSE EXECUTE field so that the characteristics do *not* stay in effect. If, for example, someone requests a small pitch, subsequent reports will also use the small pitch unless reset in the CLOSE EXECUTE statement for that device (or altered by the OPEN EXECUTE statement of the next device called).

Summary

The Device Handler is a common interface used by all VistA applications to send output to devices (usually, printers). Once you become familiar with the Device Handler, you can enhance your productivity by making use of some of the Device Handler's special features, including queuing, selecting a specific right margin or page length, and selecting a special subtype.

15 Device Handler: System Management

The Device Handler makes use of two primary Files:

- DEVICE File (#3.5)
- TERMINAL TYPE File (#3.2)

Together, these two files control most of the characteristics of devices in Kernel.

DEVICE File (#3.5)

Kernel's DEVICE file (#3.5) stores information about devices on the system. All connected volume Sets/CPUs should make use of a single DEVICE file (#3.5). Then all information concerning a particular device is stored in just one place, which facilitates device management.

Sometimes, a CPU will have an attachment point to which a device can be connected, for example, physical ports. The $I field in the DEVICE file (#3.5) entry identifies this attachment point.

Most devices (e.g., printers) are connected to the network and $I points to the name used by the underlying OS to point to the device. When using such a device, Kernel's Device Handler allows the creation and use of multiple DEVICE file (#3.5) entries for the same physical device. Each DEVICE file (#3.5) entry can contain different specifications (font, margin, page length, etc.) to format output. Each entry in the DEVICE file (#3.5), then, uniquely identifies a set of instructions to send to a particular device on the network.

Each device that Kernel Device Handler needs to communicate with should be set up as an entry in the DEVICE file (#3.5). The DEVICE file (#3.5) supports a variety of devices, including video display terminals (VDTs), commonly called cathode ray tube devices (CRTs); printers; tape drives; and operating system files (e.g., HFS devices).

The DEVICE file (#3.5) is located in the Manager's account for common reference from all associated accounts. With TaskMan's help, this information is also available to all associated processors (CPUs) in the local area network. The global locations of the device-related files are:

Table 15-1. Device-related files global locations

Device-related File Name	Global Location
DEVICE (#3.5)	^%ZIS(1,
TERMINAL TYPE (#3.2)	^%ZIS(2,
DA RETURN CODES (#3.22)	^%ZIS(3.22,

The SUBTYPE field of the DEVICE file (#3.5) points to the TERMINAL TYPE file (#3.2) to retrieve a standard set of characteristics that have been defined for vendor devices (e.g., Laser printers or VT320 CRTs).

DEVICE File Fields
DEVICE file (#3.5)
The most essential fields to populate or consider populating for DEVICE entries are:

- NAME
- $I
- VOLUME SET (CPU)
- TYPE
- SUBTYPE

The following two fields were introduced with Kernel V. 8.0:

- PRE-OPEN EXECUTE
- POST-CLOSE EXECUTE

The DEVICE file (#3.5) has many more fields where additional specific information for particular devices can be entered. Kernel provides a number of options to facilitate creating and editing device types on the Device Management menu [XUTIO]:

Figure 15-1. Device edit options

```
Device Management ...                                                  [XUTIO]
  Device Edit                                                          [XUDEV]
  Edit Devices by Specific Types ...                                [XUDEVEDIT]
    ALL           Edit All Device Fields                          [XUDEVEDITALL]
    CHAN          Network Channel Device Edit                    [XUDEVEDITCHAN]
    HFS           Host File Server Device Edit                    [XUDEVEDITHFS]
    LPD           LPD/VMS Device Edit                             [XUDEVEDITLPD]
    MT            Magtape Device Edit                              [XUDEVEDITMT]
    RES           Resource Device Edit                            [XUDEVEDITRES]
    SPL           Spool Device Edit                               [XUDEVEDITSPL]
    TRM           TRM or VTRM Device Edit                         [XUDEVEDITTRM]
```

VOLUME SET (CPU)

Setting a value in a device's VOLUME SET (CPU) field is optional. If the VOLUME SET (CPU) field is filled in, the device is assumed to be accessible only from the specified CPU. If the field is left blank, the device is assumed to be accessible from any CPU.

NOTE: In the Caché environment, where cluster mounting is used and most devices are set up on all CPUs, all such devices do not need a value for this field.

When the VOLUME SET(CPU) field is blank, the Device Handler still maintains the CPU cross-reference to support queuing and other activities. The cross-reference format involves use of periods as

delimiters. If the VOLUME SET (CPU) value were BBB, the cross-reference for the device with a $I of 75 would be "BBB.75". If the VOLUME SET (CPU) value were NULL, then ".75" would be the CPU cross-reference.

TYPE

There are twelve types of devices. A device's type is set in the DEVICE file's (#3.5) TYPE field. The twelve types are:

Table 15-2. Device types in the TYPE Field in the DEVICE file (#3.5)

Type	Description
TRM	Terminal devices (e.g., most CRTs and printers) should be associated with a corresponding device entry with a type of TRM.
OTH	Other devices that do not fit a particular category should be given a type of OTH.
MT	Magtape devices should have a type of MT.
SDP	(Obsolete) Sequential Disk Processor is a predefined allocated disk space used for sequential processing; use HFS.
SPL	Spool device is a predefined allocated disk space similar to SDP. However, access to the spool device can be achieved from multiple users simultaneously.
BAR	Bar code reader. This type identifies the device as a barcode reader.
HFS	This type and the associated functionality provides the vehicle to READ and WRITE to host level files. Instead of directing reports to a printer, the results could be placed into an OpenVMS or UNIX/Linux file. This would allow a non-M-based statistical software application or spreadsheet to use data produced by the M-based application by simply extracting data from the host file.
VTRM	Virtual Terminal Server devices are those that are associated with a dynamically created M port identification($I). A generic device entry with a device type of VTRM can be established for users who log into the system through Terminal Servers or other network protocols.
HG	Hunt Groups are groups of devices that share a purpose for printing common reports. Printers in the same Hunt Group are usually expected to be in the same general proximity. Reports can be directed to a Hunt Group or to one of its members. If a member is not available, the next available device in the Hunt Group is selected.
RES	Resources is a type used for special sequencing of tasks that do not require a particular device.
CHAN	Network Channels are high speed devices that use network protocols (e.g., TCP/IP).
IMPC	Imaging work station device (reserved for future use for VistA Imaging Project).

This listing of the device type descriptions can also be obtained by entering two question marks ("**??**") at the TYPE field while editing a device.

SUBTYPE

Use this field to select a default terminal type for the device.

REF: For a discussion of the TERMINAL TYPE file (#3.2), please refer to the "TERMINAL TYPE File (#3.2)" topic in this chapter.

SIGN-ON/SYSTEM DEVICE

If set to YES, this field identifies that this entry is the primary device among those device entries that have the same $I with the same VOLUME SET(CPU). Among those device entries that have a common $I and CPU, only one of these entries can have this field set to YES. If none of the common device entries is set to YES, the default device will be identified by the first device on the CPU cross-reference. The default device is used when the Device Handler is invoked with $I as the device to be selected.

QUEUING

You can control the degree of queuing allowed for a device with the QUEUING field. The following settings control queuing for a device:

Table 15-3. Queuing settings

Setting	Queuing	Description
0	ALLOWED	Jobs can be queued or run directly (default).
1	FORCED	Queuing is forced, unless disallowed by application.
2	NOT ALLOWED	Queuing to device is not allowed.

OPEN PARAMETERS and USE PARAMETERS Fields

Magtape, SDP (obsolete), and HFS device types use the value of the OPEN PARAMETERS field as the default if the ASK PARAMETERS flag is set. Users would then be prompted for address/parameters. If the ASK PARAMETERS flag is not set and if there is a value in the OPEN PARAMETER field, this value is used when opening the device (or file). The Device Handler also takes information from the USE PARAMETERS field when opening and using such devices as the tape drive.

Each operating system has its own way of specifying parameters. For example, under Caché, margins are set with both the OPEN and USE command.

PRE-OPEN EXECUTE, POST-CLOSE EXECUTE

These fields can be used to execute a line of M code prior to opening the device and after the device is closed.

NOTE: If you define the variable %ZISQUIT in the PRE-OPEN EXECUTE code, the device open will fail. With this variable, you can use the PRE-OPEN EXECUTE as a screen on whether the device should be opened or not.

Mixed OS Environment

NOTE: This is for Caché only.

Caché can use ECP to run in an application/data server configuration. As of Kernel Patch XU*8.0*440, support was added to allow the Device Handler to work in this mixed operating system (OS) environment. The following fields were added to the KERNEL SYSTEM PARAMETERS file (#8989.3) to provide this support:

- MIXED OS field (#.05)—Makes OpenVMS the primary system and a non-VMS system the secondary system.
- SECONDARY HFS DIRECTORY (#320.2).

Kernel Patch XU*8.0*440 updated the Enter/Edit Kernel Site Parameters option [XUSITEPARM], screen 3, shows these added fields:

15-2. Support added to allow the Device Handler to work in a mixed operating system (OS) environment; updating the MIXED OS (#.05) and SECONDARY HFS DIRECTORY (#320.2) fields

```
------------------------------------------------------------------------
                      Kernel Site Parameter edit
           DOMAIN:NXT.KERNEL.FO-OAKLAND.MED.VA.GOV

    MAX SPOOL LINES PER USER: 99999
MAX SPOOL DOCUMENTS PER USER: 99
MAX SPOOL DOCUMENT LIFE-SPAN: 60


                 MIXED OS: VMS/Linux
DEFAULT DIRECTORY FOR HFS: USER$:[TEMP]
  SECONDARY HFS DIRECTORY: /var/tmp/


DNS IP: 10.9.99.10,10.9.21.999

NEW PERSON IDENTIFIERS:

------------------------------------------------------------------------
```

Kernel Patch XU*8.0*440 also included the addition of the SECONDARY $I field (#52) in the DEVICE file (#3.5). The following example shows an HFS device using the Host File Server Device Edit option [XUDEVEDITHFS] to update Field #52:

15-3. Sample user dialogue updating the SECONDARY $I field (#52)—HFS device

```
------------------------------------------------------------------------
                      EDIT A HOST FILE SERVER DEVICE

   NAME: HFS                                  LOCATION: Host Disk File

     $I: USER$:[TEMP]MIXED.TXT
 Alt $I: /tmp/mixed.txt
SUBTYPE: P-OTHER

       ASK PARAMETERS: YES                  MARGIN WIDTH:
        ASK HOST FILE: YES                   PAGE LENGTH:
ASK HFS I/O OPERATION: NO                VOLUME SET(CPU):

   OPEN PARAMETERS: ("NWS")
  PRE-OPEN EXECUTE:
POST-CLOSE EXECUTE:

           QUEUING: ALLOWED           SUPPRESS FORM FEED: YES
_____________________________________________________________________________
```

The following example shows a NULL device using the TRM or VTRM Device Edit option [XUDEVEDITTRM] to update Field #52:

15-4. Sample user dialogue updating the SECONDARY $I field (#52)—NULL device

```
                    Edit a TRM or VTRM device

NAME: NULL                                      LOCATION: Bit Bucket

     $I: _NLA0:
 Alt $I: /dev/null
   TYPE: TERMINAL
SUBTYPE: P-OTHER

                                  SIGN-ON/SYSTEM DEVICE: NO
                                       VOLUME SET(CPU):

    ASK DEVICE: NO                          MARGIN WIDTH:
ASK PARAMETERS: NO                           PAGE LENGTH:

       QUEUING:                        SUPPRESS FORM FEED:

_____________________________________________________________________________
```

The following example shows a printer set up as an HFS device with the Terminal Type CLOSE EXECUTE, which submits the file to the OS print queue:

15-5. Sample user dialogue

```
                    EDIT A HOST FILE SERVER DEVICE

   NAME: SDD P10                              LOCATION: Printer next to Raul

     $I: USER$:[TEMP]SDD_DN2$PRT.TXT
 Alt $I:
SUBTYPE: P-HP8000 TCP/S

       ASK PARAMETERS: NO                   MARGIN WIDTH:
        ASK HOST FILE: NO                    PAGE LENGTH:
ASK HFS I/O OPERATION: NO                 VOLUME SET(CPU):

   OPEN PARAMETERS: "NWS"
  CLOSE PARAMETERS:
  PRE-OPEN EXECUTE:
POST-CLOSE EXECUTE:
           QUEUING:                    SUPPRESS FORM FEED: YES
```

OpenVMS-Specific DEVICE Fields

The DEVICE file (#3.5) can store operating system-specific information. Several fields are included, for example, in the DEVICE file (#3.5) to configure terminals and ports on Terminal Servers as part of an OpenVMS start-up command file. These are:

- LAT SERVER NODE (#61)
- LAT SERVER PORT (#62)
- VMS DEVICE TYPE (#63)
- LAT PORT SPEED (#64)
- PRINT SERVER NAME OR ADDRESS (#65)
- TELNET PORT (#66)
- REMOTE PRINTER NAME (#67)

Kernel Toolkit software distributes the XTLATSET and NVSTNSET routines that makes use of these fields.

NOTE: These fields are used by VMS and *not* Caché.

Device Security

To regulate who can use a particular device, you can use the PASSWORD and SECURITY fields.

The SECURITY field, if populated, should contain a string of characters to compare with a user's FILE MANAGER ACCESS CODE field (#3), DUZ(0), when the device is selected. Access is denied to anyone whose DUZ(0) does not contain one of the specified characters. As with other uses of DUZ(0), the at-sign ("**@**"; programmer access) will override this restriction.

The PASSWORD field, if populated, forces all users trying to log on to the device to be prompted for the matching password, before entering their Access code.

TERMINAL TYPE File (#3.2)

The TERMINAL TYPE file (#3.2) holds device vendor-specific code to characterize a terminal type. For example, escape sequences can be entered in the OPEN EXECUTE and CLOSE EXECUTE fields to set pitch or font. Every device in the DEVICE file (#3.5) *must* be assigned a terminal type, in the SUBTYPE field.

The most common fields to populate for TERMINAL TYPE file (#3.2) entries are:

- NAME
- SELECTABLE AT SIGN-ON
- RIGHT MARGIN
- FORM FEED
- PAGE LENGTH
- BACK SPACE
- OPEN EXECUTE
- CLOSE EXECUTE

The TERMINAL TYPE file (#3.2) has many more fields where additional specific information for particular terminal types can be entered. Kernel provides the following options to facilitate creating and editing terminal types:

Figure 15-6. Terminal type edit options

```
Device Management ...                                              [XUTIO]
   Terminal Type Edit                                             [XUTERM]
   Change Device's Terminal Type                                [XUCHANGE]
   List Terminal Types                                            [XULIST]
```

Terminal Type Naming Conventions

The convention for naming terminal types is as follows:

Table 15-4. Terminal type naming conventions

Terminal Type	Description
C-	Video terminals (e.g., C-VT320).
PK-	Printers with keyboards.
P-	Printers without keyboards (e.g., P-LASER).
M-	Modems.

The general format is limited to two alphabetic characters, a hyphen, and alphanumeric characters.

As mentioned previously, a spool document name cannot use this format; this is so that it can be distinguished from a device subtype in a call to the Device Handler. Confusion could arise since either can be used as the second piece of the device specification. The SPOOL DOCUMENT file (#3.51) has an input transform pattern match that guards against creation of document names in the format of device subtypes.

How Shared Device and Terminal Type Attributes are Used

The DEVICE (#3.5) and TERMINAL TYPE (#3.2) files share attribute fields for RIGHT MARGIN and PAGE LENGTH. If a value is entered for RIGHT MARGIN or PAGE LENGTH in the DEVICE file (#3.5), it will override the value from the TERMINAL TYPE file [#3.2].

When a user selects a device by responding to the device prompt with only the first required piece of information, the device identification, Device Handler retrieves parameters to characterize the device (e.g., RIGHT MARGIN) from the DEVICE file (#3.5). Furthermore, the Device Handler checks the ASK PARAMETERS flag for the selected device and, if the flag is set, prompts the user for associated parameters, presenting DEVICE file (#3.5) characteristics as the default. For terminals and virtual terminals (types TRM and VTRM, respectively), the user is prompted for the right margin. For magtape (MT), Sequential Disk Processor (SDP; obsolete), and Host File Server (HFS) devices, they can be prompted for address/parameters with the value of the OPEN PARAMETERS field (in the DEVICE file [#3.5]) as the default.

REF: For more information on MT devices, please refer to the "Special Device Issues" chapter in this manual.

For more information on HFS devices, please refer to the "Host Files" chapter in this manual.

Terminal Type Information Retained by User

User can change some terminal type attributes of their signon device by doing either of the following:

1. Changing the terminal type during the session with Edit User Characteristics.
2. Selecting a device for direct output.

Kernel uses the ^XUTL global to hold information about changes made to device characteristics of the home device during a session.

REF: For more information the ^XUTL global, please refer to the "Menu Manager: System Management" chapter in this manual.

The terminal type established for users at each signon is stored according in their NEW PERSON file (#200) entries so that, if necessary, it can be used as a default for the next signon.

Devices and Signon

Device Selection at Signon, and Virtual Terminal Devices

Every interactive user *must* be associated with a device by the Device Handler when they sign on to the VistA system. The association of device is done by matching the incoming user's $I value with the $I value of an entry in the DEVICE file (#3.5). Historically, it was practical to set up one device entry with a matching $I for each physical port.

With the move to OpenVMS, however, the $I of the user was dynamic, with many thousands of $I values possible. The Virtual Terminal device type was created as a way to have one device entry be used for signon for multiple incoming $I values. The Device Handler still checks to see if it can assign a device to an incoming process based on an exact match of $I values. If there is no direct match, however, Device Handler checks to see if the *first part* of the user's $I value matches the $I value of a virtual device entry. This way, a virtual device with a $I value of "_TNA" can service all incoming processes whose $I values *start* with the string "TNA".

Virtual devices do not need a value in the VOLUME SET (CPU) field; they should have SIGN-ON/SYSTEM DEVICE field set to YES, however, to speed up the signon device selection process.

Common device prefixes on VMS systems that could be used for virtual terminal device entries include:

- "TNA"—Telnet devices
- "RTA"—Remote processes using the "SET HOST" command
- "FTA"—Secure Shell devices

Processes on VMS systems that use Telnet usually have $I values beginning with the prefix "TNA", concatenated with an integer value and a colon (e.g., "TNA8456:"). A single virtual terminal device entry whose $I value is "TNA" will service all such processes.

Terminal Type Selection at Signon

Besides needing a device assigned at signon, users also need a terminal type. As described in the "Signon/Security: System Management" chapter in this manual, Kernel can usually determine the correct subtype without needing to prompt the user, by querying the terminal, and matching the returned string (if any) with return codes for terminals stored in the DA RETURN CODES file (#3.22).

If the user is prompted to enter a terminal type, they will need to choose one. The list of terminal types from which they can choose is screened by the SELECTABLE AT SIGN-ON field in the TERMINAL TYPE file (#3.2). Users can only choose from entries with this field set to YES. This stops users from choosing inappropriate terminal types. The setting of this field does *not* prevent terminal types from being chosen by the DA return code method, however. Make sure that all terminal types appropriate for signon have SELECTABLE AT SIGN-ON set to YES.

If the Signon/Security system cannot supply even a default, the Device Handler makes a selection according to the signon device's subtype.

Managing Display Attributes (DA) Return Codes

Figure 15-7. DA Return Code Edit option

```
Device Management ...                                                [XUTIO]
   DA Return Code Edit                                            [XU DA EDIT]
```

The DA RETURN CODES file (#3.22), stores entries for the codes returned by different terminals after Kernel asks for their display attributes at signon. This file then maps Kernel terminal types to terminal's return codes. This mapping allows sites to set up mappings for new terminals or to map different terminals to a common type. For example, a site could map all codes returned by all DEC VT type terminals to a single C-VT102 type terminal type.

The DA RETURN CODES file (#3.22) is a small static file managed by the DA Return Code Edit option [XU DA EDIT]. You can use the DA Return Code Edit option to automate the population of the DA RETURN CODES file (#3.22). When you select this option, the terminal you are using is queried and you are shown the terminal's DA code response. You are then asked for the terminal type and description for this return code. Enter the terminal type name for the terminal you are using. The option updates the DA RETURN CODES file (#3.22), and all terminals responding with this code will be recognized at signon. You can quickly populate the DA RETURN CODES file (#3.22) by using this option from several different types of terminals.

Kernel pre-populates the DA RETURN CODES file (#3.22) with a set of standard Terminal Type entries. You may need to add more entries as needed to handle all terminals at your site.

Troubleshooting

Figure 15-8. Device management—Troubleshooting options

```
SYSTEM MANAGER MENU                                                    [EVE]
  Device Management...                                                [XUTIO]
    Loopback Test of Device Port                                  [XUTLOOPBACK]
    Send Test Pattern to Terminal                                     [XUTTEST]
    Out of Service Set/Clear                                          [XUOUT]
```

Kernel provides several options on the Device Management menu [XUTIO] to aid with troubleshooting device problems, which are described in the topics that follow.

Loopback Test of Device Port Option

Use the Loopback Test of Device Port option [XUTLOOPBACK] to test an RS-232 serial data line when using a loopback connection on the line. First, disconnect the data line from the device it is attached to (if any). Then, tie pins 2 and 3 of the RS-232 serial data line together. This is a loopback connection; data sent down pin 2 (transmit) will loop back up pin 3 (receive). The Loopback Test of Device Port option sends the letters of the alphabet down the data line one at a time, and attempts to READ them back. If both lines are intact, you should see "ABCDEFGHIJKLMNOPQRSTUVWXYZ" print on the terminal from which you are testing the data line.

Send Test Pattern to Terminal Option

Use the Send Test Pattern to Terminal option [XUTTEST] to send a simple test pattern to a device. This is an easy way to verify whether a device is connected to the system. It lets you choose how many lines of the test pattern to send, and then sends that number of lines to the device. You can confirm on the device end exactly how many lines of the test pattern you receive, which can be useful when troubleshooting printer handshaking problems.

Out of Service Set/Clear Option

You can use the Out of Service Set/Clear option [XUOUT] to set a device out of order. It asks you the date on which to put the device out of order. From that date forward, the Device Handler will not allow any jobs to use the device (users will get a message that the device is out of order). To clear the out of order status, use this option again and delete the out of order date.

Device Identification and Cross-references

Devices can be selected in several ways from the "DEVICE:" prompt. Besides the NAME field (#.01), three other attributes, MNEMONIC, LOCAL SYNONYM and $I can also be used to select devices. When LOCAL SYNONYM is used, the Device Handler searches the local CPU for a match. Thus, the same LOCAL SYNONYM value (e.g., PRINTER) can be used to identify several devices, one per CPU.

When editing devices through VA FileMan, two additional fields can be used for lookup: VOLUME SET (CPU) and SIGN-ON/SYSTEM DEVICE. You can separate these values with a period delimiter, as follows:

Table 15-5. Sample period-delimited pieces used for device lookup

Period-delimited Piece	Description
CPU	All devices matching CPU.
CPU.$I	All devices matching the CPU and $I.
SYS	All SIGN-ON DEVICES.
SYS.CPU	All SIGN-ON DEVICES matching CPU.
SYS.$I	All SIGN-ON DEVICES matching $I.
SYS.CPU.$I	All SIGN-ON devices matching CPU and $I.

For example, to display all signon devices on CPU "BBB", you could do:

Figure 15-9. Displaying signon devices on a specific CPU—Sample user dialogue

```
Select DEVICE NAME: SYS.BBB
```

To display all signon devices whose $I begins with "_TNA" you could do:

Figure 15-10. Displaying signon devices with a specific $I—Sample user dialogue

```
Select DEVICE NAME: SYS._TNA
```

The following global listing shows the cross-references for a device with a $I value of 99 and an internal entry number (IEN) of 251. It is a Sign-On/System Device and has a VOLUME SET (CPU) value of AAA.

Figure 15-11. Global listing for device cross-references—$I value = 99 and IEN = 251

```
^%ZIS(1,"G","SYS.AAA.99",251) = ""
^%ZIS(1,"CPU","AAA.99",251) = ""
^%ZIS(1,"C","99",251) = ""
```

If this device is a virtual terminal with a $I of _TNA and established as a Sign-On/System Device but not given a VOLUME SET (CPU) value, the cross-reference structure would be as follows:

Figure 15-12. Global listing for virtual terminal device cross-references—$I value = _TNA and IEN = 251

```
^%ZIS(1,"G","SYS.._TNA",251) = ""
^%ZIS(1,"CPU","._TNA",251) = ""
^%ZIS(1,"C","_TNA",251) = ""
```

16 Host Files

User Interface

Host File Server (HFS) devices allow you to send output to a file maintained by your computer's operating system, rather than to a printer. You can send your output to an HFS device, if such a device type has been established on the system. Depending upon how IRM defines the HFS device, you may be prompted for a host file name and for an input/output operation:

Figure 16-1. Choosing a Host File Server (HFS) device—Sample user dialogue

```
DEVICE: HFS <Enter> DISK FILE
HOST FILE NAME: TMP.TMP// <Enter>            INPUT/OUTPUT OPERATION: ?
Enter one of the following host file input/output operation:
                R = READONLY
                N = NEWVERSION
               RW = READ/WRITE
```

Not all input/output modes are available on all systems. The possible modes for input/output operation work as follows:

Table 16-1. HFS input/output modes of operation

Input/Output Mode	Description
APPEND	Data from a WRITE operation will be appended to the file.
MIXED	Both READs and WRITEs are allowed for the specified file.
NEWVERSION	A new file will be created with a higher version number; this file can be used for WRITEs only.
READ	READs are allowed from the specified file; WRITEs are not allowed.
READONLY	READs are allowed from the specified file; WRITEs are not allowed.
READ/WRITE	Both READs and WRITEs are allowed for the specified file; if a WRITE operation is performed, output is appended to the file.
WRITE	WRITEs are allowed; output can be sent to the specified file.

System Management

To provide access to host files through the Device Handler, set up device entries of type HFS.

There are three fields in an HFS device entry that act as flags for what a user *must* enter when they use an HFS device. The fields are:

- **ASK PARAMETERS—**If this field is set to YES, the user *must* enter the correct M open parameters to open the device. This should be set it to NO if the device is accessible to non-IRM users. If it is set to YES, the default value is the current value of the OPEN PARAMETERS field.
- **ASK HFS I/O OPERATION—**If this field is set to YES, the user can choose what mode the file should be opened in (for example, READ or WRITE). If it is set to NO, files are opened in WRITE mode. This should be set to NO if the device is accessible to non-IRM users, assuming that all such users would only need to WRITE host files.
- **ASK HOST FILE—**When this field is set to YES, the user can choose what file will be opened. If it is set to NO, the default file name built into the device entry is always used. This should be set to NO in most cases if the HFS device is accessible to non-IRM users, since host files can proliferate if many users are able to create many files, and also because an HFS device opens up access to the host operating system (and the potential for overwriting vital files).

Host File Server Device Edit Option

Figure 16-2. Host File Server Device Edit option

```
Device Management...                                                [XUTIO]
  Edit Devices by Specific Types...                              [XUDEVEDIT]
     Host File Server Device Edit                              [XUDEVEDITHFS]
```

The Host File Server Device Edit option [XUDEVEDITHFS] lets you to edit Host File Server device attributes using a ScreenMan form.

Caché and GT.M HFS Device Setup

Caché and GT.M require the name of the host file to be part of the device $I and not part of the parameter list.

Table 16-2. HFS I/O operation modes for Caché and GT.M

I/O Operation Mode	Description
NEWVERSION	A new file will be created (on VMS with a higher version number); this file can be used for WRITEs only.
READONLY	READs are allowed from the specified file; WRITEs are not allowed.
READ/WRITE	Both READs and WRITEs are allowed for the specified file; if a WRITE operation is performed, output is appended to the file.

Figure 16-3. Host File Server device for Caché and GT.M—Sample settings

```
                      Name:    HFS
                        $I:    TMP.TMP
                      Type:    HFS
           Ask Parameters:     NO
            Ask Host File:     NO
Ask HFS I/O Operation:         NO
         Open Parameters:      ("NWS")
```

17 Spooling

User Interface

Spooling privileges can be granted by IRM to users who prepare and manage reports. By sending your output to the spooler, rather than to a printer, you can benefit in several ways. Since spooling saves the output online in a holding area, you can easily print multiple copies of the report at a later time. Spooling is also a good way to store the results of a time-consuming calculation (e.g., a complex VA FileMan report). By queuing to the spooler, a report involving intensive processing can be done at night when the system is relatively free. Output can then be printed during the day when the printer can be attended. Finally, when using the spooler, report processing can run to completion without printer problems interfering.

Sending Output to the Spooler

If you have been given the authority to spool, you can send output to the spooler by responding to the "DEVICE:" prompt with the name of the spool device. Devices used for spooling are commonly named SPOOL or SPOOLER.

If you do not have spooling privileges and you try to use the spool device, the spooler issues a message that authority has not been granted, as below:

Figure 17-1. Unable to send output to a spool device—Sample message

```
DEVICE: SPOOL

     You aren't an authorized SPOOLER user.
```

To send output to the spooler with a customized right margin of 96 and page length of 66, you can use the following syntax:

Figure 17-2. Specifying spooled output margin and length

```
DEVICE: SPOOL;96;66
```

After requesting the spool device, you are usually prompted for a spool document name, as shown below (Figure 17-3). The prompt is not issued, however, if the spool device has been set up to generate the spool document name itself.

Figure 17-3. Spool document name prompt

```
DEVICE: SPOOL

Select SPOOL DOCUMENT NAME:
```

To skip the "Select SPOOL DOCUMENT NAME:" prompt, you can specify the spool document name at the "DEVICE:" prompt by entering the name in the second semicolon piece. A name entered here is *not* used if the spooler is set up to generate names itself, however. Because of the format used, the Device Handler knows that a spool document name, rather than a device subtype, is being specified. Subtypes begin with one or two letters followed by a dash (e.g., P-DEC), while spool document names cannot.

Figure 17-4. Specifying the spool device and document name

```
DEVICE: SPOOL;MYDOC

DEVICE: SPOOL;P-OTHER80;MYDOC
```

If the computing environment is composed of several networked processors, you may need to specify where spooling should take place. The spooler on the current CPU should be chosen unless the output is queued.

Figure 17-5. Spooling output to a spool device on the same CPU

```
DEVICE: SPOOL
  1   SPOOL AAA
  2   SPOOL BBB
Choose 1-2>
```

If the output is queued, you can choose a spooler on another CPU and a time to schedule the job to run.

Figure 17-6. Queuing output to a spool device

```
DEVICE: Q
DEVICE: SPOOL BBB
```

Figure 17-7. Spooler Parameters at the Device prompt (summary)

```
DEVICE:  Spooler

DEVICE:  Spooler;Right Margin;Page Length

DEVICE:  Spooler;Subtype

DEVICE:  Spooler;Spool Document Name

DEVICE:  Spooler;Subtype;Spool Document Name
```

Retrieving Spooled Documents

After a spool document has been created, you can retrieve the output by using options on the Spooler Menu. This menu is distributed as part of Kernel's Common menu, a menu available to all users. Specifically, the Spooler Menu is in your User's Toolbox menu (to quickly reach the Toolbox, or any other option on the Common menu, you can enter a quotation mark plus the menu text or synonym):

Figure 17-8. Spooler Menu options

```
Select Primary Menu Option: "TBOX

Select User's Toolbox Option: Spooler Menu

Select Spooler Menu Option: ?

          Allow other users access to spool documents
          Browse a Spool Document
          Delete A Spool Document
          List Spool Documents
          Make spool document into a mail message
          Print A Spool Document
```

The List Spool Documents option lists any documents that you have created. Other users cannot read or print these documents unless you have authorized them to with the Allow other users access to spool documents option, also on the Spooler menu.

To delete spool documents, you can use the Delete A Spool Document option. Since there is a limit on the amount of spool space that any one user can consume, you may need to delete old spool documents to free up space for new ones. If you attempt to create a new document when the space limits have been exceeded, the spooler issues a message about the need to delete some documents.

Old documents are deleted automatically, on a schedule as determined by IRM. The "life span" of a spool document is a site parameter that IRM controls. IRM should inform you of the life span of spooled documents so that you are not surprised when old documents are purged.

Browsing a Spool Document

With the Browse a Spool Document option, you can view spool documents with VA FileMan's Browser. The Browser allows you to view spool documents on your terminal screen, letting you scroll backward and forward through the report, and also letting you perform simple searches within the report. In the past, when you sent a report to the terminal screen, you could only go in one direction, forward, by pressing **<Enter>** after each screen of text.

REF: For more information on using the Browser, please refer to the *VA FileMan Getting Started Manual*.

Printing Spool Documents

You can print spool documents with the Print A Spool Document option. Before selecting an output device, you are prompted for the number of copies to print. If you have been granted the ability to print to multiple devices, you can send your output to several devices for simultaneous printing. If this privilege has been granted to you, the device prompt is displayed again after you choose the first printer. Entering a NULL response to the second device prompt tells the spooler not to use any more additional printers.

To save users the time and trouble of despooling their documents, IRM can set up a spool device for auto-despooling. If you invoke such a spool device, the spool document is sent to one or more printers when the spooling process has completed. After automatic printing, the spool document remains available for reprinting as necessary (it is not automatically deleted upon despooling).

Making Spool Documents into Mail Messages

You can also be granted the ability to make spool documents into mail messages. If so, the corresponding option on the Spooler Menu will be available. You can use it to make documents into regular mail messages that can then be edited, copied, or forwarded just like other messages. After the text has been moved into a mail message, the spool document is deleted. The deletion is to allow space for new spool documents.

If you plan to make a document into a message, you should do the original output to the spool device with an appropriate margin and page length for a MailMan message. Since MailMan breaks incoming text lines at about the 75th character, a right margin of 75 may be desirable. Indicating that page breaks should not be inserted during the spooling process may also be desirable. Otherwise, the VA FileMan window command |TOP| is inserted into the text at the beginning of each page. While this automatic formatting is an advantage when printing spool documents, it is a disadvantage when creating a mail message. Page breaks will not be inserted when indicating a page length of 99999 lines, or a number greater than the document's total. So when you know your spool document will end up as a MailMan message, a suitable margin and page length request might be:

Figure 17-9. Formatting/Sending a document to a spool device to print as a MailMan message—Sample user dialogue

```
DEVICE: SPOOL;75;99999
```

To turn the spool document into a MailMan message, once your spool document completes, go to the Spooler Menu and select the appropriate option, as illustrated below:

Figure 17-10. Make spool document into a mail message option

```
Select Primary Menu Option: ^Spooler Menu

Select Spooler Menu Option: Make spool document into a mail message
```

If the number of lines in the document exceeds 500, you are asked whether the transfer process should be queued. This prompt is provided for the your convenience since queuing of a time-consuming process is usually preferred. After using the option, you can find your messages by reviewing recently delivered mail in your IN basket.

System Management

Spool Document Storage

Spool document identification is stored in the SPOOL DOCUMENT file (#3.51) in the ^XMB global. This file is for internal use by Kernel's spooler and should not be directly manipulated by IRM. It holds identifying information, such as the name of the spool document and the line count totals. The document's text is stored in another file, the SPOOL DATA file (#3.519), in the ^XMBS global. If the spool document is made into a mail message, the text is moved into the MESSAGE file (#3.9), the ^XMB global, and the corresponding entry in the SPOOL DOCUMENT file (#3.51) is deleted.

When initially creating a spooled document, output is sent to the operating system's spooling area (as defined in the spool device). Kernel's spooler moves the output into ^XMBS when the operating system's spooling process is complete. The status of the document (a field in the SPOOL DOCUMENT file [#3.51]) is then changed from Active to Ready and the document can be accessed by the user. Thus, except during spooling, the operating system's spool area should be empty.

Overflowing Spool Document Storage

When the output is moved from the operating system's spool area into the ^XMBS global, the lines are counted. If, during the count, the user's maximum line limit is reached, the transfer process is halted and a notification message is appended to the transferred text. The entry in the SPOOL DOCUMENT file (#3.51) is also marked as incomplete. Thus, the ^XMBS global is protected from growth expansion that could overflow the disk storage area.

The Kernel spooler cannot, however, count the lines of output as they are sent to the operating system's spool area. If the user's line limit is not exceeded before initiating the report, Kernel permits sending of an unlimited amount of output to the operating system's spooler. This should be considered by IRM when granting spooling privileges. Users who are allowed to spool should be trained accordingly.

Users need to anticipate the results of a process they send to the spooler. If they are not sure what to expect, they should be instructed to test the process by sending it directly to an output device. If unexpected results such as an endless loop or meaningless sort should occur, they can interrupt and cancel the process. Users should also be advised about appropriate use of processing time. Methods of efficient VA FileMan searching and sorting should be used when invoking the spooler (just as when printing directly). For example, as described in the VA FileMan documentation, the first sort-by field should be a cross-referenced field when possible and search criteria should be specified with the most likely conditions first.

Granting Spooling Privileges

Options on the Spool Management menu can be used to grant spooling privileges to users.

Figure 17-11. Edit User's Spooler Access option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Spool Management ...                                              [XU-SPL-MGR]
  Edit User's Spooler Access                                     [XU-SPL-USER]
```

The ability to invoke the spooler at the device prompt is controlled by one flag. Another flag can enable the use of more than one device when despooling. A third flag permits the conversion of a spool document into a mail message. These three flags are user-specific and are stored in the NEW PERSON file (#200). As mentioned earlier, the user-oriented spooler options are distributed as part of the Common menu, a menu available to all users. If IRM has chosen to lock the Spooler Menu or remove it from the Common menu, access to the options will need to be re-established for users who are allowed to spool via the Edit User's Spooler Access option [XU-SPL-USER], as shown below:

Figure 17-12. Edit User's Spooler Access—Sample user dialogue

```
Select Spool Management Option: Edit User's Spooler Access

Select NEW PERSON NAME: KRNUSER,SIX
ALLOWED TO USE SPOOLER: YES// <Enter>
MULTI-DEVICE DESPOOLING: YES// <Enter>
CAN MAKE INTO A MAIL MESSAGE: YES// <Enter>
```

Managing Spool Documents

The remaining options on the Spool Management menu [XU-SPL-MGR] are also found on the user-oriented Spooler Menu. They are provided on the Spool Management menu simply for convenience to IRM to access any spool document on the system. The only provision for access to all spool documents is holding the XUMGR security key. Together, these options along with the XUMGR security key permit IRM to view, print, or delete anyone's documents.

Figure 17-13. Spool Management menu options

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Spool Management ...                                              [XU-SPL-MGR]
  Delete A Spool Document                                      [XU-SPL-DELETE]
  List Spool Documents                                            [XU-SPL-LIST]
  Print A Spool Document                                         [XU-SPL-PRINT]
```

Spooler Site Parameters Edit Option

Figure 17-14. Spooler Site Parameters option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
   Spool Management ...                                            [XU-SPL-MGR]
      Spooler Site Parameters Edit                                 [XU-SPL-SITE]
```

The Spool Management menu also has the Spooler Site Parameters Edit option [XU-SPL-SITE] for setting the spooler site parameters (system-wide defaults for the spooler). The initial settings are defined when installing Kernel but can be edited afterwards.

The spooler site parameters control the total number of documents a user can create and the total number of lines for all documents. When the limits are reached, the user cannot create new documents.

The effects of the three spooler site parameter fields are as follows:

Table 17-1. Spooler site parameter fields

Spooler Site Parameter Field	Description
MAX SPOOL LINES PER USER	This field holds the MAX number of lines of spooled output a user is allowed. If the user has more than this number, then they will not be permitted to spool any more until some of their spool documents are deleted. This only controls allowing the creation of new spool documents and does not terminate a job that is running that has gone over the limit. Recommended value 9999.
MAX SPOOL DOCUMENTS PER USER	This field limits the number of spool documents that any user can have on the system. Recommended value 10-100.
MAX SPOOL DOCUMENT LIFE-SPAN	This field controls the number of days that a spooled document will be allowed to remain in the spooler before deletion by the XU-SPL-PURGE option that needs to be setup to run in the background.

Purge old Spool documents Option

Figure 17-15. Purge old spool documents option

```
PARENT OF QUEUABLE OPTIONS                          [ZTMQUEUABLE OPTIONS]
  Purge old spool documents                                 [XU-SPL-PURGE]
```

A spool document is automatically deleted when its life span (in days) is reached. The purge is carried out by the Purge old spool documents option [XU-SPL-PURGE]. This option is listed on the PARENT OF

QUEUABLE OPTIONS menu [ZTMQUEUABLE OPTIONS] along with others that should not be invoked interactively but should be scheduled to run through TaskMan.

Defining Spool Device Types

The DEVICE file (#3.5) entries for spooler device types make use of information about the underlying operating system's spooling mechanism. Examples for several operating systems are provided below.

Caché and GT.M

Caché and GT.M use an OpenVMS directory for spooling. As indicated in the VistA Cookbook for VAX sites, the directory should be established with full privileges for System, Owner, Group, and World. The directory specifications are used as the $I value.

Figure 17-16. Spool Device for Caché and GT.M

```
                Name:   SPOOL
                  $I:   VA1$:[SPOOLER]
                Type:   SPOOL
             Subtype:   P-OTHER
```

Spool Device Edit Option

The Spool Device Edit option [XUDEVEDITSPL] lets you edit Spool device attributes, using a ScreenMan form.

Figure 17-17. Spool Device Edit option

```
Device Management...                                                [XUTIO]
  Edit Devices by Specific Types...                               [XUDEVEDIT]
     Spool Device Edit                                          [XUDEVEDITSPL]
```

NOTE: The type of data entered in the $I and OPEN PARAMETERS fields depends on the type of M system you are using and the mode of access.

REF: For further details, please refer to your M system manuals.

REF: Examples are provided in the "Defining Spool Device Types" topic.

Auto-despooling

For convenience, spool devices can be defined to ensure that despooling takes place automatically, without user interaction. If the AUTO DESPOOL flag is set, one copy of the spooled output is sent to each device named in the DESPOOL DEVICES field (Multiple). Having the output automatically

despooled saves users the time and trouble of logging on and printing a spool document that may have been created the previous evening. Documents are not deleted upon despooling; they remain available to the user for subsequent printing.

Figure 17-18. Device Edit option—Sample user dialogue

```
Select Device Handler Option: Device Edit

Select DEVICE NAME: SPOOL
NAME: SPOOL// ^AUTO DESPOOL
AUTO DESPOOL: 1 <Enter> YES
Select DESPOOL DEVICES:
```

Generating Spool Document Names

Spool devices can be set up to generate the name that will identify the spool document. If this flag is set in the DEVICE file (#3.5), users of that device will not be prompted to enter the spool document name. Also, if the flag is set, any user- or developer-defined name [in IO("DOC")] is ignored. The generated name consists of the first 15 characters of the spool device's name, an underscore ("_"), and the internal entry number (IEN) of the spool document in the SPOOL DOCUMENT file (#3.51).

Figure 17-19. Generating spool document name—Sample user dialogue

```
NAME: SPOOL// ^GENERATE SPL DOC NAME
GENERATE SPL DOC NAME: YES
```

18 Special Device Issues

This chapter discusses the following special devices and device issues:

- Browser Device
- Form Feeds
- Hunt Groups
- Magtape
- Network Channel Device Support
- Resources
- SDP (obsolete)
- Slaved Printers

Browser Device

User Interface

VA FileMan's Browser allows you to view reports on your terminal screen, letting you scroll backward and forward through the report, and also letting you perform simple searches within the report. In the past, when you sent a report to the terminal screen, you could only go in one direction, forward, by pressing the **<Enter>** key after each screen of text.

If the Browser has been installed at your site and set up as a device, you can now use the Browser to view any report that asks you for an output device.

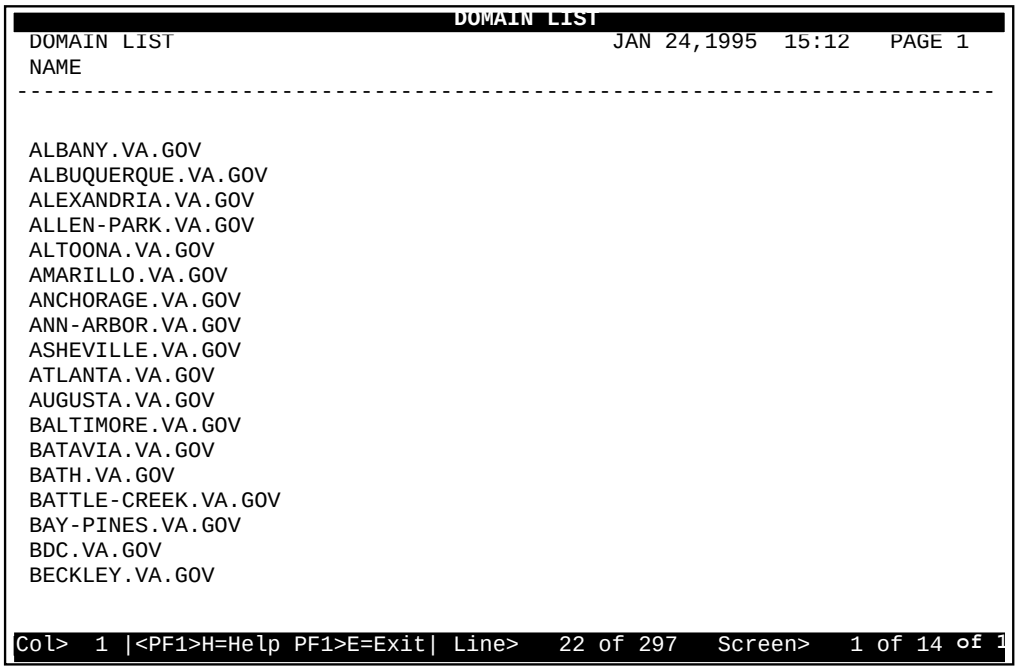
To send a report to the BROWSER device, at any device prompt, enter BROWSER as the device. You may not want to send huge reports to the BROWSER, however, since the report *must* complete before you can view its output in the Browser.

REF: For information on using the Browser and on Browser commands, please refer to the *VA FileMan Getting Started Manual.*

Figure 18-1. Print File Entries option—Sample user dialogue when sending a report to the Browser device

```
Select VA FileMan Option: Print File Entries

OUTPUT FROM WHAT FILE: DOMAIN
SORT BY: NAME// <Enter>
START WITH NAME: FIRST// <Enter>
FIRST PRINT FIELD: NAME
THEN PRINT FIELD: <Enter>
HEADING: DOMAIN LIST// <Enter>
DEVICE: HOME// BROWSER <Enter> HFS/CRT

...one moment...
```

Figure 18-2. Report displayed in Browser device—Domain List sample output



```
                                DOMAIN LIST
 DOMAIN LIST                                  JAN 24,1995  15:12   PAGE 1
 NAME
--------------------------------------------------------------------------------

 ALBANY.VA.GOV
 ALBUQUERQUE.VA.GOV
 ALEXANDRIA.VA.GOV
 ALLEN-PARK.VA.GOV
 ALTOONA.VA.GOV
 AMARILLO.VA.GOV
 ANCHORAGE.VA.GOV
 ANN-ARBOR.VA.GOV
 ASHEVILLE.VA.GOV
 ATLANTA.VA.GOV
 AUGUSTA.VA.GOV
 BALTIMORE.VA.GOV
 BATAVIA.VA.GOV
 BATH.VA.GOV
 BATTLE-CREEK.VA.GOV
 BAY-PINES.VA.GOV
 BDC.VA.GOV
 BECKLEY.VA.GOV

Col>  1 |<PF1>H=Help PF1>E=Exit| Line>   22 of 297   Screen>   1 of 14 of 1
```

System Management

You can set up VA FileMan's Browser as a device to which users can send their output.

When a user sends output to a Browser device, the Browser device performs the following steps:

1. Output is sent to a host file.
2. When the output completes, the host file is closed.
3. The contents of the host file are read back into a scratch global.
4. The host file is deleted.
5. The Browser is called, which displays the data in the global to the user, through the Browser interface.
6. When the user exits the Browser, the scratch global is deleted.

This provides a quick way to generate a report and view the report through the scrollable Browser, potentially saving paper and wear and tear on printers.

To support the Browser device, you need to set up a special terminal type (P-BROWSER), and a special device type (BROWSER).

REF: For sample entries of the special Browser terminal type and device entries for the Caché and GT.M operating systems, please refer to Figure 18-3 and Figure 18-4.

The Browser Device tests the current terminal to see whether it supports:

- A scrolling region.
- Reverse indexing.

If the terminal does not support these features, the Browser device issues a message saying that it is not selectable from the current terminal. Also, in order for the check ($$TEST^DDBRT) to work properly, the user *must* already be in the Kernel menu system or *must* have set up developer variables through the ^XUP entry point. Otherwise, the test will always fail.

Storing Host Files in a Specific Directory

By default, the temporary host files created by the Browser device are stored in the current default directory. You can optionally specify a path to a specific directory to store the temporary host files. Make sure the directory you specify exists on all nodes/CPUs where users can sign on. On DOS systems, do not specify the root directory (since there is a limit on the number of files a DOS root directory can hold). Finally, make sure you change both the OPEN PARAMETERS and POST-CLOSE EXECUTE fields in the Browser DEVICE file (#3.5) entry to specify the directory (replace DD with, for example, D:\BROW\DD).

Figure 18-3. Caché and GT.M Browser Device—TERMINAL TYPE file (#3.2) entry

```
NAME: P-BROWSER                          SELECTABLE AT SIGN-ON: NO
  RIGHT MARGIN: 80                       FORM FEED: #
  PAGE LENGTH: 99999                     BACK SPACE: $C(8)
  OPEN EXECUTE: D OPEN^DDBRZIS
  CLOSE EXECUTE: D CLOSE^DDBRZIS
  DESCRIPTION: Browser Device
```

Figure 18-4. Caché and GT.M Browser Device—DEVICE file (#3.5) Entry

```
NAME: BROWSER                            $I: DDBR.TXT
  ASK DEVICE: YES                        ASK PARAMETERS: NO
  SIGN-ON/SYSTEM DEVICE: NO              QUEUING: NOT ALLOWED
  LOCATION OF TERMINAL: HFS/CRT          ASK HOST FILE: NO
  ASK HFS I/O OPERATION: NO              MARGIN WIDTH: 80
  FORM FEED: #                           PAGE LENGTH: 99999
  BACK SPACE: $C(8)                      OPEN PARAMETERS: NEW:DELETE
  POST-CLOSE EXECUTE: D POST^DDBRZIS
  SUBTYPE: P-BROWSER                     TYPE: HOST FILE SERVER
  PRE-OPEN EXECUTE: I '$$TEST^DDBRT S %ZISQUIT=1 W $C(7),!,"Browser not selectable
from current terminal.",!
```

Form Feeds

User Interface

Most users would prefer to see their printouts without any extra blank pages before or after. Most prefer to see their reports printed on a fresh page instead of starting in the middle of the previous printout. The printing of labels should also be accomplished without unnecessary form feeds. If a printer is generating extra pages, you should contact IRM to remedy the problem.

System Management

If a particular device does not need a form feed between reports, IRM should set the SUPPRESS FORM FEED AT CLOSE field to YES in the device's DEVICE file (#3.5) entry. Label printers, for example, should have this flag set. This procedure prevents the Device Handler from issuing a form feed:

Figure 18-5. Device Edit option—Sample user dialogue

```
Select Systems Manager Menu Option: Device Handler
Select Device Handler Option: Device Edit

Select DEVICE NAME: LABEL PRINTER
NAME: LABEL PRINTER// ^SUPPRESS FORM FEED AT CLOSE
SUPPRESS FORM FEED AT CLOSE: YES
```

The Device Handler also checks the TERMINAL TYPE file (#3.2) to see if form feeds have been suppressed for that terminal type. It checks for the existence of the IONOFF variable. Thus, for certain terminal types (e.g., laser printers), IRM can set this "no form feed" variable in the corresponding terminal type's CLOSE EXECUTE field. (IONOFF can also be set by the calling program to suppress form feeds.)

Figure 18-6. Terminal Type Edit option—Sample user dialogue

```
Select Systems Manager Menu Option: Device Handler
Select Device Handler Option: Terminal Type Edit
Select TERMINAL TYPE NAME: P-DEC-LABEL
NAME: P-ZPK80// ^CLOSE EXECUTE
CLOSE EXECUTE: S IONOFF=""
```

Hunt Groups

User Interface

Hunt groups are a set of printers that IRM can set up to share print workload. If one printer in the hunt group is busy, output is directed to another one that is free. This results in less waiting time for printouts, and helps prevent any particularly long print job on one printer from holding up other output directed to the same printer. Hunt group devices are defined by IRM and would typically include similar printers located near each other. Users can send to the hunt group device or to any member of the hunt group with the same effect.

Figure 18-7. Hunt groups—Sample hunt group device name

```
DEVICE: PRTGROUP
```

The Device Handler indicates which hunt group device was selected so the user knows where to look for the output. If all members of the hunt group are busy, Device Handler displays a message with this information.

System Management

Hunt group devices can be created and managed with options on the Device Handler menu. The hunt group device type (HG type) makes use of a multiple field that holds the names of printers and other devices that can be used interchangeably.

Figure 18-8. Hunt Group Manager menu options

```
Select Systems Manager Menu Option: Device Handler

Select Device Handler Option: Hunt Group Manager

Select Hunt Group Manager Option: ?

          Edit Hunt Groups
          Delete Hunt Groups
          List Hunt Groups
          Print Hunt Groups and Associated Devices
```

If the user attempts to select a hunt group device (HG type), the Device Handler uses the B cross-reference of the multiple field to find the device with the lowest internal entry number (IEN) in the DEVICE file (#3.5). If this device cannot be opened, the Device Handler finds the hunt group member with the next higher internal number and tries again. The Device Handler discontinues the search after all hunt group members have been tried.

If the user attempts to select a printer or other device that happens to be a member of a hunt group, the Device Handler attempts to open the device. If unsuccessful, the Device Handler identifies the name of the associated hunt group device from the AHG whole-file cross-reference and proceeds as described above.

Hunt Group Device on Another CPU

If the user sends output to a device that is a member of a hunt group and is located on another CPU, the hunt group search will not succeed since the Device Handler only attempts to open devices that are on the local CPU. If the user has *queued* to the device, however, TaskMan would have the Device Handler begin searching for a device on the local CPU. If the Device Handler was unsuccessful, TaskMan would send the job to the other CPU's task global to be scheduled for processing and the effort would eventually succeed.

Queuing to a Hunt Group Device

If the user had queued to a hunt group member on the local CPU, TaskMan again calls the Device Handler to search the local CPU and, if the Device Handler is unable to open a device, returns the job to TaskMan for processing on the local CPU. (The Device Handler returns POP with a non-zero value, IO="", and IO(CPU)=destination CPU.)

NOTE: At this point, the job would be placed into the IO queue and thereafter be processed as if it was not a hunt group member.

Using a Hunt Group Device Without Using the Hunt Group

Using a device as if it was not a member of a hunt group is possible with the use of the "D" flag in the Device Handler input variable ^%ZIS.

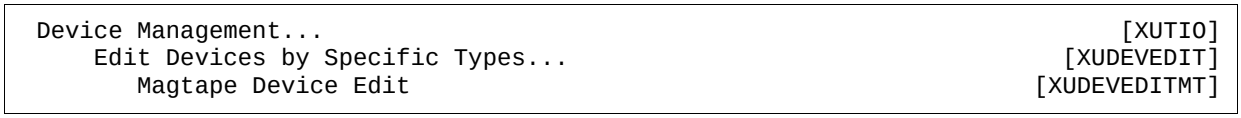
REF: For more information, please refer to the documentation for the ^%ZIS() API in the "Device Handler: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

Magtape

System Management

Figure 18-9. Edit Devices by Specific Types option



The Edit Devices by Specific Types option [XUDEVEDIT] lets you edit specific types of devices using ScreenMan.

Values entered in a Magtape device's SUBTYPE, FORM FEED, BACK SPACE, MARGIN WIDTH, and PAGE LENGTH fields may not be significant to a given application. The value of the data entered may be arbitrary. On the other hand, if the application plans to copy the output to a printer, the characteristics may need to be similar to that of the printer.

If an application intends to use these fields, be cautious about the type of data that is entered. When sent to the tape unit, some control codes will initiate tape movement or cause tape markers to be written to the mounted tape.

Data entered in the $I and OPEN PARAMETERS fields depends on the type of M system you are running, the type of tape unit, and the desired format.

REF: For examples of the type of data required in these fields, please refer to the "Device Handler: System Management" chapter in this manual.

REF: For further details, please refer to your specific M implementation manuals.

Network Channel Devices

System Management

Network channel devices are typically high speed channel devices (e.g., TCP/IP). Currently, this Network Channel Device Support exists under the Caché and GT.M operating system. In most cases, these devices are used for specialized purposes rather than for general output. For example, network mail could use such devices to move enormous amounts of mail through high speed communication channels.

The use of network channel devices requires at least two processes on each end of the communication channel, a server and a client, which can then exchange information:

- Server Process — One process *must* be available at all times. It can be actively running or triggered to run at a given moment. This process is commonly known as a server. The server waits until another process makes a request to exchange information.
- Client Process — The other process is known as the client.

The two processes can be hosted by two CPUs using network protocols.

Network Channel Device Edit

Figure 18-10. Network Channel Device Edit option

```
Device Management...                                                    [XUTIO]
    Edit Devices by Specific Types...                                [XUDEVEDIT]
       Network Channel Device Edit                                [XUDEVEDITCHAN]
```

The Network Channel Device Edit option [XUDEVEDITCHAN] allows you to edit network channel device attributes.

When editing Network Channel devices, the contents of the SUBTYPE, FORM FEED, BACK SPACE, MARGIN WIDTH, and PAGE LENGTH fields are not necessarily needed for using Network Channel devices. However, these fields are provided in case the application calling the Device Handler is not able to distinguish between a printer and a Network Channel device when sending output.

The timeout on the M open command may not be applicable with network channel devices. Therefore, it may be necessary to answer NO to the USE TIMEOUT ON OPENS field.

REF: For more information about device timeout applicability, please refer to the appropriate Caché manual.

For Network Channel devices that use TCP/IP, data is required for the OPEN PARAMETERS field. For the client device setup, this field stores the remote Internet address to which the host connects.

REF: For examples of the type of data required for these fields, please refer to the "System Management" topic under the "Network Channel Devices" topic in this chapter.

Figure 18-11. Network Channel Device Edit option—Sample output

```
                              EDIT A NETWORK CHANNEL DEVICE
NAME: SDD-DIRECT                                                    PAGE 1 OF 1
_____________________________________________________________________________

NAME: SDD-DIRECT                      LOCATION OF TERMINAL: HP-8000 near Raul
  $I: |TCP|9100                           VOLUME SET(CPU):
TYPE: NETWORK CHANNEL                SIGN-ON/SYSTEM DEVICE: NO

SUBTYPE: P-HP8000 TCP/S                       MARGIN WIDTH:
                                               PAGE LENGTH:

           ASK DEVICE: NO             USE TIMEOUT ON OPENS:
       ASK PARAMETERS: NO                     OPEN TIMEOUT:
      OPEN PARAMETERS: ("10.6.21.138":9100:"M")

             USE LOCK:
```

The GLOBAL LOCK field (3.5) in the DEVICE file (#3.5) stores a YES/NO Set of Codes. This is important, especially if the application expects that only one client at a given time is able to open the device. If this field is set to YES an M lock on ^%ZIS(“lock”,IO) will be obtained before the device is opened. It will remain until a call to ^%ZISC to close the device. It can be used with any type of device.

Resources

System Management

A resource is a type of device that can only be used by tasks. They cannot be used for input or output (I/O). As such, they are not available for user selection at the device prompt. The purpose of a resource is to provide a mechanism of limiting the number of concurrent jobs that can run at any one time.

When creating a task, a task can request the resource as an input variable for the ^%ZTLOAD call. The resource itself, as defined in the DEVICE file (#3.5), has a field called RESOURCE SLOTS that determines how many jobs can simultaneously own it as a resource.

Device Handler and TaskMan work together to provide resource device functionality. The RESOURCE file (#3.54), stored in the translated ^%ZISL global, regulates processing and is for internal use only. The NAME field (#.01) holds the $I of the resource device. Other fields hold information on jobs currently using the resource, information that is cleared when the resource is closed.

The RESOURCE file (#3.54) supports processing by maintaining a count of the number of available "slots." The ability to open and close resources is accomplished by decrementing and incrementing this count.

Limiting Simultaneous Running of a Particular Task

Resources make it possible for you to control the number of a particular kind of non-I/O task that runs at any one time. If you have a particular job and you want no more than three running versions of it at any one time, you can queue the job (through the ^%ZTLOAD interface) to a resource that had a RESOURCE SLOTS setting of 3.

Running Sequences of Tasks

Resources also make it possible to run non-I/O tasks in sequential order. Non-I/O tasks ordinarily can run simultaneously because they do not compete for the ownership of I/O devices. If you instead queue such tasks to the same resource, and the resource has a RESOURCE SLOT setting of 1, TaskMan will run the tasks one at a time and in the order queued. In this way, the results of one process can be used by another. This sequential processing might be appropriate, for example, for the processing of physician orders or other nested tasks involving code execution.

An additional enhancement to resource devices, called SYNC FLAGs, allows TaskMan to run the next task waiting for a resource only if the previous task using that resource has completed successfully. You can use SYNC FLAGs to ensure that subsequent jobs run only if previous jobs have completed successfully.

Creating Resource Devices

Figure 18-12. Resource Device Edit option

```
SYSTEMS MANAGER MENU ...                                                [EVE]
   Device Management ...                                               [XUTIO]
      Resource Device Edit                                          [XUDEVEDITRES]
```

The Resource Device Edit option [XUDEVEDITRES] provides a facility for editing resource devices. Software that uses a resource should include in its installation instructions the way the new resource should be defined in the DEVICE file (#3.5). IRM can then create one or more resource-type (RES) entries.

Figure 18-13. Resource device—Sample output

```
NAME:  ZZRES       $I:  ZZRES
   LOCATION OF TERMINAL:  NA   RESOURCE SLOTS:  1
   TYPE:  RESOURCE
```

The installation instructions should indicate the number of resource slots. Sequential processing should use a value of 1. The name and $I should probably use the same value and be namespaced according to VistA conventions.

SDP (Obsolete)

Though the Sequential Disk Processors (SDP)SDP entry is still found in the DEVICE file (#3.5), it is obsolete and users should now use Host File Server (HFS) devices.

REF: For more information on HFS devices, please refer to the "Host Files" chapter in this manual.

Slaved Printers

User Interface

If your terminal has an auxiliary printer port with a printer directly attached, you can send output normally destined for the CRT terminal directly to a printer. Output for the terminal is redirected from the host computer through the terminal's auxiliary port to the printer. Such printers are commonly called slaved printers or slaved devices.

If slaved printing is available from your terminal, you can send a printed report to your slaved printer, by entering the device name that corresponds to your slaved printer like this:

Figure 18-14. Slaved Printer—Sample user dialogue

```
DEVICE:  SLAVELA50
```

You can consult your local IRM to find out if slaved printing devices are available.

System Management

There are two modes of slaved printing:

- **Auto Print Mode (a.k.a. Copy Print Mode)—**When Auto Print Mode is toggled on, output is displayed on the terminal as well as printed on the printer. Special escape sequences and control characters, such as those that are normally used to adjust fonts/pitches, are *not* passed to the printer; however, those used for actions like carriage return, line feed, and form feed are passed on to the printer.
- **Printer Controller Mode (a.k.a. Transparent Print Mode)—**When Printer Controller Mode is on, output is only printed on the printer; nothing is displayed on the terminal. All escape sequences and control characters are passed to the printer. This mode is preferable to Auto Print Mode, especially when compressed mode printing is desired.

The following are the escape sequences used to toggle the slaved printing modes for DEC VT220/VT320 terminals:

Table 18-1. Escape sequences used to toggle the slaved printing modes for DEC VT220/VT320 terminals

Mode	Escape Sequence
Auto print mode on.	ESC [?5i
Auto print mode off.	ESC [?4i
Printer controller mode on.	ESC [5i
Printer controller mode off.	ESC [4i

Device and Terminal Type File Entries

To use a slaved printer through the Device Handler, two DEVICE file (#3.5) entries along with corresponding TERMINAL TYPE file (#3.2) entries *must* be made for the following:

- Home Device
- Slaved Printer

One pair of DEVICE/TERMINAL TYPE entries is needed to describe the home (i.e., CRT) terminal attributes including the codes to open and close the printer port. The OPEN PRINTER PORT and CLOSE PRINTER PORT fields of the TERMINAL TYPE file (#3.2) can be used to store the appropriate codes.

Another pair of DEVICE/TERMINAL TYPE entries is needed to describe the attributes of the slaved printer including escape codes to adjust fonts/pitches. The OPEN EXECUTE and CLOSE EXECUTE fields of the TERMINAL TYPE file (#3.2) can be used to hold such codes. Additionally, the device entry for the slaved printer *must* have a value of 0 (zero) entered into the $I field. This $I value identifies the DEVICE file (#3.5) entry as one for a slaved device.

The following examples show the setup for a home device, and the setup for slaved printers

Figure 18-15. Home Device example (VT320)—DEVICE file (#3.5) entry

```
NAME: TELNET DEVICE                          $I: _TNA
  ASK DEVICE: YES                         ASK PARAMETERS: NO
  VOLUME SET(CPU): KDE                    SIGN-ON/SYSTEM DEVICE: YES
  LOCATION OF TERMINAL: Network           MARGIN WIDTH: 80
  FORM FEED: #,$C(27,91,50,74,27,91,72) PAGE LENGTH: 24
  BACK SPACE: $C(8)                       SUBTYPE: C-VT320
  TYPE: VIRTUAL TERMINAL
  TIMED READ (# OF SECONDS): 400
```

Figure 18-16. Home Device example (VT320)—TERMINAL TYPE file (#3.2) entry

```
NAME: C-VT320                          SELECTABLE AT SIGN-ON: YES
  FORM FEED: #,$C(27,91,50,74,27,91,72) RIGHT MARGIN: 80
  PAGE LENGTH: 24                      BACK SPACE: $C(8)
  DESCRIPTION: Digital Equipment Corporation VT-320 video
  OPEN PRINTER PORT: W *27,"[5i"
  CLOSE PRINTER PORT: W *27,"[4i"
```

Figure 18-17. Slaved Printer example: DEC LA50—DEVICE file (#3.5) entry

```
NAME: SLAVELA50                        $I: 0
  ASK DEVICE: YES                      ASK PARAMETERS: YES
  SLAVED FROM DEVICE: TRM
  LOCATION OF TERMINAL: SLAVE DEVICE FOR LA50
  MARGIN WIDTH: 132                    FORM FEED: #
  PAGE LENGTH: 64                      SUBTYPE: P-LA50
  TYPE: TERMINAL
```

Figure 18-18. Slaved Printer example: DEC LA50—TERMINAL TYPE file (#3.2) entry

```
NAME: P-LA50                           RIGHT MARGIN: 132
  FORM FEED: #                         PAGE LENGTH: 64
  OPEN EXECUTE: W *27,"[4w"            CLOSE EXECUTE: W *27,"[0w"
  DESCRIPTION: LA50 132 COL/16.5 CPI
```

Figure 18-19. Slaved Printer example: Epson LQ870—DEVICE file (#3.5) entry

```
NAME: SLAVELQ870                       $I: 0
  ASK DEVICE: YES                      ASK PARAMETERS: YES
  SLAVED FROM DEVICE: TRM
  LOCATION OF TERMINAL: SLAVE DEVICE FOR LQ870
  MARGIN WIDTH: 132                    FORM FEED: #
  PAGE LENGTH: 64                      SUBTYPE: P-LQ870
  TYPE: TERMINAL
```

Figure 18-20. Slaved Printer example: Epson LQ870—TERMINAL TYPE file (#3.2) entry

```
NAME: P-LQ870                          RIGHT MARGIN: 132
  FORM FEED: #                         PAGE LENGTH: 64
  OPEN EXECUTE: W *15                  CLOSE EXECUTE: W *18
  DESCRIPTION: EPSON LQ870 PRINTER--CONDENSED
```

Use of Slaved Printer: Processing Steps

The Device Handler manages output to slaved printers using the following steps:

1. Execute the OPEN PRINTER PORT code of the home device's terminal type.
2. Execute the OPEN EXECUTE code of the slaved printer's terminal type.

3. When the application closes the device, execute the CLOSE EXECUTE code of the slaved printer's terminal type.
4. Execute the CLOSE PRINTER PORT code of the home device's terminal type.

Queuing to Slaved Printers

If queuing to a slaved device is desired, then the SLAVE FROM DEVICE field of the DEVICE file (#3.5) *must* be used. This field is a pointer to the DEVICE file (#3.5). Data *must* be entered in this field for the entry for the slaved printer. This data should point to the home device entry unless the slaved printer is attached to a terminal on a Terminal Server (i.e., a virtual terminal).

If queuing to a slaved device is being performed from a virtual terminal, then a third device entry *must* be established that fully describes the home device with a type of TRM. This device should be entered into the SLAVE FROM DEVICE field.

NOTE: When queuing to a slaved device from a terminal on a Terminal Server, the user *must* be fully logged off the computer system and logged off the port by the time the queued task is scheduled to run.

IV TaskMan

19 TaskMan: User Interface

The Kernel TaskMan (TM) software allows you to run tasks (e.g., VA FileMan prints and sorts) in the background and lets you continue working without interruption.

Creating Tasks

VistA runs in a multiprocessing environment, which means the computer can work on more than one job at a time. Each job the computer works on consumes a part of the computer's resources. Initially, you have only one job, your interactive terminal session, with which to do your work. TaskMan, however, allows you to claim more of the computer's resources by allowing you to schedule additional jobs to run in the background.

Background Jobs

You can queue additional tasks to run through TaskMan. Once started, these additional tasks (called background tasks) can run at the same time as the foreground jobs and without further dialogue with the people who started them. Appropriate use of background tasks can cut your frustration by reducing the amount of time you *must* wait for the computer to do lengthy, repetitious work that does not need human intervention. Every task queued to run in the background reduces time spent waiting and also uses the computer's resources more efficiently.

Queuing Output

Most users use TaskMan by queuing reports, labels, and other kinds of output. Because output involves no dialogue once it has begun and because it requires you to wait while it prints, it makes an ideal candidate for queuing. You can queue most output when the computer asks you to select a device to which the output should be sent. The series of prompts and responses to queue a job to a device usually looks something like this:

Figure 19-1. Queuing output—Sample user dialogue

```
DEVICE:  QUEUE TO PRINT ON

DEVICE:

Requested time to print:  NOW// <Enter>

Request queued.
```

Answer with name of the output device here.

After you answer this series of prompts, the output is queued for TaskMan to start at the requested time, and you can continue with other work while TaskMan prints the output. When many tasks need the same device at the same time, TaskMan runs them in order based on the time they were requested.

Other Sources of Tasks

An application can create other kinds of tasks without your interaction. The application might offer to queue other kinds of work like large filing or complex data analysis jobs. Sometimes applications queue tasks without asking. For example, the delivery of MailMan messages is performed by a job running as a task. If that task is not running when someone uses the MailMan options, MailMan automatically uses their foreground job to queue the task without asking them. Although people may knowingly or unknowingly queue these other kinds of tasks, output remains the most common kind of work to queue.

Working with Tasks

Figure 19-2. TaskMan User option

```
System Command Options ...                                          [XUCOMMAND]
User's Toolbox ...  "TBOX"                                         [XUSERTOOLS]
   TaskMan User                                                     [XUTM USER]
```

TaskMan also allows you to examine or modify your own tasks. You can do this by using the TaskMan User option [XUTM USER], located in the User's Toolbox menu on your Common menu. This option lets you monitor or manipulate one task at a time.

Selecting Tasks

When you choose the TaskMan User option, it first asks you to select a task with which to work. TaskMan displays the "Select TASK:" prompt. If you enter a single question mark ("**?**"), you get some general help about the option; if you enter two question marks ("**??**"), you can get a list of every task that you have queued to run. Typically, you would enter two question marks at this prompt so that you can get a listing of your individual tasks, listed by task number. You then choose a task from the list of tasks to work with. Using the TaskMan User option looks like the following:

Figure 19-3. TaskMan User option—Sample user dialogue

```
Select User's Toolbox Option: TaskMan User


Select TASK: ??


Please wait while I find your tasks...searching...finished!

------------------------------------------------------------------------------
1: (Task #161325) ZTSK2^XMA02, Queued print for KRNUSER,TWELVE.  Device VER$LW.
   KRN,KDE.  From TODAY at 14:22,  By you.  Scheduled for TODAY at 20:00
------------------------------------------------------------------------------
2: (Task #161776) ZTSK^DIP4, DEVICE LIST.  Device VER$LW.  KRN,KDE.
   From TODAY at 14:22,  By you.  Scheduled for TODAY at 22:00
------------------------------------------------------------------------------

End of listing.  Press RETURN to continue: <Enter>


Select TASK: 161776 <Enter> DEVICE LIST

               Taskman User Option

                    Display status.
                    Stop task.
                    Edit task.
                    Print task.
                    List own tasks.
                    Select another task.

               Select Action (Task # 161776):
```

You can select tasks either by task number or list number. In the list of tasks, the list number is at the left hand side of the each task listing, and is followed by the task number for each task (in parentheses). The rest of the information helps identify where the task came from and what it will do.

Tasks in the Task List

You can only select tasks that are still in TaskMan's task list. When a task finishes running, it usually removes itself from the task list. Thus, you should not get a listing of every task you have run in the last year! Tasks that do not clean up their entries usually get cleaned out by TaskMan several days after they

complete. You should only have to select tasks that are still actively waiting to start, currently running, or encountered some kind of problem while running.

Display Status of Tasks

Once you've selected a task to work with, you can ask to see the status of that task, using the Display status option ("D"). TaskMan uses a task's status to try to explain how soon the task will run and why. The possible normal statuses for a task include:

- Scheduled for <date and time>.
- Being inspected by TaskMan.
- Waiting for a partition.
- Being prepared.
- Currently running.
- Completed <date and time>.

NOTE: Please keep in mind that TaskMan can only "guess" whether a task is currently running.

One of the following messages may show up if the task needs some system resource not currently available:

- Waiting for hunt group(s) <list of hunt groups>.
- Waiting for device <name of device>.
- Waiting for the link to <name of CPU> to be restored.

When you display the status of a task waiting for a device, TaskMan shows you how many tasks are in line for that device ahead of your task. Additional statuses exist for tasks that have encountered some kind of problem. For each situation it lists a different explanation of the problem. For example, if you use the Stop task option to stop a task, its status shows up as "Stopped by you."

Stopping Tasks

Under certain conditions, you may want to stop a task. The TaskMan User option allows you to do this through the Stop task option ("ST"). Your ability to stop a task depends on the task's status, however. If the task has already been stopped, is finished, or it encountered a problem while running and you try to stop it, the Stop task option tells you that the task has already stopped. If the task has not yet started running, on the other hand, you can always stop it. If the task has started running, the Stop task option will succeed in stopping it only if the developer who wrote the task has designed the task to be stopped by a user. At any rate, it does not cause any problems if you try to stop a running task.

To stop a task, use the Stop task option. Once you stop a task, it remains in the TASKS file (#14.4) until you edit it to run again or until TaskMan purges it from the Task list.

Editing Tasks

The Edit task option ("E") lets you edit a task's output device, description, and run time.

The task *must* be unscheduled before it can be edited. The Edit task option asks if it's OK to unschedule the task. To edit the task, answer YES. But once the task is unscheduled, it will not run unless you reschedule it by finishing each step of editing the task.

NOTE: You *cannot* edit a task that is already running.

Once the task is unscheduled, you can update the following task settings:

- When the task should start.
- Which device it should use (and whether a device is needed).
- What the description of the task should be.

Once you've had a chance to modify these three settings, you're asked whether the task should be rescheduled as shown (see Figure 19-4). If you answer YES, the task is updated to reflect the changes you specified. If you answer NO, however, no settings are changed, but the task remains unscheduled (and will not run until you use Edit Task to reschedule it).

Figure 19-4. Edit Task option—Sample user dialogue

```
Before you edit the task I'll make sure it's not scheduled, okay?  YES//  <Enter>
  Task ready for editing.

  Currently, this task requests output device VER$LW.
  Do you want to change the output device for this task? NO// Y
  Select Task's Output Device (^ for none): P236

  When should this task run?:  AUG 16, 2004@22:00// <Enter>

  Task's purpose: DEVICE LIST// <Enter>

161776: DEVICE LIST.  P236.  Next run time: AUG 16, 2004@22:00.

  Shall I reschedule this task as shown? YES// <Enter>
  Task rescheduled.
```

Listing and Printing Tasks

You can use the List own tasks option ("L") to review your tasks. This option displays the same list as that given when you enter two question marks ("**??**") at the "Select Task:" prompt.

The Print task option lets you print out the description of the task that you have currently selected.

Selecting Another Task

Once in the TaskMan User option, you can choose to work with a different task by using the Select another task option ("SE"). Enter another task number to work with a different task. If you're not sure what task you want to work with, you can get a list of all of your tasks by entering two question marks ("**??**").

Summary

Most output in VistA is performed by creating tasks that run in the background. Once you become familiar with TaskMan's queuing system, you can increase productivity by using some of TaskMan's special features, including listing your future tasks, displaying a task's status, stopping a running task, and editing a future task's run time and output device.

20 TaskMan: System Management—Overview

Kernel's TaskMan module provides a standardized system for initiating and managing background processing. Since TaskMan handles all background processes, system managers have a unified set of controls that apply to all background processes on their systems.

Most of TaskMan's processing does not involve interaction with users, rendering its operation virtually invisible. The explanations that follow provide information about the operation of TaskMan.

TaskMan's Division of Labor

TaskMan uses a three-step system to start and manage background processing:

1. **Queuers**

   Foreground jobs cannot directly start any background jobs. Instead, they call the TaskMan Application Program Interface (API) to file requests in the TASKS (#14.4) and SCHEDULE files. The program code calling the TaskMan API is called a Queuer. The TASKS file (#14.4) is VA FileMan-compatible. The SCHEDULE file is *not* VA FileMan compatible.

   REF: For a description of the TASKS (#14.4) and SCHEDULE file structure, please refer to the "Troubleshooting" topic in the "TaskMan: System Management—Operation" chapter in this manual.

2. **Manager**

   A TaskMan program called the Manager runs at all times in the background. The Manager monitors the SCHEDULE file. As needed, it initiates background jobs (called Submanagers) to perform the work requested by the foreground jobs.

3. **Submanagers**

   Each background job request is picked up by a TaskMan process called the Submanager. The Submanager is the job that actually runs each task. Submanagers handle contention for partitions and I/O devices by running the waiting tasks in order, first the oldest tasks and then the more recent ones.

Queuers

Tasks run by TaskMan begin with code in an software application that decides to perform some work in the background. This code is a queuer. Most applications in VistA respond to a user's request to queue some output, but other decisions may be involved. Two commonly used queuers are programs that create report output (by using the TaskMan API) and options that are scheduled through the OPTION SCHEDULING file (#19.2).

Programs that Use the TaskMan API

One commonly used queuer is an application's call to the TaskMan API to queue tasks. In this process the queuer defines the task and its environment. Applications are not allowed to do direct manipulation of the ^%ZTSCH and ^%ZTSK globals.

The TaskMan API consists of entry points that allow developers to create, manipulate, and inquire about tasks. The most widely used entry point, ^%ZTLOAD, lets developers queue tasks, which involves creating and scheduling them. First, an application sets the variables that ^%ZTLOAD needs to define the desired task. In turn, ^%ZTLOAD uses that information to create an entry in the TASKS file (#14.4). ^%ZTLOAD then sets up a simple cross-reference to the new task in the SCHEDULE file, thereby finishing the queuing process.

After queuing the task, ^%ZTLOAD quits, returning control back to the queuer and leaving the next step in the process to the Manager routines.

Option Scheduling through the OPTION SCHEDULING File (#19.2)

Another commonly used queuer is the OPTION SCHEDULING file (#19.2). Menu Manager and TaskMan work together to allow certain options to be run as TaskMan tasks. These special options can be scheduled to run just once, or they can be set up to run over and over based on a rescheduling cycle. Such cycles can even include running the task whenever the computer system boots up.

Manager

For tasks to run, at least one CPU in a configuration needs to run a Manager. Only one Manager process needs to run per CPU; the site determines how many CPUs should be configured to run a Manager. The Manager's job is to route the tasks created by queuers. It normally runs at all times in the manager UCIs. It repeats the same loop of code all day long; every 2 seconds it looks for overdue tasks, every 15 seconds it checks the environment and performs some cleanup.

The environment check allows the system manager to control the Manager even at its busiest. All of the commands to which the Manager responds (described later) take effect here, between every task processed.

The Manager looks for overdue tasks in the schedule list, comparing the current time to the start time of the tasks listed. If an overdue task is found, the Manager removes it from the schedule list and inspects it. If the task is defined with a complete task record, the Manager places it in a list of tasks ready to run. The Manager places a task on one of several different lists depending on whether the task needs ownership of a currently unavailable I/O device. As its final step in processing each overdue task, the Manager checks the number of Submanagers available to process tasks and starts up new submanagers, if needed. The Manager uses the JOB command (or %SPAWN if the Manager is running in a DCL context on a Caché system).

The only variation on this scheme happens when the Manager finds a task bound for a different Volume Set. Depending on the system configuration, such tasks may need to be run by the Manager running on that other Volume Set. In this case, the current Volume Set's Manager copies the task over to the Volume Set on which the task should run and marks it as moved in the current TASKS file (#14.4). In this

process, the task is assigned a new task number, and the Manager on that other Volume Set handles the task from there. If during this process the Manager discovers that the link between the two Volume Sets has dropped, it saves the task in a list of tasks waiting for that Volume Set and checks periodically to see whether it has been restored. When the link recovers, the Manager will send, in sequence, all the waiting tasks to the other Volume Set.

The Manager never actually runs the task but merely places it in a list as a task now available to be run by a Submanager.

Submanagers

Submanagers are the processes that actually run tasks. A Manager starts Submanagers whenever more are needed to handle the current workload of tasks, and they only last as long as they are needed. Submanagers loop back and forth between finding new tasks to run and running them.

To run each task, the Submanager first removes the task from the list of waiting tasks on which it reside (e.g., the Job or the I/O list). Then it looks up the task's entry in the TASKS file (#14.4), unloading all of the information about the task. If the task needs a device, the Submanager calls the Device Handler to get ownership of it and issues a USE command for it. Then the Submanager sets up the partition for the task and does the following:

- Sets the priority.
- Cleans out unwanted variables.
- Sets up requested variables.
- Prints a page header on the device if one was requested, etc.

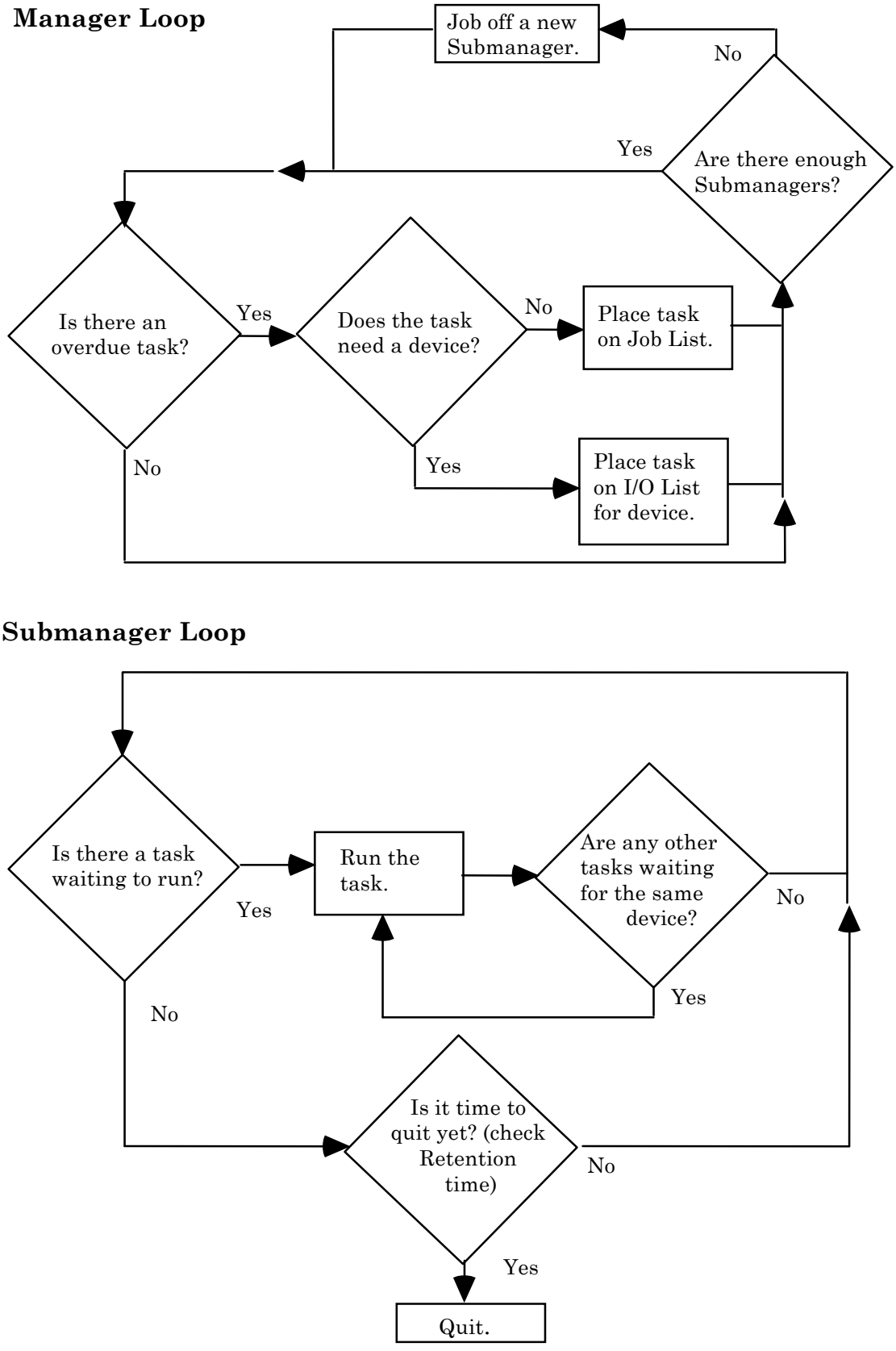
Next, the Submanager starts the task running at the task's entry point. The Submanager uses a DO command and runs the task's entry point in its own partition. When the task finishes, the Submanager cleans up after the task:

- Closes the output device.
- Performs any commands left for it by the task, etc.

Running completely without user interaction, each task performs the work it was created to do and then quits, returning control to the Submanager that started it. The task may leave instructions for its Submanager, such as to requeue the task so that it runs again later or to delete the task's entry from the TASKS file (#14.4), but the task itself finishes before the Submanager continues.

After Submanagers have run all available tasks, they wait an interval before quitting. This period, called Submanager retention time, allows the Submanager to keep its partition open for new tasks for a while so that the Manager need not start a new Submanager. Every time a new task shows up during the retention time, the Submanager starts its main loop over again, returning to retention again only after all new tasks have been run. When the Submanagers eventually reach the end of their retention time, they quit.

Figure 20-1. TaskMan Manager and Submanager process flow diagram



TaskMan's Files

The two central files that facilitate task processing are:

- TASKS file (#14.4)
- SCHEDULE file (*not* VA FileMan-compatible)

TaskMan is configured by three configuration Files:

- VOLUME SET (#14.5)
- UCI ASSOCIATION (#14.6)
- TASKMAN SITE PARAMETERS (#14.7)

These files and the TaskMan routines fall within TaskMan's namespace (ZTM), and numberspace. TaskMan user interface routines have been moved to the XUTM namespace beginning with Kernel V. 8.0 (they were previously in the ZTM namespace).

TaskMan also relies upon software components outside of its direct control. As an integral part of Kernel, TaskMan accesses several files controlled by other Kernel modules and calls many software entry points as a whole. TaskMan's main external relation, however, is with VistA software applications through the queuers and the tasks they use.

TaskMan Globals: ^%ZTSCH and ^%ZTSK

^%ZTSCH holds the SCHEDULE file, and ^%ZTSK the TASKS file (#14.4). Every environment controlled by a single Manager needs each of these globals in its library UCI. % globals are used to make these files accessible to all the UCIs in that environment so a single Manager's influence spans all of those UCIs. When the environment spans Volume Sets, ^%ZTSCH and ^%ZTSK are translated across the Volume Sets included. They are never replicated because TaskMan updates them so frequently.

The ^%ZTSK global is mostly defined by VA FileMan (beginning with Kernel V. 8.0), but the ^%ZTSCH is not. Historically these globals were not VA FileMan-compatible. Now the inquire, search, and print capabilities of VA FileMan can be used to study the TASKS file (#14.4). At present, all edit access to these globals is restricted to the TaskMan options that edit the tasks in various ways.

REF: For a description of the structure of ^%ZTSCH and ^%ZTSK, please refer to the "Troubleshooting" topic in the "TaskMan: System Management—Operation" chapter in this manual.

SCHEDULE File

The SCHEDULE file holds all of the lists and nodes that TaskMan uses to manage itself and to schedule tasks. Some of these lists are:

- Schedule List (or Time Queue)
- Waiting List (or IO Queue)
- Job List
- Compute Server Job List (or C List)
- Link List
- Status List
- Run Node
- Taskman Error Log
- Error Screens

The SCHEDULE file's function is split between identifying the status of active tasks and of TaskMan itself.

REF: For more information on these lists, please refer to the "TaskMan: System Management—Operation" chapter in this manual.

Most of the lists in the SCHEDULE file describe tasks, as follows:

- Schedule List—Sorts all scheduled tasks by time, according to when they are supposed to begin running.
- Waiting List—Stores each task whose running was delayed because its I/O device was busy.
- Job List—Holds those tasks that can begin running immediately.
- Link List—Stores tasks whose running is delayed because of a dropped link to another Volume Set.
- Task List—Describes all actively running tasks.
- Compute Server Job List—Describes all tasks waiting to start on a Compute Server (cross-CPU queuing).

The role of tracking the status of TaskMan itself is split between lists of information and individual nodes and flags. The Status List is where the Manager keeps track of its current condition; it is a list because IRM may choose to run more than one Manager in the same TaskMan environment. The RUN Node is a place where TaskMan stamps the current time; this node reveals when TaskMan stops running. The Taskman Error Log is a simple list in which TaskMan stores each error that occurs either within TaskMan itself or within the tasks that it runs. The Error Screens are screens that can be established by IRM to prevent the recording of certain errors.

These lists and nodes, as well as others not described here, are the primary data structures that TaskMan uses to schedule and run tasks.

TASKS File (#14.4)

The TASKS file (#14.4), unlike the SCHEDULE file, contains the tasks themselves.

Every task run by TaskMan is described by an entry in the TASKS file (#14.4). Each entry is subscripted by a unique internal number, and ^%ZTSK(-1) always equals the number of the most recently created task. The lists and nodes in ^%ZTSCH store the tasks' numbers that are scheduled to run. Each task's entry consists of a ^%ZTSK(task #, 0) node that contains most of the essential information about the task, several decimal nodes (.1, .2, .25, and .26) that store the remainder of the critical information, and a number of storage nodes under ^%ZTSK(task#,.3) that store the names and values of parameters that TaskMan creates for the task. Left unchecked, this file tends to grow.

REF: For a description of the various means of controlling this growth, please refer to the "TaskMan: System Management—Operation" chapter in this manual.

Other Files

The TASKS (#14.4) and SCHEDULE files, taken together, describe all the information about tasks on the system. A few more files are needed, however, to describe everything about how tasks are managed on the system.

The following three files are stored in ^%ZIS:

- The VOLUME SET file (#14.5)—Describes the computer system's Volume Sets and how they are configured into TaskMan environments.
- The UCI ASSOCIATION file (#14.6)—Lists all the UCIs on the system and which Volume Sets they belong to. In more complicated systems, it is also used to describe how the UCIs in different environments correspond with one another.
- The TASKMAN SITE PARAMETERS file (#14.7)—Lets the system manager divide up the environments by both CPU and Volume Set. This allows a fine degree of control over such parameters as priority, partition size, and retention time.

Taken together, these files give IRM precise and powerful control over TaskMan's behavior.

Other minor pieces of information are scattered throughout other Kernel files, especially the DEVICE (#3.5) and OPTION SCHEDULING (#19.2) files.

System Configuration Terminology

TaskMan operates close to the level of the system architecture. It *must* be capable of starting tasks in all the environments within a computer system. This means it *must* know about those environments; consequently, the options, routines, files, and documentation somehow *must* refer to that architecture.

One problem presented by system configuration is terminology. Such system architecture features as UCIs, directories, Volume Sets, and namespaces are not part of the ANSI M standard, so different vendors use different terminology. Although it would be ideal for Kernel to use a universal terminology, none exists. For historical reasons, Kernel has settled on a terminology based on that of **DSM-11** that includes the following terms:

Table 20-1. TaskMan system configuration terminology

Term	Definition
UCI	User Class Identifier. This is roughly equivalent to a "directory" or an "account". A UCI refers to the environment limited to a particular set of routines and globals. In Caché terms, this is a "namespace."
Manager UCI	Roughly equivalent to a "system UCI" or a "library UCI." This is where the vendor's system management routines are kept, and where all %-namespaced routines and globals reside. Currently, all Kernel % routines and globals are mapped back to the production account.
Volume Set	On current systems, we just set this to the string “ROU”. This is the critical definition, since this is what affects how TaskMan starts background jobs.
CPU	Also known as a "node" or "computer", this designates a source of computing power and partitions. It is used both for controlling TaskMan's behavior with parameters and for sending tasks to specific CPUs.
Mounted Volume Set	Obsolete; no longer used.

The TaskMan chapters that follow make use of this terminology.

TaskMan Security Key

The TaskMan module comes with one security key, ZTMQ. The ZTMQ security key does not completely lock any options. Instead, it affects the behavior of the following three options:

- Dequeue Tasks [XUTM DQ]
- Requeue Tasks [XUTM REQ]
- Delete Tasks [XUTM DEL]

Those who use these options without holding this security key can manipulate only their own tasks. Only the holder of the ZTMQ security key can use these options to manipulate any task on the system.

21 TaskMan: System Management—Configuration

This chapter discusses the many issues surrounding the configuration of TaskMan.

Defining TaskMan Environments

The part of configuring TaskMan for a system that requires the most creativity is deciding how to divide the system's UCIs, Volume Sets, and CPUs into TaskMan environments. A TaskMan environment is the collection of UCIs from which entries can be made directly into a given Manager's TASKS (#14.4) and SCHEDULE files and that are within that Manager's reach. This requires looking at the system in terms of queuing and starting tasks. There are a number of options available. Many different configurations are possible.

One type of configuration has CPUs sharing the same Volume Set. Since this type of environment shares a single Volume Set among multiple CPUs, they also share a single TASKS (#14.4) and SCHEDULE file. However, the reach of Managers cannot span CPUs. Therefore, you *must* decide which CPUs in that environment run Managers, or whether some of them should rely on the other CPUs to run their tasks for them. Alpha clusters in VA are typically configured with Managers on only one or a few CPUs.

A different configuration allows you to limit the number of places TaskMan runs. In this scenario, you pick certain CPUs to run TaskMan and give them Managers and files to do the job. To have background processing support, the remaining Volume Sets need to be able to queue to one of the Managers on the system. This entails translating the TASKS (#14.4) and SCHEDULE files of that Manager so they are visible to the unsupported Volume Set. To tell TaskMan that the one Volume Set runs no tasks but is instead supported by the other, you *must* configure the VOLUME SET file (#14.5) as described later in this topic.

Another possible configuration is to allow tasks to run everywhere, which requires that you place Managers within reach of every UCI and that you define your TaskMan environments accordingly. Under this configuration every CPU needs its own Manager, and its own TASKS (#14.4) and SCHEDULE files.

One other configuration to keep in mind, of course, is to have a standalone environment disconnected from the rest of the computer system. Such environments make excellent test areas for developers. They are configured the same regardless of the configuration of the main system.

Configuring TaskMan

TaskMan's three configuration files *must* be setup to properly reflect your system's layout. The three files are:

- TASKMAN SITE PARAMETERS (#14.7)
- VOLUME SET (#14.5)
- UCI ASSOCIATION (#14.6)

There are three options on the Edit TaskMan Parameters menu, one to edit each of the three configuration files.

Because the TASKMAN SITE PARAMETERS (#14.7) allows you to define parameters (e.g., TaskMan Job Limit) separately for each CPU on your system; you are able to optimize TaskMan's behavior individually for each CPU.

You no longer need to stop and then restart TaskMan in order to change the TASKMAN JOB LIMIT on a CPU. Cross-references on the relevant fields locate every TaskMan on your system and inform them that they need to update their TaskMan parameter information. Thus, within a minute or so of making the changes, TaskMan on that CPU should be operating with the new value.

TaskMan's Reach

The key issue that defines TaskMan's configuration is its "reach," those places where TaskMan can start background jobs. TaskMan's reach extends to:

- All UCIs a Submanager can access directly after using Kernel's UCI switching facilities.
- All other Managers to whose TASKS (#14.4) and SCHEDULE files a given Manager can WRITE using extended global reference.
- All UCIs on Print Servers with link access to the current Volume Set.

TaskMan's reach does *not* include other sites on a wide area network, because they *cannot* be accessed through either UCI switching or through extended global reference. There are ways to simulate such a reach through the use of server options, however. For purposes of TaskMan configuration, we generally think in terms of the reach of a single Manager, which can only run tasks in the UCIs it can reach.

TASKMAN SITE PARAMETERS File (#14.7)

Figure 21-1. Site Parameters Edit option

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Task Manager ...                                                         [XUTM MGR]
   Taskman Management Utilities ...                                     [XUTM UTIL]
      Edit Taskman Parameters ...                                 [XUTM PARAMETER EDIT]
         Site Parameters Edit                                             [XUTM BVPAIR]
```

System managers *must* enter one set of site parameters into the TASKMAN SITE PARAMETERS file (#14.7) for each Manager that runs in a different Volume Set/CPU. This set of parameters tells each Manager how it should process tasks. The parameters are organized both by Volume Set and by CPU. This allows two CPUs that share a Volume Set to be treated differently if one is more powerful than the other.

BOX-VOLUME PAIR

The BOX-VOLUME PAIR field identifies a Volume Set and the CPU on which it is available. It contains the name of a Volume Set concatenated to the CPU ("box") name: first the Volume Set name and then the CPU name. For example, if the Volume Set name is "KRN" and the name of the CPU (e.g., box) is "ISC6A1," then the BOX-VOLUME PAIR would be "KRN:ISC6A1."

For systems on which each CPU tends to have a unique Volume Set, and vice versa, you can enter just the Volume Set name (e.g., "PSA" or "AAA"). This field's value for the current process can be found by doing GETENV^%ZOSV and checking the fourth ^-piece of Y. Since the Volume Set and CPU are identified, the TaskMan site parameters can be tuned for each specific Volume Set and CPU affected. Systems running Managers on more than one CPU need one entry for each CPU where a manager is running.

LOG TASKS?

Set the LOG TASKS? field to YES to make tasks log in and out through the signon log the way interactive users do. How to set this is up to the individual site; it does consume space and resources.

TASK PARTITION SIZE

The TASK PARTITION SIZE field is used to assign partition sizes for tasks. The value from this field is plugged directly into the JOB command used to create new submanagers. If this field is left blank, all tasks receive the operating system's current default value. This field should only be used by system managers who thoroughly understand how their vendor's version of M handles partition sizes with the JOB command.

SUBMANAGER RETENTION TIME

The SUBMANAGER RETENTION TIME number determines how many seconds submanagers should wait while looking for new tasks. The purpose of this field is to reduce the number of JOB commands needed to process a site's tasks. By keeping old Submanagers around to run new tasks, new process creation is significantly reduced.

TASKMAN JOB LIMIT

If there are more active processes on the system than the number stored in the TASKMAN JOB LIMIT field, TaskMan will not create new submanagers to handle tasks. Task processing will be left to existing Submanagers until the number of processes falls back below this number. This number should be slightly lower than the MAX SIGNON ALLOWED field of the VOLUME SET field (Multiple) in the KERNEL SYSTEM PARAMETERS file (#8989.3) so that the system manager still has room to sign on when TaskMan is using its greatest number of partitions.

TASKMAN HANG BETWEEN NEW JOBS

The TASKMAN HANG BETWEEN NEW JOBS field sets a delay between the creation of new submanagers, in seconds. It is useful as a throttle. For systems, this delay spaces out the use of the JOB command to avoid slowing users' response time when the Manager needs to JOB off many new processes in rapid succession.

For systems that create new processes cheaply, this delay is unnecessary. This delay also becomes less important when a high Submanager retention time is used since higher retention times reduce the likelihood that TaskMan will need to create new processes.

Be sure not to combine a high TASKMAN HANG BETWEEN NEW JOBS value with a low SUBMANAGER RETENTION TIME value, since that increases the number of jobs per day TaskMan has to start and can cause busy systems to fall behind. The number should be the lowest value that prevents the problem and can be left blank for systems with efficient JOB commands.

MODE OF TASKMAN

The MODE OF TASKMAN field determines how each CPU (BOX-VOLUME pair entry) should process tasks. You can set it to one of four values:

- **General Processor ("G"):** The G type should be selected when the TASKS (#14.4) and SCHEDULE files are seen by only one Volume Set. For example, VA's Alpha clusters have several CPUs, but each of them runs on the same Volume Set. The Manager on a G type runs tasks created on the same Volume Set, and tasks from any other Volume Set that explicitly requests the G type's Volume Set. The G type sends tasks from another Volume Set that did not explicitly request its Volume Set back to the originating Volume Set, however.

  To transfer tasks to a G type, TaskMan uses extended global references to copy the task to the destination TASKS (#14.4) and SCHEDULE files and then removes the task from its own side. Submanagers started on a G-type processor process tasks in the Partition Waiting List and the Busy Device Waiting List.

- **Print Server ("P"):** The P type should be selected when multiple Volume Sets map to the same TASKS (#14.4) and SCHEDULE files, and you want to run the Manager on the Volume Set/CPU in question.

  Like the G type, the Manager on a P type runs tasks created on the same Volume Set and tasks from any other Volume Set/CPU that explicitly request the P type's Volume Set/CPU. Unlike the G type, however, the P type also runs tasks from other Volume Sets that did not make an explicit Volume Set request. Tasks are transferred to a P type in the same way as to a G type, and Submanagers behave the same.

- **Compute Server ("C"):** The C type should be selected when multiple Volume Sets map to the same TASKS (#14.4) and SCHEDULE files (as with the P type), but when the Volume Set/CPU in question runs users (not tasks). The Manager will not start on a C type. Tasks that explicitly request to run on a C type are transferred to it by being placed in the Link Waiting List; a Submanager is then jobbed across to the C type Volume Set/CPU. Submanagers started on a C type only process tasks in the Link Waiting List for their Volume Set.

- **Other Non-TaskMan ("O"):** Neither the Manager nor the Submanager will run on O types. Tasks sent from or to an O type are rejected.

Because of the field's crucial role in guiding TaskMan's behavior, the field is required.

ENVIRONMENT FOR DCL

The ENVIRONMENT FOR DCL field only has meaning to DSM for OpenVMS and Caché systems. It is set to the OpenVMS username of the DSM environment manager account. Setting it to this username causes the Manager to use %SPAWN to SUBMIT Submanagers to run. This method requires that certain DCL command files exist, along with a TASKMAN OpenVMS user account and directory.

REF: For descriptions of the needed setups, please refer to the "Running TaskMan with a DCL Context" topic in this chapter.

If the field is empty, the Manager starts submanagers with the JOB command instead.

LOAD BALANCE ROUTINE

If you are running multiple Managers (one per node), use this field to set up load balancing between the Managers on each node. It should be set to the name of an extrinsic function that returns a load rating for the node.

REF: For more information on load balancing, please refer to the "Multiple TaskMan Managers and Load Balancing" topic in this chapter.

VOLUME SET File (#14.5)

Figure 21-2. Volume Set Edit option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Task Manager ...                                                  [XUTM MGR]
   Taskman Management Utilities ...                               [XUTM UTIL]
      Edit Taskman Parameters ...                          [XUTM PARAMETER EDIT]
         Volume Set Edit                                          [XUTM VOLUME]
```

TaskMan knows about a system's configuration from the values entered into the VOLUME SET file (#14.5) using the Volume Set Edit option [XUTM VOLUME]. The information stored in this file strongly affects TaskMan's behavior. If you inaccurately describe your system, you will usually notice very quickly as TaskMan begins processing tasks in a consistently incorrect way.

You need to make one entry in this file for each Volume Set that tasks can be queued to or from. These entries are only used when:

- A Manager is running on the Volume Set and *must* look up information about its own environment.
- The Volume Set is a required volume, in which case every Manager *must* check access to it when they start up.
- A task needs to run on the Volume Set, in which case the Manager *must* look up how to get the task there.

The following is what we have set up for FORUM:

21-3. Sample Volume Set setup on FORUM

```
VOLUME SET (14.5)

VOLUME SET: ROU                          INHIBIT LOGONS?: NO
  LINK ACCESS?: NO                       TASKMAN FILES UCI: VAH
  DAYS TO KEEP OLD TASKS: 1              TYPE: GENERAL PURPOSE VOLUME SET
  SIGNON/PRODUCTION VOLUME SET: Yes

UCI ASSOCIATION (14.6)

Empty


TASKMAN SITE PARAMETERS (14.7 )

BOX-VOLUME PAIR: ROU:FORFORUM1           LOG TASKS?: NO
  SUBMANAGER RETENTION TIME: 60          TASKMAN JOB LIMIT: 400
  TASKMAN HANG BETWEEN NEW JOBS: 1       MODE OF TASKMAN: GENERAL PROCESSOR
  OUT OF SERVICE: NO                     MIN SUBMANAGER CNT: 10
  LOAD BALANCE ROUTINE: $$CACHE1()       Auto Delete Tasks: Yes
  Manager Startup Delay: 30
```

The value of ^%ZOSF("VOL") is "FOR".

VOLUME SET

The VOLUME SET field should be set to the name of a Volume Set. It is used in extended global references to reach this Volume Set and can be used in UCI-switching software to move Submanagers between UCIs. If you are unsure how your Volume Sets are named, you can look at the value of ^%ZOSF("VOL") in the Volume Set in question.

TYPE

The TYPE field is used to help resolve where tasks should run; it should properly identify the type of the Volume Set. Typically it should be set to the same value as the MODE OF TASKMAN field for all BOX-VOLUME PAIRs associated with this Volume Set, in the TASKMAN SITE PARAMETERS file (#14.7). This field *must* be filled in for all Volume Sets. This field can have the following values:

- GENERAL PURPOSE VOLUME SET
- PRINT SERVER
- COMPUTE SERVER
- OTHER NON-TASKMAN VOLUME SET

These values have the same meanings as the equivalent values for the MODE OF TASKMAN field in the TASKMAN SITE PARAMETERS file (#14.7), as described previously in the "Where Tasks Run" topic in this chapter. GENERAL PURPOSE VOLUME SET for Volume Sets is the rough equivalent of the MODE OF TASKMAN value GENERAL PROCESSOR for BOX-VOLUME PAIRs.

The FILE SERVER value has been removed; Volume Sets for File Servers should be set to a TYPE of OTHER NON-TASKMAN VOLUME SET.

INHIBIT LOGONS?

Setting the INHIBIT LOGONS? field to YES causes TaskMan to notify Signon that logons are now prohibited and to enter a PAUSE state (stopping processing of tasks) until logons are allowed again. Under ordinary circumstances, system managers should leave this field as NULL or NO.

LINK ACCESS

The LINK ACCESS field should always be set to NULL or YES for the usual kinds of configurations used in VistA. Answer NO to tell TaskMan that this Volume Set cannot be accessed by other Volume Sets using the local network links. Tasks that request a Volume Set without link access are rejected by TaskMan. Such Volume Sets are usually PC workstations linked into the larger network. They can access the core computers, but cannot be accessed themselves.

Some system managers may wish to have a completely isolated computer for testing. They can cut it off from the rest of the world by making entries for all the other Volume Sets and setting this field to NO for each of them. This explicitly tells TaskMan it cannot reach the other Volume Sets.

OUT OF SERVICE? (Obsolete)

This field is obsolete and should only be set to NULL; use the TYPE field).

REQUIRED VOLUME SET? (Obsolete)

This field is obsolete and should only be set to NULL.

TASKMAN FILES UCI

The TASKMAN FILES UCI field should be set to the name of the UCI that holds the ^%ZTSCH and ^%ZTSK globals (usually the manager UCI). The answer should not contain a comma and Volume Set name (e.g., "VAH,PSA"), just the UCI name (e.g., "VAH"). This field is required.

TASKMAN FILES VOLUME SET

The TASKMAN FILES VOLUME SET field should be set to the name of the Volume Set that holds ^%ZTSCH and ^%ZTSK.

A NULL value means this Volume Set holds its own TaskMan files, which is usually the case.

REPLACEMENT VOLUME SET

The REPLACEMENT VOLUME SET field should be set to the name of a Volume Set to which tasks can be sent if this Volume Set is unavailable. A REPLACEMENT VOLUME SET should be essentially equivalent in features to the current one, since tasks that would normally run on the current one will be running on the REPLACEMENT VOLUME SET instead. For many Volume Sets, no other Volume Set is equivalent, and tasks should wait for the link to be restored rather than run elsewhere. If tasks that need this Volume Set should wait, leave the field blank.

DAYS TO KEEP OLD TASKS

This number stored in the DAYS TO KEEP OLD TASKS field is used by the XUTM QCLEAN option to decide which tasks to delete. The decision only affects inactive tasks, as explained in the discussion of the XUTM QCLEAN option. Values in this field cannot inadvertently cause TaskMan to delete scheduled or running tasks. If the field contains no value, XUTM QCLEAN keeps the last seven days' tasks. A value of 0 here keeps your file very clean.

UCI ASSOCIATION File (#14.6)

Figure 21-4. UCI Association Table Edit option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Task Manager ...                                                   [XUTM MGR]
   Taskman Management Utilities ...                                [XUTM UTIL]
      Edit Taskman Parameters ...                          [XUTM PARAMETER EDIT]
         UCI Association Table Edit                                [XUTM UCI]
```

There are two different kinds of entries made into the UCI ASSOCIATION file (#14.6) using the UCI Association Table Edit option [XUTM UCI].

Entries with only the first two fields (FROM UCI and FROM VOLUME SET) filled in identify the valid UCIs on the system for TaskMan. Every kind of VistA site needs one entry of this kind for each UCI to which tasks can be queued or from which tasks are created. Caché sites only need to fill in the first two fields (FROM UCI and FROM VOLUME SET).

REF: For a sample configuration, please refer to the "Sample Configuration: Standardized VA Caché and GT.M Configuration" topic in this chapter.

Entries with all four fields completed collectively build a UCI ASSOCIATION TABLE.

A complete UCI ASSOCIATION TABLE tells TaskMan which UCI to use for tasks that *must* switch Volume Sets in order to reach an I/O device. This situation arises when an I/O device is located in a different Volume Set than the Volume Set where the task was created. In such situations, the Manager knows exactly where the task originated and knows to which Volume Set it *must* be moved, but it does not know in which UCI on that Volume Set it should run the task. A UCI ASSOCIATION TABLE entry supplies the missing information by linking equivalent UCIs together. When building a full UCI ASSOCIATION TABLE, you can omit entries where the UCIs on both Volume Sets have the same name because TaskMan assumes that same-named UCIs are equivalent if no entry is present.

FROM UCI

The FROM UCI field should be set to the name of a UCI on your system. For entries requiring only two fields, this catalogues all the UCIs on your system (and there should be an entry for each). For four-field entries, this represents a UCI from which tasks are being transferred in order to reach their I/O device. Enter only the UCI name (e.g., "VAH"). Do *not* include the Volume Set name (e.g., "VAH,ROU").

FROM VOLUME SET

The FROM VOLUME SET field should be set to the name of the Volume Set that holds the UCI identified in the entry's FROM UCI field. For four-field entries, this represents the Volume Set from which tasks are being transferred in order to reach their I/O device. Every Volume Set listed in this field should be described in the VOLUME SET file (#14.5).

TO VOLUME SET

The TO VOLUME SET field is only used for entries that build a UCI Association Table. For such entries, it should be the name of the Volume Set to which tasks are being transferred in order to reach their I/O devices.

TO UCI

As with TO VOLUME SET, the TO UCI field is only used for entries that build a UCI Association Table. For such entries, it should be the name of the UCI to which tasks will be transferred whenever they *must* be moved from the UCI on the first Volume Set to the second Volume Set in order to reach their I/O devices. As with the From UCI field, the Volume Set name should not be included.

Sample Configuration: Standardized VA Caché and GT.M Configuration

Sites that run Managers on their satellites should make the appropriate TASKMAN SITE PARAMETERS file (#14.7) entries for each satellite and adjust their TaskMan Job Limit to reflect each satellite's individual capacity.

Figure 21-5. VOLUME SET file (#14.5) standardized VA Caché and GT.M configuration

```
VOLUME SET                      You need one entry, for ROU
TYPE                            GENERAL PURPOSE VOLUME SET
INHIBIT LOGONS?                 Blank or NO
LINK ACCESS?                    Blank or NO
OUT OF SERVICE?                 Blank or NO
REQUIRED VOLUME SET?            Blank or NO
TASKMAN FILES UCI               VAH
TASKMAN FILES VOLUME SET        Leave this blank
REPLACEMENT VOLUME SET          Leave this blank
DAYS TO KEEP OLD TASKS          Up to you; can leave blank
SIGNON/PRODUCTION VOLUME SET    Yes
```

Figure 21-6. UCI ASSOCIATION file (#14.6) standardized VA Caché and GT.M configuration

```
FROM UCI                        1 entries: VAH
FROM VOLUME SET                 ROU
TO VOLUME SET                   Blank
TO UCI                          Blank
```

NOTE: Can leave this empty.

Figure 21-7. TASKMAN SITE PARAMETERS file (#14.7) standardized VA Caché and GT.M configuration

```
BOX-VOLUME PAIR                 ROU:FORFORUM1
                                Your answer should be the volume set name
                                concatenated with the ":" concatenated with
                                the name of the Cache Configuration.
LOG TASKS?                      Blank or NO (unless TaskMan is running in a
                                DCL context, in which case set to YES)
DEFAULT TASK PRIORITY           Blank
TASK PARTITION SIZE             Blank
SUBMANAGER RETENTION TIME       60
TASKMAN JOB LIMIT               400 (2-5 lower than Max Signons)
TASKMAN HANG BETWEEN NEW JOBS   1
MODE OF TASKMAN                 GENERAL PROCESSOR
ENVIRONMENT FOR DCL             Blank
OUT OF SERVICE                  Blank
MIN SUBMANAGER CNT              2
LOAD BALANCE ROUTINE            Blank
Auto Delete Tasks               Yes
Manager Startup Delay           30
```

Manager Startup

You may want to configure your system so that, on CPUs where the Manager should run, a Manager starts up every time the CPU starts up. Otherwise, you will need to manually start up the Manager each time you start up those nodes that should run the Manager.

For most sites, only one Manager is needed to cover each environment. Therefore, this topic focuses on starting up only a single Manager.

Neither the Manager nor the Submanagers will start up on a BOX-VOLUME PAIR pair of the wrong type, so pay attention to how you fill in the MODE OF TASKMAN field of the TASKMAN SITE PARAMETERS file (#14.7). If you want the Manager to start, you *must* make sure this field is set to either a Print Server or a General Processor.

Getting the Manager to start up when the system does is accomplished in the VA by the ZSTU routine in the "%SYS" namespace. This routine is provided by Enterprise Product Support (EPS).

Multiple TaskMan Managers and Load Balancing

TaskMan supports the running of multiple Manager processes (but only one Manager process should run per CPU). Running multiple Managers is probably useful only at large sites; at a large site, doing this can enable tasks to be processed more quickly than if only one CPU runs a Manager. An added bonus with multiple Managers is that if one CPU running a Manager becomes unavailable, Manager(s) will still run on the other CPU(s), with no further re-configuration required.

Configuration for Multiple Managers(

Each node that runs a TaskMan Manager *must* have its own entry (BOX-VOLUME PAIR) in the TASKMAN SITE PARAMETERS file (#14.7).

Each CPU *must* share access to a common ^%ZTSK and ^%ZTSCH global, and have access to the same devices. Because of this, all CPUs *must* run the same M implementation.

Starting Up, Pausing, and Stopping Multiple Managers

You will need to start a Manager on each CPU where a Manager should run. Whatever steps you follow to start a single Manager, you will need to repeat for any additional nodes on which you want to run additional Managers.

The options that place TaskMan in a WAIT state and stop TaskMan are not CPU-specific; they will affect all running Managers across the system.

Load Balancing

The LOAD BALANCE ROUTINE field in the TASKMAN SITE PARAMETERS file (#14.7) holds the name of a function that returns a CPU's load rating. This field is only useful if you are running multiple TaskMan Managers.

To use load balancing, enter a routine name in the LOAD BALANCE ROUTINE field for each participating CPU's BOX-VOLUME PAIR entry. Kernel Patch XU*8.0*355 added the following routine for TaskMan load balancing in Caché:

M Implementation	Kernel Load Balance Routine
Caché	$$CACHE2(@com-file,logical-name) in ^ZTM6

If the com-file value is set, that com-file will be run each time TaskMan gets the balance value. The logical-name defaults to "VISTA$METRIC" or uses the value entered. The normal way would be to have $$CACHE2() in the field and use the following two scripts:

- GET_METRIC.COM—This script sets the logical "VISTA$METRIC." It can be run by TaskMan or from the TM$<node> batch queue with the METRIC_SCHEDULE.COM script.
- METRIC_SCHEDULE.COM—This script takes a parameter of the number of seconds to reschedule itself. It defaults to 15 seconds and runs under the "SYSTEM" user.

NOTE: These scripts are located in the same directory as the TaskMan in DCL files.

Use of TaskMan in DCL is optional.

It is all right to run multiple TaskMan Managers without using load balancing; it is also all right if load balancing is set up and only one Manager is running (that Manager automatically takes all jobs itself). If one Manager's CPU has the LOAD BALANCE ROUTINE field filled in, and another running Manager's CPU does not, the Managers acts as if no load balancing is taking place. In short, the only ramification from various combinations of Managers with the LOAD BALANCE ROUTINE field filled in or not is that load balancing might not take place.

The load balancing routine *must* be an extrinsic function that returns a positive value. The CPU with the highest value is the one that will run new tasks.

Cache Algorithms:

- $$Cache2()s:—Returns the TCPIP metric.
- $$Cache1()—Returns the Available jobs.

Each CPU performing load balancing compares its current CPU capacity with that of the other nodes running Managers. If the current CPU has a lower rating than the other CPUs, it puts itself in a BALANCE state and waits to let the other CPUs take up the load before running more jobs itself.

Submanagers will try and wait until there node is running before testing if they should exit.

Monitor Taskman Option

On a system where multiple managers are running, the Monitor Taskman option [XUTM ZTMON] shows a combined view of the operation of multiple managers.

If the current node (the one where you are running the Monitor Taskman option) has a lower rating than other nodes, Monitor TaskMan will show that the current node is in a BALANCE state.

Device Handler's Influence on TaskMan

Certain DEVICE file (#3.5) fields strongly affect TaskMan's behavior. System managers should keep these effects in mind as they configure their systems' devices.

VOLUME SET (CPU)

If the VOLUME SET (CPU) field is not filled in, TaskMan considers this device to be available from all Volume Sets. If it is filled in, TaskMan makes sure all tasks that need this device start on the designated Volume Set.

TYPE

Any tasks that *must* wait for HFS- or SPL-type devices are rescheduled for ten minutes in the future, instead of being placed in a list of waiting tasks. This is because these lists are checked through repeated opens, which may contaminate the output of these two special types of devices.

PRIORITY AT RUN TIME

The PRIORITY AT RUN TIME field overrides the default priority that system managers can establish for tasks using the Site Parameters Edit option on the Edit TaskMan Parameters menu.

TASKMAN PRINT A HEADER PAGE?

If the TASKMAN PRINT A HEADER PAGE? field is set to YES for the device being opened by the Submanager, a header page is printed. The header page distributed with TaskMan is very simple, and system managers can substitute their own locally written header pages. To do this, you *must* rename your header page routine as ^%ZTMSH, the name of the one distributed with TaskMan.

Whenever you install new versions of Kernel, it overwrites ^%ZTMSH with the default copy, so you should maintain your local version by doing the following:

- Keep your local header page routine saved somewhere under a local name.
- After each Kernel install, re-save the locally named copy as ^%ZTMSH.

The following example shows an alternative to the default header page distributed with Kernel:

Figure 21-8. Customized Header Page routine

```
%ZZTMSH     ;SEA/RDS-Local: Sample Header Page ;3/9/92 11:17 ;
            ;;1.0;Local;;
            ;
LOCAL       ;Print The Local Header Page
            ;
B           ;build text lines
            S X1=$P($G(^VA(200,DUZ,0)),U) I X1="" S X1="name unknown"
            S X2=$P($G(^VA(200,DUZ,5)),U,2) I X2="" S X2="unlisted mail stop"
            S X3=$P($G(^VA(200,DUZ,.13)),U,2) I X3="" S X3="unlisted phone number"
            S ZZLINE1=$$FORMAT(" "_X1_"  ("_X2_")  "_X3_"",IOM)
            S ZZLINE2=$$FORMAT(" "_ZTDESC_" ",IOM)
            S ZZLINE3=$$FORMAT(" "_ION_"  "_$$HTE^XLFDT($H)_"",IOM)
            ;
D           ;display each line three times
            F X=1:1:3 W !,ZZLINE1
            W ! F X=1:1:3 W !,ZZLINE2
            W ! F X=1:1:3 W !,ZZLINE3
            Q
            ;
FORMAT(ZZTEXT,ZZIOM)  ;local extrinsic function
            ;input: text to be formatted, and margin width
            ;output: text filled out to margin width -3 with *characters
            N ZZ1,ZZFILLED
            S ZZ1=ZZIOM-3-$L(ZZTEXT)\2
            S $P(ZZFILLED,"*",ZZ1*2+1)=""
            S $P(ZZFILLED,"*",ZZ1+1)=ZZTEXT
            I $L(ZZFILLED)+3-ZZIOM S ZZFILLED=ZZFILLED_"*"
            Q ZZFILLED
```

Figure 21-9. Customized Header Page

```
   ************ XUUSER,ONE  (OIFO)  FTS 555-5555 ************
   ************ XUUSER,ONE  (OIFO)  FTS 555-5555 ************
   ************ XUUSER,ONE  (OIFO)  FTS 555-5555 ************

   ********************** SAMPLE TASK ***********************
   ********************** SAMPLE TASK ***********************
   ********************** SAMPLE TASK ***********************

   *********** LAT DEVICE  Jun 30, 1992@14:34:01 ************
   *********** LAT DEVICE  Jun 30, 1992@14:34:01 ************
   *********** LAT DEVICE  Jun 30, 1992@14:34:01 ************
```

Running TaskMan with a DCL Context

When run from a DCL context, TaskMan runs as an OpenVMS user. The Manager runs as a job that originates from a node-specific OpenVMS batch queue(and, by default, submits new Submanagers to the same queue as needed.

One advantage to running TaskMan from a DCL context is that it allows jobs to be queued to specific CPUs. When a program calls ^%ZTLOAD, it can request that the job run on a specific CPU/node in your cluster (via the ZTCPU input variable). Unless you are running TaskMan in a DCL context (on Caché systems only), this request will probably fail (and possibly cause the task not to run). When TaskMan runs with a DCL context, however, the Manager can submit the job as a new submanager to a given CPU's TaskMan batch queue.

Depending on the %ZTSK and %ZTSCH mapping, multiple Cache environments on the same CPU can each run TaskMan in a DCL context. Although TaskMan in each Cache environment shares the same account, directory, DCL command files, and batch queue, jobs will run in the environment specified in each environment's ENVIRONMENT FOR DCL site parameter.

NOTE: Kernel Patch XU*8.0*355 added the $$CACHE2 routine for TaskMan load balancing and provides support for DCL context in Caché.

Setup for Running TaskMan in a DCL Context in a Cache/VMS Environment

The following steps show you how to set up TaskMan to run in a DCL context in Cache/VMS (see Kernel Patch XU*8.0*355).

NOTE: The following is just an example an will have to be modified for your site. You will need to adjust the UIC [100,20] to match your system and indicate the location of the TaskMan directory.

1. Create TASKMAN that runs the TaskMan jobs:

Figure 21-10. Create TASKMAN

```
ADD TASKMAN/OWNER="SYSTEM MANAGER" -
/ACCOUNT=CACHE -
/PRIV=(NETMBX,TMPMBX) -
/DEFPRIV=(NETMBX,TMPMBX) -
/DEVICE=USER$/DIR=[TASKMAN]/LGICMD=LOGIN.COM -
/FLAGS=(DisCtlY,DisWelcome,DisReport,DisForce_Pwd_Change,DisPwdDic,DisPwdHis) -
/PASS=TASK$MAN/UIC=[100,20]
```

2. Create the TASKMAN directory:

Figure 21-11. Create the TASKMAN directory

```
Define/SYSTEM DHCP$TASKMAN USER$:[TASKMAN]
```

3. Create the system logical name for the directory with the COM files.

NOTE: Be sure to also add to the STARTUP$LOGICALS.COM file

Figure 21-12. Create system logical name for the directory with the COM files

```
Define/SYSTEM DHCP$TASKMAN USER$:[TASKMAN]
```

4. Create the queues, as explained in this manual.

NOTE: Be sure to also add to the STARTUP$DEFINE_QUEUES.COM file.

TaskMan will submit jobs to the queue TM$<node>. Because we use "run loginout" to detach the execution, we do not need a large JOB limit here.

Figure 21-13. Create system logical name for the directory with the COM files

```
INIT/QUEUE/BATCH/OWNER=[TASKMAN] -
/prot=(S:M,O:D,G:R,W:S)/JOB=5/AUTOSTART_ON=isfva2:: TM$isfva2
```

5. Load the following DCL command files into the [TASKMAN] directory:

- GET_METRIC.COM
- LOGIN.COM
- METRIC_SCHEDULE.COM
- ZTM2WDCL.COM
- ZTMS2WDCL.COM

These command files are located in the cache-taskman subdirectory in the Anonymous FTP site.

NOTE: Get the files in ASCII mode.

Figure 21-14. Sample user dialogue to retrieve DCL command files

```
ISC6A2$SET DEF USER$:[TASKMAN]
ISC6A2$ftp ftp.fo-oakland.med.va.gov
220 ISC6A2.ISC-SF.MED.VA.GOV FTP Server (Version 5.3) Ready.
Connected to FTP.FO-OAKLAND.MED.VA.GOV.

Name (FTP.FO-OAKLAND.MED.VA.GOV:fort): anonymous
331 Guest login OK, send ident as password.
Password: xxxxxxxxxx
230 Guest login OK, access restrictions apply.
FTP> cd cache-taskman
FTP> ls
150 Opening data connection for USR$:[ANONYMOUS.CACHE-TASKMAN]*.*;*

GET_METRIC.COM
LOGIN.COM
METRIC_SCHEDULE.COM
ZTM2WDCL.COM
ZTMS2WDCL.COM

FTP> ascii
200 TYPE set to ASCII.
FTP> get get_metric.com
FTP> get login.com
FTP> get metric_schedule.com
FTP> get ztm2wdcl.com
FTP> get ztms2wdcl.com
FTP> bye
221 Goodbye.
```

NOTE: Repeat for each node in the TASKMAN SITE PARAMETERS file (#14.7).

6. Edit TaskMan Parameters:

Figure 21-15. Sample user dialogue to edit TaskMan parameters

```
Select Edit Taskman Parameters Option: SIT <Enter> e Parameters Edit

Select TASKMAN SITE PARAMETERS BOX-VOLUME PAIR: ISC
       1   ISC:ISCISC6A1
       2   ISC:ISCISC6A2
CHOOSE 1-2: 1 <Enter>  ISC:ISCISC6A1
  ...
VAX ENVIROMENT FOR DCL: ISC6A1
  ...
Balance Interval: 30// <Enter>
LOAD BALANCE ROUTINE: $$CACHE2("@DHCP$TASKMAN:GET_METRIC.COM")

LOAD BALANCE ROUTINE: $$CACHE2()
```

namespace:configname

node name

Have TaskMan call the script.

Submit the METRIC_SCHEDULE.COM file

How to Restart TaskMan when Running in a DCL Context

To manually restart TaskMan when TaskMan is running in a DCL context, you can either:

- Sign in as OpenVMS user TASKMAN and DO RESTART^ZTMB.
- Sign in from an OpenVMS account that has the OPER and SYSPRV privileges and DO RESTART^ZTMB. This submits the Manager to run under the username TASKMAN.

In either case, however, do *not* use the Restart TaskMan option in the Kernel menus; it is not compatible with TaskMan in a DCL context.

Figure 21-16. ZTM2WDCL.COM Command File

```
$!----------------------------------------------------------------------
$! ZTM2WDCL.COM - Cache Run Taskman in a DCL Context
$! * KERNEL 8 *
$!
$!  P1 is the Cache config that taskman should start in.
$!  P2 is the namespace that taskman should start in.
$!  P3 = null to START and 1 to RESTART
$!
$! This file is submitted to the queue to run and it
$!  builds and runs the TMP_pid.* files
$!
$! Build the file to run, can't pass arguments with RUN
$ pid = F$GETJPI("","PID")
$ infile="TMP_" + pid + ".ZTM"
$ outfile = "TMP_" + pid + ".log"
$ SAY = "write output"
$!
$ entry="START"
$ if p3 .eq. 1 then entry="RESTART"
$!
$! open and build the input file
$ OPEN/write output 'infile'
$ SAY "$! Taskman temp file to run the Manager"
$ SAY "$! Delete this file if it is not open."
$ SAY "$ set verify"
$ SAY "$ csession ""''p1'"" ""-U"" ""''p2'"" ""''entry'^%ZTM0"""
$ SAY "$ exit"
$ Close output
$!
$! If a log file is needed change _NLA0: to 'outfile
$ name = "ZTMS_" + pid
$  run sys$system:loginout.exe -
       /input='infile -
       /output=_NLA0: -
       /detach /process='name
$!
$!      Wait for loginout to run it then delete the file.
$ wait 00:01
$!
$ del TMP_*.ZTM;1
$ exit
```

Figure 21-17. ZTMS2WDCL.COM Command File

```
$!------------------------------------------------------------------------
$! ZTMS2WDCL.COM - Cache Start Submanager with a DCL Context
$! * KERNEL 8 *
$! p1 is the Cache config name
$! p2 is the namespace to start.
$! p3 is NOT used. (VOL for DSM)
$!
$! This file is submitted to the queue to run and it
$!  builds and runs the TMP_pid file
$!
$! Build the file to run, can't pass arguments with RUN
$ pid = F$GETJPI("","PID")
$ infile = "TMP_" + pid + ".ZTMS"
$ outfile = "TMP_" + pid + ".log"
$ SAY = "write output"
$!
$! open and build the input file
$ OPEN/write output 'infile'
$ SAY "$! Taskman temp file to run a submanager"
$ SAY "$! Delete this file if it is not open."
$ SAY "$ set verify"
$ SAY "$! ''P1' and ''P2'"
$ SAY "$ csession ""''p1'"" ""-U"" ""''p2'"" ""START^%ZTMS"""
$ SAY "$ exit"
$ Close output
$!
$! If a log file is needed change _NLA0: to 'outfile
$ name = "ZTMS_" + pid
$  run sys$system:loginout.exe -
       /input='infile -
       /output=_NLA0: -
       /detach /process='name
$!
$!      Wait for loginout to run it then delete the file.
$ wait 00:01
$!
$ del TMP_*.ZTMS;1
$ exit
```

Figure 21-18. Example of OpenVMS User TASKMAN on ALPHA AXP Systems

```
Username: TASKMAN                          Owner:
Account:                                   UIC:    [50,20] ([DEV,TASKMAN])
CLI:      DCL                              Tables: DCLTABLES
Default:  USER$:[TASKMAN]
LGICMD:   LOGIN
Flags:  DisCtlY Restricted DisWelcome DisReport
Primary days:   Mon Tue Wed Thu Fri
Secondary days:                    Sat Sun
No access restrictions
Expiration:            (none)    Pwdminimum:  6   Login Fails:     0
Pwdlifetime:        180 00:00    Pwdchange:  19-NOV-1992 14:12
Last Login: 20-NOV-1992 10:34 (interactive), 20-NOV-1992 10:44 (non-
interactive)
Maxjobs:         0  Fillm:       300  Bytlm:        64000
Maxacctjobs:     0  Shrfillm:      0  Pbytlm:           0
Maxdetach:       0  BIOlm:       300  JTquota:       4096
Prclm:          14  DIOlm:       900  WSdef:         2048
Prio:            4  ASTlm:       600  WSquo:         4096
Queprio:         0  TQElm:        10  WSextent:     16384
CPU:        (none)  Enqlm:      4096  Pgflquo:     100000
Authorized Privileges:
  CMKRNL TMPMBX OPER NETMBX
Default Privileges:
  CMKRNL TMPMBX OPER NETMBX
```

Figure 21-19. Example of OpenVMS TASKMAN Queue

```
ISC6A1$ SH QUE/FULL TM$ISC6A1

 Batch queue TM$ISC6A1, available, on ISC6A1:
   /BASE_PRIORITY=4 /JOB_LIMIT=50 /OWNER=[DEV,TASKMAN]
 /PROTECTION=(S:E,O:D,G:R,W:W)

ISC6A1$
```

22 TaskMan: System Management—Operation

This chapter describes how to operate TaskMan. This chapter discusses the following:

- TaskMan Management Menu
- Taskman Management Utilities
- Scheduling Options
- Taskman Error Log Menu
- Troubleshooting

TaskMan Management Menu

The Taskman Management menu [XUTMGR] is the main point of entry into the TaskMan options. It contains the following options:

Schedule/Unschedule Options

One-time Option Queue

Taskman Management Utilities ...

List Tasks

Dequeue Tasks

Requeue Tasks

Delete Tasks

Print Options that are Scheduled to run

Cleanup Task List

Print Options Recommended for Queueing

The TaskMan Management Utilities submenu and the scheduling-related options are discussed later in this chapter. The options for listing, dequeuing, requeuing, deleting, and cleaning up tasks are discussed first.

List Tasks Option

Figure 22-1. List Tasks Option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   List Tasks                                                      [XUTM INQ]
```

Beginning with Kernel V. 8.0, the TASKS file (#14.4) (in ^%ZTSK) is VA FileMan compatible (i.e., you can use VA FileMan to print out information about a task). However, the List Tasks option [XUTM INQ] also provides a way to examine tasks in the TASKS file (#14.4). The List Tasks option allows you to choose between several useful ways of selecting tasks. When you choose this option, it presents you with the following menu:

Figure 22-2. List Tasks option submenu options

```
                    List Tasks Option

                         All your tasks.
                         Your future tasks.
                         Every task.
                         List of tasks.
                         Unsuccessful tasks.
                         Future tasks.
                         Tasks waiting for a device.
                         Running tasks.

                    Select Type Of Listing:
```

Several choices only appear on the list when there are tasks in those collections to be displayed. Remember, the TASKS file (#14.4) can be Volume Set/CPU-specific. This means that the option can only display tasks from the TASKS file (#14.4) on the current Volume Set/CPU.

Holders of the ZTMQ security key see a slightly different list of selections. Instead of "All your tasks" and "Your future tasks" they see "All of one user's tasks" and "One user's future tasks." These two selections are generic versions of those available to normal users. They allow the holder to see any user's tasks and start by prompting the holder for the user whose tasks should be shown. Other than that, they are identical to the selections used by normal users.

Although each submenu option choice shows a different set of tasks, the format for the output is the same. Here is a sample display from the All your tasks suboption:

Figure 22-3. All your tasks suboption—Sample of TaskMan tasks running

```
All tasks that you created...

2572: ALIVE^XINDEX, XINDEX of 1 routine.  Device QMS-17P.  VAH,KXX.
      From TODAY at 10:55,  By you.  Scheduled for TODAY at 12:05

End of listing.  Press RETURN to continue:
```

In the upper left-hand corner of each entry is the task number. What follows the task number is either an option name (e.g., XUTM QCLEAN) or a routine entry point (e.g., ERROR^ZTMZT) depending on whether the task was a queued routine or a queued option. This is generally followed by a description of the task. The device to which the task was queued (if any), along with the account in which the task was/is scheduled to run, complete the first line. The next line contains the time the task was created followed by an identification of the creator. In the case of tasks that requeue themselves, this date and time represents when the task was last requeued.

When the creator's DUZ number is not listed in the NEW PERSON file (#200), the phrase "USER #" followed by the DUZ is substituted. Finally, the status of the task is shown.

REF: For a list and description of the status messages, please refer to the "Troubleshooting" topic in this chapter.

Each of these submenu options are described in the topics that follow.

All your tasks Option

The All your tasks option (see Figure 22-3) displays every task in the TASKS file (#14.4) on the current Volume Set/CPU that you created. If you have no tasks scheduled, the option gives you the message "You have no tasks in this Volume Set's TASKS file."

Your future tasks Option

The Your future tasks option displays those tasks you created that are currently scheduled to run. If there are none, the option tells you.

"Every task" lists every task in the TASKS file (#14.4).

List of tasks Option

The List of tasks option allows you to list one or more tasks by task number. You can specify individual tasks separated by commas along with ranges of tasks using a hyphen.

Unsuccessful tasks Option

The Unsuccessful tasks option lists three kinds of tasks:

- Rejected by the Manager's validation process.
- Encountered an error while they were running.
- Unscheduled through the Dequeue Tasks option.

Future tasks Option

The Future tasks option shows all tasks that are in the Schedule List or the Waiting List. It does not show the tasks that are in the Job List. In other words, it shows all tasks that are scheduled to run but not those that are currently being run or those that are ready to be run. "Future Tasks" is not offered by the List Tasks option if the Schedule List and Waiting List are empty (an unlikely occurrence at most sites).

Tasks waiting for a device Option

The Tasks waiting for a device option shows just the Waiting List, which can be a useful way of isolating problem printers. If there are no tasks currently waiting for output devices to become available, the List Tasks option will not show this choice.

Running tasks Option

The Running tasks option shows tasks that are currently running.

REF: For a discussion of how TaskMan knows a task is running, please refer to the "Troubleshooting" topic in this chapter.

Dequeue Tasks Option

Figure 22-4. Dequeue Tasks option

```
SYSTEMS MANAGER MENU ...                                          [EVE]
Taskman Management ...                                        [XUTM MGR]
   Dequeue Tasks                                               [XUTM DQ]
```

The Dequeue Tasks option [XUTM DQ] allows you to unschedule a task so that the task still exists in the TASKS file (#14.4) but is no longer in the Schedule, Waiting, or Job List. The process of unscheduling a task is called "dequeuing". This option allows you to dequeue any one task or range of tasks. A task that you dequeue has a status of NOT QUEUED in a List Tasks display.

The option first prompts you for the task number. Entering one question mark ("**?**") gets you a short explanatory message, but entering two question marks ("**??**") puts you in the List Tasks option to find the task you are interested in dequeuing. When you leave the List Tasks option, you automatically return to the task number prompt.

If you enter the number of a nonexistent task, List Tasks tells you and then prompts you for another task number. If you enter the number of a task that does exist, the option displays the task and asks you if you are sure. Answering NO returns you to the task number prompt, whereas a YES dequeues the task and then returns you to the task number prompt.

You can also enter a list of tasks to be dequeued. The list can include single tasks separated by commas and ranges of tasks consisting of two numbers separated by a hyphen. After you enter the list, you are asked if you want to know the actual number of tasks in the list. You are then asked if you want a display of the actual tasks that are about to be dequeued.

Only holders of the ZTMQ security key can dequeue any task. Others can only dequeue their own tasks as identified by their DUZ.

Requeue Tasks Option

Figure 22-5. Requeue Tasks option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Requeue Tasks                                                   [XUTM REQ]
```

A benefit of the Dequeue Tasks option is that it is completely non-destructive. If you dequeue a task and subsequently change your mind, you can use the Requeue Tasks option [XUTM REQ] to requeue the task exactly the way that it was. You can also use this option to change some of the details of a task that is already queued.

As with XUTM DQ, you are first prompted for a Task Number with the same help available. Here, you can only enter a single task, *not* a range. The task is then displayed, and you are asked for a new run time with the default being either the original or current run time (whichever applies). The next question is "Do you wish to requeue this task to a device?", with the default depending on whether the task originally requested an output device. If you answer YES, the option asks you to specify an output device using the original output device (if there was one) as a default. The option also allows you to adjust the task's priority.

The task is requeued according to your specifications. Requeuing involves completely dequeuing the task so that your task does *not* run twice, making the changes you requested, and placing the task back on the Schedule List. Notice that the task is not dequeued until after you specify the changes you want to make. If you want to modify a task that may start running soon, it is usually a good idea to dequeue it first.

The ZTMQ security key affects this option in two ways

- Users who do not hold the security key are limited to requeuing only their own tasks.
- Users are not prompted to change the priority.

Delete Tasks Option

Figure 22-6. Delete Tasks option

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Taskman Management ...                                                  [XUTM MGR]
   Delete Tasks                                                         [XUTM DEL]
```

The Delete Tasks option [XUTM DEL] has the same structure as the Dequeue Tasks option. The only difference is that where dequeuing a task just removes it from the lists (unschedules it), the Delete Tasks option also deletes the task from the TASKS file (#14.4). When you have deleted a task, there is no reference to that task anywhere in TaskMan's files.

Only holders of the ZTMQ security key can delete any task. Others can only delete their own tasks as identified by their DUZ.

Cleanup Task List Option

Figure 22-7. Cleanup Task List option

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Taskman Management ...                                                  [XUTM MGR]
   Cleanup Task List                                                [XUTM TL CLEAN]
```

You can use the Cleanup Task List option [XUTM TL CLEAN] to remove a task entry from a task list for a job that is no longer running. This might happen when a process is forcibly exited, but TaskMan still believes the task is running. You can use this option to tell TaskMan which tasks you forcibly exited. TaskMan then removes those tasks from its list of running tasks.

Taskman Management Utilities

A submenu on the Taskman Management menu, called TaskMan Management Utilities menu [XUTM UTIL], provides several options to set up, monitor, and modify the TaskMan environment.

The Taskman Management Utilities menu contains the following options:

Monitor Taskman

Check Taskman's Environment

Edit Taskman Parameters ...

Restart Task Manager

Place Taskman in a WAIT State

Remove Taskman from WAIT State

Stop Task Manager

Taskman Error Log ...

Clean Task File

SYNC flag file control

These options are discussed in the topics that follow.

Monitor Taskman Option

Figure 22-8. Monitor Taskman option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                                [XUTM UTIL]
      Monitor Taskman                                             [XUTM ZTMON]
```

The Monitor Taskman option [XUTM ZTMON] gives you a screen of information about the current state of TaskMan and offers you several ways to get more information. The monitor focuses on the current state of the Manager itself and on the contents of the SCHEDULE file.

As you use this option, you will acquire an intuitive understanding of how these lists should look and behave when your system is healthy. Spending the time using this option to get that intuition will save you troubleshooting time by helping you to notice problems sooner.

RUN Node

The first section of the Monitor TaskMan screen reports whether the Manager is currently running on your machine, and if so, whether or not it is being delayed. This is accomplished by comparing TaskMan's RUN Node to the M $HOROLOG variable. Under normal circumstances they should be within 15 seconds of each other, though certain conditions can cause a difference of up to two minutes. Any difference greater than that, however, is a sign that the Manager is being delayed, typically by a problematic device or a recurring error. Of course, the Manager is also likely to fall behind if the system is saturated to the point where all of the jobs on the system are slow. The last line of the first section evaluates the difference and guesses at the Manager's current condition. The $HOROLOG values are translated into an external format for your convenience in understanding the values.

Figure 22-9. Sample Monitor TaskMan screen

```
Checking TaskMan.  Current $H=54180,45147  (MAY 04, 1989 @12:32:27)
                     RUN NODE=54180,45145  (MAY 04, 1989 @12:32:25)

  TaskMan is current.

  Checking the Status List:
       TaskMan job 4 status 54180,45145^RUN^Main Loop.
       There are 3 idle submanagers

  Checking the Schedule List:
       TaskMan has 29 tasks in the Schedule List.
       None of them are overdue.

  Checking the IO Lists:  Last TM scan: 54180,45146^_TNA9995:
   Device: _TNA9995: is not available, and there are 7 tasks waiting.

  Checking the Job List:
       There are no tasks waiting for partitions.
       For KDE:ISC6V2 there are 2 tasks.   Not responding


  Checking the Task List:
       There are 5 tasks currently running.

  Enter monitor action: UPDATE//
```

Status List

The Status List is where each Manager periodically reports its current status. The job number of the Manager is reported both for ease of location on a system status report and also to distinguish between multiple Managers (if there are more than one). Under normal circumstances, the Manager removes its entry from the Status List when it shuts down, but if a Manager stops abnormally (e.g., RJD or FORCEX) its entry is usually left on the list. The list is updated and cleaned out whenever a new Manager is started or restarted.

The status of a Manager consists of three parts:

- Date and time—This date and time should equal the RUN node's date and time, and like that node, it should be close to the current $HOROLOG.
- Manager's state.
- Description of special circumstances.

The Manager can be in one of five states at any given time:

- BALANCE
- ERROR
- PAUSE
- RUN
- WAIT

RUN is the normal state, with a description of "Main Loop."

The Manager's status is the most important piece of information the monitor gives, and it should always be the first thing checked when troubleshooting problems.

REF: For a detailed list and description of the possible state messages, please refer to the "Troubleshooting" topic in this chapter.

Schedule List

The Schedule List always shows the number of tasks currently scheduled to run and checks the times for which they are scheduled to determine whether any of them should already have started. When many tasks are queued to run at the same time, it is not unusual for the Manager to be a little late in sending off the last few.

When most of the tasks on the Schedule List are overdue, however, the Manager is probably having problems keeping up. This is *not* a normal condition. If the problem is not a recurring error or a difficult output device, the most likely culprit is your default setup in the TASKMAN SITE PARAMETERS file (#14.7). Another possible problem is that TaskMan is trapping many errors or trying to access a very slow link between Volume Sets. If the problem is error trapping, the Status List should regularly show the Manager in an Error state. Also, remember that if the machine is saturated, all of the jobs on the system, including the Manager, will run slowly.

IO List

The IO List first shows the last time ($H) a Submanager checked the list and the last device checked. The check generally shows how many tasks are waiting for each device in the IO List. The occasional remark "Allocated" means that a Submanager has already noticed that the device is available and has allocated the device to a task using the Device Allocation List. Devices should only be allocated for a short time before the Submanager opens the device, making it unavailable.

Understanding how the IO List works can make this particular check very useful. Submanagers handle the Device IO Lists. Unusual behavior in these lists usually points to device or Submanager problems.

There are three fundamental things to look for with this check:

- When a device becomes available—The Submanagers should notice and start a task running on that device. If the Submanagers do not do this, it is probably time to start looking for problems with the Submanagers.
- When a device is allocated—A Submanager should quickly make it unavailable. If this fails to occur, the Submanagers may be having problems. There can be extenuating circumstances (e.g., the system being very slow) that explain these occurrences.
- When many tasks are backed up waiting for the same device—Sometimes it is just because that device is busy. However, sometimes the device is off-line or out of paper.

Job List

The Job List is where tasks wait for partitions, so if many tasks are backed up here you know the Submanagers are not picking them up. This can be caused by any of the following:

- A slow system.
- TaskMan reaching its job limit.
- TaskMan assigning tasks a priority that is too low for them to run.

Systems that are too busy will back up in this list, not the Schedule List. The Compute Server Job List is checked here and will let you know about tasks waiting to run on other CPUs and if the Submanagers are not starting.

Task List

The Task List is where TaskMan keeps track of the tasks it has started running. Entries are set into this list when the Submanagers start their tasks and are cleared when the tasks quit or cause errors to be trapped. KILLing a task by forcing its process to exit in the middle of execution (using such vendor-specific tools as RJD, RESJOB, FORCEX, KILLJOB, etc.) does not give the Submanager a chance to clear the task from the Task list, so the Task List can become inaccurate. If you frequently KILL jobs but want to keep your Task List accurate, you will need to manually remove the obsolete entries. The exit action of the KILL off a users' job option [XURESJOB] will help you identify and remove from the list of running tasks those you have forcibly exited.

Monitor Action Prompt

After summarizing the status of the Manager and the principal lists of the SCHEDULE file, the monitor offers you a choice of actions. They are displayed if you enter a single question mark ("**?**") at the "Enter monitor action:" prompt:

Figure 22-10. TaskMan monitor actions

```
Enter <RET> to update the monitor screen.
Enter ^ to exit the monitor.
Enter E to inspect the TaskMan Error File.
Enter S to see a system status listing.
Enter ? to see this message.
Enter ?? to inspect the tasks in the monitor's lists.
```

These actions (see Figure 22-10) attempt to bring together those utilities used most often in response to seeing a monitor screen. Updating is the most commonly used choice since you often want to watch how the lists change over time. The TASKMAN ERROR file needs to be easily accessible, not only in case the Manager enters an Error state, but also if a task that should take a long time to run leaves the Job List but never shows up in the Task List. This usually means the task hit an error and quit, which can be confirmed or disproved by a quick glance at the Taskman Error Log. The System Status Report can be used to verify that tasks, Submanagers, and the Manager are indeed running as the monitor suggests.

Some actions at the Monitor Action prompt are not accessible when monitoring TaskMan from the manager's account (using the direct-mode utility D ^ZTMON).

Inspecting the Tasks in the Monitor's Lists

If you are in a non-library account, you can directly inspect the contents of the various lists. Do this by entering two question marks ("**??**") at the "Enter monitor action:" prompt. You get the following list of choices:

Figure 22-11. Options for inspecting tasks in the TaskMan Monitor's Lists

```
  Help For Monitor Taskman Option

       Schedule List.
       Waiting Lists.
       One Waiting List.
       Job List.
       Task List.
       Link Lists.

  Select Type Of Listing:
```

These listings use the same format as that of the List Tasks option, and show you the contents of the lists at the time you look at them. The One Waiting List listing prompts you to select a device, and the help for that prompt lets you see those devices that have tasks waiting. Many of these lists change very quickly.

Thus, it is not unusual to enter the help with the intention of seeing the task that was shown by the main screen to be in the Job List, only to be informed by the help software that the Job List is now empty. These kinds of experiences are simply part of troubleshooting TaskMan.

While these monitor actions are useful, there are still times when you *must* leave the monitor to follow up on information you saw there. For example, you may want to check the list of unsuccessful tasks or to list a specific task; both these actions require using the List Tasks option.

Taken as a whole, the checks that make up the monitor can save you a lot of time in trying to evaluate TaskMan's status. The example shown in Figure 22-9 is of a healthy, and not very busy, Manager. Monitors at sites usually show considerably more activity, especially in the Waiting Lists.

Check Taskman's Environment Option

Figure 22-12. Check Taskman's Environment option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                                [XUTM UTIL]
      Check Taskman's Environment                                [XUTM CHECK ENV]
```

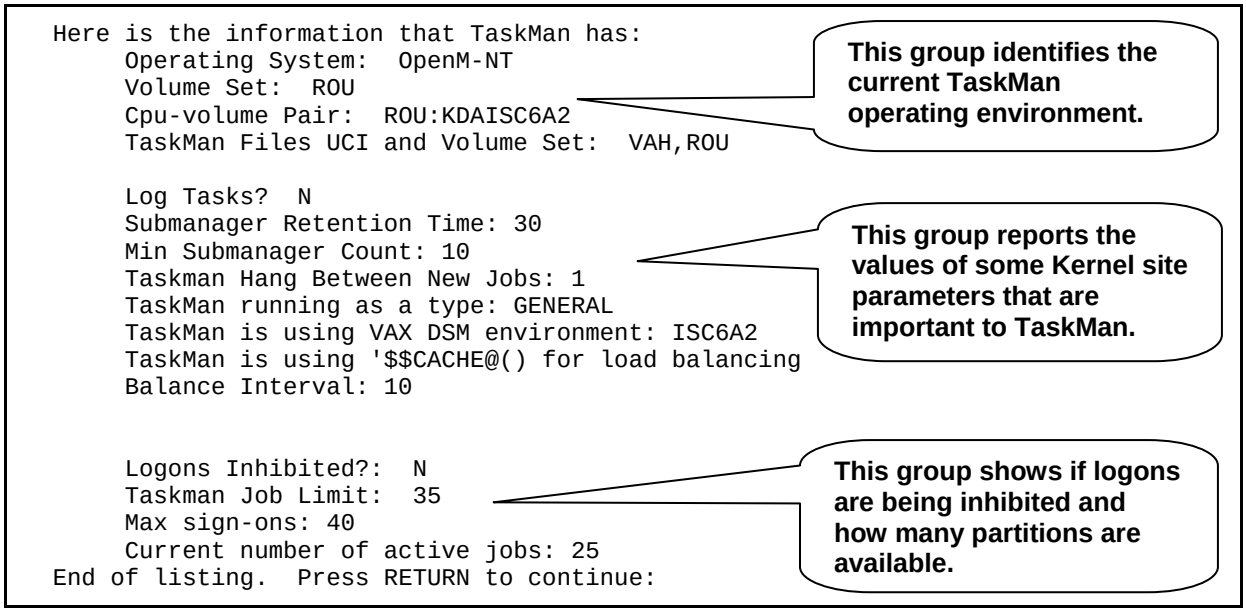
The Check Taskman's Environment option [XUTM CHECK ENV] presents two screens of information about TaskMan's environment on the current CPU. The first screen (see Figure 22-13) performs all of the checks that the Manager does whenever it starts, restarts, or encounters an error. The second screen (see Figure 22-14) shows what values the Manager is using for its definition variables. This information can be very useful in pinpointing startup problems, in verifying that the Manager is using the information you want it to use and in getting a general feel for how you have defined your system's task management.

Figure 22-13. Check TaskMan's Environment option—First screen

```
Checking Task Manager's Environment.

Checking TaskMan's globals...
     ^%ZTSCH is defined!
     ^%ZTSK is defined!
     ^%ZTSK(0) is defined!
     ^%ZIS(14.5,0) is defined!
     ^%ZIS(14.6,0) is defined!
     ^%ZIS(14.7,0) is defined!

Checking the ^%ZOSF nodes required by TaskMan...
     All ^%ZOSF nodes required by TaskMan are defined!

Checking the links to the required volume sets...
     There are no volume sets whose links are required!

Checks completed...TaskMan's environment is okay!

Press RETURN to continue or '^' to exit:
```

This first screen (see Figure 22-13) goes through each step that the Manager goes through when it starts or restarts and reports the results. If your Manager is failing to start, this screen should identify any problem with the environment.

Figure 22-14. Check TaskMan's Environment option—Second screen



The second screen (see Figure 22-14) reports more information about the current TaskMan environment. The first group of four items identifies the current TaskMan operating environment. The next group of items reports the values of some Kernel site parameters that are important to TaskMan.

REF: These parameters, as well as all the other parameters that TaskMan uses, are described in detail in the "TASKMAN SITE PARAMETERS File (#14.7)" topic in the "TaskMan: System Management—Configuration" chapter in this manual.

The last four items show if logons are being inhibited and how many partitions TaskMan currently has to work with. These values show how busy your system is, as well as how busy it can become. Their importance is also described in the discussion of parameters.

Restart Task Manager Option

Figure 22-15. Restart Task Manager option

```
SYSTEMS MANAGER MENU ...                                          [EVE]
Taskman Management ...                                        [XUTM MGR]
   Taskman Management Utilities ...                           [XUTM UTIL]
      Restart Task Manager                                   [XUTM RESTART]
```

The Manager generally starts automatically when your system comes up. If the Manager crashes or is stopped, you can use the Restart Task Manager option [XUTM RESTART] to restart it. The option first checks the RUN node and calculates whether it thinks the Manager is currently running. If this option believes the Manager is running, it will ask you if you are sure you want to restart another TaskMan; you *must* answer YES to start the Manager. If XUTM RESTART thinks the Manager has stopped, it will ask

you for confirmation before jobbing out a new Manager. If XUTM RESTART believes the Manager to be active when you know for sure that it has failed, you can invoke XUTM STOP to prove to XUTM RESTART that the Manager really has stopped. Then you will be able to restart it.

Place Taskman in a WAIT State Option

Figure 22-16. Place Taskman in a WAIT State option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                               [XUTM UTIL]
      Place Taskman in a WAIT State                               [XUTM WAIT]
```

The WAIT state (as described in the "Troubleshooting" topic in this chapter) is a condition in which the Manager does nothing but wait for you to release it. Putting a stop to the Manager's activities without actually shutting down the Manager can often be very useful. For example, with the Manager in a WAIT state, you can look at the tasks after they are queued but before the Manager has a chance to validate them. This can help you isolate problems caused by the queuing process from those caused by the validation process. Another time you may want to create a WAIT state is before restarting a manager that has stopped. This prevents the Manager from processing any tasks when it first starts up; the Manager will check out its environment and then WAIT for your command to continue. The Place Taskman in a WAIT State option [XUTM WAIT] gives you a way to switch the Manager's activities on and off without having to completely shut down and restart the Manager.

When you select the XUTM WAIT option, you are also prompted with the question "Should active submanagers shut down after finishing their current tasks?". If you answer YES, the Submanagers on the current Volume Set/CPU will quit when they finish a task instead of recycling. If you answer NO, the Manager enters a WAIT state and the Submanagers continue with their business. If you also want to keep the Submanagers from searching the Waiting List and the Job List for tasks, you need to explicitly say so at this prompt. This inhibition of the Submanagers' recycling remains in effect either until you remove the WAIT state or until a new Manager starts or restarts, whichever comes first.

Remove Taskman from WAIT State Option

Figure 22-17. Remove Taskman from WAIT State option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                               [XUTM UTIL]
      Remove Taskman from WAIT State                               [XUTM RUN]
```

The Remove Taskman from WAIT State option [XUTM RUN] simply undoes the effects of XUTM WAIT, allowing the Manager to process tasks and allowing the Submanagers to recycle (if recycling had been inhibited).

Stop Task Manager Option

Figure 22-18. Stop Task Manager option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                               [XUTM UTIL]
      Stop Task Manager                                           [XUTM STOP]
```

The Stop Task Manager option [XUTM STOP] gives you a clean way to stop the Manager from within the menu system. This option also asks if you want the Submanagers to shut down when they finish what they are doing.

NOTE: The WAIT state takes precedence. While the Manager is in a WAIT state, not even XUTM STOP affects it until after you invoke XUTM RUN to release it from the WAIT state; after it is released, it shuts down.

This option should always be used to shut down TaskMan, rather than simply KILLing the TaskMan process, which can leave the TaskMan globals in an improper state and even lose tasks.

SYNC flag file control Option

Figure 22-19. SYNC flag file control option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                             [XUTM MGR]
   Taskman Management Utilities ...                               [XUTM UTIL]
      SYNC flag file control                                      [XUTM SYNC]
```

With the SYNC flag file control option [XUTM SYNC], for any SYNC FLAG entry, you can remove it from the file and delete all waiting tasks with the same SYNC FLAG. You can also choose START NEXT, which resumes running the series of tasks associated with that SYNC FLAG. This is useful when one task in a series of tasks that is synchronized with SYNC FLAG fails.

Clean Task File Option

The TASKS file (#14.4) grows every time a new task is queued. While the SAC requires applications to delete their tasks' entries when they complete, it is possible that older applications may not do this. Other tasks abort with errors; still others are rejected. The result is that ^%ZTSK is always growing. Options are available that clean up the ^%ZTSK global.

Figure 22-20. Clean Task File option

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Taskman Management ...                                                  [XUTM MGR]
   Taskman Management Utilities ...                                    [XUTM UTIL]
      Clean Task File                                                 [XUTM CLEAN]
```

In unusual circumstances, you may need to clean the ^%ZTSK global manually. Kernel provides the called Queuable Task Log Clean Up option to regularly clean up the TASKS file (#14.4) in the background.

Only rarely will you not be able to rely on the queued cleanup to perform this function. However, when necessary, you can use the interactive Clean Task File option [XUTM CLEAN]. First, XUTM CLEAN asks you if you are sure you want to clean out the old entries from the TASKS file (#14.4). If you respond that you are, the option asks you how far back you want to keep old entries. The default is to keep old entries going back a week and to delete the older ones. After you provide this value, the option queues a task to do the cleanup. XUTM CLEAN cannot be queued.

Queuable Task Log Clean Up Option

The Queuable Task Log Clean Up option [XUTM QCLEAN], resides on the ZTMQUEUABLE OPTIONS menu. This option allows you to purge all of the entries for tasks that are no longer queued (for whatever reason) and to purge the Taskman Error Log. It is very useful to be able to queue the cleanup to run automatically each night; XUTM QCLEAN has been distributed to provide this feature. XUTM QCLEAN should not be run interactively; indeed, it is not available from any of TaskMan's menus. To queue this option, use Schedule/Unschedule Options to queue it to run.

The date XUTM QCLEAN starts purging the TASKS file (#14.4) is controlled by the DAYS TO KEEP OLD TASKS parameter in the VOLUME SET file (#14.5). A value of seven days is recommended. XUTM QCLEAN does not need an output device; therefore, you can leave that field blank. Once set up, the task automatically runs periodically, cleaning out inactive task entries that are older than the time period specified in the DAYS TO KEEP OLD TASKS parameter. If you want to run this on all of your machines, create an entry in the OPTION SCHEDULING file (#19.2) for each machine on which you want to run it.

Scheduling Options

TaskMan lets you, the site manager, schedule options that run regularly as tasks. Menu Manager and TaskMan work together to give you this ability. All you have to do is tell TaskMan which option you want to queue and how you want to queue it.

Which Options to Queue

The first requirement for queuing regards the option type. Only the run, print, and action types of options can be queued. The second requirement is that the option (if a run or action type) *must* not involve user input! There is nothing to prevent you from queuing an option of the wrong type or from queuing one that prompts the user for input, but doing so results in a failed task. You *must* be conscious of the nature of the task when you consider creating one that performs an option. If the option itself will not run in the background, then queuing it is pointless. Even options that themselves queue tasks probably cannot be queued, because most of these ask the user for an output device or a run time.

Software applications can make recommendations for scheduling of options. This is a great help to site managers.

REF: Recommendations for scheduling Kernel options can be found in the *Kernel Installation Guide* and the *Kernel Technical Manual*.

PARENT OF QUEUABLE OPTIONS Menu

Some options that are intended to be queued are not intended to be run interactively, so placing such options on a user menu could cause problems. The PARENT OF QUEUABLE OPTIONS menu [ZTMQUEUABLE OPTIONS], a menu-type option, has no parent in the menu tree and is intended to be used as the parent of all such options.

Printing Options Recommended to Run and Scheduled to Run

Figure 22-21. Print Options Recommended for Queueing and Print Options that are Scheduled to run options

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                            [XUTM MGR]
  Print Options Recommended for Queueing          [XUTM BACKGROUND RECOMMENDED]
  Print Options that are Scheduled to run               [XUTM BACKGROUND PRINT]
```

The Print Options Recommended for Queueing option [XUTM BACKGROUND RECOMMENDED] displays all options in the OPTION SCHEDULING file (#19.2) that are recommended for scheduling by the option's developer.

The Print Options that are Scheduled to run option [XUTM BACKGROUND PRINT] lists all currently scheduled options on your system. By comparing these two reports, you can see if any options recommended for scheduling are not scheduled on your system (and vice-versa).

Schedule/Unschedule Options

Figure 22-22. Schedule/Unschedule Options option

```
SYSTEMS MANAGER MENU ...                                              [EVE]
Taskman Management ...                                            [XUTM MGR]
   Schedule/Unschedule Options                                 [XUTM SCHEDULE]
```

The Schedule/Unschedule Options option [XUTM SCHEDULE] is a straightforward VA ScreenMan edit option, and allows you to schedule and unschedule options. After you select the option to schedule, you are prompted for information about the task you want to set up. You can edit the following fields in the OPTION SCHEDULING file (#19.2):

- QUEUED TO RUN AT WHAT TIME
- DEVICE FOR QUEUED JOB OUTPUT
- QUEUED TO RUN ON VOLUME SET
- RESCHEDULE FREQUENCY
- TASK PARAMETERS
- SPECIAL QUEUEING

The cross-references on these fields make calls to TaskMan's API to update the TASKS file (#14.4) and ^%ZTSCH.

NOTE: In order to queue a task, its SCHEDULING RECOMMENDED field (#209) in the OPTION file (#19) *must* be set to **YES**.

QUEUED TO RUN AT WHAT TIME

To queue an option, select the option and enter a time at least two minutes in the future into the QUEUED TO RUN AT WHAT TIME field. When you enter a time (and date) for the task to run, the task is immediately put on the Schedule List for that time.

How to Delete a Regularly Scheduled Task

Deleting a scheduled task is as simple as entering the at-sign ("@") at the QUEUED TO RUN AT WHAT TIME field. TaskMan then searches the current TASKS file (#14.4) for the task that corresponds to the entry in the OPTION SCHEDULING file (#19.2) and deletes it.

If your system has multiple copies of the TaskMan globals, you *must* use Schedule/Unschedule Options on the same Volume Set/CPU where your task originated, when you delete the task. Otherwise, the future

task in the TASKS file will *not* be found (and deleted) when you enter an at-sign ("**@**") in the QUEUED TO RUN AT WHAT TIME field.

How to Requeue a Regularly Scheduled Task

Requeuing merely involves placing a new value in the QUEUED TO RUN AT WHAT TIME field. When you do this, the currently scheduled task is deleted (exactly as described above when deleting a scheduled task). Then, a new task is created at the new time to replace the previously scheduled task.

If your system has multiple copies of the TaskMan globals, you *must* use Schedule/Unschedule Options on the same Volume Set/CPU where your task originated, when you requeue the a task. Otherwise, the existing future task in the TASKS file will not be found (and deleted) when you enter a new time in the QUEUED TO RUN AT WHAT TIME field.

DEVICE FOR QUEUED JOB OUTPUT

This field is where you can give the task an output device. For print (Report) type options this is obviously mandatory; for run or action types you need to consider if the option needs an output device. Modifying this value for an already-scheduled task merely causes a direct change to the currently scheduled task.

Tasks with an output device are assigned a process name of "Task ####" where "####" is the task number; tasks with no output device are assigned a process name of "BTask ####" (with B meaning background).

QUEUED TO RUN ON VOLUME SET

This field lets you designate a Volume Set or CPU for the task other than your current one. This field is only useful for options that do not have a device selected because most devices are tied to a CPU, and thus, the task *must* run on the CPU that has that device.

Modifying this value for an already-scheduled task merely causes a direct change to the currently scheduled task.

Running a task on each CPU for a given option may at times be useful (consider XQBUILDTREEQUE). In such cases, make multiple entries in the OPTION SCHEDULING file (#19.2), and use the QUEUED TO RUN ON VOLUME SET field to specify the Volume Set/CPU where each scheduled task should run.

If you leave the DEVICE FOR QUEUED JOB OUTPUT field blank, the task that performs the option runs without a device (or tries to). If you also leave the QUEUED TO RUN ON VOLUME SET field blank, the task runs on the current CPU without a device. If you fill in both fields, TaskMan uses the value of the QUEUED TO RUN ON VOLUME SET field (unless overridden by the VOLUME SET [CPU] field in the DEVICE file [#3.5] entry of the selected device).

RESCHEDULE FREQUENCY

Whenever a task starts running an option, it looks to see what is in the RESCHEDULE FREQUENCY field. If the field is blank, the option does not reschedule itself. If you have filled in this field, the task uses the value you placed in the field to figure out when you want it to run next. Then it updates the QUEUED TO RUN AT WHAT TIME field to reflect the new scheduled time. When this field is updated, the next task in the sequence is scheduled.

If you change the existing value in the RESCHEDULE FREQUENCY field, the new increment is used beginning after the next time the option runs.

There are several formats you can use in this field:

- Every "**n**" seconds.
- Hours.
- Days.
- months (incremental).
- A particular day of the month.
- A list of times every "**n**" months.

REF: For a list of the code formats for the RESCHEDULE FREQUENCY field, please refer to the "SPECIAL QUEUEING" topic that follows.

For the incremental scheduling frequencies (every n seconds, hours, days, or months), the increment is added to the scheduled date and time in the QUEUED TO RUN AT WHAT TIME field to determine when the task should run next. As of Kernel V. 8.0, if the incremented time is in the past, however, TaskMan keeps adding the increment until a future time is reached, only then does it reschedule the task.

TASK PARAMETERS

The TASK PARAMETERS field provides a way to pass data to a scheduled option. TASK PARAMETERS holds a string that is passed to scheduled jobs through the ZTQPARAM variable. Ideally, the developer of an option that uses the TASK PARAMETERS string should describe the format and meaning of the string in the option's DESCRIPTION field.

SPECIAL QUEUEING

Use the SPECIAL QUEUEING field (#9) in the OPTION SCHEDULING file (#19.2) to designate which option is scheduled to be run by TaskMan.

NOTE: In order to queue a task, its SCHEDULING RECOMMENDED field (#209) in the OPTION file (#19) *must* be set to **YES**.

Valid values are:

Table 22-1. Special Queueing field settings

Value	Option Description
S	**STARTUP—**TaskMan will queue the job to run whenever the TaskMan/computer is started (i.e., at System Boot). If you want to the run the startup option on multiple CPUs, make multiple entries in the OPTION SCHEDULING file (#19.2), and use the QUEUED TO RUN ON VOLUME SET field to specify on what Volume Set/CPU each should run.
SP	**STARTUP/PERSISTENT—**TaskMan will queue the job as it does for "STARTUP. It will mark it as a "PERSISTENT" task to be restarted if it stops unexpectedly.
P	**PERSISTENT—**TaskMan will run it on its normal schedule, marking it as Persistent. TaskMan will restart the task if it stops unexpectedly. If the task completes in a normal fashion it is treated like any other regularly scheduled task and it is rescheduled based on the value in the RESCHEDULING FREQUENCY field (#6) in the OPTION SCHEDULING file (#19.2).

Table 22-2. Option Scheduling frequency code formats

Code	Frequency
nS	Every n seconds.
nH	Every n hours.
nD	Every n days.
nM	Every n months.
day[@time]	Day of week (for Day codes, see Table 22-3).
D[@time]	Every weekday.
E[@time]	Every weekend day (Sat,Sun).
nM(entry[,entry[,...]])	Every n months, at each entry in the parameter list; the entries in the parameter list (for every n months only) can be: **Entry Format** — **Frequency** dd[@time] — Day of month (e.g., 15). nday[@time] — Nth day of week in month (e.g., 1W,3W). L[@time] — Last day of month. Lday[@time] — Last specific DAY in month, (e.g., LM,LT,LW...).

Table 22-3. Day codes used in Option Scheduling frequency code formats

Day Code	Description
M	Monday
T	Tuesday
W	Wednesday
R	Thursday
F	Friday
S	Saturday
U	Sunday

Table 22-4. Examples of Option Scheduling frequency code formats

Code	Frequency
12H	Every 12 hours.
14D	Every 14 days.
1M(1,15)	First and 15th of the month.
1M(L@23:45)	Last day of the month at 11:45 pm.
1M(LS)	The last Saturday of the month.
3M(15@12:00,L@12:00)	Noon (on the 15th and last days), every 3 months.
W@4pm	Each Wednesday at 4 pm.
D	Each weekday.

Problems With Scheduled Options

Once an option has been put on a schedule, it stays on that schedule unless one of the following happens:

- You delete the task.
- The running task aborts while setting up the next task in the sequence; the schedule sequence is broken.
- You dequeue the task that is scheduled to run the option. You *must* either requeue the task or use the Schedule/Unschedule Options option to start the cycle over.
- You change the value in the RESCHEDULE FREQUENCY field. The new increment is used beginning after the next time the option runs.
- You change the value in the QUEUED TO RUN AT WHAT TIME field. The currently scheduled task will be unscheduled and a new one will be scheduled for the time you specify.

Another peculiarity in this process involves using a monthly scheduling frequency. What should happen if on January 31st you queue an option and give it a monthly scheduling frequency? Other months lack a

31st day. In this situation, the task pretends there is a 31st day in every month. To avoid this, you can use the RESCHEDULE FREQUENCY code 1M(L@time).

One-time Option Queue Option

Figure 22-23. One-time Option Queue option

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Taskman Management ...                                                  [XUTM MGR]
   One-time Option Queue                                           [XU OPTION QUEUE]
```

To run the One-time Option Queue option [XU OPTION QUEUE] at a special time one day without affecting its established schedule, use the One-time Option Queue option. It queues a task to run once, without affecting the option's normal schedule in any way. This lets you handle the condition where you have an option queued to run periodically and you would like to queue it once to run at an irregular time without affecting its normal periodic schedule.

Taskman Error Log Menu

The Manager and Submanagers keep track of all errors caused by their own software or by the tasks they start. They log their own errors in two places:

- ERROR LOG file (#3.075)
- Taskman Error Log

Those errors caused by tasks are also recorded in the entries of the tasks themselves and can be seen with any of the various task listing options (List Tasks, TaskMan User, etc.). Just as there are options to display and purge the ERROR LOG file (#3.075), there are options to do the same for the Taskman Error Log.

When the XUTM QCLEAN option cleans tasks from the TASKS file (#14.4), it also cleans any corresponding entries in the Taskman Error Log since it is hard to make sense of an error log entry without the task data.

Kernel strongly recommends that you report new errors to your OIFOs and follow up to ensure expeditious patching. If you do this, over time the number of errors occurring on your system will diminish. This also improves the value of the various error logging systems as indicators of significant events deserving investigation.

Allocation and store errors are often not logged in Kernel's ERROR LOG file (#3.075) because the process of logging errors is complicated and usually requires the use of local variables. Local variables take up space and there is no excess space when these errors occur. However, TaskMan makes its simple entries in the Taskman Error Log prior to calling the Kernel error logging utility. Thus, these errors are often recorded in the Taskman Error Log, but not Kernel's. You are encouraged to carefully monitor both places.

Show Error Log Option

Figure 22-24. Show Error Log option

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Taskman Management ...                                                   [XUTM MGR]
   Taskman Management Utilities ...                                     [XUTM UTIL]
      Taskman Error Log ...                                            [XUTM ERROR]
         Show Error Log                                             [XUTM ERROR SHOW]
```

The Show Error Log option [XUTM ERROR SHOW] displays the errors currently stored in the Taskman Error Log, showing the date and time that the error occurred in a readable format and showing the error message. After the listing, the option gives the number of errors in the error log.

Errors stored in the Taskman Error Log historically are also cross-referenced to the TASKS file (#14.4), linking tasks to the errors they cause.

Clean Error Log Over Range Of Dates Option

Figure 22-25. Clean Error Log Over Range Of Dates

```
SYSTEMS MANAGER MENU ...                                                    [EVE]
Taskman Management ...                                                   [XUTM MGR]
   Taskman Management Utilities ...                                     [XUTM UTIL]
      Taskman Error Log ...                                            [XUTM ERROR]
         Clean Error Log Over Range Of Dates           [XUTM ERROR LOG CLEAN RANGE]
```

After prompting for a "First date to purge:" and a "Final date to purge:", the Clean Error Log Over Range Of Dates option [XUTM ERROR LOG CLEAN RANGE] removes the entries for all errors that occurred on and between the two dates. It prints the number of entries removed. If the first date is not earlier than the final date, no entries are removed.

Use this option to delete all but recent errors that deserve your attention. It is better to resolve specific kinds of errors as you encounter them. However, if there is a period during which you cannot resolve them fast enough to keep the log clean, this option will help you focus on the recent ones.

Purge Error Log Of Type Of Error Option

Figure 22-26. Purge Error Log Of Type Of Error option

```
SYSTEMS MANAGER MENU ...                                                [EVE]
Taskman Management ...                                                [XUTM MGR]
   Taskman Management Utilities ...                                  [XUTM UTIL]
      Taskman Error Log ...                                         [XUTM ERROR]
         Purge Error Log Of Type Of Error                 [XUTM ERROR PURGE TYPE]
```

With the Purge Error Log Of Type Of Error option [XUTM ERROR PURGE TYPE] you can delete from the Taskman Error Log all entries for an error of a specific type. In fact, this option uses the M contains operator ("[**[**"); therefore, it removes every error whose message contains your input as a substring. For example, you can remove every error that occurred in a certain routine or even every error whose message contains a "Q." After performing the purge, the option shows you how many entries were removed.

This option is the best way to keep the log clean. As you resolve certain kinds of errors and prevent them from happening again, you can remove all errors of that kind from the log. This leaves behind only those errors you have not resolved, helping you focus on the problems that remain.

Delete Error Log Option

Figure 22-27. Delete Error Log option

```
SYSTEMS MANAGER MENU ...                                                [EVE]
Taskman Management ...                                                [XUTM MGR]
   Taskman Management Utilities ...                                  [XUTM UTIL]
      Taskman Error Log ...                                         [XUTM ERROR]
         Delete Error Log                                       [XUTM ERROR DELETE]
```

The Delete Error Log option [XUTM ERROR DELETE] completely deletes all errors in the Taskman Error Log. If the error log is cleaned and purged as described above, you will rarely need to use this option.

Troubleshooting

The information given in this topic cannot be used by application developers in their code. It is provided to help site managers troubleshoot problems with tasks and TaskMan. Consider this topic a reference to TaskMan's global structure and messages.

SCHEDULE File

^%ZTSCH holds the non-VA FileMan-compatible SCHEDULE file, which consists of independent lists and nodes. This is where TaskMan processes tasks. This structure is not supported for use by application software. All task manipulation *must* be done through approved options and entry points. These structures *must* be free to change from version to version to easily adapt and meet the changing needs of VistA. On the following pages is an example of a global that contains one of each type of node used by TaskMan:

The initial node was used to create ^%ZTSCH before TaskMan was active, so that the global type and protection could be assigned.

Table 22-5. ^%ZTSCH (SCHEDULE File) nodes

^%ZTSCH Node	Description
^%ZTSCH(next run time, task #)	This node stores the Schedule List. The task # corresponds to an entry in the TASKS file (#14.4), and the next run time is computed from the value in the sixth **^**-piece of the entry's 0 node (and is the total number of seconds contained in the next run time's $H translation). If the Schedule List entry equals a device name, the entry was not created through the Program Interface.
^%ZTSCH("C")	This node stores the Compute Server Job List (C list). This list holds tasks that are ready to be run by Submanagers on specific Compute Servers. A Submanager cross-Volume Set jobbed to a Compute Server only runs tasks under this list for the Compute Server on which it is running, and does not process the Device Waiting List or the Job List. The Volume Set, next run time, task #, and device $IO are stored here.
^%ZTSCH("DEV")	This node stores the Device Allocation List. This list is used by TaskMan to coordinate its allocation of devices to tasks. The presence of a node indicates that TaskMan has already allocated this device to a specific task that has not yet gained ownership of it. It tells TaskMan not to give the device to another task. When the task for whom the allocation node was established gains ownership of the device or fails due to possession by some interactive job, the node is KILLed off. The $H value is used in case the task fails to remove its own node for some reason; after two minutes TaskMan KILLs the node on its next idle loop.
^%ZTSCH("ER")	This node stores the Taskman Error Log.

^%ZTSCH Node	Description
^%ZTSCH("ES")	This node stores the Error Screens.
^%ZTSCH("IDLE")	This node is used to ensure that the Manager's idle loop activities are spaced out correctly in case multiple Managers are being run in the same environment.
^%ZTSCH("IO")	This node stores the Device Waiting List. The device $IO value is the value for the task's device and should not be the $IO of a spool or host file device. The run time subscript (the total number of seconds contained in the run time's $H translation) prioritizes the tasks that should have started the longest time ago. The Submanagers use the top node to space out access to the list, and the last device so that only one Submanager at a time is checking the list, and so that checks that find all devices still busy are followed by a short waiting period before the list is checked again.
^%ZTSCH("JOB")	This node stores the Job List. This list holds tasks that are ready to be run by Submanagers. The run time is the total number of seconds contained in the run time's $H translation, and task # and device $IO are what you would expect.
^%ZTSCH("LINK")	This node stores the Link Lists. The LINK node itself is only present when a link is down. It is used to time the checks that occur every fifteen minutes. The second level nodes should always be present with the current information on each of the CPUs and Volume Sets.
^%ZTSCH("LOAD", load rating)	This node is used to balance the CPU load among the various Managers that work out of the current TASKS and Schedule files. It identifies the CPU that most recently checked its rating and decided to run. Managers more loaded (a lower rating) than this one wait to allow this Manager to pick up more of its share of the load.
^%ZTSCH("LOADA")	This node stores the Load List. This list records the ratings for all the CPUs with Managers processing this TASKS file. The first **^**-piece, which flags the Managers that decide to wait to balance the load, is used to tell the Submanagers on those CPUs that they, too, should wait.
^%ZTSCH("LOGRSRC")	This node flags whether Submanagers should log resources for the capacity management software. This node is set for every Volume Set whenever the LOG RESOURCE USAGE? field of the KERNEL SYSTEM PARAMETERS file (#8989.3) is edited. A cross-reference keeps the ^%ZTSCH("LOGRSRC") node in synchronization with the LOG RESOURCE USAGE? field.
^%ZTSCH("NO-OPTION")	if set, this node stops the Submanagers from running any scheduled options. This is for the KIDS install process.

^%ZTSCH Node	Description
^%ZTSCH("RUN")	This node is where the Manager periodically stamps the current time, leaving a way to determine whether it is currently active. Invoking the XUTM STOP option removes this node (see Figure 22-28).
^%ZTSCH("STARTUP", UCI, option #)	This node holds the Startup List. This list holds the internal number of all options that are specially queued to run every time the Manager starts up. The $HOROLOG value reflects when the option was placed on this list.
^%ZTSCH("STATUS", $J of Manager)	This node holds the Status List. This list holds the periodically updated entries for each Manager active on your machine and reflects each Manager's own perception of its current state.
^%ZTSCH("STOP")	This node prevents Submanagers from running. While it is present, Managers will not start new Submanagers, Submanagers waiting for tasks quit immediately, and those currently running tasks quit as soon as the tasks finish.
^%ZTSCH("SUB")	This node counts the number of Submanagers waiting for new tasks. It is updated regularly by Submanagers as they run tasks. The Manager uses this value to decide whether to JOB out new Submanagers and adjusts its value during the idle loop whenever it believes it to be inaccurate.
^%ZTSCH("TASK", task #)	This node holds the tasks TaskMan believes are currently running. Since entries are cleaned up when tasks quit or encounter errors, those that are forcibly exited by the system manager are left on the list even though they are not running. The Manager clears the list whenever the system starts up, and the system manager can manually remove inaccurate entries by using the exit action of the KILL off a users' job option [XURESJOB]. The task data stored at each node allows TaskMan to list the tasks even when they clean out their TASKS file (#14.4) records when they start instead of when they quit.
^%ZTSCH("UPDATE", $J of Manager)	This node, records when the Manager last updated its local information about the site parameters. This node is KILLed whenever the Manager should update (e.g., site parameters are changed).
^%ZTSCH("WAIT")	This node, puts the Manager into a WAIT state.

Figure 22-28. ^%ZTSCH Global Structure

```
^%ZTSCH= ""
^%ZTSCH(next run time, task #)= ""
^%ZTSCH(next run time, task #)= (D1) device IOP value
^%ZTSCH("C", volume set)= count
^%ZTSCH("C", volume set, next run time, task #) = device $IO
^%ZTSCH("DEV", device $IO)= $H when device was allocated for a specific
                                        ==>task
^%ZTSCH("ER")= "A1" or ""
^%ZTSCH("ER", $H when error happened)= error message
^%ZTSCH("ER", $H when error happened, 0)= context of error
^%ZTSCH("ES", error screen, 0)= ""
^%ZTSCH("ES", error screen, 1)= screened errors count
^%ZTSCH("IDLE")= $H when the Manager's idle loop checks were last performed
^%ZTSCH("IO")= $H when device waiting list was last checked without finding
               ==> an available device^ $IO of last device tried
^%ZTSCH("IO", device $IO)=device type
^%ZTSCH("IO", device $IO, run time, task #)= ""
^%ZTSCH("JOB", run time, task #) = device $IO
^%ZTSCH("LINK")= "" or $H when dropped link was last checked
^%ZTSCH("LINK", volume set)= 1 if link has dropped
^%ZTSCH("LINK", volume set, next run time, task #)= ""
^%ZTSCH("LOAD", load rating) = cpu ^ $H when rating was checked
^%ZTSCH("LOADA", cpu) = whether TM should wait ^ load rating ^ $H
                                  ==>when rating was checked ^ $J of Manager
^%ZTSCH("LOGRSRC") = ""

^%ZTSCH("NO-OPTION")=""
^%ZTSCH("RUN")= $H when Manager last checked in
^%ZTSCH("STARTUP", UCI, option #)= $H when option was first queued for
                                   ==>startup
^%ZTSCH("STATUS", $J of Manager)= $H when Manager last checked in [1] ^
                                 ==>status [2] ^ description of status [3]
^%ZTSCH("STOP")= ""
^%ZTSCH("SUB")= count of Submanagers waiting for tasks
^%ZTSCH("TASK", task #)= (A2) entry point [1] ^ (A3) routine [2] ^ (A4)
                      ==>option # [3] ^ (A5) option name [4] ^ (C6)
                      ==>description [5] ^ device name [6] ^ (E1) UCI [7] ^
                      ==>(C3) creation time [8] ^ (C1) creator DUZ or (C2)
                      ==>creator name [9] ^ $J of running task [10] ^ $H
                      ==>when task actually started running [11]
^%ZTSCH("UPDATE", $J of Manager)= $H when the Manager last updated its
                                 ==>parameters
^%ZTSCH("WAIT")= ""
```

TASKS File (#14.4)

^%ZTSK holds this partially-VA FileMan-compatible file of tasks. It is structured with a descriptor node followed by sequential entries. The data dictionary for this file is 14.4, TASKS. It is a read-only file. The TASKS file (#14.4) has no cross-references, not even a top-level B cross-reference, and its descriptor node is updated by the purge option (XUTM QCLEAN).

Each entry itself contains a zero node and several decimal nodes followed by a number of storage nodes. Like the SCHEDULE file, the TASKS file (#14,4) is *not* available for direct manipulation or examination by application software. Site managers, however, can print out information on entries in the TASKS file (#14,4) using VA FileMan.

The following diagram (Figure 22-29) describes the nodes 0 through .26 for each entry in the TASKS file (#14.4):

Figure 22-29. TASKS File (#14.4) nodes (1 of 2)

```
^%ZTSK(task #, 0)= (#.01) Entry Point [1F] ^ (#2) Routine Name [2F] ^ (#3) User
          ==>[3P:200] ^ (#4) Requested UCI [4F] ^ (#5) Creation Time ($H)
          ==>[5F] ^ (#6) Scheduled Run Time ($H) [6F] ^ (#7) Type of Task
          ==>[7F] ^ (#8) Option Number [8N] ^ (#9) Option Name [9F] ^ (#10)
          ==>Creator Name [10F] ^
          ==> (#11) Creation UCI [11F] ^ (#12) Creation Volume Set [12F] ^
          ==>(#13) RESERVED [13F] ^ (#14) Requested Volume Set [14F] ^ (#15)
          ==>Priority [15N] ^ (#16) Original Create date ($H) [16F]
^%ZTSK(task #, .01)= (#21) Original Destination UCI [1F] ^ (#22) Original
          ==>Destination Volume [2F] ^
^%ZTSK(task #, .02)= (#31) Current Destination UCI [1F] ^ (#32) Current
          ==>Destination Volume Set [2F] ^ (#33) Hop Count [3N] ^
^%ZTSK(task #,.03)= (#41) Task Description [E1,240F]^%ZTSK(D0,.04)= (#42) Schedule
Time Seconds [1N] ^
^%ZTSK(task #, .1)= (#51) Status Code [1F] ^ (#52) Last Update $H [2F] ^ (#53)
          ==>Status Notes [3F] ^ (#54) Job [4N] ^  ^  ^  ^ (#59.8) Remember
          ==>Untill [8F] ^  ^ (#59.1) Stop Flag [10F]^
^%ZTSK(task #, .12, (#71) Error Count [1N] ^ (#72) Error $H [2F] ^ (#73) Error
          ==>Message [3F] ^
^%ZTSK(task #, .2)= (#81) Device IOP value [1F] ^ (#82) $IO value [2F] ^ (#83)
          ==>Device Type [3F] ^ (#84) Device Sub-Type [4F] ^ (#85) Device
          ==>%IS modifier [5F] ^ (#86) Host File Address [6F] ^ (#87) Sync Flag
[7F] ^ (#88) IO
           ==>Reschedule Count [8N] ^
^%ZTSK(task #, .21)= (D8) device file entry # [1] ^
^%ZTSK(task #, .25)= (D7) device parameters [1] ^
^%ZTSK(task #, .26)= (D10) hunt group name [1] ^
^%ZTSK(task #, .26, $IO values of hunt group members)= ""
```

The remaining nodes of each entry are used to pass variables to the task. If the task has been manipulated only using TaskMan's Program Interface, then the entries look like this:

Figure 22-30. TASKS File (#14.4) nodes (2 of 2)

```
^%ZTSK(task #, .3, "name")= (F2) value of saved variable
^%ZTSK(task #, .3, "array(", node #)= (F2) value of saved variable
^%ZTSK(task #, .3, "array", node #)= (F2) value of saved variable
```

The distinguishing characteristic here is the fact that the variables to be passed are all subscripted under the .3-node.

Task Status Codes

This topic lists the various codes that may be found in the first ^-piece of the .1 node, the text displayed for that code by the List Tasks option, and the meaning of that code. These codes are set into the tasks at every point in processing where the status changes, along with a time stamp and an explanation where necessary.

Several of the codes correspond to the status of the SCHEDULE file entry for the task. If all applications used the Program Interface, the status code would always agree with the task's real status. In fact, many applications still directly manipulate ^%ZTSCH and ^%ZTSK, and they often neglect to update the status codes. Whenever the SCHEDULE file disagrees with the status code, the SCHEDULE file is correct. This is the reason many of the codes listed in Table 22-6 have multiple meanings.

Status codes 1 through 6 represent one of two common paths a task takes through TaskMan. The other common path replaces code 3 with A, where the task's device is not immediately available.

Table 22-6. TaskMan task status codes

Status Code	Description
0	Incomplete or still being created.
1	Scheduled for <date and time>. TaskMan uses this status in every option and entry point that schedules a task. If the task fails or errors out and TaskMan cannot trap the error, this status has a different meaning: "Stopped irregularly while scheduled."
2	Being inspected by TaskMan. The Manager sets this status when the time comes for a task to run. As it removes the task from the SCHEDULE file, it sets this code into the task.
3	Waiting for a partition. When the Manager places a task in the Job list of the SCHEDULE file, it gives the task this code. If the task fails or errors out, and TaskMan cannot trap the error, this status has a different meaning: "Stopped irregularly while waiting for a partition."
4	Being prepared. The Submanager gives a task this code when it removes the task from the Job list or Busy Device Waiting list in order to run it.
5	Currently running. The Submanager gives a task this status just before it starts the task at its entry point. If the task fails or errors out, and TaskMan cannot trap the error, this status has a different meaning: "Started running <date & time> and stopped irregularly."
6	Completed <date and time>. The Submanager gives a task this status after the task quits.

Status Code	Description
A	Waiting for device <device name or $I> or hunt group <hunt group name>. The Manager or the Submanager gives a task this status when it places the task in the Busy Device Waiting list. If the task fails or errors out and TaskMan cannot trap the error, this status has a different meaning: "Stopped irregularly while waiting for a device."
B	Rejected. <rejection message>. The Manager or the Submanager gives a task this status if it fails one of the basic validation tests. (The rejection messages are contained in the next topic.)
C	Error <date and time>. <error message>. The Submanager gives a task this status if it traps an error after starting the task. The error message records the vendor-specific $ZE text.
D	Stopped by user. The Manager or the Submanager gives a task this status if, when TaskMan removes the task from the SCHEDULE file for processing, it finds that the user has asked the task to stop. The Submanager also assigns this status if, just before starting the task, it finds the stop request has been made. Finally, the Submanager gives a task this status if the task uses the ZTSTOP output variable to report that it stopped in response to a user's request.  **REF:** For an explanation of ZTSTOP, please refer to the description of $$S^%ZTLOAD API in the "TaskMan: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address: http://vista.med.va.gov/kernel/apis/index.shtml
E	Interrupted while running. At startup, the Manager gives this status to any task listed in the Task list of the SCHEDULE file as still running.
F	Unscheduled by <user name or "you">. The Dequeue Tasks [XUTM DQ] and TaskMan User [XUTM USER] options and the DQ^%ZTLOAD entry point use this status for tasks they unschedule.
G	Waiting for the link to <volume set name> to be restored. The Manager uses this status for tasks that would have been transferred to a different TaskMan environment and deleted from this one, if the local area network link to the remote environment were functioning properly. If the task fails or errors out, and TaskMan cannot trap the error, this status has a different meaning: "Stopped irregularly while waiting for a link."
H	Edited without being scheduled. The Requeue Tasks [XUTM REQ] and TaskMan User [XUTM USER] options and the REQ^%ZTLOAD entry point use this status when edited tasks are not subsequently rescheduled.

Status Code	Description
I	Discarded by TaskMan because its record was incomplete. The Manager or the Submanager uses this status for tasks listed in the SCHEDULE file that lack critical information in the corresponding TASKS file (#14.4) entries.
J	Currently being edited. This status has been set aside for possible use in future versions of TaskMan.
K	Created without being scheduled. The ^%ZTLOAD entry point uses this status for tasks when the application passes ZTDTH="@". Kernel Toolkit utility ^%ZTMOVE uses this value for the tasks it creates to transfer routines between Volume Sets manually.
L	Preparing this task caused the Submanager an error <date and time>. <error msg>. The Submanager uses this status when it traps an error after claiming a task but before starting it. The Manager does not yet record a corresponding status for the analogous situation. Tasks that never start, that are left with a status of 2, have usually caused the Manager an error while it tried to examine them.
M	Waiting for a partition on a Compute Server. The Manager gives a task this code when it places the task in the Compute Server Job List. If the task fails or errors out, and TaskMan cannot trap the error, this status has a different meaning: "Stopped irregularly while waiting for a partition on a Compute Server."

Task Rejection Messages

Under certain conditions TaskMan can avoid trapping obvious errors by checking the tasks themselves for internal consistency. Whenever it finds tasks with bad data, it rejects them. This involves unscheduling them, setting their status codes to "B", and adding a brief explanatory message. These messages can help identify bugs in application queuing software, in the local system configuration, or in TaskMan itself.

Table 22-7. TaskMan rejection messages

TaskMan Rejection Message	Description
BAD DESTINATION UCI	The Manager rejects a task for this reason under three different conditions: • If the task is bound for the Manager's own Volume Set, whatever value has been passed for the destination UCI *must* be a valid UCI on the current Volume Set. If ^%ZOSF("UCICHECK") rejects the UCI, TaskMan rejects the task. • If the task is bound for a different Volume Set and the destination UCI is not listed in the UCI ASSOCIATION file (#14.6) under that Volume Set, the UCI *must* be accepted as a valid UCI on the current Volume Set so TaskMan can use File #14.6 to determine where the task should run. If ^%ZOSF("UCICHECK") rejects the UCI, TaskMan rejects the task. • If the task is bound for a different Volume Set and that Volume Set's link is down and its REPLACEMENT VOLUME SET is the current Volume Set, TaskMan rejects the task.
BAD DESTINATION VOLUME SET	Every task's destination Volume Set *must* be listed in the VOLUME SET file (#14.5).
BAD IO DEVICE <$I>	If a port goes bad while many tasks wait for it in the Busy Device Waiting list, TaskMan traps an error whenever the port is tested for availability. When the Submanager traps such an error, it rejects every task waiting for that device.
INVALID OUTPUT DEVICE	The Manager performs a lookup on the devices that tasks request. If the ^%ZIS call indicates that the device does not exist, then TaskMan rejects the task.
INVALID ROUTINE NAME	If a task's entry point is in a %-routine, the Manager tests for that routine's existence in the library UCI. If the routine does not exist there, TaskMan rejects the task.
NO DESTINATION UCI	When older applications bypassed the Program Interface, they sometimes scheduled tasks without specifying the destination UCI. The Manager rejects all such tasks.
NO LINK ACCESS TO VOLUME SET	If the VOLUME SET file (#14.5) entry for a task's destination Volume Set indicates there is no link access to that Volume Set, the task is rejected.
NO ROUTINE AT DESTINATION	If a task's entry point is in a non-%-routine, then the check for

TaskMan Rejection Message	Description
	the routine's existence is done by the Submanager prior to starting the task.

TaskMan State Messages

When the Manager does not run, all background processing grinds to a halt. For this reason, the Manager's condition is of vital importance to system managers. When problems are detected with background processing at a site, checking the Manager's condition should be the first step. The Manager periodically records its state in the Status List. The Monitor TaskMan option [XUTM ZTMON] displays this list near the top of the screen. The various states and their meanings are described in the topics that follow.

BALANCE State

The Manager lists itself in this state if other Managers (that are processing the same files) appear to have more CPU capacity available than the current Manager. While in the BALANCE state, the Manager does not process any tasks or start any new Submanagers. The Manager removes itself from the BALANCE state when it appears to have at least as much CPU capacity as the active Manager. In general, when many Managers are working out of the same TASKS (#14.4) and SCHEDULE files, most of them will be in the BALANCE state at any given time, with only the one or two least loaded Managers actually processing tasks.

REF: For more information about TaskMan load balancing, please refer to the "Multiple TaskMan Managers and Load Balancing" topic in the "TaskMan: System Management—Configuration" chapter in this manual.

ERROR State

The Manager lists itself in this state after trapping errors. On some systems the process of recording an error is slow, so the presence of a distinct state helps identify the source of delay to the system manager. A troubleshooter who sees this state for TaskMan should immediately check the TaskMan Error list to see what kind of error is being recorded. Because TaskMan's code is structured as a series of nested loops, it can very easily generate thousands of errors a day under certain conditions.

PAUSE State

The PAUSE state means that some external condition is preventing the Manager from processing tasks. The description always indicates the cause. While in the PAUSE state, the Manager waits until the problem is resolved, checking once every 60 seconds. The pause states are as follows:

Table 22-8. TaskMan PAUSE states

PAUSE State	Description
The following required ^%ZOSF nodes are undefined, <list of nodes>	When the Manager starts, restarts, or recovers from a trapped error, its first order of business is to drop through some setup code that checks TaskMan's environment. If any critical ^%ZOSF nodes are missing, it enters a PAUSE state and waits until the system manager restores the nodes.
Required link to <volume set name> is down	The other key check in the setup code is to ensure that all Volume Sets listed in the VOLUME SET file (#14.5) as required can actually be reached. The Manager tests each required link and enters the PAUSE state if any tests cause an error. The Manager remains in the PAUSE state, periodically testing the links, until they are restored.
Logons Inhibited	When the system manager sets the INHIBIT LOGONS? field of the VOLUME SET file (#14.5), TaskMan enters a PAUSE state and waits until the flag is cleared.
No Signons Allowed	The system manager can use the software switch to stop logons, which places TaskMan in the PAUSE state.

RUN State

The RUN state indicates that the Manager is going about its business in a relatively normal manner, managing background tasks on your system.

Table 22-9. TaskMan RUN states

RUN State	Description
Start	The Manager sets this value before and after executing the setup code at system startup.
Setup	The Manager identifies when it executes the setup code to test its environment.
Restart	The Manager sets this value after executing the setup code during a restart.
Main Loop	This should be the Manager's usual state. This indicates the Manager is executing the main loop that checks the environment, processes the Schedule list, and performs idle loop activities when appropriate.
TaskMan Job Limit	When the total number of processes on the Manager's CPU exceeds the TaskMan Job Limit given in the VOLUME SET file (#14.5), the Manager

RUN State	Description
Reached	can continue to process the Schedule list but cannot start any new Submanagers.

WAIT State

While in the WAIT state, the Manager does not react to changes in its environment. It does not process tasks, enter PAUSE states, or even stop after the Stop TaskMan option has been used.

You have two options (described above) that let you create or undo the WAIT state. TaskMan cannot enter this state on its own; it can only be initiated manually. This is essentially a tool for you to tightly control the processing of tasks on your machines. The description for this state always reads "TaskMan Waiting".

V KIDS

23 KIDS: System Management—Installations

Kernel Installation and Distribution System (KIDS) was introduced with Kernel V. 8.0. Previously, software was exported using a utility called DIFROM, and installed by running INIT routines that the DIFROM utility created. KIDS is the replacement for DIFROM, and introduces significant revisions to the software distribution and installation processes. This chapter introduces KIDS, and describes some of the changes to the software export process.

The following definitions apply throughout the KIDS documentation:

Table 23-1. KIDS-related terms and definitions

Term	Definition
Transport Global	An exported software application, stored in a global. KIDS exports software (i.e., package) based on its definition in a build entry. The transport global also contains the build entry and the PACKAGE file (#9.4) entry (if any) for a given software application.
Build Entry	An entry in the BUILD file (#9.6) that defines the parts of a software application to export. Also known as a build.
Component	An element of one of the following types: template (PRINT, SORT, and INPUT); form; function; bulletin; help frame; routine; option; security key; and protocol.
Distribution	A Host File Server (HFS) file containing transport global(s). If a distribution contains multiple transport globals, KIDS treats them as a single installation when installing from the distribution.
Package	A cohesive set of files, data, and components that together form a set of computing activities related to a functional area (i.e., software).

KIDS Options

To get to the KIDS: Kernel Installation & Distribution System menu [XPD MAIN] (locked with the XUPROG security key) choose the Programmer Options menu option [XUPROG] on the Kernel Systems Manager Menu [EVE], as shown below:

Figure 23-1. KIDS menu options

```
Select Systems Manager Menu Option: Programmer Options


KIDS   Kernel Installation & Distribution System ...                    [XPD MAIN]
             **> Locked with XUPROG
   PG     Programmer mode                                              [XUPROGMODE]
             **> Locked with XUPROGMODE
          Delete Unreferenced Options                          [XQ UNREF'D OPTIONS]
          Error Processing ...                                            [XUERRS]
          General Parameter Tools ...                             [XPAR MENU TOOLS]
          Global Block Count                                       [XU BLOCK COUNT]
          List Global                                                      [XUPRGL]
             **> Locked with XUPROGMODE
          Routine Tools ...                                    [XUPR-ROUTINE-TOOLS]
          Test an option not in your menu                          [XT-OPTION TEST]
             **> Locked with XUMGR
Select Programmer Options Option: kids <Enter>  Kernel Installation & Distribution
    System


          Edits and Distribution ...                        [XPD DISTRIBUTION MENU]
          Utilities ...                                               [XPD UTILITY]
          Installation ...                                  [XPD INSTALLATION MENU]
             **> Locked with XUPROGMODE
          Patch Monitor Main Menu ...                  [XTPM PATCH MONITOR MAIN MENU]
          Patchman ...                                  [XPD AUTOMATIC PATCHING MENU]
```

As indicated by its name (i.e., KIDS = Kernel Installation and Distribution System), KIDS supports two major functions:

- **Distribution**
- **Installation** (discussed in this chapter)

REF: In addition, KIDS also provides other utilities. For more information on KIDS utilities, please refer to the "KIDS: System Management—Utilities" chapter in this manual.

Distributions

The distribution related options are located on the Edits and Distribution menu [XPD DISTRIBUTION MENU] (see Figure 23-2). The distribution portion of KIDS allows developers to:

- Define the contents of a software application in a build entry.
- Create transport globals from build entries.
- Export transport globals by creating distributions.

Figure 23-2. Edits and Distribution menu options

```
Select Kernel Installation & Distribution System Option: Edits and Distribution


          Create a Build Using Namespace
          Copy Build to Build
          Edit a Build
          Transport a Distribution
          Old Checksum Update from Build
          Old Checksum Edit
          Routine Summary List
          Version Number Update

Select Edits and Distribution Option:
```

REF: For a description on how application developers use the KIDS build and distribution options, please refer to the "KIDS: Developer Tools" chapter in the *Kernel Developer's Guide*.

Installations

The installation related options are located on the Installation menu [XPD INSTALLATION MENU] (see Figure 23-3). The installation portion of KIDS allows sites to:

- Load transport globals from KIDS distributions.
- Load transport globals from KIDS PackMan messages.
- Print out the contents of loaded transport globals before installing them.
- Compare the contents of loaded transport globals to the current system before installing them.
- Install loaded transport globals.

Figure 23-3. Installation menu options

```
Select Kernel Installation & Distribution System Option: Installation


   1      Load a Distribution
   2      Verify Checksums in Transport Global
   3      Print Transport Global
   4      Compare Transport Global to Current System
   5      Backup a Transport Global
   6      Install Package(s)
          Restart Install of Package(s)
          Unload a Distribution

Select Installation Option:
```

KIDS introduced two files into Kernel:

- BUILD file (#9.6)
- INSTALL file (#9.7)

KIDS also makes use of the existing PACKAGE file (#9.4), but its role in exporting and installing software is diminished.

Build Entries and the BUILD File (#9.6)

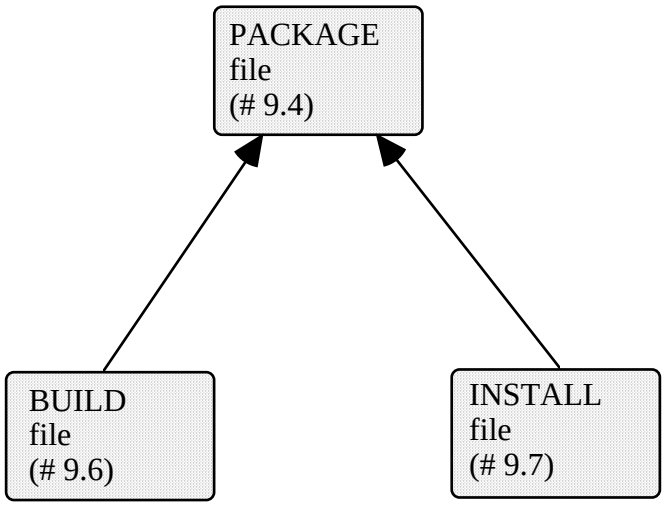
Build entries, stored in the BUILD file (#9.6), are where developers define a software application. This build entry defines the set of files, data, components, installation questions, national software information, pre- and post-install routines, and other settings that comprise the exported software.

Software components are no longer tied to namespace, as they were previously with DIFROM and the PACKAGE file (#9.4). Developers can select any components available on the current system and include them in their build entries as software components.

The format of the NAME field (#.01) of a build entry *must* be the software name concatenated with a space, and then a version number. This means that there is a separate entry for every version of a software application that a developer exports.

Also, a software application's build entry is sent to installing sites as part of the software; after an installation, the site can examine the build entry to see the software definition.

Figure 23-4. KIDS file diagram



INSTALL File (#9.7)

The INSTALL file (#9.7) stores a record of each installation a site performs. The INSTALL file (#9.7) allows KIDS to store a separate installation entry for each installation. A new version of software no

longer overwrites the installation information of a previous version, and developers' installation history no longer overwrites the sites' installation history. The national PACKAGE file (#9.4) is now static at its top level.

The three main items recorded in the INSTALL file (#9.7) for each installation are the installing site's answers to installation questions, any installation output, and the installation's timing information.

Changes in the Role of the PACKAGE File (#9.4)

The PACKAGE file (#9.4) still plays a role in installations with KIDS, albeit a diminished one. KIDS provides a link from the build entry of a package to the PACKAGE file, so that developers can link a package to a PACKAGE file (#9.4) entry.

The top level of a PACKAGE file (#9.4) entry for a package now stores static package information. The only part of the PACKAGE file (#9.4) entry that installations update automatically now is the VERSION field (Multiple). A patch sent with KIDS does *not* transport the entire PACKAGE file (#9.4) entry. It only sends the information that is needed to update the PACKAGE file (#9.4). Patch installations will update the PATCH APPLICATION HISTORY field (Multiple), which is within the VERSION field (Multiple). KIDS saves patch names along with their sequence numbers in this multiple. Most other fields have been designated for removal at the top level of the PACKAGE file (#9.4). The PACKAGE file (#9.4) now stores mainly static software information that is not version specific, as well as the patch history of the software.

Transport Mechanism: Distributions

Distributions are the mechanism KIDS uses to export software. They are more flexible than the previous mechanism (INIT routines).

Distributions are usually in the form of an HFS file. The developer creates transport globals from build entries. KIDS stores transport globals in a global. KIDS can WRITE the global (in a format readable only by KIDS) to an HFS file; the HFS file is the distribution. The HFS file can then be distributed by a variety of methods, including FTP (file transfer protocol), diskette, and tape. For example, if your system is a PC, you can also move the Transport Global to a new medium (i.e., to multiple floppy disks so you can install on other PCs):

- Select the Load a Distribution option (*Do not* run the Environment Check routine).
- Under the Utilities Menu, select the Convert Loaded Package for Redistribution option.
- Under the Edits and Distribution Menu, select the Transport a Distribution option.
- When you are prompted to "Enter a Host File:", enter the floppy drive and file name. For example:

```
Enter a Host File: A:\KRN8.KID)
```

One advantage to using distributions over INIT routines is that there is no limit to the size of a software application you can export. Another advantage is that during installations, you no longer have to overwrite a software application's existing routines with the new routines before running the installation.

Alternatively, a KIDS distribution can be sent via a PackMan message in MailMan. But transporting software as host files, especially large ones, avoids slowing down MailMan.

Two Kinds of Distributions

KIDS supports two kinds of distributions:

- **Standard Distribution—**This type of distribution contains transport globals for what are traditionally thought of as software applications, including files, data, and all components. A standard distribution can contain one or more transport globals. If there is more than one transport global, KIDS treats each one as a single installation unit.
- **Global Distribution—**This type of distribution contains one transport global only, and that transport global can export M globals only.

The transport globals in both types of distributions also contain the corresponding build entry, and (if linked to a PACKAGE file [#9.4] entry) the corresponding PACKAGE file (#9.4) entry. However, a patch sent with KIDS does *not* transport the entire PACKAGE file (#9.4) entry. It only sends the information that is needed to update the PACKAGE file (#9.4).

What Happens to DIFROM?

Developers should no longer use the DIFROM entry point to export software. Developers should use KIDS. The DIFROM method is still supported, but only for the support of sites that use standalone VA FileMan (VA FileMan without Kernel).

REF: For more information on using DIFROM, please refer to the *VA FileMan Programmer Manual*.

Installing Standard Distributions

As noted previously, KIDS supports two types of distributions:

- Standard
- Global

This topic describes how KIDS installations work when installing standard distributions.

Installation Sequence

KIDS installs standard distributions in three phases:

1. Loading transport globals from the distribution.
2. Answering installation questions for each transport global.
3. Installing each transport global in the distribution.

Phase 1: Loading Transport Globals from a Distribution or PackMan Message

1. Using the Load a Distribution option, the installer chooses the HFS file from which to load distributions. If loading from a PackMan message, choose the message and invoke the INSTALL/CHECK MESSAGE PackMan option.
2. For each transport global, KIDS makes an entry in the INSTALL file (#9.7) for the transport global.
3. KIDS loads transport globals from distribution into ^XTMP.
4. KIDS prompts the user to see if they want to run the environment check for each transport global (if unsuccessful, the process quits here; the developer may or may not KILL INSTALL file (#9.7) entries and transport globals from ^XTMP.)
5. The installer can print the contents of the transport global, compare the contents to the current system, and verify checksums of the transport global.

Phase 2: Answering Installation Questions for Transport Globals in a Distribution

1. Using the Install Package(s) option, the installer selects a distribution to install by choosing an entry from the INSTALL file (#9.7).
2. KIDS runs the environment check for the first transport global; the environment check can allow KIDS to install the transport global, cancel installation of the transport global, or cancel installation of all transport globals in the distribution.
3. The installer answers pre-installation questions for the first transport global.
4. The installer answers standard KIDS questions for the first transport global.
5. The installer answers post-installation questions for the first transport global.
6. The installer repeats Steps #2-5 for the remaining transport globals, if there are any more transport globals to process.

7. The installer chooses a device for the installation to run on. The installer can queue the installation or run it directly; entering a caret ("^") aborts the installation.

Phase 3: KIDS Installation of Software

1. KIDS disables any options and protocols the site has asked to be disabled for this install. However, KIDS does *not* disable options and protocols which have an Action of USE AS LINK FOR MENU ITEMS.
2. KIDS waits for the time period (from 0 to 60 minutes) the site specifies, if they chose to disable options and protocols.
3. KIDS suspends the running of queued options by TaskMan for this install, if the site chooses to do so.
4. The pre-install routine is run for the first transport global.
5. All components are installed for the first transport global.
6. The post-install routine is run for the first transport global.
7. KIDS repeats Steps 4-6 for any remaining transport globals to install in the distribution.
8. Options and protocols that were disabled for this install (if any) are re-enabled.
9. Queued options are removed from suspense (if the site chose to suspend queued options).

Installation Menu

The KIDS Installation Menu [XPD INSTALLATION MENU] contains the following options:

Figure 23-5. KIDS Installation menu options

```
Select Kernel Installation & Distribution System Option: Installation
          **> Locked with XUPROGMODE

   1      Load a Distribution                              [XPD LOAD DISTRIBUTION]
   2      Verify Checksums in Transport Global                [XPD PRINT CHECKSUM]
   3      Print Transport Global                               [XPD PRINT INSTALL]
   4      Compare Transport Global to Current System        [XPD COMPARE TO SYSTEM]
   5      Backup a Transport Global                                    [XPD BACKUP]
   6      Install Package(s)                                  [XPD INSTALL BUILD]
          Restart Install of Package(s)                     [XPD RESTART INSTALL]
          Unload a Distribution                           [XPD UNLOAD DISTRIBUTION]
```

The number next to the options indicates the order of the option entries you should follow when performing a KIDS installation.

Loading a Standard Distribution

The first step in installing a standard distribution is to load the transport globals from the Distribution. The Load a Distribution option [XPD LOAD DISTRIBUTION] does the following:

- Lists what transport globals are contained in the distribution and asks you if you want to continue.
- Creates entries in the INSTALL file (#9.7) for each transport global in the distribution that passed its environment check.
- Loads transport globals from the distribution (HFS file) into the ^XTMP global (if you answer YES to continue).
- Prompts the user to see if they want to run the environment check for each transport global. If a transport global does not pass its environment check, KIDS may purge it from ^XTMP; otherwise, the transport global stays in ^XTMP. KIDS tells you the result of each environment check.
- Checks the version number of the incoming software against any existing software of the same name at the site. If the incoming version number is not greater than the existing version, KIDS aborts the installation for the transport global in question.
- Echoes the name of the first transport global to pass environment check (i.e., "Use transport global name to install this Distribution"). The name of the first transport global to pass its environment check is the name you use to install the distribution, in the next phase.

Loading a distribution is the first of three phases to install VistA software. The second phase is answering installation questions, including scheduling the installation; the third and final phase is the actual running of the installation.

When loading from a PackMan message, load the distribution using the INSTALL/CHECK MESSAGE PackMan option in MailMan. For KIDS PackMan messages, this option through MailMan is equivalent to the Load a Distribution option [XPD LOAD DISTRIBUTION].

Figure 23-6. Load a Distribution option—Sample user dialogue

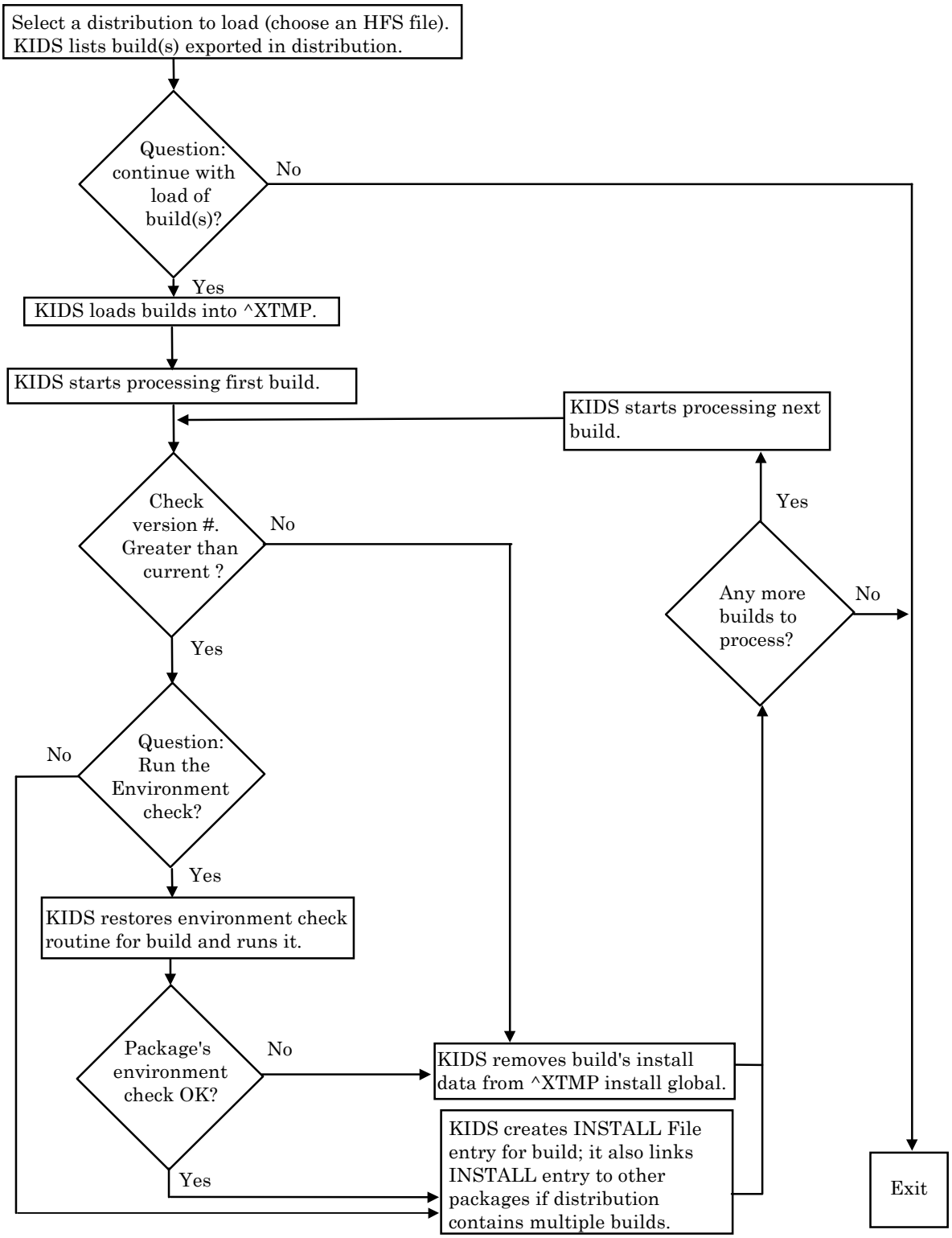
```
Select Installation Option: Load a Distribution
Enter a Host File: ZXG_EXPT.DAT

Distribution saved on Oct 13, 2004@09:29:08
Comment: TEST PKGS

This Distribution contains Transport Globals for the following Package(s):
     TEST 2.1

Want to Continue with Load? YES// <Enter>
Loading Distribution...

Want to RUN the Environment Check Routine? YES// <Enter>
   TEST 2.1

Use INSTALL NAME: TEST 2.1 to install this Distribution.

Select Installation Option:
```

When the Distribution is Split Across Diskettes

Distributions can come in a single host file (see Figure 23-6); alternatively, they can come on diskettes, with the host file split up among the diskettes. If you are installing from a distribution that is spread across diskettes, the Load a Distribution option [XPD LOAD DISTRIBUTION] will ask you for subsequent diskettes (e.g., "Insert the next diskette, #2, and Press the return key", etc.). Insert the appropriate disk and press the **<Enter>** key, and continue until the distribution is loaded.

Loading Transport Globals from a Distribution

Figure 23-7. Loading transport globals from a distribution—Flowchart



Verifying Checksums in a Transport Global

You can verify the checksums for a loaded transport global in advance of installing from it, using the Verify Checksums in Transport Global option [XPD PRINT CHECKSUM]. This option verifies all checksums of routines in the transport global, reporting any discrepancies. In the future, the ability to verify checksums will be extended to other KIDS components besides routines.

NOTE: As of Kernel Patch XU*8.0*369, the integrity checking CHECK1^XTSUMBLD API supports the Compare local/national checksums report option [XU CHECKSUM REPORT].

NOTE: As of Kernel Patch XU*8.0*393, KIDS was modified to send a message to a server on FORUM when a KIDS build is sent to a Host File Server (HFS) device. This message contains the checksums for the routines in the patch. The server on FORUM matches the message with a patch if the sending domain is authorized on FORUM. There is no longer a need for developers to manually include routine checksums (either CHECK^XTSUMBLD and/or CHECK1^XTSUMBLD) in the patch description. The patch module will include the before and after CHECK1^XTSUMBLD values in the Routine Information section at the end of the patch document.

With changes in the National Patch Module (NPM) on FORUM, when the patch is released the checksums for the routines are moved to the ROUTINE file (#9.8) on FORUM. The checksum "before" values will come from the FORUM ROUTINE file (#9.8) and are considered the GOLD standard for released checksums. The local site's Compare local/national checksums report option [XU CHECKSUM REPORT] uses the FORUM ROUTINE file (#9.8) as its source to create reports showing any routines that do *not* match..

This patch also modified the KIDS BUILD file (#9.6) by adding the TRANSPORT BUILD NUMBER field (#63) used to store a build number that is incremented each time a build is made. This build number is added to the second line of each routine in the 7th "**;**" piece. This makes it easy to tell if a site is running the current release during testing and afterword. The leading "B" found in the checksum tells the code what checksum API to use.

Printing Loaded Transport Globals

Once you have loaded transport globals from a standard distribution onto your system, you can print out the definitions of the transport globals, using the Print Transport Global option [XPD PRINT INSTALL]. This way, you can see every component exported in each transport global, before you install them.

Figure 23-8. Print Transport Global option—Sample printed transport global

```
PACKAGE: ZXG DEMO 1.0                                                    PAGE 1
------------------------------------------------------------------------------
NATIONAL PACKAGE:
DESCRIPTION:

ENVIRONMENT CHECK : ZXGENV
PRE-INIT ROUTINE : ZXGPRE
POST-INIT ROUTINE: ZXGPOS
------------------------------------------------------------------------------

ROUTINE:
   ZXGC00                                          SEND TO SITE
   ZXGC01                                          SEND TO SITE
   ZXGC02                                          SEND TO SITE
   ZXGCMOVE                                        SEND TO SITE
   ZXGCTEST                                        SEND TO SITE
   ZXGCTW1                                         SEND TO SITE
   ZXGCWE                                          SEND TO SITE
   ZXGCXMP1                                        SEND TO SITE
   ZXGCXMPL                                        SEND TO SITE
   ZXGDEMO                                         SEND TO SITE
   ZXGKC                                           SEND TO SITE
   ZXGLMSG                                         SEND TO SITE
   ZXGLOAD                                         SEND TO SITE
   ZXGTMP                                          SEND TO SITE

INSTALL QUESTIONS:
     SUBSCRIPT: PRE1
DIR(0)=YA^^
DIR("A")=Do you want to run the pre-install conversion?
DIR("B")=YES
DIR("?")=Answer YES to run the pre-install conversion, NO to skip it...
```

Comparing Loaded Transport Globals to the Current System

When you have loaded transport global(s) from a standard distribution onto your system, you can also compare a transport global to the matching software already installed on your system (if any), using the Compare Transport Global to Current System option [XPD COMPARE TO SYSTEM]. This way, you can compare the software you are about to install with the current version of the software on your system.

When this option finds differences, it notes the change by displaying the differences between the current software and the transport global on two lines, one line labeled * OLD * and the other * NEW *.

NOTE: Pointers are converted to FREE TEXT when exporting VA FileMan entries, so these converted free pointers show up as differences when using the compare feature.

Figure 23-9. Compare Transport Global to Current System option—Sample comparison output

```
Compare ZXP 1.0 to current site
------------------------------------------------------------------


 Routine: ZUVXD


 File # 3.2 Data Dictionary


 File # 3.2 Data

* OLD *   ^%ZIS(2,9,8) =
$C(27)_"[A"^$C(27)_"[B"^$C(27)_"[C"^$C(27)_"[D"^3^^$C(27)_"[L"
* NEW *   ^%ZIS(2,9,8) = $C(27)_"[A"^$C(27)_"[B"^$C(27)_"[C"^$C(27)_"[D"^3
* OLD *   ^%ZIS(2,44,13) = ^$C(26)^^^^$J("",X)_$C(27,93,($X+32-X))
* NEW *   ^%ZIS(2,44,13) = ^$C(26)^^^^
* OLD *   ^%ZIS(2,60,8) =
$C(27)_"[A"^$C(27)_"[B"^$C(27)_"[C"^$C(27)_"[D"^3^^$C(27)_"[L"
* NEW *   ^%ZIS(2,60,8) = $C(27)_"[A"^$C(27)_"[B"^$C(27)_"[C"^$C(27)_"[D"^3
* ADD *   ^%ZIS(2,93,21) = ^


HELP FRAME


BULLETIN
```

This option was updated with Kernel Patch XU*8.0*393 to add a side-by-side comparison in columnar format, which only works if Kernel Toolkit Patch XT*7.3*93 has also been installed, as shown below:

Figure 23-10. Compare Transport Global to Current System option—Sample comparison output in columnar format

```
Select Kernel Installation & Distribution System Option:

   1      Load a Distribution
   2      Verify Checksums in Transport Global
   3      Print Transport Global
   4      Compare Transport Global to Current System
   5      Backup a Transport Global
   6      Install Package(s)
          Restart Install of Package(s)
          Unload a Distribution

Select Installation Option: 4 <Enter> Compare Transport Global to Current System
Select INSTALL NAME: XU*8.0*381 <Enter> Loaded from Distribution
Loaded from Distribution  9/14/06@12:39:52
     => DEMO COMPARE  ;Created on Sep 14, 2006@12:39:17

This Distribution was loaded on Sep 14, 2006@12:39:52 with header of
   DEMO COMPARE  ;Created on Sep 14, 2006@12:39:17
   It consisted of the following Install(s): XU*8.0*381

     Select one of the following:

          1         Full Comparison
          2         Second line of Routines only
          3         Routines only
          4         Columnar Routine compare

Type of Compare: 4 <Enter> Columnar Routine compare
DEVICE: HOME// <Enter> Telnet terminal

Compare XU*8.0*381 to current site     Routines Only
------------------------------------------------------------------------------

Compare of routines from KIDS XU*8.0*381, and disk

Routine XU8P381 not on disk
------------------------------------------------------------------------------
Routine XUTMTP
   KIDS                                    Disk
------------------------------------------------------------------------------
1{XUTMTP ;SEA/RDS - TaskMan:ToolKit} 1{XUTMTP ;SEA/RDS - TaskMan: ToolKit}
 {, Print, Part 1 ;04/18/2006  16:19} {, Print, Part 1 ;04/24/2003  11:06}
                         ^                                          ^
2{ ;;8.0;KERNEL;**20,86,169,242,381*}2{ ;;8.0;KERNEL;**20,86,169,242**;Ju}
                                  ^                                       ^
------------------------------------------------------------------------------
```

Backing Up Transport Globals

The Backup a Transport Global option [XPD BACKUP] creates a MailMan message that will back up all current routines on your system that would be replaced by a KIDS patch. This option is under the

Installation menu of the KIDS menu. It works on a patch that has been loaded on your system, but not installed.

Running Installations

Once you've loaded the transport global(s) from a standard distribution, you can install them. Do this using the Install Package(s) option [XPD INSTALL BUILD].

When you load a distribution, KIDS tells you which transport global name to use to install the distribution (e.g., "Use PACKAGE 1.0 to install this Distribution"). This will always be the first transport global to successfully load from the distribution. When you use the Install Package(s) option [XPD INSTALL BUILD], select the transport global name reported when you loaded the original distribution. Once you've done that, you can answer the installation questions for each transport global in the distribution.

Processing Each Transport Global

When you select a distribution to install, the Install Package(s) option processes the installation questions for each transport global in the distribution. For each transport global, you're asked:

- Pre-Install questions.
- Standard KIDS Questions.
- Post-Install Questions.
- Whether to disable any options or protocols. By typing three question marks ("**???**") at this prompt KIDS will list all of the options and protocols it will disable. If you answer YES, all incoming options and protocols are disabled. You are also prompted to add to or delete from the list of options and protocols to disable. However, KIDS does not disable options and protocols which have an Action of USE AS LINK FOR MENU ITEMS. All scheduled options on the system are also disabled. Finally, you are asked a time period for installation:

  ```
  Delay Install(Minutes): (0-60): 0//"
  ```

  You can delay before starting the installation after disabling options and protocols from 0 to 60 minutes. This is to allow users already in (disabled) options time to exit the options before the installation starts.

Scheduling Installations

The final question you are asked when using the Install Package(s) option to load software is upon what device to run the installation. Your choices at the "DEVICE: " prompt are:

- Run the installation directly by selecting a device without queueing. The installation runs immediately, on the device you specify.
- Queue the installation.
- Abort the installation of the distribution by entering a caret ("^").

When the Installation is Queued

If you queued the installation, you can look up the installation task in TaskMan. A KIDS installation task looks like:

Figure 23-11. Queued KIDS installation—Sample installation task

```
3: (Task #1179950) EN^XPDIJ, KIDS install.  Device VER$LW.  KRN,KDE.
   From TODAY at 16:24,  By you.  Scheduled for TODAY at 22:00
```

You can cancel a queued installation (before it has started) by deleting the task. KIDS also allows you to restart an install if the install is queued and you get an error during the installation.

Re-answering Installation Questions

If you queued an installation, you can re-answer installation questions, if you so choose, using the Install Package(s) option. To be able to re-answer the questions, however, you need to locate the task that was queued for the installation and delete it first. Once you delete the installation's queued task, you can re-answer the install questions. When you re-answer questions, your answers from the previous time come up as default responses.

Also, if you abort an installation after answering its installation questions (i.e., by entering a caret ["^"]), your responses will again be used as the defaults the next time you try to install.

Information Stored in the INSTALL File (#9.7)

KIDS exports the definition of a software application in the BUILD file (#9.6). KIDS records installations of software in the INSTALL file (#9.7). The installation records in the INSTALL file (#9.7) provide a record of the start time, timing for each checkpoint, and completion time (if any) for an installation.

When an installation aborts, the contents of the INSTALL file (#9.7) determine where the install will start up again when you use the Restart Install of Package(s) option (checkpoint information is stored in the INSTALL file [#9.7]).

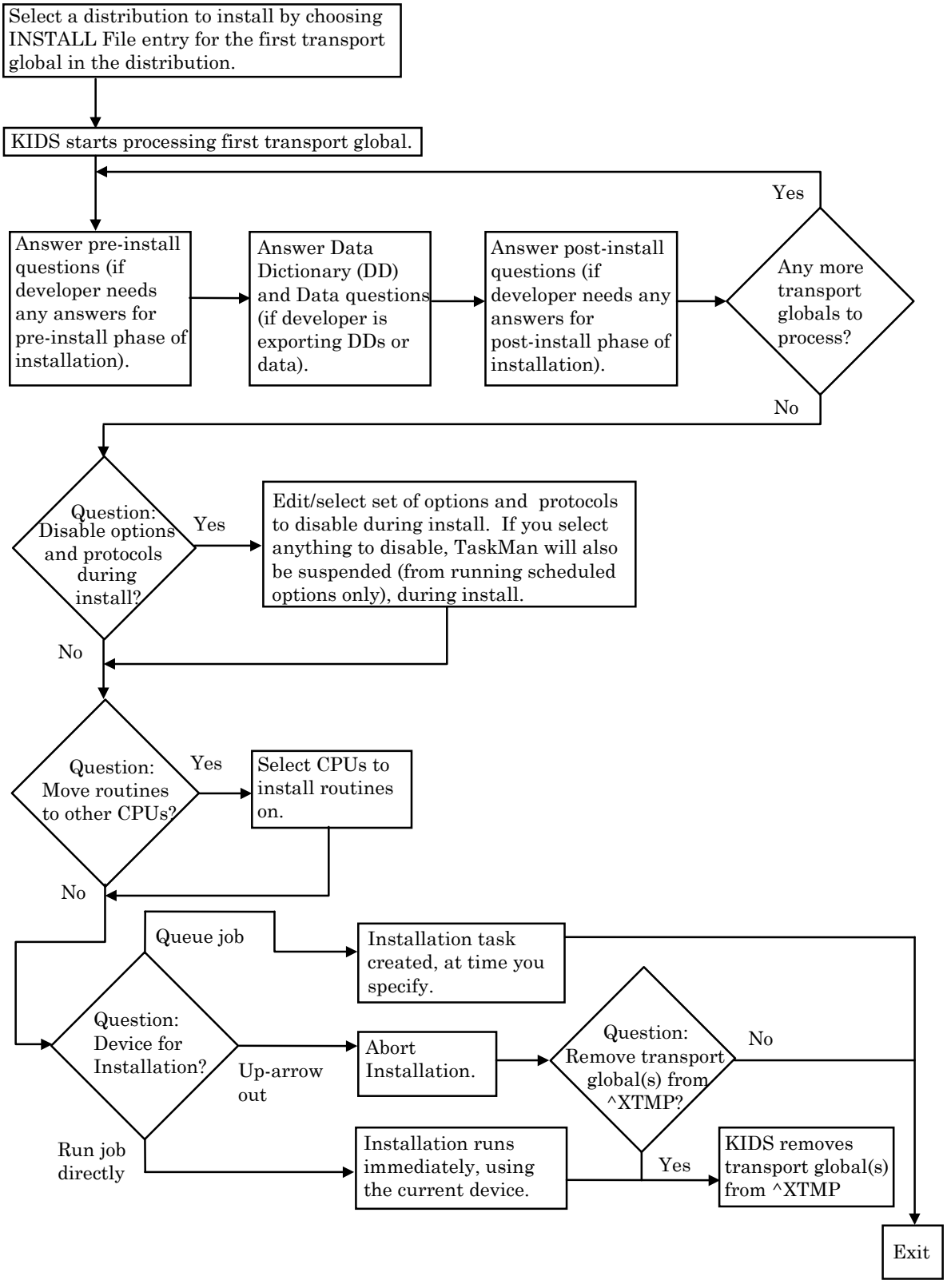
As well as being sent to the installation's principal device, all output from the installation is also stored in the INSTALL file (#9.7), in the MESSAGES word-processing-type field.

The installation questions (and your answers to them) are stored in the INSTALL ANSWERS field (Multiple) of the INSTALL file (#9.7).

You can print entries from the INSTALL file (#9.7) with the Install File Print option.

Answering Installation Questions for a Distribution

Figure 23-12. Answering installation questions for a distribution—Flowchart

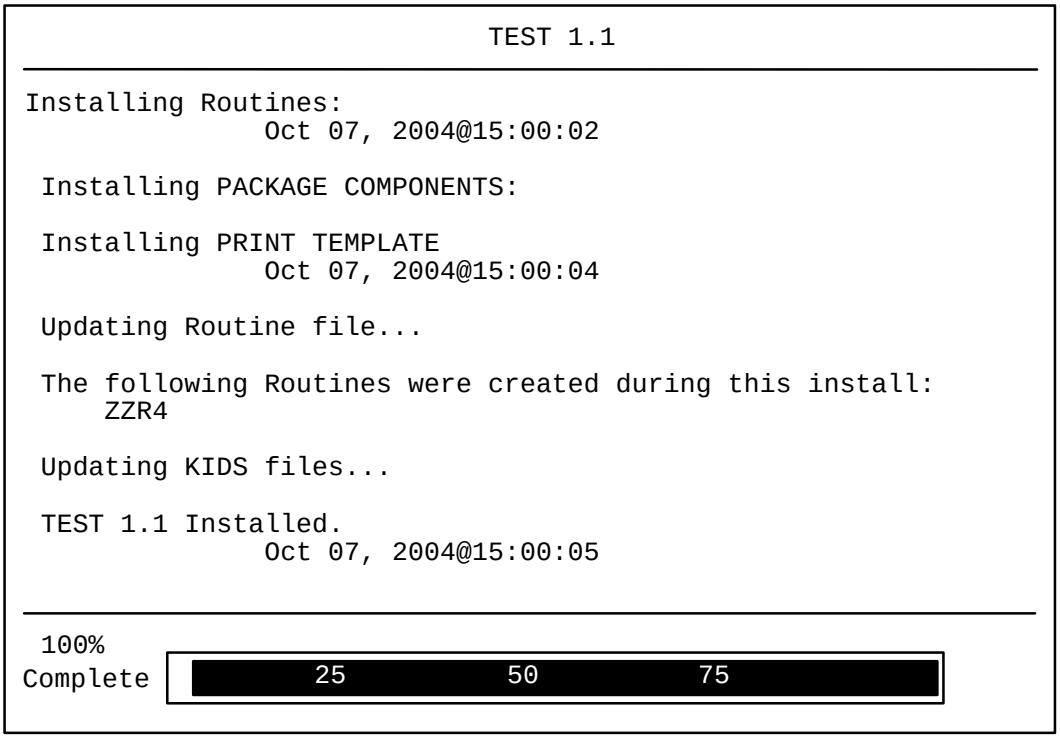


Installation Progress

If the device selected for output is a VT100-compatible (or higher) terminal, KIDS displays the installation output in a virtual window on the terminal. Below the virtual window, a progress bar graphically illustrates the percentage complete that the current part of the installation has reached. KIDS is able to report progress for the installation of files and for all components (PRINT templates, forms, help frames, routines, options, etc.) KIDS lists those compiled cross-references, INPUT templates, and PRINT templates that were created during the install process. KIDS does not show progress for installing data, nor for pre- and post-install tasks.

On all other devices, progress is reported using dots.

Figure 23-13. Installation progress—Sample output



Once the Installation Finishes

When the installation runs, its output is sent to the device you specified when you answered the installation questions. If, for example, you queued the installation to a printer, the output is sent to the printer.

You can find out whether an installation finished by looking up the entry in the INSTALL file (#9.7) for that installation (use the Install File Print option). You should check whether an installation completed successfully or not. If the install completed successfully, the STATUS field in the INSTALL file (#9.7) entry will be set to "Install Completed." If the install errored out, the STATUS field in the INSTALL file

(#9.7) entry will still be set to "Install Started." If it errored out, you need to find out what went wrong, and restart the installation.

REF: For information on restarting an installation, please refer to the "Restarting Aborted Installation" topic that follows.

If you disabled scheduled options, options, and protocols, KIDS should have re-enabled those (unless the install errored out).

You should refer to the instructions that came with the software you installed to see what post-installation tasks, if any, you should perform.

Restarting Aborted Installations

A feature of KIDS is the ability to restart an aborted installation. KIDS uses a checkpoint system to keep track of how many phases of an installation it completed. When an installation aborts for some reason, you can restart the installation (using the Restart Install of Package(s) option [XPD RESTART INSTALL]). KIDS does *not* automatically re-run the entire installation from the beginning; instead, it re-runs the installation only from the last completed checkpoint.

As well as some standard checkpoints built into KIDS (e.g., completion of pre-install, completion of each component type, and completion of post-install), KIDS lets developers create checkpoints for use within their pre- and post-install routines. So depending on how the developer has designed a pre- or post-install, it is possible that, when re-started, the pre- or post-install does not have to be re-run in its entirety either (if the error occurred there). Instead, KIDS only re-runs the pre- or post-install from the last completed developer checkpoint (if any) within the pre- or post-install.

Before restarting an installation, you should try to determine what caused the installation to abort. If an error occurred, any error messages will be in the INSTALL file (#9.7) entry, in the MESSAGES word-processing-type field. Once you've fixed the problem, you can use the Restart Install Of Package(s) option [XPD RESTART INSTALL] to continue with the installation. KIDS also allows you to restart an install if the install is queued and you get an error during the installation.

Recovering from an Aborted Distribution Load

If you encounter an error while loading a distribution (using the KIDS option to load a distribution from the export medium into the ^XTMP global), you will be unable to re-load the distribution until you clear out what was stored during the aborted load attempt.

To clear out the previously loaded distribution, use the Unload a Distribution option [XPD UNLOAD DISTRIBUTION]. To unload a distribution, enter the name of the *first* transport global that was loaded when you loaded the distribution. The entries in the INSTALL file (#9.7) for all transport globals in the distribution will be removed, and the transport globals themselves will be purged from the ^XTMP global.

Once you delete entries in the INSTALL file (#9.7) and entries in the ^XTMP global with the Unload a Distribution option [XPD UNLOAD DISTRIBUTION], you should be able to reload the distribution in question. If the install was already started and you choose to unload the distribution, you first *must* edit

the INSTALL file (#9.7) and set the STATUS field to Load From Distribution (i.e., 0) prior to using the Unload a Distribution option [XPD UNLOAD DISTRIBUTION].

Installing Global Distributions

The second type of distribution supported by KIDS is called a global distribution. This type of distribution, unlike standard distributions, is used to only export globals.

You still use the Load a Distribution option to install global distributions. Unlike loading a standard distribution, however, KIDS installs global distributions immediately from the Load a Distribution option. Also, there is no queueing of the installation.

A global distribution can only contain one transport global, and the transport global can only export globals. You know that the distribution you're installing is a global distribution rather than a standard distribution, because when you load it with the Load a Distribution option, KIDS will indicate the following:

Figure 23-14. KIDS Global distribution—Sample message

```
This is a Global Distribution. It contains Global(s) that will
update your system at this time. The following Global(s) will be installed:
```

The Load a Distribution option lists each global that will be installed from the distribution. Each global in the list is marked OVERWRITE or REPLACE:

- OVERWRITE—Load the global *without* purging the site's version of the global beforehand.
- REPLACE—Purge the site's version of the global first, and then load the global.

You are given two chances to abort the installation of the global distribution. If you answer YES to both questions, the globals in the global distribution are installed immediately.

Purging the BUILD and INSTALL Files

Each KIDS installation adds one entry to the BUILD (#9.6) and INSTALL (#9.7) files for every transport global installed from the distribution.

REF: For information about purging these files, please refer to the discussion of the Purge Build or Install Files option in the "Purge Build or Install Files" topic in the "KIDS: System Management—Utilities" chapter in this manual.

Figure 23-15. Installation of a global distribution—Load a Distribution option

```
Select Installation Option: Load a Distribution
Enter a Host File: [DMANAGER]XGGLOBAL.DAT

KIDS Distribution save on Jan 26, 2004@12:58:25
Comment: GLOBAL PACKAGE

This Distribution contains the following Transport global(s):
     GLOBAL PACKAGE 1.0

This is a Global Distribution. It contains Global(s) that will
update your system at this time. The following Global(s) will be installed:
^XGRON(1)     Overwrite
^XGRON("PX")  Replace
^XGRON("TX")  Overwrite

If you continue with the Load, the Global(s) will be
Installed at this time.

Want to Continue with Load? YES// <Enter>
Loading Distribution...


Globals will now be installed, OK? YES// <Enter>


Installing Globals...
               Jan 26, 2004@13:04:16


 GLOBAL PACKAGE 1.0 Installed.
               Jan 26, 2004@13:04:17


Select Installation Option:
```

Alpha/Beta Tracking

Kernel provides a mechanism for tracking and monitoring installation and option usage during the alpha and beta testing phases of VistA software applications. This tool is primarily intended for application developers to use in monitoring the testing process at local test sites.

NOTE: In VA terminology "Alpha" and "Beta" testing are defined as follows:

- Alpha Testing—VistA test software application is running in a site's Test account.
- Beta Testing—VistA test software application is running in a site's Production account.

Alpha/Beta Tracking provides the following services to both developers and IRM personnel:

- Notification when a new alpha or beta software version is installed at a site.
- Periodic option usage reports for alpha or beta options being tracked.
- Periodic listings of errors in the software's namespace that are currently in alpha or beta test at the site.

The following options are provided on the Alpha/Beta Test Option Usage Menu [XQAB MENU], which is located on the Operations Management menu [XUSITEMGR]. These options allow developers and IRM personnel to monitor Alpha/Beta Tracking at a site:

- Errors Logged in Alpha/Beta Test (QUEUED) option [XQAB ERROR LOG XMIT]
- Actual Usage of Alpha/Beta Test Options option [XQAB ACTUAL OPTION USAGE]
- Low Usage of Alpha/Beta Test Options option [XQAB LIST LOW USAGE OPTS]
- Print Alpha/Beta Errors (Date/Site/Num/Rou/Err) option [XQAB ERR DATE/SITE/NUM/ROU/ERR]
- Send Alpha/Beta Usage to Programmers option [XQAB AUTO SEND]

REF: For more detailed information about and description of the Alpha/Beta Tracking functionality (e.g., starting, stopping, and monitoring options), please refer to the "Alpha/Beta Tracking" topic in Chapter 14, "KIDS: Developer Tools," in the *Kernel Developer's Guide*.

24 KIDS: System Management—Utilities

KIDS provides the following utility options:

Figure 24-1. KIDS Utilities menu options

```
Kernel Installation and Distribution System...                          [XPD MAIN]
  Utilities...                                                        [XPD UTILITY]
    Build File Print                                                [XPD PRINT BUILD]
    Install File Print                                         [XPD PRINT INSTALL FILE]
    Convert Loaded Package for Redistribution                     [XPD CONVERT PACKAGE]
    Display Patches for a Package                           [XPD PRINT PACKAGE PATCHES]
    Purge Build or Install Files                                    [XPD PURGE FILE]
    Rollup Patches into a Build                                  [XPD ROLLUP PATCHES]
    Update Routine File                                          [XPD ROUTINE UPDATE]
    Verify a Build                                                 [XPD VERIFY BUILD]
    Verify Package Integrity                                    [XPD VERIFY INTEGRITY]
```

These utilities can be used both by developers and by sites who install software created by KIDS.

Build File Print Option

The Build File Print option [XPD PRINT BUILD] prints out the build entry for a software application. It lists the complete definition of the software, including all files, components, install questions, and the environment, pre-install, and post-install routines.

Figure 24-2. Build File Print option—Sample output

```
PACKAGE: ZXG DEMO 1.0                                                PAGE 1
-------------------------------------------------------------------------------
NATIONAL PACKAGE:
DESCRIPTION:
Package containing demonstration of ZXG* functions.

ENVIRONMENT CHECK : ZXGENV
PRE-INIT ROUTINE : ZXGPRE
POST-INIT ROUTINE: ZXGPOS


                                     UP    SEND  DATA               USER
                                     DATE  SEC.  COMES  SITE   RSLV  OVER
FILE #      NAME                     DD    CODE  W/FILE DATA   PTS   RIDE
-------------------------------------------------------------------------------

662105      ZXG DEMO                 YES   YES   NO


PRINT TEMPLATE:
   ZXG PRINT    FILE #662105                SEND TO SITE

ROUTINE:
   ZXGC00                                   SEND TO SITE
   ZXGC01                                   SEND TO SITE
   ZXGC02                                   SEND TO SITE
   ZXGC03                                   SEND TO SITE
   ZXGC04                                   SEND TO SITE
   ZXGC05                                   SEND TO SITE
   ZXGC06                                   SEND TO SITE
   ZXGC07                                   SEND TO SITE
   ZXGC08                                   SEND TO SITE

OPTION:
   ZXG TEST                                 SEND TO SITE

INSTALL QUESTIONS:
```

Install File Print Option

The Install File Print option [XPD PRINT INSTALL FILE] prints out the results of an installation, as stored in the INSTALL file (#9.7). Use this option to check on the status of an installation in progress or to print out the results of a completed installation.

Figure 24-3. Install File Print option—Sample output

```
PACKAGE: ZXG DEMO 1.0                                          PAGE 1
                                          COMPLETED          ELAPSED
------------------------------------------------------------------------
STATUS: Install Completed                DATE LOADED: FEB 07, 2004@07:51:59
NATIONAL PACKAGE:

INSTALL STARTED: FEB 07, 2004@07:52:14      07:52:23            0:00:09

ROUTINES:                                   07:52:15            0:00:01

PRE-INIT CHECK POINTS:
XPD PREINSTALL STARTED                      07:52:15
XPD PREINSTALL COMPLETED                    07:52:15

FILES:
ZXG DEMO                                    07:52:16            0:00:01

PRINT TEMPLATE                              07:52:17            0:00:03
OPTION                                      07:52:21            0:00:02

POST-INIT CHECK POINTS:
XPD POSTINSTALL STARTED                     07:52:21
XPD POSTINSTALL COMPLETED                   07:52:21

INSTALL QUESTION PROMPT                                            ANSWER

XPZ1   Want to DISABLE Scheduled Options, Options and Protocols    NO
MESSAGES:

 Install Started for ZXG DEMO 1.0 :
               Feb 07, 2004@07:52:14

 Installing Routines:
               Feb 07, 2004@07:52:15

 Running Pre-Install Routine: ^ZXGPRE

 Installing Data Dictionaries:
               Feb 07, 2004@07:52:16

 Installing PACKAGE COMPONENTS:

 Installing PRINT TEMPLATE

 Installing OPTION
               Feb 07, 2004@07:52:21

 Running Post-Install Routine: ^ZXGPOS

 Updating Routine file...
```

```
Updating KIDS files...

ZXG DEMO 1.0 Installed.
              Feb 07, 2004@07:52:23
```

Convert Loaded Package for Redistribution Option

Use the Convert Loaded Package for Redistribution option [XPD CONVERT PACKAGE] to add software to an existing distribution.

A KIDS distribution can transport one or more software applications. What if you want to add additional software to an existing distribution? For example, suppose you have a distribution for a software application. Further suppose that patches are transported as individual KIDS software, and you want to add all existing patches to the software's distribution? The Convert Loaded Package for Redistribution option [XPD CONVERT PACKAGE] lets you do this.

In Figure 24-4 and Figure 24-5, distributions for a software application (i.e., ZXG 1.0) and a patch (i.e., ZXG*1.0*1) are both loaded. The Convert Loaded Package for Redistribution option is used to build a new distribution combining both original distributions.

Follow these steps to create a new distribution from existing distributions:

1. Load the original distributions (there is no need to install them, however).

   In this example, we would load the distributions for ZXG 1.0 and ZXG*1.0*1 (but we would not install them).

2. Use the Convert Loaded Package for Redistribution option. It lets you choose loaded transport globals, and transfers them into a format ready for export. Also, it creates build entries for each software application contained in the distributions. This allows you to create a new distribution containing the transport globals from the existing distributions. Patch XU*8.0*44 added the "Want to make the Transport Globals Permanent? NO//" prompt, answering YES to this prompt flags the global so that it is not deleted after the transportation. This provides a "Gold" account or library of software and patches that are included in a Transport Global.

In this example, we would first convert the loaded distribution ZXG 1.0 into a form ready to re-distribute:

Figure 24-4. Convert Loaded Package for Redistribution—Sample user dialogue (1 of 2)

```
Select Utilities Option: Convert Loaded Package for Redistribution
Select INSTALL NAME: ZXG 1.0 <Enter>         Loaded from Distribution

This distribution was loaded on Feb 28,2004@08:15:05 with header of

It consisted of the following Install(s):
ZXG 1.0

Want to make the Transport Globals Permanent? NO//YES
Want to continue with the conversion of the package(s)? NO//YES
  ** DONE **

Select Utilities Option:
```

Then we would convert the patch distribution, ZXG*1.0*1, into a form ready to re-distribute:

Figure 24-5. Convert Loaded Package for Redistribution—Sample user dialogue (2 of 2)

```
Select Utilities Option: Convert Loaded Package for Redistribution
Select INSTALL NAME: ZXG*1.0*1 <Enter>         Loaded from Distribution

This distribution was loaded on Feb 28,2004@08:15:35 with header of

It consisted of the following Install(s):
ZXG*1.0*1

Want to make the Transport Globals Permanent? NO//YES
Want to continue with the conversion of the package(s)? NO//YES
  ** DONE **
```

3. Create the new distribution with the Transport a Distribution option. Select each build from the original distributions that you want to be part of the new distribution. For each build that you select, you should be told that the transport global already exists and be asked if you want to use this transport global. Answer YES in each case to use the current transport global.

   Once you have selected all of the builds for the new distribution, go ahead and create the new distribution.

In this example, we create a new distribution containing both ZXG 1.0 (the original software application) and ZXG*1.0*1 (an added software application):

Figure 24-6. Transport a Distribution—Sample user dialogue

```
Select Edits and Distribution Option: Transport a Distribution

Enter the Package Names to be transported. The order in which they are entered
will be the order in which they are installed.

First Package Name: ZXG 1.0 <Enter>  **Transport Global exists**
    Use this Transport Global? YES
Another Package Name: ZXG*1.0*1 <Enter>  **Transport Global exists**
    Use this Transport Global? YES
Another Package Name: <Enter>

Order
  1    ZXG 1.0    **will use current Transport Global**
  2.   ZXG*1.0*1    **will use current Transport Global**

OK to continue? NO//YES

Enter a Host File: ZXG1.KID
Header Comment: PATCHED DISTRIBUTION ZXG 1.0

     ZXG 1.0...
     ZXG*1.0*1...

Package Transported Successfully
```

NOTE: Changing a distribution's build entries before redistributing is *not* recommended.

Display Patches for a Package Option

The Display Patches for a Package option [XPD PRINT PACKAGE PATCHES] prints all patches installed for a software application. It displays the Date Installed and who installed the patches. It optionally will print the description of the patch. All the displayed information comes from the PACKAGE file (#9.4).

Figure 24-7. Display Patches for a Package option—Sample user dialogue

```
Select Utilities Option: Display Patches for a Package
Select PACKAGE NAME: KERNEL
Select VERSION: 8.0// <Enter>          07-29-95
Do you want to see the Descriptions? NO// <Enter>
DEVICE: HOME// <Enter>  SYSTEM



PACKAGE: KERNEL     Oct 09, 2004 1:32 pm                          PAGE 1
PATCH #             INSTALLED                    INSTALLED BY
-------------------------------------------------------------------
VERSION: 8.0        JUL 29, 2004                 KRNUSER,TEN

   28               APR 25, 2004                 KRNUSER,NINE
   20 SEQ #23       FEB 09, 2004                 KRNUSER,NINE
   32 SEQ #24       MAY 15, 2004                 KRNUSER,NINE
   23 SEQ #25       MAY 17, 2004                 KRNUSER,TEN
   39 SEQ #26       JUL 19, 2004                 KRNUSER,ELEVEN
   26 SEQ #27       JUN 01, 2004                 KRNUSER,TEN
   27 SEQ #28       JUN 13, 2004                 KRNUSER,NINE
   24 SEQ #29       JUN 30, 2004                 KRNUSER,TEN
   40 SEQ #30       AUG 28, 2004                 KRNUSER,ELEVEN
   41 SEQ #31       AUG 29, 2004                 KRNUSER,TEN
   29 SEQ #32       AUG 30, 2004                 KRNUSER,NINE
```

Purge Build or Install Files Option

Each KIDS installation adds one entry to the BUILD (#9.6) and INSTALL (#9.7) files for every transport global installed from the distribution. You can use the Purge Build or Install Files option [XPD PURGE FILE] to purge entries in these files.

The first question the option asks is which file to purge, the BUILD (#9.6) or INSTALL file (#9.7). Choose one of these files.

The next question asked is the number of versions to retain.

Versions to Retain

When you choose to retain some number entries for a software application, the option *must* decide which entries are most recent. The Purge Install or Build Files option uses numeric order based on software version number to decide which entries are the most recent. When there are multiple entries for the same version number (for example, alpha or beta installs took place), the following order of precedence is used:

1. Released Version is the most recent (version number contains no letters, such as 8.0)
2. Beta Test Version (version number contains V, such as 8.0V10)
3. Alpha Test Version (version number contains T, such as 8.0T10)

Selecting Software Names for Purging

After versions to retain, the next prompt is "Package Name." You can enter a partial or full software application name. You will continue to be prompted for additional software names until you simply press the **<Enter>** key without making any further entries at the "Package Name" prompt.

- **Packages (Software)—**To select software entries for purging, at the "Package Name" prompt, enter a partial or full software application name. You can optionally enter partial or full version numbers. The list of candidates for purging will contain all entries (excluding patch entries) whose first characters match all characters in the software name that you specify. If you enter "ALL", all software (but *not* patches) will be selected for purging.
- **Patches—**Patches are a special case. To select patch entries for purging, you *must* enter the full namespace of the patch, the full version number, and an asterisk. You can optionally add a partial or full patch number after the asterisk. The list of candidates for purging will contain all entries whose first characters match all characters in the string you specify.

Figure 24-8. Purge or Install Files option—Sample user dialogue

```
Select Utilities Option: Purge or Install Files

      Select one of the following:

           B         Build
           I         Install

Purge from what file: B
Versions to Retain:  (0-100): 1// 0
Package Name: ALL// ZXG
Another Package Name: <Enter> ...

Package(s) in Build file, Don't retain any versions          Page 1
-------------------------------------------------------------------
ZXG 1.0
ZXG 2.0
ZXG 3.0

OK to DELETE these entries? NO// YES

Select Utilities Option:
```

Purging Selected Entries

Based on the software name you enter and the number of entries you ask to retain, the option lists the software it finds to purge. If you answer YES to the "OK to DELETE these entries? NO//" prompt, the option purges the listed entries.

Reasons to Retain BUILD and INSTALL File Entries

- **BUILD file—**Entries in the BUILD file (#9.6) are created by the software developers and identify every component in the software. BUILD file (#9.6) entries also contain the checksums for a software application's components. You may want to retain the build entry for the most recent versions of installed software, so that you can verify the checksums of the loaded software against its original checksums.
- **INSTALL file—**Each entry in the INSTALL file (#9.7) contains a record of the installation for a given software application. This information is useful as a record of each installation.

Rollup Patches into a Build Option

The Rollup Patches into a Build option [XPD ROLLUP PATCHES] finds all the patches for a software application and add their individual BUILD file (#9.6) definitions to the software's BUILD file (#9.6) definition. This will enable you to create a single BUILD file (#9.6) entry that contains the definition for the patched software.

KIDS checks the BUILD file (#9.6) and lists all KIDS patches with a matching software name and version number. The list of patches is not necessarily displayed in patch sequence number.

This list only includes KIDS patches. Also, it does *not* include any pre- or post-install routines. You can use the Edit a Build option to further modify the build and add any additional patches.

Figure 24-9. Rollup Patches into a Build option—Sample user dialogue

```
Select Utilities Option:  Rollup Patches into a Build

Rollup patches into Build:    KERNEL 8.0T20 <Enter>     KERNEL
This package already contains the following patches:
   XU*8.0T20*4

The following patches can be rolled into Package RON 8.0T20
     XU*8.0T20*5
     XU*8.0T20*6
     XU*8.0T20*7
     XU*8.0T20*8
     XU*8.0T20*11

OK to continue? YES// <Enter>
...SORRY, HOLD ON.....................................................
...............Done.
```

Update Routine File Option

The Update Routine File option [XPD ROUTINE UPDATE] updates the ROUTINE file (#9.8) to match the routine set stored on the current system.

Ideally, the ROUTINE file (#9.8) would contain an entry for every routine on the current system. However, the ROUTINE file (#9.8) does *not* get updated automatically when routines are added to or deleted from the system. But KIDS needs the ROUTINE file (#9.8) so that it can store the list of routines in a software application as pointers to the ROUTINE file (#9.8) (rather than relying on namespace alone).

Developers should use this option to update the ROUTINE file (#9.8) before editing the routine component in a build entry, to ensure that all the routines they want to include in a software application can be selected by the routines' matching entries in the ROUTINE file (#9.8).

If you answer YES to the question "Want me to clean up the Routine file before updating?", the option goes through the ROUTINE file (#9.8) and deletes any entries across all namespaces that have no matches with an actual routine on the current system. As of Kernel Patch XU*8.0*393, however, any routine that has been marked in the CHECKSUM REPORT field (#6) in the ROUTINE file (#9.8) as "National" will *not* be deleted during the clean up the Routine File phase of the update.

Then, the Update Routine File option re-populates the ROUTINE file (#9.8) with all routines currently on the system for the namespaces you enter (you can exclude parts of a namespace if you want, as well).

Figure 24-10. Update Routine File option—Sample user dialogue

```
Select Utilities Option: Update Routine File


Routine Namespace: XU
Routine Namespace: -XUI
Routine Namespace: <Enter>

NAMESPACE  INCLUDE                 EXCLUDE
           -------                 -------
           XU                      XUI

OK to continue? YES// <Enter>

Want me to clean up the Routine File before updating? YES// <Enter>
...SORRY, THIS MAY TAKE A FEW MOMENTS...    ...Done.
```

Verify a Build Option

The Verify a Build option [XPD VERIFY BUILD] checks whether a build entry's listed components actually exist on the current system. This is useful for developers who are preparing to create a transport global. They can check that there are actual components on the system matching the components requested in the build entry, in advance of trying to create a transport global. Therefore, developers should use the Verify a Build option *before* creating transport globals from build entries.

For any component in the build entry that does not actually exist on the system, the option outputs a one-line message identifying the missing component, with the appellation **NOT FOUND**. The developer is also prompted with "Do you want to remove the missing Files? NO//". This allows you to verify if the missing component should in fact be removed from the build. If the missing component is required, the developer should create the missing component for the build entry before creating a transport global.

Figure 24-11. Verify a Build option—Sample user dialogue

```
Select Utilities Option:  Verify a Build
Select BUILD NAME:    XU*8.0*11 <Enter>     KERNEL
 File #8995  ** NOT FOUND **
Do you want to remove the missing Files? NO// <Enter>

  ** DONE **

Select Utilities Option:
```

Verify Package Integrity Option

You can use the Verify Package Integrity option [XPD VERIFY INTEGRITY] to compare checksums of software components on the system against the checksums of the components when they were originally transported. Any discrepancies are reported. Currently, routines are the only components that are checked, but checksums will be extended to other software components in the future.

The checksums of components for the currently installed software are verified against checksums stored in the BUILD file (#9.6) entry for the software. If the most recent version of the BUILD file (#9.6) entry for a software application has been purged, the Verify Package Integrity option will no longer be able to verify checksums for the loaded software. Because of this, in most cases you should *not* purge the most recent build entry for a software application.

NOTE: As of Kernel Patch XU*8.0*369, the integrity checking CHECK1^XTSUMBLD API supports the Compare local/national checksums report option [XU CHECKSUM REPORT].

NOTE: As of Kernel Patch XU*8.0*393, KIDS was modified to send a message to a server on FORUM when a KIDS build is sent to a Host File Server (HFS) device. This message contains the checksums for the routines in the patch. The server on FORUM matches the message with a patch if the sending domain is authorized on FORUM. There is no longer a need for developers to manually include routine checksums (either CHECK^XTSUMBLD and/or CHECK1^XTSUMBLD) in the patch description. The patch module will include the before and after CHECK1^XTSUMBLD values in the Routine Information section at the end of the patch document.

With changes in the National Patch Module (NPM) on FORUM, when the patch is released the checksums for the routines are moved to the ROUTINE file (#9.8) on FORUM. The checksum "before" values will come from the FORUM ROUTINE file (#9.8) and are considered the GOLD standard for released checksums. The local site's Compare local/national checksums report option [XU CHECKSUM REPORT] uses the FORUM ROUTINE file (#9.8) as its source to create reports showing any routines that do *not* match..

This patch also modified the KIDS BUILD file (#9.6) by adding the TRANSPORT BUILD NUMBER field (#63) used to store a build number that is incremented each time a build is made. This build number is added to the second line of each routine in the 7th "**;**" piece. This makes it easy to tell if a site is running the current release during testing and afterword. The leading "B" found in the checksum tells the code what checksum API to use.

VI Toolkit

This section provides descriptive information about the set of software utilities furnished by Kernel Version 8.0 and Kernel Toolkit Version 7.3 (a.k.a. "Toolkit"), describing how these tools can be used for the management and definition of development projects.

The major areas of the Kernel Toolkit described in this section are listed below:

- **Multi-Term Look-Up (MTLU)**

  Multi-Term Look-Up (MTLU) utilities provide a method of enhancing the lookup capabilities of associated VA FileMan files. Multi-Term Look-Up (MTLU) is an adaptation of a tool developed by the Indian Health Service (IHS) which was originally made generic by the Albany Office of Information Field Office (OIFO). MTLU does the following:

  – Tests ICD diagnosis and procedure codes, CPT codes, and other commonly used references that have been entered in the LOCAL LOOKUP file (#8984.4). Optionally, terms or phrases can be entered into the LOCAL KEYWORD (#8984.1), LOCAL SHORTCUT (#8984.2), or LOCAL SYNONYM (#8984.3) files.

  – Prints a list of shortcuts, keywords, or synonyms from a specified reference file in the LOCAL LOOKUP file (#8984.4).

  – Adds or deletes a reference file from a site's LOCAL LOOKUP file (#8984.4).

  – Enters new or edit existing shortcuts, keywords, or synonyms to the LLOCAL LOOKUP file (#8984.4).

- **Routine Tools**

  Routine Tools provide a set of generic tools to aid the VistA development community and IRM in analysis, writing, and testing of code. These tools are used by VistA developers to support distinct tasks. Routine Tools do the following:

  – Promote standard program interfaces.

  – Check adherence to programming standards and correct syntax with the XINDEX utility.

  – Provide standard error trapping, storing, and reporting.

  – Customize and tunes site parameters for local requirements.

  – Provide M function libraries.

  – Provide a portable routine and global editor.

  – Provide a Kermit file transfer utility.

  – Provide a Multi-Term Look-Up (MTLU) utility for enhanced VA FileMan lookups.

  – Provide software project management utilities.

- **Verification Tools**

  Verification Tools are a set of generic tools to aid the VistA development community and IRM in reviewing M code. These tools are used by VistA developers to support distinct tasks. Verification Tools provide the following:

  – Tools used for comparison of routines and data dictionaries.

  – A tool used to record routine text indicated in the file used to maintain changes in routines.

Where applicable, each major area of Kernel Toolkit is described first in terms of its user interface then in terms of system management implications, showing the menu that can be used to accomplish the task at hand.

REF: Kernel and Kernel Toolkit Application Program Interfaces (APIs) are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

NOTE: The *Parameter Tools Supplement to Patch Description (Patch XT*7.3*26)* explains the functions available with the use of the Parameter Tools, provides information on the Kernel PARAMETERS file (#8989.5), and describes the associated Application Program Interfaces (APIs).

REF: This documentation can be downloaded from the VHA Software Document Library (VDL) Web site at:

http://www.va.gov/vdl/application.asp?appID=12

The following Kernel Toolkit chapters were removed from the "Toolkit" section of this manual because they are superseded by subsequent software and documentation:

- **Duplicate Record Merge**

  The Kernel Toolkit "Duplicate Record Merge" documentation is superseded by the *Duplicate Record Merge: Patient Merge* software/documentation (i.e., Kernel Toolkit Patch XT*7.3*23).

  The Duplicate Record Merge functionality provides a developer Merge Shell with options that allow users to check data files for duplicate entries and merge those entries if any are found. These options provide functionality to combine duplicate records based on conditions established in customized applications. The Merge Shell was originally developed by Indian Health Service (IHS) to support their Multi-Facility Integration Project.

  REF: The *Duplicate Record Merge: Patient Merge* documentation is made available online in both Microsoft Word (DOC) and Adobe Acrobat (PDF) formats. It can be downloaded from the VHA Software Document Library (VDL) at the following Web address:

  http://www.va.gov/vdl/application.asp?appid=2

- **Capacity Management**

  The Kernel Toolkit "Capacity Management" documentation is superseded by the following software/documentation:

  – Capacity Management (CM) Tools Version 2.0

  – Resource Usage Monitor (RUM) Version 2.0

  – Statistical Analysis of Global Growth (SAGG) Version 1.8

REF: The Capacity Management-related documentation is made available online in both Microsoft Word (DOC) and Adobe Acrobat (PDF) formats. It can be downloaded from the VHA Software Document Library (VDL) at the following Web address:

- Capacity Management (CM) Tools:
  http://www.va.gov/vdl/application.asp?appid=129
- Resource Usage Monitor (RUM):
  http://www.va.gov/vdl/application.asp?appid=130
- Statistical Analysis of Global Growth (SAGG):
  http://www.va.gov/vdl/application.asp?appid=115

NOTE: Kernel Toolkit Patch XT*7.3*102 removed all options, routines, and files associated with the following menus and options:

- VPM VAX/ALPHA Capacity Management ...
- Move Host File to Mailman
- Response Time Log Options ...

The following namespace options and routines are also removed:

- XUCM*
- XUCS*
- XURTL*
- XTCM DISK2MAIL(option)
- XTCMXTCMFILN (routine)

Data dictionaries and data have been deleted for the following VA FileMan compatible files:

- Global ^XUCM:
  - CM DAILY STATISTICS (#8986.6)
  - CM DISK DRIVE RAW DATA (#8986.5)
  - CM METRICS (#8986.4)
  - CM NODENAME RAW DATA (#8986.51)
  - CM SITE DISKDRIVES (#8986.35)
  - CM SITE NODENAMES (#8986.3)
  - CM SITE PARAMETERS (#8986.095)
  - VPM RESPONSE TIME DATA (#8986.098)
- Global ^%ZRTL:
  - RESPONSE TIME (#3.091)
  - RT DATE_UCI,VOL (#3.092)
  - RT RAWDATA (#3.094)

Data has been deleted for the following non-VA FileMan compatible global:

- ^%ZRTL(3)
- ^%ZRTL("RTH")

NOTE to System Managers: The ^XUCM and %ZRTL globals can be removed from your database after installation of this patch; however, please make sure no local routines access these globals before doing so.

25 Multi-Term Look-Up (MTLU)

Overview

This chapter contains an introduction and functional description, site implementation instructions for Multi-Term Look-Up (MTLU), and the option documentation.

Introduction to Multi-Term Look-Up (MTLU)

Many medical information systems depend on the standardized encoding of diagnoses and procedures for reports, searches, and statistics. The ICD DIAGNOSIS (#80), ICD OPERATION/PROCEDURE (#80.1), and CPT (#81) files are among some of the more critical files. The Multi-Term Look-Up utility increases the accessibility of the information in these files by associating user-supplied words or phrases with terms found in a more descriptive, free-text field.

Multi-Term Look-Up allows:

- Local setup of virtually any reference file.
- Modification of the behavior of the "special" lookup by defining shortcuts, synonyms, or keywords.

MTLU integrates with any software that uses a reference file that has been entered into a site's LOCAL LOOKUP file (#8984.4).

Functional Description

The Multi-Term Look-Up (MTLU) utility provides a method of enhancing the lookup capabilities of associated software applications. This utility is comprised of the following options:

- The Multi-Term Lookup (MTLU) option [XTLKLKUP] is used to test ICD diagnosis and procedure codes, CPT codes, and other commonly used references that have been entered in the LOCAL LOOKUP file (#8984.4). Optionally, terms or phrases may be entered into the LOCAL KEYWORD (#8984.1), LOCAL SHORTCUT (#8984.2), or LOCAL SYNONYM (#8984.3), files.
- The Print Utility option [XTLKPRTUTL] is used to print a list of shortcuts, keywords, or synonyms from a specified reference file in the LOCAL LOOKUP file (#8984.4). This list can be sorted alphabetically by name or numerically by code.
- The Delete Entries from Look-Up option [XTLKMODPARK]is used to delete a reference file from a site's LOCAL LOOKUP file (#8984.4). This option should be used as an IRM/developer utility and can only be accessed by holders of the XTLKZMGR security key.
- The Add Entries To Look-Up File option [XTLKMODPARS]is used to add reference files to a site's LOCAL LOOKUP file (#8984.4). This option should be used as an IRM/developer utility and can only be accessed by holders of the XTLKZMGR security key. In order to add entries with this option, DUZ(0) *must* be set to an at-sign ("@"; programmer access).

- The Add/Modify Utility option [XTLKMODUTL] is used to enter new or edit existing shortcuts, keywords, or synonyms to the LOCAL LOOKUP file (#8984.4) as described below:
  - The Shortcuts option [XTLKMODSH] is used to enter new or edit existing shortcuts to the LOCAL LOOKUP file (#8984.4).
  - The Keywords option [XTLKMODKY] is used to enter new or edit existing keywords to the LOCAL LOOKUP file (#8984.4).
  - The Synonyms option [XTLKMODSY] is used to enter new or edit existing synonyms to the LOCAL LOOKUP file (#8984.4).

Usage Considerations

MTLU provides users and developers with the ability to perform specialized lookups on database files using standard VA FileMan calls. These files typically comprise a number or "term" in the .01 field and a longer description or definition in some other field.

In the simplest application of MTLU, a special lookup routine (XTLKDICL) is defined in the file's data dictionary (DD), then a MUMPS cross-reference is applied to the description/definition field. Options are available to fully configure a file for use with MTLU. FileMan is used to create/build the cross-reference. To set the cross-reference, text from the selected field is passed to a tokenizing routine (XTLKTOKN). Trivial words are filtered by an expanded Key Word In Context (KWIC), then each remaining token is added to the cross-reference.

To request a lookup, users and developers can pass in words or phrases. Their input is similarly tokenized. However, only terms associated with *all* tokens entered are found. Input can be generalized using partial words or fewer words as well as lexical variants. For example, using the FileMan Inquire to File Entries option on the ICD DIAGNOSIS file (#80) one could first enter "MALIG". MTLU informs the user which terms apply to the search, "MALIG/MALIGNANT", and that 447 matches are found. To be more specific, the user might enter "MALIG LIP" to request all malignancies associated with the lip. In this case, only 12 matches are found. The user can further screen searches by using the Not-Sign (**'**) before a word or phrase. To request all malignancies of the lip *except* those of the lower lip, one could enter "MALIG LIP 'LOWER" and obtain 10 matches. Though the term "malignancies" may not exist in the lookup file, MTLU might still produce a match. When a term contains a suffix that does not produce a match, MTLU removes the suffix and continues the search.

REF: For more information on the Inquire to File Entries option, please refer to the "How to Display and Print File Data" chapter in the *VA FileMan User Manual*.

Three additional files are supplied that can dramatically alter the predictable behavior described above. They are checked in the following order against the user's entry:

1. LOCAL SHORTCUT file (#8984.2): Shortcuts are used to point to a single term. They can be a word or phrase. MTLU checks the user's entry against this file first for an exact match. If found, the lookup displays only the associated entry. A single shortcut cannot point to multiple terms.
2. LOCAL SYNONYM file (#8984.3): Synonyms can be associated with many terms in a file because they can be associated with multiple "tokens" rather than a specific term. For example, CANCER can be defined as a synonym of "MALIG", "TUMOR", and "LEUKEMIA". When the user enters CANCER, the lookup finds *all* terms associated with the three tokens as if each had been entered separately. Compared with the example above, CANCER returns 534 matches. CANCER LIP returns the same 12 matches as MALIG LIP.
3. LOCAL KEYWORD file (#8984.1): A keyword or phrase can be associated with a single term, much like a shortcut; however, it can also be associated with multiple terms, and multiple keywords can be associated with the same term.

The term SMOKER can be used as a synonym or keyword. As a keyword, one can associate it with a few *specific* diseases. As a synonym, properly selected tokens might result in a display of all smoking-related diseases.

Recall that MALIG results in 447 matches. If this were used as a shortcut to a single entry, MTLU would display only that entry and the remaining 446 would never be displayed.

These files add some control over the behavior of certain lookups. However, developers should use extreme caution when placing entries in these files to ensure that results are predictable and appropriate for both users and other VistA software developers.

The decision to populate them for a given lookup file depends on whether or not a commonly used word or phrase results in any matches during a lookup. If not, it is a candidate. The LOCAL KEYWORD (#8984.1), LOCAL SHORTCUT (#8984.2), and LOCAL SYNONYM (#8984.3) files should only be populated with common words or phrases.

In the event that a search produces no matches, MTLU continues with a standard FileMan search.

User Interface

Multi-Term Look-Up Menu Options

The following is a description of the Multi-Term Lookup Main Menu [XTLKUSER2] which can be selected from the Application Utilities menu [XTMENU]. The options are described in the same order as they appear on the screen:

Figure 25-1. Multi-Term Lookup Main Menu options

```
Application Utilities ...                                              [XTMENU]
   Multi-Term Lookup Main Menu ...                                   [XTLKUSER2]
     Multi-Term Lookup (MTLU)                                         [XTLKLKUP]
     Print Utility                                                   [XTLKPRTUTL]
     Utilities for MTLU ... <Locked with XTLKZMGR>                 [XTLKUTILITIES]
     Delete Entries From Look-up <Locked with XTLKZMGR>              [XTLKMODPARK]
     ST Add Entries To Look-Up File <Locked with XTLKZMGR>           [XTLKMODPARS]
     Add/Modify Utility...                                            [XTLKMODUTL]
```

Most MTLU options are described using the following methods:

- Introduction—A detailed description of the option is given. The introduction usually contains any necessary special instructions.
- Process Chart—The step-by-step flow of the option is illustrated, showing the various choices allowed at each prompt.
- Examples—In most cases, there is an example of what might appear on the screen when using the particular option. If the option produces a hardcopy output, an example of the output is usually given.

The phrase "You will be prompted for a device at this step" appears in the process chart when a device is asked for. A Standard Device Chart is shown on the next page. It provides assistance in answering prompts related to device selection.

The MTLU Process Charts do *not* contain documentation of the system's response to erroneous input. In certain instances, in order to preserve the integrity of previously entered data, the system does *not* allow the entry of a caret (^, sometimes referred to as an up-arrow). This might not be documented.

The following chart provides assistance in answering prompts related to device selection:

Standard Device Chart

Figure 25-2. Standard Device Chart

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
1	DEVICE:	Device name/number from your DEVICE file (#3.5) for report to print on	3
		'Q'UEUE to have report queued to print at a Later date/time	2
		<Enter> for report to Print on your screen	3
		Up-arrow <^>	6
2	DEVICE:	Device name/number from your DEVICE file (#3.5) for report to print on	3
		Up-arrow <^>	6
3	RIGHT MARGIN: 132//	*<Enter> to accept default, different RIGHT MARGIN Value, or up-arrow <^>	6
		*The next step depends on what you entered in Step 1: Device name/number	4
		"Q"	5
		<Enter> (The report appears on your screen)	6
4	WANT TO FREE UP THIS TERMINAL? NO//	<Enter> to accept default	6
		'Y'ES to free up terminal during report processing and to exit from the system	5
		Up-arrow <^>	6
5	REQUESTED TIME TO PRINT: NOW//	*<Enter> to accept default	6
		*Later date/time for report process to begin	6
		Up-arrow <^>	6
		*If <Enter> or later date/time is entered, the following message appears: "REQUEST QUEUED!"	
6	Return to the menu.		

Using the Multi-Term Lookup (MTLU) Option

The Multi-Term Lookup (MTLU) option [XTLKLKUP] is used to test the ICD diagnosis and procedure codes, CPT codes, and other commonly used references that have been entered in the LOCAL LOOKUP file (#8984.4) and have been associated with a shortcut, synonym, or keyword.

The system searches for entries in the following order: shortcut, synonym, then keyword. If you are entering a multi-term narrative (phrase), you can enter double spaces between each term to avoid a search of the LOCAL SHORTCUT file (#8984.2). When searching for a keyword phrase, the system searches for each word in the phrase and then displays all common entries. For example, if the keyword is FRACTURE FEMUR, the system searches for FRACTURE and then FEMUR and displays only those codes with a diagnosis containing both keywords or synonyms of those words.

The following process chart shows the prompts and steps involved in using the Multi-Term Lookup (MTLU) option:

Figure 25-3. Multi-Term Lookup (MTLU) option process chart

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
1	Lookup on which file?:	Name of entry in LOCAL LOOKUP file (#8984.4).....	2
		<?> for list of entries.....	1
		<Enter> or up-arrow <^>.....	4
2	NARRATIVE:	Existing shortcut, synonym, or keyword.....	3
	If a word, phrase, or symbol is entered that the system cannot identify, the following appears: "Narrative contained no usable words. The following word(s) was not used in this search: {word(s)} Search was unsuccessful."		
	The selected code or description is displayed. The system searches in the following order: shortcut, synonym, then keyword. If more than one entry is found, they are displayed, and you are prompted to select one. If only one entry is found, the following appears:		
3	OK? Y//	<Enter> to accept default.....	4
		'N'O.....	4
4	Return to the menu.		

The following is an example of what might appear on your screen when using the Multi-Term Lookup (MTLU) option:

Figure 25-4. Multi-Term Lookup (MTLU) option—Sample user entries

```
Lookup on which file?: ICD DIAGNOSIS

NARRATIVE: DIABETES MELLITUS
( DIABETES|DIABETIC MELLITUS )
....

The following 3 matches were found:

   1: 250.00  (250.00)
      DIABETES UNCOMPL ADULT/NIDDM
   2: 250.40  (250.40)
      DIAB RENAL MANIF ADULT/NIDDM
   3: 775.0  (775.0)
      INFANT DIABET MOTHER SYN

Select 1-3: 2
```

Using the Print Utility Option

The Print Utility option [XTLKPRTUTL] is used to print a list of shortcuts, keywords, or synonyms from a specified reference file in the LOCAL LOOKUP file (#8984.4). Both the shortcut and keyword lists can be sorted alphabetically by name or numerically by code. The synonym list, however, only prints alphabetically.

Since these lists can be long and the generation time consuming, it is suggested you queue the report to a device during off hours.

The following process chart shows the prompts and steps involved in using the Print Utility option:

Figure 25-5. Print Utility option process chart

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
1	Select one of the following: SH Shortcuts KE Keyword SY Synonyms Print which file?:	SH for Shortcuts.....2 KE for Keywords.....2 SY for Synonym.....3	
2	Select one of the following: A Alphabetic C Code Sort By?:	'A'lphabetic.....3 'C'ode.....3	
3	Print {Shortcuts, Keywords, or Synonyms} for which file?:	Name of entry in LOCAL LOOKUP file (#8984.4).....4 <?> for list of entries.....3 <Enter> or up-arrow <^>.....5	
4	You will be prompted for a device at this step.....1		
5	Return to the menu.		

The following is an example of what might appear on your screen when using the Print Utility option (an example of the output generated by this option is provided following the computer dialogue):

Figure 25-6. Print Utility option—Sample user entries and sample output

```
     Select one of the following:

          SH        Shortcuts
          KE        Keywords
          SY        Synonyms

      Print which file?: SH <Enter>  Shortcuts

     Select one of the following:

          A         Alphabetic
          C         Code

      Sort By?: A <Enter> lphabetic


Print Shortcuts for which file?: CPT
DEVICE:HOME// <Enter>    RIGHT MARGIN: 80// <Enter>

                                                    Sample output.

Shortcuts of the CPT file sorted by Name        NOV 23, 1994  13:36  PAGE 1
FREQUENTLY USED NARRATIVE                 ENTRY
------------------------------------------------------------------------

DREAM                                     01200
NIGHT                                     02400
SLEEP                                     01100
```

Using the Utilities for MTLU Option

The following is a list of the options and their descriptions that comprise the Utilities for MTLU menu [XTLKUTILITIES]. This option can only be accessed by holders of the XTLKZMGR security key:

- The Delete Entries From Look-Up option [XTLKMODPARK] deletes entries from the LOCAL LOOKUP file (#8984.4). In order to do this, there cannot be any shortcuts, synonyms, or keywords associated with the file to be deleted. This option should be used as an IRM/developer utility and can only be accessed by holders of the XTLKZMGR security key.
- The Add Entries To Look-Up File option [XTLKMODPARS] sets entries in the LOCAL LOOKUP file (#8984.4). This option should be used as an IRM/developer utility and can only be accessed by holders of the XTLKZMGR security key. In order to add entries with this option, DUZ(0) *must* be set to an at-sign ("@"; programmer access).
- The Add/Modify Utility option [XTLKMODUTL] is used to make or edit entries in the LOCAL KEYWORD (#8984.1), LOCAL SHORTCUT (#8984.2), and LOCAL SYNONYM (#8984.3) files.

Delete Entries from Look-Up Option

The Delete Entries From Look-Up option [XTLKMODPARK] is used to delete a reference file from a site's LOCAL LOOKUP file (#8984.4).

All shortcuts, synonyms, and keywords associated with the reference file you wish to delete *must* be canceled before you attempt to delete the file.

It should be noted that when a reference file is "killed" through this option, all variable pointers from the LOCAL KEYWORD (#8984.1) and LOCAL SHORTCUT (#8984.2) files are deleted. The special lookup routine for the file is also deleted.

Only holders of the XTLKZMGR security key, can access this option.

NOTE: Due to the brevity of this option, no process chart has been provided.

The following is an example of what might appear on your screen when using the Delete Entries From Look-Up option:

Figure 25-7. Delete Entries From Look-Up option—Sample user entries

```
Select LOCAL LOOKUP NAME: PROCEDURE MODIFIERS

Are you sure you want to delete PROCEDURE MODIFIERS? YES

Deleting from Local Lookup file.....
Deleting variable pointers from Local Keyword and Shortcut files.
Deleting special lookup routine from PROCEDURE MODIFIERS DD.
```

Add Entries To Look-Up File Option

The Add Entries To Look-Up File option [XTLKMODPARS] is used to add/edit reference files to a site's LOCAL LOOKUP file (#8984.4). Examples of files that a site might wish to enter in their LOCAL LOOKUP file (#8984.4) include: ICD DIAGNOSIS (#80), ICD OPERATION/PROCEDURE (#80.1), and CPT (#81).

Only holders of the XTLKZMGR security key, can access this option. In order to add entries with this option, DUZ(0) *must* be set to an at-sign ("@"; programmer access).

The process chart below shows the prompts and steps involved in using the Add Entries To Look-Up File option:

Figure 25-8. Add Entries To Look-Up File option process chart (1 of 2)

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
1	Select LOCAL LOOKUP NAME:	Name of new reference file you wish to enter in LOCAL LOOKUP file (#8984.4) <?> for file list Name of existing file <Enter> or up-arrow <^>	2 1 8 12
2	ARE YOU ADDING {reference file name} AS A NEW LOCAL LOOKUP (THE nTH)?	'Y'ES 'N'O	3 1
3	LOCAL LOOKUP NAME: {reference file name}//	<Enter> to accept default Other file name	4 4
4	LOCAL LOOKUP DISPLAY PROTOCOL:	Entry point for routine to determine the display format <Enter> to accept the internal default display format	5 5
	If the entry made at this step is not the same as the cross reference in the description field of the file, the software still functions, but it only uses the keywords entered in the LOCAL LOOKUP file (#8984.4).		

*Required field

Figure 25-9. Add Entries To Look-Up File option process chart (2 of 2)

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
* 5	INDEX:	Cross reference to be used to create new key-words.....6	
	NOTE: The following message is displayed : "...Ok, will now setup KEYWORD and SHORTCUT file DD's to allow terms for {reference file name} entries..."		
* 6	PREFIX: M//:	Letter(s) to be used to identify a variable pointer.....7	
7	The following reminder message is displayed: <REMINDER> Using 'Edit File', set the lookup routine, XTLKDICL, in {reference file name} DD	1	
	The selected file is displayed.		
8	...OK? YES//	<Enter> to accept default .....9 'N'O.....1	
9	LOCAL LOOKUP NAME: {reference file name}//	<Enter> to accept default.....10 Correct file name.....10	
10	LOCAL LOOKUP DISPLAY PROTOCOL: {protocol}//	<Enter> to accept default.....11 Correct entry point for routine to set display format.....11 <Enter> (no default) to accept the internal Default display format.....11	
11	INDEX: {index}//	<Enter> to accept default.....12 correct cross reference to be used to create new Keywords.....12	
12	Return to the menu.		

*Required field

The following is an example of what might appear on your screen when using the Add Entries To Look-Up File option:

Figure 25-10. Add Entries To Look-Up File option—Sample user entries

```
Select LOCAL LOOKUP NAME: PROCEDURE MODIFIERS
   ARE YOU ADDING 'PROCEDURE MODIFIERS' AS A NEW LOCAL LOOKUP (THE 4th)? Y <Enter>
(YES)
   LOCAL LOOKUP NAME: PROCEDURE MODIFIERS// <Enter>
   LOCAL LOOKUP DISPLAY PROTOCOL: <Enter>
INDEX: AIHS
...Ok, will now setup KEYWORD and SHORTCUT file DD's
   to allow terms for 'PROCEDURE MODIFIERS' entries...
PREFIX: M// <Enter>
  <REMINDER> Using 'Edit File', set the lookup routine, XTLKDICL, in PROCEDURE
MODIFIERS DD
Select LOCAL LOOKUP NAME: <Enter>
```

Add/Modify Utility Option

The Add/Modify Utility option [XTLKMODUTL] is used to enter new or edit existing shortcuts, keywords, or synonyms to the LOCAL LOOKUP file (#8984.4).

A shortcut is a word or phrase which recognizes one specific code or procedure. If you are adding a shortcut whose text duplicates the first part of an existing entry, you *must* enclose the new shortcut word or phrase in double quotes to prevent the system from matching it to existing terms.

A keyword is a word or phrase which corresponds to several related codes or procedures. Keywords are typically terms commonly used to describe a clinical entity. Entering a series of keywords separated by single spaces results in all of the keywords being added to the specified code.

A synonym is a word entered to expand the lookup capability of an existing term or terms in the LOCAL LOOKUP file (#8984.4). Synonyms would be used in cases where several words within the text of codes or procedures have the same diagnostic meaning (e.g., CANCER and MALIGNANCY). A synonym can be entered for an existing keyword or for a word in the diagnostic description or procedure (e.g., the term CANCER might be matched to the synonyms MALIGNANCY, LEUKEMIA, and CARCINOMA). When CANCER is referenced in the Multi-Term Lookup (MTLU) option, it recognizes all the codes and descriptions associated with MALIGNANCY, LEUKEMIA, and CARCINOMA.

NOTE: A synonym replaces the original word in the lookup process; therefore, to retain the original word in the search, it *must* be matched to itself as well as to other synonyms.

Words used as a shortcut should never be repeated as synonyms or keywords. Since the system searches for shortcuts first and stops when one is found, it cannot find duplicated words in the LOCAL SYNONYM (#8984.3) or LOCAL KEYWORD (#8984.1) files. Since searching all files for each word is time consuming, the search is done in this order so as to speedup the search process.

Since the add/modify functions for Shortcuts, Keywords, and Synonyms are considered separate options, a process chart for each is provided. The charts on the following pages show the prompts and steps involved in using the following options:

Figure 25-11. Add/Modify Utility menu options

```
Select Add/Modify Utility Option: ??

            SH Shortcuts                                                [XTLKMODSH]
            KE Keywords                                                 [XTLKMODKY]
            SY Synonyms                                                 [XTLKMODSY]
```

The Shortcuts option [XTLKMODSH], one of the three selections within the Add/Modify Utility option, is described below.

The following process chart shows the prompts and steps involved in using the Add/Modify Utility option when adding or editing a shortcut:

Figure 25-12. Add/Modify Utility option—Shortcuts process chart (1 of 2)

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
1	SH Shortcuts KE Keywords SY Synonyms Select Add/Modify Utility Option:	SH for Shortcuts <Enter> or up-arrow <^>	2 11
2	Additions/Modifications to Shortcuts in which file?	Name of entry in local reference file <?> for list of entries <Enter>	3 2 1
3	Select LOCAL SHORTCUT FREQUENTLY USED NARRATIVE:	New text you wish to use as a shortcut Existing shortcut term <Enter>	4 8 1
4	ARE YOU ADDING {'text'} AS A NEW LOCAL SHORTCUT? An at-sign (@) entered at this	'Y'ES 'N'O or <Enter> step deletes the entire entry.	5 3
5	LOCAL SHORTCUT FREQUENTLY USED NARRATIVE: {shortcut}//	<Enter> to accept default Other text	6 6
6	LOCAL SHORTCUT ENTRY:	Name or number of entry in LOCAL LOOKUP file (#8984.4) you wish your shortcut to reference	7

Figure 25-13. Add/Modify Utility option—Shortcuts process chart (2 of 2)

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
7	If the selected number/name corresponds to more than one entry, they are shown and you are prompted to choose one. If there is only one corresponding entry, it is displayed and the following appears:		
	"...OK? YES//	<Enter> to accept default.....2 'N'O.....6	
8	LOCAL SHORTCUT FREQUENTLY USED NARRATIVE:{shortcut}//	<Enter> to accept default.....9 Correct shortcut term.....9	
9	LOCAL SHORTCUT ENTRY: {code}//	<Enter> to accept default.....2 Correct code.....10	
	The selected code is displayed.		
10	...OK? YES//	<Enter> to accept default.....2 'N'O.....9	
11	Return to the menu.		

The Keywords option [XTLKMODKY], one of the three selections within the Add/Modify Utility option, is described below.

The following process chart shows the prompts and steps involved in using the Add/Modify Utility option when adding or editing a keyword:

Figure 25-14. Add/Modify Utility option—Keywords process chart

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
1	SH Shortcuts KE Keywords SY Synonyms Select Add/Modify Utility Option:	KE for Keywords..... <Enter> or up-arrow <^>.....	2 7
2	Additions/Modifications to Keywords in which file?	Name of entry in local reference file..... <?> for list of entries..... <Enter>.....	3 2 1
3	Which code in the {file name} file?	Code for which you wish to enter a keyword.....	4
4	Select LOCAL KEYWORD NAME:	New text you wish to use as a keyword..... Existing keyword term..... <Enter>.....	5 6 1
5	ARE YOU ADDING {'text'} AS A NEW LOCAL KEYWORD?	'Y'ES..... 'N'O or <Enter>.....	6 1
	An at-sign (@) entered at this step deletes the entire entry.		
6	LOCAL KEYWORD NAME: {keyword}//	<Enter> to accept default..... Correct keyword term.....	2 2
7	Return to the menu.		

The Synonyms option [XTLKMODSY], one of the three selections within the Add/Modify Utility option, is described below.

The following process chart shows the prompts and steps involved in using the Add/Modify Utility option when adding or editing a synonym:

Figure 25-15. Add/Modify Utility option—Adding or editing a synonym process chart (1 of 2)

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
1	SH Shortcuts KE Keywords SY Synonyms Select Add/Modify Utility Option:	SY for Synonyms <Enter> or up-arrow <^>	2 9
2	Additions/Modifications to Synonyms in which file?	Name of entry in local reference file <?> for list of entries <Enter>	3 2 1
	The entry made at this step must be in all upper case letters.		
3	Select LOCAL SYNONYM TERM:	New text you wish to use as a synonym Existing synonym term <Enter>	4 7 1
4	ARE YOU ADDING {'text'} AS A NEW LOCAL SYNONYM?	'Y'ES 'N'O	5 3
	An at-sign (@) entered at this step deletes the entire entry.		
5	LOCAL SYNONYM TERM: {synonym}//	<Enter> to accept default Other text	6 6
6	LOCAL SYNONYM Select SYNONYM:	Existing term in LOCAL LOOKUP file (#8984.4) for which you are entering a synonym	2

Figure 25-16. Add/Modify Utility option—Adding or editing a synonym process chart (2 of 2)

STEP	AT THIS PROMPT...	IF USER ANSWERS WITH...	THEN STEP
7	TERM: {term entered at Step 3}//	<Enter> to accept default.....8 Correct synonym term.....8	
	The entry made at this step must be in all upper case letters.		
8	Select SYNONYM: {term synonym was entered for}//	<Enter> to accept default.....2 Correct term.....2	
9	Return to the menu.		

The following are examples of what might appear on your screen when using the Add/Modify Utility option. The first example is for a new shortcut entry, the second example shows a new keyword entry, and the third shows the editing of an existing synonym entry.

Example 1

Illustration of a new Shortcut entry.

Figure 25-17. Shortcut option—Sample user entries

```
   SH     Shortcuts
   KE     Keywords
   SY     Synonyms

Select Add/Modify Utility Option: SH <Enter> Shortcuts


Additions/Modifications to Shortcuts in which file? CPT

Select LOCAL SHORTCUT FREQUENTLY USED NARRATIVE: COUGH
   ARE YOU ADDING 'COUGH' AS A NEW LOCAL SHORTCUT? Y <Enter> (YES)
   LOCAL SHORTCUT FREQUENTLY USED NARRATIVE: COUGH// <Enter>
   LOCAL SHORTCUT ENTRY: 31659

     Searching for a CPT  31659          BRONCHOSCOPIC PROCEDURES
         ...OK? YES// <Enter> (YES)
```

Example 2

Illustration of a new Keyword entry.

Figure 25-18. Keyword option—Sample user entries

```
   SH     Shortcuts
   KE     Keywords
   SY     Synonyms

Select Add/Modify Utility Option: KE <Enter> Keywords


Additions/Modifications to Keywords in which file?: CPT

Which code in the CPT file?: 11044 <Enter>    CLEANSING TISSUE/MUSCLE/BONE
Select LOCAL KEYWORD NAME: TISSUE SKIN
   ARE YOU ADDING 'TISSUE SKIN' AS A NEW LOCAL KEYWORD? Y <Enter> (YES)
   LOCAL KEYWORD NAME: TISSUE SKIN// <Enter>
```

Example 3

Illustration of editing an existing Synonym entry.

Figure 25-19. Synonym option—Sample user entries

```
   SH     Shortcuts
   KE     Keywords
   SY     Synonyms

Select Add/Modify Utility Option: SY <Enter> Synonyms


Additions/Modifications to Synonyms in which file?: CPT

Select LOCAL SYNONYM TERM: SLEEP
TERM: SLEEP// <Enter>
Select SYNONYM: DREAM// NIGHT
```

Systems Management

Implementation of Multi-Term Look-Up (MTLU)

This is how a user would configure a new file to be used with MTLU. The file you select would typically contain a free text field that more completely describes the record entry. Users would then use a cross-reference on this text field to perform lookups. MTLU is distinguished from FileMan in that users can enter a narrative or phrase, rather than a single term. The cross-reference can be either a VA FileMan Key Word In Context (KWIC) cross-reference, or you can create a custom MUMPS cross-reference that calls the routine, ^XTLKWIC (shown below). The ICD DIAGNOSIS file (#80) is used as an example.

REF: Multi-Term Look-Up (MTLU) Application Programming Interfaces (APIs) are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

Once you are in VA FileMan, do the following:

Figure 25-20. VA FileMan Utility Functions option—Sample user entries

```
Select OPTION: UTILITY FUNCTIONS
Select UTILITY OPTION: CROSS-REFERENCE A FIELD

MODIFY WHAT FILE: ICD DIAGNOSIS// <Enter>   ICD DIAGNOSIS
                                          (12535 entries)
Select FIELD: DESCRIPTION

CURRENT CROSS-REFERENCE IS MUMPS 'D' INDEX OF FILE
CHOOSE E (EDIT)/D (DELETE)/C (CREATE): C
WANT TO CREATE A NEW CROSS-REFERENCE FOR THIS FIELD? NO// Y <Enter> (YES)
CROSS-REFERENCE NUMBER: 2// <Enter>
Select TYPE OF INDEXING: REGULAR// MUMPS
WANT CROSS-REFERENCE TO BE USED FOR LOOKUP AS WELL AS FOR SORTING? YES// N <Enter>
(NO)
SET STATEMENT: S %="^ICD9(""AIHS"",I,DA)" D S^XTLKWIC
KILL STATEMENT: S %="^ICD9(""AIHS"",I,DA)" D K^XTLKWIC
INDEX: AC// AIHS
...

NO-DELETION MESSAGE: <Enter>
DESCRIPTION: <Enter>
  Edit? NO// <Enter>

DO YOU WANT TO CROSS-REFERENCE EXISTING DATA NOW? YES// Y <Enter> (YES)
...EXCUSE ME, THIS MAY TAKE A FEW MOMENTS...
```

Figure 25-21. Add Entries To Look-Up File—Sample user entries

```
>D ^XUP

Setting up programmer environment
Terminal Type set to: C-VT100

Select OPTION NAME: APP <Enter> LICATION UTILITIES  XTMENU     Application
Utilities


          Multi-Term Lookup Main Menu ...

Select Application Utilities Option: Multi <Enter>-Term Lookup Main Menu


          Multi-Term Lookup (MTLU)
          Print Utility
          Utilities for MTLU ...

Select Multi-Term Lookup Main Menu Option: Util <Enter> ities for MTLU


   KL     Delete Entries From Look-up
   ST     Add Entries To Look-Up File
          Add/Modify Utility ...

Select Utilities for MTLU Option: ST <Enter> Add Entries To Look-Up File
Select LOCAL LOOKUP NAME: ICD DIAGNOSIS
  ARE YOU ADDING 'ICD DIAGNOSIS' AS A NEW LOCAL LOOKUP (THE 3RD)? Y <Enter> (YES)
  LOCAL LOOKUP NAME: ICD DIAGNOSIS// <Enter>
  LOCAL LOOKUP DISPLAY PROTOCOL: DSPLYD^XTLKKWLD
INDEX: AIHS
...Ok, will now setup KEYWORD and SHORTCUT file DD's
   to allow terms for 'ICD DIAGNOSIS' entries...
PREFIX: M// ?
     Answer must be a unique prefix, 1-10 characters in length

PREFIX: M// D
  <REMINDER> Using 'Edit File', set the lookup routine, XTLKDICL, in ICD
DIAGNOSIS DD
Select LOCAL LOOKUP NAME: <Enter>
```

Enter the "Variable Pointer" prefix.

If *all* references to a file (by all packages) are to behave as MTLU lookups, add the special lookup routine, ^XTLKDICL, to the file's DD using the FileMan Edit File option.

REF: For more information on the Edit File option, please refer to the "Utility Functions" chapter in the *VA FileMan User Manual*.

Figure 25-22. VA FileMan Edit File option—Sample user entries

```
VAH,MTL>D Q^DI


VA FileMan 20.0


Select OPTION: UT <Enter> ILITY FUNCTIONS
Select UTILITY OPTION: ED <Enter> IT FILE

MODIFY WHAT FILE: ICD DIAGNOSIS// <Enter>
NAME: ICD DIAGNOSIS// <Enter>
DESCRIPTION: <Enter>
  1>Contains all valid ICD diagnosis codes.
EDIT Option: <Enter>
Select APPLICATION GROUP: <Enter>
PROGRAMMER: <Enter>
VERSION: 9// <Enter>
DATA DICTIONARY ACCESS: <Enter>
READ ACCESS: <Enter>
WRITE ACCESS: <Enter>
DELETE ACCESS: <Enter>
LAYGO ACCESS: <Enter>
AUDIT ACCESS: <Enter>
DD AUDIT? NO// <Enter>
ASK 'OK' WHEN LOOKING UP AN ENTRY? YES// <Enter>  (YES)
POST-SELECTION ACTION: <Enter>
LOOK-UP PROGRAM: XTLKDICL
CROSS-REFERENCE ROUTINE: <Enter>


Select UTILITY OPTION:<Enter>
```

NOTE: The developer might elect to use MTLU only in selected instances. This is accomplished by *not* adding the special lookup routine to the file's DD. After the file has been added to the LOCAL LOOKUP file (#8984.4), you can make a developer call to LKUP^XTLKMGR.

REF: Multi-Term Look-Up (MTLU) Application Programming Interfaces (APIs) are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

26 Routine Tools

Overview

Kernel Toolkit provides an Application Programming Interface (API) that includes developer utilities for working with routines and globals. This chapter describes routine tools exported with Kernel Toolkit. These tools are useful to IRM staff and VistA software developers.

REF: Kernel Toolkit Application Programming Interfaces (APIs) are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

Most of the tools are available as options on the Routine Tools menu [XUPR-ROUTINE-TOOLS] located on the Programmer Options menu [XUPROG], which is locked with the XUPROG security key. Some subordinate menu options are locked with the XUPROGMODE and/or XUPROG security keys as an extra level of security.

Routines can be edited, analyzed by flow charting, printed, compared, deleted, and moved by using an option or its corresponding direct mode utility.

REF: Kernel Toolkit APIs and direct mode utilities are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*.

The Routine Tools menu is shown below:

Figure 26-1. Routine Tools menu options

```
SYSTEMS MANAGER MENU ...                                                  [EVE]
  Programmer Options ... <locked with XUPROG>                          [XUPROG]
   Routine Tools ...                                             [XUPR-ROUTINE-TOOLS]
     %Index of Routines                                                [XUINDEX]
     Compare local/national checksums report                    [XU CHECKSUM REPORT]
     Compare routines on tape to disk                            [XUPR-RTN-TAPE-CMP]
     Compare two routines                                       [XT-ROUTINE COMPARE]
     Delete Routines <locked with XUPROGMODE>                          [XTRDEL]
     Flow Chart Entire Routine                                          [XTFCR]
     Flow Chart from Entry Point                                        [XTFCE]
     Group Routine Edit <locked with XUPROGMODE>                     [XTRGRPE]
     Input routines <locked with XUPROG>                         [XUROUTINE IN]
     List Routines                                                   [XUPRROU]
     Load/refresh checksum values into ROUTINE file             [XU CHECKSUM LOAD]
     Output routines                                             [XUROUTINE OUT]
     Routine Edit <locked with XUPROGMODE>                       [XUPR RTN EDIT]
     Routines by Patch Number                                   [XUPR RTN PATCH]
     Variable changer <locked with XUPROGMODE>                [XT-VARIABLE CHANGER]
     Version Number Update <locked with XUPROGMODE>             [XT-VERSION NUMBER]
```

These options are documented below, grouped by routine type.

User Interface

Analyzing Routines

XINDEX—%Index of Routines Option

Kernel Toolkit's XINDEX utility (formerly known as %INDEX utility), %Index of Routines option [XUINDEX], is a verification tool for developers. It reviews a routine according to the 1995 ANSI M Standard and according to the VA *Programming Standards and Conventions (SAC)*.

XINDEX reports any errors that it finds. The reported error codes (error flags) are as follows:

- S—Standards violation (according to VA Standards)
- W—Warning violation (according to VA Conventions)
- F—Fatal M error (hard MUMPS error)

The XINDEX utility creates a cross-referenced list of global references and routines invoked by a selected list of routines, a build, or a package. While XINDEX will show all routines that are called, it will *not* identify all global references in the routines. After running XINDEX, use the %RFIND utility to find entries that were missed by XINDEX.

XINDEX can also be used to verify parts of a software application that contain M code. Compiled templates can be included. The M code used in data dictionaries, functions, and options can also be reviewed. When using XINDEX to review an entire software application, it is best to queue the report for an off-peak time since processing is intensive.

The XINDEX utility can be called directly (i.e., >D ^XINDEXFigure 26-2) or via an option on the Programmer Options menu [XUPROG].

Figure 26-2. XINDEX—%Index of Routines option; direct mode utilities sample user entries

```
>D ^XINDEX


                 V. A.  C R O S S  R E F E R E N C E R  7.3
                 UCI: NXT CPU: NXT    JUL 06, 2006@15:14:12

All Routines? No => <Enter> No

Routine: XDRMAIN
Routine: <Enter>
1 routine

Select BUILD NAME: XT*7.3*102  <Enter> TOOLKIT
Include the compiled template routines: N// <Enter>

Print more than compiled errors and warnings? YES// <Enter>

Print summary only? NO// <Enter>

Print routines? YES// <Enter>

Print (R)egular,(S)tructured or (B)oth?  R// <Enter>
Print the DDs, Functions, and Options? YES// <Enter>

Print errors and warnings with each routine? YES// <Enter>

Save parameters in ROUTINE file? NO// <Enter>
Index all called routines? NO// <Enter>
```

Enter the software name you want to check with XINDEX.

Or enter "S" for an indented report.

Or enter YES to store the parameters.

The following is a list of the error conditions that the XINDEX utility flags. The 1995 ANSI M Standard is taken into account with new syntax and other checks.

Figure 26-3. XINDEX—List of the error conditions that the XINDEX utility flags

```
1    F - UNDEFINED COMMAND (rest of line not checked).
2    F - Non-standard (Undefined) 'Z' command.
3    F - Undefined Function.
4    F - Undefined Special Variable.
5    F - Unmatched Parenthesis.
6    F - Unmatched Quotation Marks.
7    F - ELSE Command followed by only one space.
8    F - FOR Command did not contain '='.
9    W - QUIT Command followed by only one space.
10   F - Unrecognized argument in SET command.
11   W - Invalid local variable name.
12   W - Invalid global variable name.
13   F - Blank(s) at end of line.
14   F - Missing LABEL, REFERENCED in this routine.
15   W - Duplicate label.
16   F - Error in pattern code.
17   W - First line tag NOT routine name.
18   W - Line contains a CONTROL (non-graphic) character.
19   S - Line is longer than 245 bytes.
20   S - View command used.
21   F - General Syntax Error.
22   S - Exclusive Kill.
23   S - Unargumented Kill.
24   S - Kill of an unsubscripted global.
25   S - Break command used.
26   S - Exclusive or Unargumented NEW command.
27   S - $View function used.
28   S - Non-standard $Z special variable used.
29   S - 'Close' command should be invoked through 'D ^%ZISC'.
30   S - TAG+OFFSET syntax.
31   S - Non-standard $Z function used.
32   S - 'HALT' command should be invoked through 'G ^XUSCLEAN'.
33   S - Read command doesn't have a timeout.
34   S - 'OPEN' command should be invoked through ^%ZIS.
35   S - Routine exceeds SACC maximum size of 5000 (|).
36   S - Should use 'TASKMAN' instead of 'JOB' command.
37   F - Tag is not valid.
38   F - Call to this MISSING LABEL (see INVOKED BY list).
39   S - Kill of a protected variable (|).
40   S - Space where a command should be.
41   S - Star or pound READ used.
42   F - Null line (no commands or comment).
43   F - Invalid or wrong number of arguments to a function.
44   S - Version number not found on 2nd line of routine.
45   S - Set to a '%' global.
46   F - Quoted string not followed by a separator.
47   S - Lowercase command(s) used in line.
48   F - Missing argument to a command post-conditional.
49   F - Command missing an argument.
50   S - Extended reference.
51   F - Block structure mismatch
52   F - Reference to routine '^|'. That isn't in this UCI.
53   F - Bad Number
54   S - Access to SSVN's restricted to Kernel.
```

Must be manually checked by the programmer.

Flow Chart Entire Routine Option

The Flow Chart Entire Routine option [XTFCR] generates a flow chart, showing the processing performed within an entire routine.

The following corresponding direct mode utility can be used in programmer mode:

```
>D ^XTFCR
```

Flow Chart From Entry Point Option

The Flow Chart from Entry Point option [XTFCE] generates a flow chart of the processing performed from a specified entry point to its termination of processing. It also allows the user to expand the code in other routines or entry points referenced by DO or GOTO commands.

The following corresponding direct mode utility can be used in programmer mode:

```
>D ^XTFCE
```

Editing Routines

Group Routine Edit Option

The Group Routine Edit option [XTRGRPE] calls the XTRGRPE routine to edit a group of routines. Once several routines are identified, the Kernel Toolkit ^%Z editor is called. This option is locked with XUPROGMODE.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D ^XTRGRPE
```

Routine Edit Option

The Routine Edit option [XUPR RTN EDIT] invokes the ^%Z editor. The ^%Z editor can be used to edit a group of routines with the Group Routine Edit option. This allows developers at an external site (e.g., on the site manager's staff) to edit M routines. This option is locked with the XUPROGMODE security key.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>X ^%Z
```

REF: For more information on the ^%Z Editor, please refer to the "^%Z Editor" topic in Chapter 0, "**Error! Not a valid result for table.**," in this manual.

Routines by Patch Number Option

The Routines by Patch Number option [XUPR RTN PATCH] allows users to print routines associated with a patch. When prompted, enter a list of routines. The output is sorted by patch number.

Variable Changer Option

The Variable Changer option [XT-VARIABLE CHANGER] runs the XTVCHG routine, which changes all occurrences of one variable to another. This option is locked with the XUPROGMODE security key.

CAUTION: This option changes DOs and GOTOs also, but it does not change the target of the DOs and GOTOs. For example, if you request to change all occurrences of "TAG" to "TAGS", "DO TAG" would be changed to "DO TAGS". However, the actual Line Label called TAG would not be changed.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D ^XTVCHG
```

Version Number Update Option

The Version Number Update option [XT-VERSION NUMBER] is used to update version numbers of one or more routines. This option runs the XTVNUM routine to update or set the version number into a set of routines. This option is locked with the XUPROGMODE security key.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D ^XTVNUM
```

Printing Routines

List Routines Option

The List Routines option [XUPRROU] uses the %ZTPP utility to print a listing of entire routines.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D ^%ZTPP
```

Comparing Routines

Compare local/national checksums report Option

The Compare local/national checksums report option [XU CHECKSUM REPORT] compares checksums for routines to the values in the ROUTINE file (#9.8). It produces a report listing routines that differ by the following criteria:

- Patch or version, where the version or patch may be correct but checksums are off
- Local routines being tracked
- Information is not on record for a patch (e.g., test patches)

Nationally released routine checksums are sent by Master File Updates to the local ROUTINE file (#9.8) automatically. Local sites may also record checksums in the CHECKSUM VALUE field in the ROUTINE file (#9.8). To compare local routines that are being tracked, the CHECKSUM REPORT field should be set to "Local – report."

NOTE: As of Kernel Patch XU*8.0*369, the integrity checking CHECK1^XTSUMBLD API supports the Compare local/national checksums report option [XU CHECKSUM REPORT].

NOTE: As of Kernel Patch XU*8.0*393, KIDS was modified to send a message to a server on FORUM when a KIDS build is sent to a Host File Server (HFS) device. This message contains the checksums for the routines in the patch. The server on FORUM matches the message with a patch if the sending domain is authorized on FORUM. There is no longer a need for developers to manually include routine checksums (either CHECK^XTSUMBLD and/or CHECK1^XTSUMBLD) in the patch description. The patch module will include the before and after CHECK1^XTSUMBLD values in the Routine Information section at the end of the patch document.

With changes in the National Patch Module (NPM) on FORUM, when the patch is released the checksums for the routines are moved to the ROUTINE file (#9.8) on FORUM. The checksum "before" values will come from the FORUM ROUTINE file (#9.8) and are considered the GOLD standard for released checksums. The local site's Compare local/national checksums report option [XU CHECKSUM REPORT] uses the FORUM ROUTINE file (#9.8) as its source to create reports showing any routines that do *not* match..

This patch also modified the KIDS BUILD file (#9.6) by adding the TRANSPORT BUILD NUMBER field (#63) used to store a build number that is incremented each time a build is made. This build number is added to the second line of each routine in the 7th "**;**" piece. This makes it easy to tell if a site is running the current release during testing and afterword. The leading "B" found in the checksum tells the code what checksum API to use.

Compare Routines on Tape to Disk Option

The Compare Routines on Tape to Disk option [XUPR-RTN-TAPE-CMP] compares routines and displays the differences. This option reads a standard Caché %RO Host File Server (HFS) file and compares the routines on the HFS file with a routine with the same name in the current account.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D TAPE^XTRCMP
```

NOTE: While it is still called a "TAPE" compare, it is actually comparing a routine in an Host File Server (HFS) file to an installed routine.

Compare Two Routines Option

The Compare Two Routines option [XT-ROUTINE COMPARE] is used to compare two routines with different names that are located in the same account and display/print the differences (using MailMan's PackMan compare utilities).

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D ^XTRCMP
```

Deleting Routines

Delete Routines Option

The Delete Routines option [XTRDEL] can be used to delete one or more routine(s). The wildcard syntax can be used to delete a set, such as **ABC*** to delete all those routines beginning with the letters **ABC**. This option is locked with the XUPROGMODE security key.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D ^%ZTRDEL
```

Load and Save Routines

The Input Routines and Output Routines options can be used to move routines from one UCI to another. These make use of operating system-specific utilities such as %RR for routine restore and %RS for routine save.

Input Routines Option

The Input Routines option [XUROUTINE IN] loads routines from an external device. This option is locked with the XUPROG security key.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D ^%RR (OS-specific)
```

Output Routines Option

The Output Routines option [XUROUTINE OUT] outputs routines to an external device, such as a host file.

The corresponding direct mode utility can be used in programmer mode as follows:

```
>D ^%RS (OS-specific)
```

Load/refresh checksum values into ROUTINE file Option

The Load/refresh checksum values into ROUTINE file option [XU CHECKSUM LOAD] can be used to update the ROUTINE file (#9.8) with the latest checksum values from FORUM.

REF: Kernel Toolkit Application Programming Interfaces (APIs) are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

27 Verification Tools

Overview

Kernel Toolkit provides an Application Programming Interface (API) that includes developer utilities for working with routines and globals. This chapter describes the verification tools exported with Kernel Toolkit that are useful to Information Resource Management (IRM) staff and developers for reviewing Veterans Health Information Systems and Technology Architecture (VistA) software.

Kernel Toolkit Application Programming Interfaces (APIs) are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

Verification tools can be accessed through one of three methods:

- Programmer Options menu
- Operations Management menu
- Direct Mode utilities

User Interface

Verifier Tools Menu

The Verifier Tools Menu contains options which are available as tools for verification during program development. These options are located on the Verifier Tools Menu [XTV MENU], which is located on the Systems Manager Menu. These tools are useful for developers to:

- Record the text of the routines indicated in the file used to maintain changes in routines.
- Compare one or more current routines to previous versions.

The Verifier Tools Menu [XTV MENU] consists of the following options that are described below:

Figure 27-1. Verifier Tools Menu options

```
SYSTEMS MANAGER MENU ...                                              [EVE]
  Verifier Tools Menu ...                                          [XTV MENU]
     Update with current routines                                [XTVR UPDATE]
     Routine Compare - Current with Previous                    [XTVR COMPARE]
```

Routine Compare

Update with Current Routines Option

The Update with Current Routines option [XTVR UPDATE] records the text of the routines indicated in the file used to maintain changes in routines. Only the last version entered is kept intact; previous entries reflect only the changes in lines added and/or deleted to make the next version. This option is used to record the current routine structure so that it can be compared with future versions of the routine using the Routine Compare - Current with Previous option [XTVR COMPARE].

After editing the routine, the Update with Current Routines option can again be used to store changes. Rather than storing all minor changes, the user can choose to wait and use the Update with Current Routines option only after extensive edits have been made. Lines are compared and changes, including inserted or deleted lines, are recorded. (Alteration of the routine's second line are usually insignificant and are ignored.) The Update with Current Routines option can be used whenever the developer would like a new "snapshot" of the routine. The XTV ROUTINE CHANGES file (#8991) holds each new snapshot as a new version. This filing method does not, however, alter the actual version number of the routine itself.

Routine Compare - Current with Previous Option

The Routine Compare - Current with Previous option [XTVR COMPARE] is used to compare one or more current routines to previous versions. To use the routine compare utility, copies of the selected routines *must* first be stored in the XTV ROUTINE CHANGES file (#8991), stored in the ^XTV(8991, global. This is achieved by use of the Update with Current Routines option [XTVR UPDATE] on the Verifier Tools Menu. Routines can be specified one by one or as a group with the wildcard syntax (e.g., XQ*). Any initialize routines are automatically excluded. Differences between the current version and the indicated number of prior versions are noted. The user is prompted for the number of previous versions from which to begin the listing. An entire history or just a brief display of recent modifications can be obtained.

Miscellaneous Verification Tools

Programmer Options Menu

The Programmer Options menu [XUPROG] comprised of the following options:

Figure 27-2. Programmer Options menu options: Toolkit verification tools

```
SYSTEMS MANAGER MENU ...                                              [EVE]
  Programmer Options ...                                            [XUPROG]
    **> Locked with XUPROG
   KIDS Kernel Installation & Distribution System ...                [XPD MAIN]
          **> Locked with XUPROG
   PG   Programmer mode                                           [XUPROGMODE]
          **> Locked with XUPROGMODE
        Calculate and Show Checksum Values                     [XTSUMBLD-CHECK]
        Delete Unreferenced Options                          [XQ UNREF'D OPTIONS]
        Error Processing ...                                          [XUERRS]
        General Parameter Tools ...                             [XPAR MENU TOOLS]
        Global Block Count                                       [XU BLOCK COUNT]
        List Global                                                     [XUPRGL]
          **> Locked with XUPROGMODE
        Routine Tools ...                                     [XUPR-ROUTINE-TOOLS]
        Test an option not in your menu                          [XT-OPTION TEST]
          **> Locked with XUMGR
```

Tools found on the Programmer Options menu that can be of use for verification purposes include:

- Calculate and Show Checksum Values [XTSUMBLD-CHECK]
- Error Processing [XUERRS]

These options are described below.

Calculate and Show Checksum Values Option

The Calculate and Show Checksum Values option [XTSUMBLD-CHECK] gives developers the ability to check the value of a routine at any given time. It does not regenerate NTEG routines and can safely be used anytime.

This option calls the CHCKSUM^XTSUMBLD API to calculate and show the checksum value for one or more routines in the current account. This value is referenced in the Patch Module description for routine patches.

Kernel Toolkit Patch XT*7.3*94, deployed the CHECK1^XTSUMBLD API and the new logic CheckSum: %^ZOSF("RSUM1"). Kernel Toolkit Patch XT*7.3*100 included the CHECK1^XTSUMBLD routine into the Calculate and Show Checksum Values option [XTSUMBLD-CHECK].

The CHECK1^XTSUMBLD API is more accurate than the old integrity checking utility (CHECK^XTSUMBLD). CHECK1^XTSUMBLD determines the current checksums for selected routine(s), the functionality of which is shown as follows:

- Any comment line with a single semi-colon is presumed to be followed by comments and only the line tag will be included.
- Line 2 will be excluded from the count.
- The total value of the routine is determined (excluding exceptions noted above) by multiplying the ASCII value of each character by its position on the line and position of the line in the routine being checked.

The corresponding direct mode utility can be used in programmer mode:

```
>D CHCKSUM^XTSUMBLD
```

The integrity checking utility CHCKSUM^XTSUMBLD supports the Compare local/national checksums report option [XU CHECKSUM REPORT], as released with Kernel Patch XU*8.0*369.

The modification, CHECK1^XTSUMBLD, to the integrity checking utility CHCKSUM^XTSUMBLD fixes the problem in which the old CheckSum output is the same CheckSum value, even if some lines were swapped within a routine.

Error Processing—Kernel Error Trapping and Reporting

Technical personnel who have entered programmer mode with D ^XUP, might choose to record an error encountered with D ^%ZTER. The error log can be displayed with D ^XTER, or with the corresponding option. Also, the error log can be purged with D ^XTERPUR. Errors can also be purged from within the menu system with an option that is locked with the XUPROGMODE security key.

The corresponding direct mode utilities can be used in programmer mode as follows:

- Record an Error

  ```
  >D ^%ZTER
  ```

- Display Error Trap

  ```
  >D ^XTER
  ```

- Purge Error Log

  ```
  >D ^XTERPUR
  ```

REF: For more information on Error Processing, please refer to Chapter 13,"Error Processing," in this manual.

Kernel Toolkit Application Program Interfaces (APIs) are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

28 Miscellaneous Programmer Tools

Programmer Options Menu

User Interface

Figure 28-1. Programmer Options menu options: Toolkit miscellaneous tools

```
SYSTEMS MANAGER MENU ...                                                [EVE]
  Programmer Options ... <locked with XUPROG>                          [XUPROG]
   KIDS Kernel Installation & Distribution System ...                [XPD MAIN]
           <locked with XUPROG>
   PG Programmer mode <locked with XUPROGMODE>                      [XUPROGMODE]
      Calculate and Show Checksum Values                         [XTSUMBLD-CHECK]
      Delete Unreferenced Options                              [XQ UNREF'D OPTIONS]
      Error Processing ...                                             [XUERRS]
      General Parameter Tools ...                                 [XPAR MENU TOOLS]
      Global Block Count                                           [XU BLOCK COUNT]
      List Global <locked with XUPROGMODE>                                [XUPRGL]
      Routine Tools ...                                          [XUPR-ROUTINE-TOOLS]
      Test an option not in your menu <locked with XUMGR>            [XT-OPTION TEST]
```

Delete Unreferenced Options

The Delete Unreferenced Options option [XQ UNREF'D OPTIONS] is used to examine those options which are not on any menu, are not used as primary or secondary options, and are not tasked to run. The user may then decide in each case whether to delete the unreferenced option.

Global Block Count Option

The Global Block Count option [XU BLOCK COUNT] can be used to count the number of data blocks in a global.

Listing Globals Option

The List Global option [XUPRGL] is found on the Programmer Options menu, locked with the XUPROG key. This option is also locked with the XUPROGMODE key as an extra level of security.

It can be used to list the contents of a global to the screen. It makes use of operating system-specific utilities such as %G, the global lister.

The option is locked with the XUPROGMODE security key

The corresponding direct mode utility can be used in programmer mode. For example:

```
>D ^%G (OS-specific)
```

Test an option not in your menu Option

The Test an option not in your menu option [XT-OPTION TEST] is used for in-house testing of options, only. It allows the selection of an option from the OPTION file (#19) and then executes it. This option is locked with the XUMGR security key.

CAUTION: No security checks are performed in the XT-OPTION TEST option; therefore, it should only be given to programmers.

REF: Kernel Toolkit Application Programming Interfaces (APIs) are documented in the "Toolkit: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

^%Z Editor

User Interface

The ^%Z editor (routine editor) is installed in the Manager account as the ^%Z global by ZTMGRSET during installation. (It can also be installed with D ^ZTEDIT.) To use the editor, load the routine (it *must* pre-exist) and then X ^%Z. The following example creates a one-line routine in Caché and then calls the ^%Z Editor.

Figure 28-2. Calling the ^%Z Editor—Sample user entries

```
>ZR

>ZZTEST <Enter> ;ID/SITE;test routine;
>ZS ZZTEST

>ZL ZZTEST X ^%Z

%Z Editing: ZZTEST  Terminal type: C-VT100
Edit:
```

The editor fills in the third "<space>;" piece with the date/time that the routine is filed.

Enter ".F" (dot-file) at the edit prompt to change files. When saving with dot-file, an edit comment can be entered. This text is stored in the EDIT HISTORY multiple in the ROUTINE file (#9.8) as programmer documentation. The following example shows how an entire routine can be displayed by entering the ZP print command followed by a space at the M prompt. Dot-file (.File) is then used to file. A dot is then used to exit. (The dot exit does not automatically file changes.)

Figure 28-3. ^%Z Editor—Displaying a routine using the ZP command

```
>ZL ZZTEST X ^%Z

%Z Editing: ZZTEST  Terminal type: C-VT100
Edit: ZP<SPACE> <Enter>
ZZTEST   ;test routine

Length:  20 <Enter>  Line: ZZTEST
ZZTEST ;test routine
Edit: .Insert after: ZZTEST// <Enter>
Line:    ;next line
Line:    Q
Line: <Enter>
Edit: .File ZZTEST
Edit comment:
  1> This text is stored in the Routine file's Edit History multiple. <Enter>
  2> <Enter>
EDIT Option: <Enter>
Edit: . <Enter>
>
```

First, either a <tab> or line label is entered.

Routines are filed by the name used when loading, not by the first line tag. If a ROUTINE file (#9.8) exists, then the routine is added if not already there, and an entry is made of the date/time and DUZ of the user that filed it. When filing, the editor updates the third piece of the first line of the routine with the date/time.

When editing, a question mark ("**?**") can be entered to provide help. The dot commands are listed first. They provide the usual break, join, insert, and remove functions. The **+n** method of selecting lines to edit is also noted. The line tag can be used along with a number (e.g., TAG+3) to reach a particular line. A minus sign ("**-**") will back up lines. And the asterisk ("*****") can be entered to reach the last line.

Figure 28-4. ^%Z Editor—Listing edit commands

```
>X ^%Z
Edit: ?
.ACTION menu              .BREAK line              .CHANGE every
.FILE routine             .INSERT after            .JOIN lines
.MOVE lines               .REMOVE lines            .SEARCH for
.TERMinal type            .XY change to/from replace-with
. -TO EXIT THE EDITOR
""+n Absolute line n    +n To advance n lines   -n To backup n lines_
 use '*' to get last line

^NAME - to edit a GLOBAL node        *NAME - to edit a LOCAL variable
MUMPS command line (mumps command <space> or Z command <space>)
```

Help displays information about editing in line mode. A complete line is displayed and various keys can be used to navigate. The **<Spacebar>** moves forward by words, the period moves forward by characters, and the **<CTRL H>** command key sequence moves backwards by characters. Upon reaching the desired location, the **<Delete>** key can be used to remove characters. To enter characters, the character "E" *must* first be entered as an insert/delete toggle. Pressing the **<Enter>** key reverses the toggle and allows navigation. Pressing the **<Enter>** key again moves back to the beginning of the line.

Figure 28-5. ^%Z Editor—Line mode help information

```
In the line mode,
Spacebar moves to the next space or comma. Dot to the next char.
'>' To move forward 80 char or to end of line.
Backspace to back up one char. E to enter new char's at the cursor.
CR to exit enter mode, return to start of line or EDIT prompt.
D to delete from the cursor to the next space or comma.
Delete (Rub) to delete the char under the cursor.
CTRL-R to restore line and start back at the beginning.
```

Replace mode editing can be invoked by entering dot-XY at the edit prompt. This method allows easy string substitution, as in the VA FileMan line editor. Entering a question mark at the next edit prompt displays the following help:

Figure 28-6. ^%Z Editor—Replace mode editing help information

```
In the replace/with mode,
SPECIAL <REPLACE> STRINGS:
  END    -to add to the END of a line
  ...    -to replace a line
  A...B  -to specify a string that begins with "A" and ends with "B"
  A...   -to specify a string that begins with "A" to the end of the line
CTRL-R to restore line.
```

The ACTION menu provides additional functions. Save and restore lines can be used to move lines within one routine or from one routine to another. To copy lines to another routine, first save the lines, then load and edit the other routine, and restore the lines.
When patching a routine, the ACTION menu can be used to calculate checksums. Before filing changes, the new checksum can be displayed and compared with the patch report for verification of editing. The following figure shows how to reach the ACTION menu with dot-A (.A).

Figure 28-7. ACTION menu—Sample user entries

```
Edit: .A
Action: ?
Bytes in routine          Checksum                Restore lines
Save lines                Version #
Action: C
                    Checksum is 4971725
Action: <Enter>
Edit: <Enter>
```

Global nodes and local variables may also be edited with the ^%Z editor. Editing occurs directly, so the idea of filing does not apply. The editor must then be exited with a dot, not with a dot-file, since filing should not take place.

29 XGF Function Library: System Management

The XGF Function Library supports developers designing text-based applications. The functions in this library support cursor positioning, overlapping text windows, video attribute control, and keyboard escape processing, all in a text-mode environment.

If you intend to make simple interface enhancements for an existing text-mode application, then you may find the XGF Function Library useful. The XGF Function Library provides the following functionality:

- Text-mode overlapping windows.
- Text-mode cursor positioning by screen coordinate.
- Text-mode video attribute control (bold, blink, etc.).
- Keyboard reader using M escape processing (thereby making use of keystrokes like **<UP-ARROW>** ("↑"), **<DOWN-ARROW>** ("↓"), **<PREV>** ("←"), **<NEXT>** ("→"), etc.).

The XGF Function Library may *not* be appropriate if you need:

- A full graphical user interface (GUI) front end for your application.
- Support for non-ANSI VT-compatible display devices.

To use the XGF Function Library, your system *must* use an M implementation that complies with the 1995 ANSI M standard. At a minimum, the M implementation *must* support the following features to use the XGF Function Library:

Table 29-1. Minimum M implementation features required for the XGF Function Library

Feature	Example
SET into $EXTRACT	`S X="this is a string",$E(X,1,4)="that"`
Reverse $ORDER	`S X=$O(^TMP(""),-1)`
Two argument $GET	`K Y S X=$G(Y,"DEFAULT")`
Skipping parameters	`D TAG^ROUTINE(,P2,,P4)`
$NAME	`W $NA(^TMP($J))`
SET $X and $Y	`S $X=10`

This XGF Function Library supports terminals that are ANSI-compatible and at least VT100-compatible. As a result, this software does not support QUME QVT102/QVT102A terminals.

The XGF Function Library Application Program Interfaces (APIs) are documented in the "XGF Function Library: Developer Tools" chapter in the *Kernel Developer's Guide*. Kernel and Kernel Toolkit APIs are also available in HTML format at the following Web address:

http://vista.med.va.gov/kernel/apis/index.shtml

Glossary

ALPHA TESTING	In VA terminology, Alpha testing is when a VistA test software application is running in a site's account.
AUTO MENU	An indication to Menu Manager that the current user's menu items should be displayed automatically. When AUTO MENU is not in effect, the user *must* enter a question mark at the menu's select prompt to see the list of menu items.
BETA TESTING	In VA terminology, Beta testing is when a VistA test software application is running in a Production account.
CAPACITY MANAGEMENT	The process of assessing a system's capacity and evaluating its efficiency relative to workload in an attempt to optimize system performance. Kernel provides several utilities.
CHECKSUM	A numeric value that is the result of a mathematical computation involving the characters of a routine or file.
CIPHER	A system that arbitrarily represents each character as one or more other characters. (See also: ENCRYPTION.)
COMMON MENU	Options that are available to all users. Entering two question marks ("**??**") at the menu's select prompt will display any SECONDARY MENU OPTIONS available to the signed-on user along with the common options available to all users.
COMPILED MENU SYSTEM (^XUTL GLOBAL)	Job-specific information that is kept on each CPU so that it is readily available during the user's session. It is stored in the ^XUTL global, which is maintained by the menu system to hold commonly referenced information. The user's place within the menu trees is stored, for example, to enable navigation via menu jumping.
COMPUTED FIELD	This field takes data from other fields and performs a predetermined mathematical function (e.g., adding two columns together). You will not, however, see the results of the mathematical function on the screen. Only when you are printing or displaying information on the screen will you see the results for this type of field.
DEVICE HANDLER	The Kernel module that provides a mechanism for accessing peripherals and using them in controlled ways (e.g., user access to printers or other output devices).
DIFROM	VA FileMan utility that gathers all software components and changes them into routines (namespaceI* routines) so that they can be exported and installed in another VA FileMan environment.

DOUBLE QUOTE (")	A symbol used in front of a Common option's menu text or synonym to select it from the Common menu. For example, the five character string **"TBOX** selects the User's Toolbox Common option.
DR STRING	The set of characters used to define the DR variable when calling VA FileMan. Since a series of parameters may be included within quotes as a literal string, the variable's definition is often called the DR string. To define the fields within an edit sequence, for example, the developer may specify the fields using a DR string rather than an INPUT template.
DUZ(0)	A local variable that holds the FILE MANAGER ACCESS CODE field (#3) of the signed-on user.
ENCRYPTION	Scrambling data or messages with a cipher or code so that they are unreadable without a secret key. In some cases encryption algorithms are one directional, that is, they only encode and the resulting data cannot be unscrambled (e.g., Access and Verify codes).
FILE ACCESS SECURITY SYSTEM	Formerly known as Part 3 of the Kernel Inits. If the File Access Security conversion has been run, file-level security for VA FileMan files is controlled by Kernel's File Access Security system, not by VA FileMan Access codes (i.e., FILE MANAGER ACCESS CODE field [#3] in the NEW PERSON file [#200]).
FORCED QUEUING	A device attribute indicating that the device can only accept queued tasks. If a job is sent for foreground processing, the device will reject it and prompt the user to queue the task instead.
GO-HOME JUMP	A menu jump that returns the user to the primary menu presented at signon. It is specified by entering two carets ("^^") at the menu's select prompt. It resembles the "Rubber-band Jump" but *without* an option specification/name following the carets.
HELP PROCESSOR	A Kernel module that provides a system for creating and displaying online documentation. It is integrated within the menu system so that help frames associated with options can be displayed with a standard query at the menu's select prompt.
HOST FILE SERVER (HFS)	A procedure available on layered systems whereby a file on the host system can be identified to receive output. It is implemented by the Device Handler's HFS device type.
HUNT GROUP	An attribute of an entry in the DEVICE file (#3.5) that allows several devices to be used interchangeably; useful for sending network mail or printing reports. If the first hunt group member is busy, another member can stand in as a substitute.
INIT	Initialization of an software application. INIT* routines are built by VA FileMan's DIFROM and, when run, recreate a set of files and other software components.

JUMP	In VistA applications, the Jump command allows you to go from a particular field within an option to another field within that same option. You can also Jump from one menu option to another menu option without having to respond to all the prompts in between. To jump, type a caret ("^") and then type the name of the field or option you wish to jump to. (See also GO-HOME JUMP, PHANTOM JUMP, RUBBER-BAND JUMP, or UP-ARROW JUMP.)
JUMP START	A logon procedure whereby the user enters the "Access code;Verify code;option" to go immediately to the target option, indicated by its menu text or synonym. The jump syntax can be used to reach an option within the menu trees by entering "accesscode;verifycode;option".
KERMIT	A standard file transfer protocol. It is supported by Kernel and can be set up as an alternate editor.
MANAGER ACCOUNT	A UCI that holds vendor shared routines.
MENU CYCLE	The process of first visiting a menu option by picking it from a menu's list of choices and then returning to the menu's select prompt. Menu Manager keeps track of information (e.g., the user's place in the menu trees) according to the completion of a cycle through the menu system.
MENU MANAGER	The Kernel module that controls the presentation of user activities (e.g., menu choices or options). Information about each user's menu choices is stored in the Compiled Menu System, the ^XUTL global, for easy and efficient access.
MENU SYSTEM	The overall Menu Manager logic as it functions within the Kernel framework.
MENU TEMPLATE	An association of options as pathway specifications to reach one or more final destination options. The final options *must* be executable activities and not merely menus for the template to function. Any user can define user-specific MENU templates via the corresponding Common option.
MENU TREES	The menu system's hierarchical tree-like structures that can be traversed or navigated, like pathways, to give users easy access to various options.
PAC	**Programmer Access Code**. An optional user attribute that can function as a second level password into programmer mode.
PART 3 OF THE KERNEL INIT	See FILE ACCESS SECURITY SYSTEM.
PATTERN MATCH	A preset formula used to test strings of data. Refer to your system's M Language Manuals for information on Pattern Match operations.

PHANTOM JUMP	Menu jumping in the background. Used by the menu system to check menu pathway restrictions.
PRIMARY MENUS	The list of options presented at signon. Each user *must* have a PRIMARY MENU OPTION in order to sign on and reach Menu Manager. Users are given primary menus by IRM. This menu should include most of the computing activities the user will need.
PROGRAMMER ACCESS	Privilege to become a developer on the system and work outside many of the security controls of Kernel. Accessing programmer mode from Kernel's menus requires having the at-sign security code ("@"), which sets the variable DUZ(0)=@.
PROTOCOL	An entry in the PROTOCOL file (#101). Used by the Order Entry/Results Reporting (OE/RR) software to support the ordering of medical tests and other activities. Kernel includes several protocol-type options for enhanced menu displays within the OE/RR software.
QUEUING	Requesting that a job be processed in the background rather than in the foreground within the current session. Kernel's TaskMan module handles the queuing of tasks.
QUEUING REQUIRED	An option attribute that specifies that the option *must* be processed by TaskMan (the option can only be queued). The option can be invoked and the job prepared for processing, but the output can only be generated during the specified time periods.
RESOURCE	A method that enables sequential processing of tasks. The processing is accomplished with a RES device type designed by the application developer and implemented by IRM. The process is controlled via the RESOURCE file (#3.54).
RUBBER-BAND JUMP	A menu jump used to go out to an option and then return, in a bouncing motion. The syntax of the jump is two carets ("^^", uppercase-6 on most keyboards) followed by an option's menu text or synonym (e.g., ^^Print Option File). If the two carets are not followed by an option specification, the user is returned to the primary menu. (See also: GO-HOME JUMP.)
SCHEDULING OPTIONS	A way of ordering TaskMan to run an option at a designated time with a specified rescheduling frequency (e.g., once per week).
SCROLL/NO SCROLL	The Scroll/No Scroll button (also called Hold Screen) allows the user to "stop" (No Scroll) the terminal screen when large amounts of data are displayed too fast to read and "restart" (Scroll) when the user wishes to continue.

SECONDARY MENU OPTIONS	Options assigned to individual users to tailor their menu choices. If a user needs a few options in addition to those available on the primary menu, the options can be assigned as secondary options. To facilitate menu jumping, secondary menus should be specific activities, not elaborate and deep menu trees.
SECURE MENU DELEGATION (SMD)	A controlled system whereby menus and security keys can be allocated by people other than IRM staff (e.g., application coordinators) who have been so authorized. SMD is a part of Menu Manager.
SERVER OPTION	An entry in the OPTION file (#19). An automated mail protocol that is activated by sending a message to the server with the "S.server" syntax. A server option's activity is specified in the OPTION file (#19) and can be the running of a routine or the placement of data into a file.
SIGNON/SECURITY	The Kernel module that regulates access to the menu system. It performs a number of checks to determine whether access can be permitted at a particular time. A log of signons is maintained.
SPECIAL QUEUEING	An option attribute indicating that TaskMan should automatically run the option whenever the system reboots.
SPOOLER	An entry in the DEVICE file (#3.5). It uses the associated operating system's spool facility, whether it's a global, device, or host file. Kernel manages spooling so that the underlying OS mechanism is transparent. In any environment, the same method can be used to send output to the spooler. Kernel will subsequently transfer the text to a global for subsequent despooling (printing).
SYNONYM	A field in the OPTION file (#19). Options can be selected by their menu text or synonym. (See also: MENU TEXT.)
TASKMAN	The Kernel module that schedules and processes background tasks (also called Task Manager).
TIMED READ	The amount of time Kernel will wait for a user response to an interactive READ command before starting to halt the process.
UP-ARROW JUMP	In the menu system, entering a caret ("^"; sometimes referred to as an up-arrow) followed by an option specification/name accomplishes a jump to the target option without needing to take the usual steps through the menu pathway.
XINDEX	A Kernel utility used to verify routines and other M code associated with a software application. Checking is done according to current ANSI MUMPS standards and VistA programming standards. This tool can be invoked through an option or from direct mode (>D ^XINDEX).

Z EDITOR (^%Z)	A Kernel tool used to edit routines or globals. It can be invoked with an option, or from direct mode after loading a routine with >X ^%Z.
ZOSF GLOBAL (^%ZOSF)	The Operating System File—a manager account global distributed with Kernel to provide an interface between VistA software and the underlying operating system. This global is built during Kernel installation when running the manager setup routine (ZTMGRSET). The nodes of the global are filled-in with operating system-specific code to enable interaction with the operating system. Nodes in the ^%ZOSF global can be referenced by VistA application developers so that separate versions of the software need not be written for each operating system.

REF: For a comprehensive list of commonly used infrastructure- and security-related terms and definitions, please visit the Glossary Web page at the following Web address:

http://vaww.vista.med.va.gov/iss/glossary.asp

For a comprehensive list of acronyms, please visit the Acronyms Web site at the following Web address:

http://vaww.vista.med.va.gov/iss/acronyms/index.asp

Index

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z