

# **List Manager 1.0**

## **Developer's Guide**



**March 2024**

**Department of Veterans Affairs (VA)**  
**Office of Information and Technology (OIT)**  
**Software Product Management (SPM)**

## Revision History

Date	Revision	Description	Author
03/19/2024	3.1	Updates: <ul style="list-style-type: none"> <li>• Section <a href="#">2.3.3</a>:Removed references to VALMINIT and to now use KIDS to install List Manger.</li> <li>• Section 508 conformance updates:               <ul style="list-style-type: none"> <li>○ Marked all decorative images throughout.</li> <li>○ Changed all absolute URLs to relative URLs throughout.</li> </ul> </li> <li>• Update reference to the Network File System (NFS; formerly known as the Anonymous Directories).</li> </ul>	VistA Infrastructure Shared Services (VISS) Development Team
06/30/2023	3.0	Updates: <ul style="list-style-type: none"> <li>• Reformatted document to follow current documentation standards and style guidelines.</li> <li>• Added image and table captions throughout.</li> <li>• Verified document is Section 508 conformant.</li> <li>• List Manager Patch VALM*1.0*10: Updated the <a href="#">EN^VALM2()</a>: <a href="#">Generic Selector</a> API: Updated the <b>Options</b> input parameter.</li> </ul> <b>ListMan 1.0</b>	VISS Development Team
04/09/2012	2.0	Updates: Miscellaneous. <b>ListMan 1.0</b>	VistA Infrastructure (VI) Development Team
07/1995	1.0	Initial List Manager (ListMan ) 1.0 Developer's Guide. <b>ListMan 1.0</b>	VistA Infrastructure (VI) Development Team

## Patch Revisions

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

# Table of Contents

Revision History .....	ii
List of Figures.....	vi
List of Tables.....	vi
Orientation.....	viii
<b>1 Introduction .....</b>	<b>1</b>
<b>2 Getting Started .....</b>	<b>2</b>
2.1 List Manager Main Screen.....	2
2.2 List Manager Workbench—^VALMWB .....	3
2.3 Installation and Setup .....	4
2.3.1 Major List Manager Components.....	4
2.3.2 Package Requirements.....	4
2.3.3 Installation.....	5
2.3.4 Terminal Type Attributes for List Manager Users.....	5
<b>3 How to Make a List Manager Application.....</b>	<b>6</b>
3.1 Define List Template.....	6
3.1.1 Create a New List Template .....	6
3.1.2 Create an Outline Routine .....	6
3.1.3 Edit the List Template .....	7
3.1.4 Edit the Outline Routine .....	8
3.1.5 What Comes Next?.....	9
3.2 Define List Array .....	10
3.2.1 Routine to Create List .....	10
3.2.2 Array to Store the List.....	10
3.2.3 Build the List Array Yourself.....	11
3.2.4 Build the List Array Using List Manager's API.....	11
3.3 Define List Actions .....	14
3.3.1 How To Define an Action .....	14
3.3.2 How to Select List Items .....	15
3.3.3 Using the Entire Screen .....	16
3.3.4 When Your Action Completes.....	16
3.4 Define List Menu .....	17
3.4.1 Steps to Set Up Your Application's Menu .....	17
3.4.2 Hidden Menu .....	18
3.4.3 Columnar Arrangement of Menu Items.....	18
3.4.4 Sub-Menus .....	18
3.4.5 Overriding the Default Action .....	19
3.5 Fine Tune Your Application.....	19
3.5.1 Entry Selection and Light Bar Scrolling.....	19

3.5.2	Setting Video Attributes in Your List Line.....	19
3.5.3	Updating Items in the List .....	20
3.5.4	When the User Is In Scrolling Mode (Not Screen Mode) .....	20
3.5.5	Scroll-Locking Columns .....	20
3.5.6	Browsing Word-Processing Fields .....	21
3.5.7	Long Lists .....	21
3.5.8	Calling List Manager and Other Programs from Actions .....	21
3.6	Export Your List Manager Application.....	21
3.6.1	Protocols.....	22
3.6.2	List Templates .....	22
3.6.3	Before Kernel 8.0.....	22
3.7	Example Code .....	23
3.7.1	LIST TEMPLATE PROTOCOL MENU.....	23
3.7.2	PROTOCOL Menu.....	24
3.7.3	PROTOCOL Action.....	25
3.7.4	DISPLAY TYPE .....	26
3.7.5	Application Code Examples .....	27
<b>4</b>	<b>List Template Reference.....</b>	<b>31</b>
4.1	Fields .....	31
4.1.1	Demographics Fields .....	31
4.1.2	Protocol Information Fields .....	31
4.1.3	List Region Fields .....	32
4.1.4	Other Fields .....	33
4.1.5	MUMPS Code Related Fields .....	34
4.1.6	Caption Line Information Fields .....	37
<b>5</b>	<b>Application Programming Interfaces (APIs) .....</b>	<b>39</b>
5.1	List Manager Variables .....	39
5.2	Kernel Video Variables .....	41
5.3	List Manager Generic Action Protocols.....	42
5.4	General APIs .....	43
5.4.1	EN^VALM(): Load a ListMan Template/Application .....	43
5.4.2	SHOW^VALM: Display Menu to User .....	43
5.4.3	PAUSE^VALM1: Pause the Screen.....	44
5.4.4	RANGE^VALM1: Change Date Range.....	44
5.4.5	EN^VALM2(): Generic Selector .....	44
5.5	List Line Text APIs.....	48
5.5.1	FLDUPD^VALM1(): Update Caption Field .....	48
5.5.2	\$\$\$SETFLD^VALM1(): Insert Text in a String .....	49
5.5.3	\$\$\$SETSTR^VALM1(): Set Up String for Display.....	49
5.5.4	FLDTEXT^VALM10(): Inserts Text in a Column .....	50

5.5.5	SET^VALM10(): Construct Initial List Array .....	51
5.6	List Line Video APIs.....	51
5.6.1	CNTRL^VALM10(): Set Video Attributes .....	51
5.6.2	FLDCTRL^VALM10(): Activate Video Control Sequences.....	52
5.6.3	RESTORE^VALM10(): Restores Video Attributes .....	53
5.6.4	SAVE^VALM10(): Save Current Video Attributes .....	54
5.6.5	SELECT^VALM10(): Highlights/Unhighlights Line in List.....	54
5.6.6	WRITE^VALM10(): Re-Write Line to Screen .....	55
5.7	Screen Control APIs .....	55
5.7.1	CHGCAP^VALM(): Changes Label on Caption Header.....	55
5.7.2	CLEAR^VALM1: Clean Up Screen after Error Occurs.....	56
5.7.3	FULL^VALM1: Sets Screen to Full Scrolling Region .....	56
5.7.4	INSTR^VALM1(): Inserts Text on Display Screen.....	56
5.7.5	RE^VALM4: Re-Displays List Header and List Areas .....	57
5.7.6	CLEAN^VALM10: Kills Data and Video Control Arrays .....	57
5.7.7	KILL^VALM10(): Deletes Video Attributes .....	58
5.7.8	MSG^VALM10(): Post Message to “Message Window”.....	58
5.8	Conversion APIs .....	59
5.8.1	\$\$FDATE^VALM1(): Returns Date in “MM/DD/YY” Format.....	59
5.8.2	\$\$FDTTM^VALM1(): Returns Date/Time in “MM/DD/YY@HH:MM” Format 59	
5.8.3	\$\$FTIME^VALM1(): Returns Date/Time in “MMM DD, YYYY@HH:MM” Format .....	60
5.8.4	\$\$LOWER^VALM1(): Converts String from Uppercase to Lowercase.....	60
5.8.5	\$\$NOW^VALM1: Returns Value of “NOW” in External Format.....	61
5.8.6	\$\$UPPER^VALM1(): Converts String from Lowercase to Uppercase .....	61
	Index .....	62

## List of Figures

Figure 1: Sample List Manager Display.....	2
Figure 2: List Manager Workbench .....	3
Figure 3: Using the Workbench to Set Up an Outline Routine for Your Application— System Prompts and User Entries.....	6
Figure 4: Outline Routine Subroutines .....	8
Figure 5: Sample NEW PERSON File Entries Line Display .....	13
Figure 6: Sample M Code Selecting Single Entry Using EN^VALM2.....	16
Figure 7: Sample Protocol Menu Attached to a List Template with TYPE of PROTOCOL .....	23
Figure 8: Sample Protocol Menu .....	24
Figure 9: Sample PROTOCOL Action .....	25
Figure 10: Sample DISPLAY Type.....	26
Figure 11: Sample List Manager Application Codes .....	27
Figure 12: Sample Stub Routines When Adding New List Templates with the Workbench .....	30
Figure 13: EN^ZZVALM2T API—Option “S” Example: System Prompts and User Entries .....	46
Figure 14: EN^ZZVALM2T API—Option “SO” Example: System Prompts and User Entries .....	47
Figure 15: EN^ZZVALM2T API—Option “L” Example: System Prompts and User Entries .....	47
Figure 16: EN^ZZVALM2T API—Option “LO” Quit Example: System Prompts and User Entries .....	48

## List of Tables

Table 1: Documentation Symbol Descriptions.....	ix
Table 2: Sample List Manager Display Key.....	2
Table 3: Package Requirements .....	4
Table 4: Terminal Type Attributes for List Manager Users .....	5
Table 5: List Template Field Categories.....	7
Table 6: Outline Routine Tags.....	9
Table 7: Setting DIR(0) Input Variable to Select Items.....	15
Table 8: VALMBCK Variable Settings When Returning to the List Manager from a Protocol Action .....	16
Table 9: Sample Column Width Settings.....	17
Table 10: List Manager Variables.....	39
Table 11: Kernel Video Variables.....	41

Table 12: List Manager Generic Action Protocols ..... 42

# Orientation

## How to Use this Manual

This manual provides advice and instruction about ListMan 1.0 Application Programming Interfaces (APIs), Direct Mode Utilities, and other information for Veterans Health Information Systems and Technology Architecture (VistA) application developers.

## Intended Audience

The intended audience of this manual is the following stakeholders:

- Software Product Management (SPM)—VistA legacy development teams.
- System Administrators—System administrators at Department of Veterans Affairs (VA) regional and local sites who are responsible for computer management and system security on the VistA M Servers.
- Information Security Officers (ISOs)—Personnel at VA sites responsible for system security.
- Product Support (PS).

## Disclaimers

### Software Disclaimer

This software was developed at the Department of Veterans Affairs (VA) by employees of the Federal Government in the course of their official duties. Pursuant to title 17 Section 105 of the United States Code this software is *not* subject to copyright protection and is in the public domain. VA assumes no responsibility whatsoever for its use by other parties, and makes no guarantees, expressed or implied, about its quality, reliability, or any other characteristic. We would appreciate acknowledgement if the software is used. This software can be redistributed freely provided that any derivative works bear some notice that they are derived from it.



**CAUTION: Kernel routines should *never* be modified at the site. If there is an immediate national requirement, the changes should be made by emergency Kernel patch. Kernel software is subject to FDA regulations requiring Blood Bank Review, among other limitations. Line 3 of all Kernel routines states:**

**Per [VA Directive 6402](#) (pending signature), this routine should not be modified.**



**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).

## Documentation Disclaimer

This manual provides an overall explanation of using ListMan; 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 SharePoint sites and websites for a general orientation to VistA. For example, visit the Office of Information and Technology (OIT) Software Product Management (SPM) Intranet Website.



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

## Documentation Conventions

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. [Table 1](#) gives a description of each of these symbols:

**Table 1: Documentation Symbol Descriptions**

Symbol	Description
	<b>NOTE / REF:</b> Used to inform the reader of general information including references to additional reading material.
	<b>CAUTION / RECOMMENDATION / DISCLAIMER:</b> 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) begin with either “000” or “666”.
  - Patient and user names are formatted as follows:
    - *<Application Name/Abbreviation/Namespace>*PATIENT,<N>
    - *<Application Name/Abbreviation/Namespace>*USER,<N>

Where:

- *<Application Name/Abbreviation/Namespace>* is defined in the Approved Application Abbreviations document.
- <N> represents the first name as a number spelled out and incremented with each new entry.

For example, in ListMan (**VALM**) test patient and user names would be documented as follows:

VALMPATIENT,ONE;VALMPATIENT,TWO; VALMPATIENT,THREE; ...  
VALMPATIENT,14; etc.

VALMUSER,ONE; VALMUSER,TWO; VALMUSER,THREE; ...  
VALMUSER,14; etc.

- “Snapshots” of computer online displays (i.e., screen captures/dialogues) and computer source code is shown in a *non*-proportional font and may be enclosed within a box.
  - User’s responses to online prompts are **boldface** and (optionally) highlighted in yellow (e.g., **<Enter>**).
  - Emphasis within a dialog box is **boldface** and (optionally) highlighted in blue (e.g., **STANDARD LISTENER: RUNNING**).
  - Some software code reserved/key words are **boldface** with alternate color font.
  - 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 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 is 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
```

- The following conventions are used with regards to APIs:
  - The following API types are documented:

- **Supported:**

- This applies where any VistA application may use the attributes/functions defined by the Integration Control Registration (ICR); these are also called “Public”. An example is an ICR that describes a standard API. The package that creates/maintains the Supported Reference *must* ensure it is recorded as a Supported Reference in the ICR database. There is no need for other VistA packages to request an ICR to use these references; they are open to all by default.

- **Controlled Subscription:**

- Describes attributes/functions that *must* be controlled in their use. The decision to restrict the Integration Control Registration (ICR) is based on the maturity of the custodian package. Typically, these ICRs are created by the requesting package based on their independent examination of the custodian package’s features. For the ICR to be approved the custodian grants permission to other VistA packages to use the attributes/functions of the ICR; permission is granted on a one-by-one basis where each is based on a solicitation by the requesting package.



Private APIs are *not* documented.

- Headings for developer API descriptions (e.g., supported for use in applications and on the Database Integration Committee [DBIC] list) include the routine tag (if any), the caret (^) used when calling the routine, and the routine name. The following is an example:

```
EN1^XQH
```

- For APIs that take input parameter, the input parameter is labeled “required” when it is a required input parameter and labeled “optional” when it is an optional input parameter.

- For APIs that take parameters, parameters are shown in lowercase and variables are shown in uppercase. This is to convey that the parameter name is merely a placeholder; M allows you to pass a variable of any name as the parameter or even a string literal (if the parameter is *not* being passed by reference). The following is an example of the formatting for input parameters:

```
XGLMSG^XGLMSG(msg_type, [.]var[, timeout])
```

- Rectangular brackets [ ] around a parameter are used to indicate that passing the parameter is optional. Rectangular brackets around a leading period [.] in front of a parameter indicate that you can optionally pass that parameter by reference.
- All APIs are categorized by function. This categorization is subjective and subject to change based on feedback from the development community. In addition, some APIs could fall under multiple categories; however, they are only listed once under a chosen category.

APIs within a category are first sorted alphabetically by Routine name and then within routine name are sorted alphabetically by Tag reference. The \$\$, ^, or ^% prefixes on APIs is ignored when alphabetizing.

- 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 (e.g., CamelCase).

## Entry Points

For Application Programming Interface (AI) entry points that take input variables, the input variable is labeled optional if it is optional; otherwise, it is a required variable.

For entry points that take parameters, parameters are listed in lowercase. This is to convey that the listed parameter name is merely a placeholder. MUMPS (M) allows you to pass a variable of any name as the parameter or even a string literal (if the parameter is *not* being passed by reference).

The following is an example of the documentation format for input parameters:

```
D XGLMSG^XGLMSG(msg_type, [.]var[, timeout])
```

Rectangular brackets [ ] around a parameter are used to indicate that passing the parameter is optional. Rectangular brackets around a leading period in front of a parameter indicate that you can optionally pass that parameter by reference.

## Documentation Navigation

This document uses Microsoft® Word’s built-in navigation for internal hyperlinks. To add **Back** and **Forward** navigation buttons to the toolbar, do the following:

1. Right-click anywhere on the customizable Toolbar in Word (*not* the Ribbon section).
2. Select **Customize Quick Access Toolbar** from the secondary menu.
3. Select the drop-down arrow in the “Choose commands from:” box.
4. Select **All Commands** from the displayed list.
5. Scroll through the command list in the left column until you see the **Back** command (circle with arrow pointing left).
6. Select/Highlight the **Back** command and select **Add** to add it to your customized toolbar.
7. Scroll through the command list in the left column until you see the **Forward** command (circle with arrow pointing right).
8. Select/Highlight the **Forward** command and select **Add** to add it to the customized toolbar.
9. Select **OK**.

You can now use these **Back** and **Forward** command buttons in the Toolbar to navigate back and forth in the Word document when selecting hyperlinks within the document.



**NOTE:** This is a one-time setup and is automatically available in any other Word document once you install it on the Toolbar.

## How to Obtain Technical Information Online

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



**NOTE:** Methods of obtaining specific technical information online is indicated where applicable under the appropriate section.

## 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** [DILIST] option on the **Data Dictionary Utilities** [DI DDU] menu in VA FileMan to print formatted data dictionaries.



**REF:** For details about obtaining data dictionaries and about the formats available, see the “List File Attributes” section in the “File Management” section in the *VA FileMan Advanced User Manual*.

## Assumptions

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

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

## Reference Materials

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

- *List Manager 1.0 Developer’s Guide* (this manual)
- ListMan VA Intranet Website.

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

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, which is freely distributed by [Adobe® Systems Incorporated](#).

VistA documentation can be downloaded from the [VA Software Document Library \(VDL\)](#).



**REF:** List Manager manuals are located on the [List Manager application on the VDL](#).

Unredacted VistA documentation and software can be downloaded from the Network File System (NFS; formerly known as the Anonymous Directories).

# 1 Introduction

The Veterans Health Information Systems and Technology Architecture (VistA) List Manager (aka ListMan) provides a generic method of presenting lists of items to terminal users. Its core functions are:

- Display a list of items.
- Users can browse through the list.
- Users can select one or more items from the list.
- Users can execute an action for selected list items.
- You can use List Manager recursively within an action.

The *List Manager Developer's Guide* is designed to provide the Department of Veterans Affairs (VA) developer with “how to” information on creating applications using List Manager. This manual is a full reference for creating List Manager applications.

List Manager was originally developed as an interface for the Scheduling module of VistA's MAS V. 5.2 package. Since then it has been used as an interface for a number of other applications, including Text Integration Utility (TIU).



## 2 Getting Started

### 2.1 List Manager Main Screen

[Figure 1](#) is an illustration of the components of a typical List Manager display. The screen is divided into three regions:

- Header Area
- List Area
- Action Area

Figure 1: Sample List Manager Display

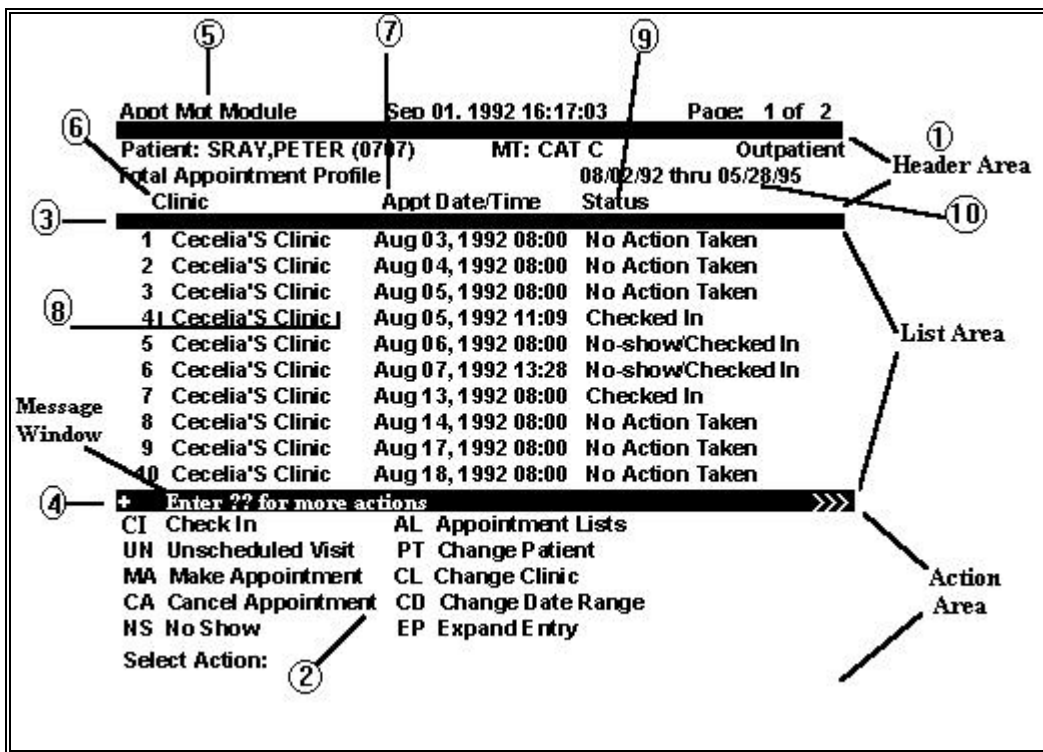


Table 2: Sample List Manager Display Key

Key	Controlled By
1	<a href="#">Header Code</a>
2	<a href="#">Expand Code</a>
3	<a href="#">Top Margin</a>
4	<a href="#">Bottom Margin, Right Margin</a>
5	<a href="#">Screen Title</a>

Key	Controlled By
6	<a href="#">Caption Line Columns</a>
7	<a href="#">Column</a>
8	<a href="#">Array Name</a>
9	<a href="#">Display Text</a>
10	<a href="#">Date Range Limit</a>

You are only allowed to directly **WRITE** to the “Action Area.” The List Manager controls the other two areas. However, you can modify the contents of the “Header Area” and “List Area” by using calls in the [List Manager API](#), and by changing the HEADER and LIST arrays passed to the List Manager.

## 2.2 List Manager Workbench—^VALMWB

The List Manager Workbench ([Figure 2](#)) allows the development of a List Manager application without having to move from one development tool to another. Load the Workbench by calling the ^VALMWB routine.

Figure 2: List Manager Workbench

```

List Manager Workbench      Jan 21, 1993 16:14:15      Page:      1 of      2
Template: VALM OPTION DEMO

Demographics                List Region
Template Name: VALM OPTION DEMO      Top Margin:      5
Entity Name: Option                Bottom   "   : 14
Screen Title: Option List           Right    "   : 240

Protocol Information         Other Fields
Type of List: PROTOCOL           OK to Transport?: OK
Protocol Menu: VALM DEMO         Use Cursor Control: YES
Print Protocol: VALM DEMO PRINT   Allowable Number of Actions: 2
Hidden Menu: VALM HIDDEN ACTIONS   Date Range Limit:

+ Enter ?? for more actions >>>>
DE Demographic Edit      MC MUMPS Code Edit      PE Protocol Edit
PI Protocol Information  CE Caption Edit         RN Run List
LR List Region Edit      CL Change List Template IT Input Template
OF Other Fields          EA Edit All             RO Routine Editor
Select Tool:Next Screen//

```

The Workbench allows you to edit all of the data for the following:

- [List Template](#)
- [Action Protocols](#)
- [Menu Protocols](#)
- Input Templates
- Routines

In short, every part of a List Manager application.

You can run a List Template from the Workbench. When you run a template, you are prompted for any “setup” code to initialize variables. This is needed if the template is *not* a top-level template. After “running” the template, you are returned to the Workbench. The Workbench itself is a List Manager application.

We *recommend* that you do all List Template development using the Workbench. As new features become available, the Workbench will automatically present them to you.

## 2.3 Installation and Setup

### 2.3.1 Major List Manager Components

The major List Manager components include:

- LIST TEMPLATE (#409.61) file.
- PROTOCOL (#101) file.
- Routines in the **VALM\*** namespace (List Manager routines).
- Routines in the **XQOR\*** namespace (Protocol Processing routines).

### 2.3.2 Package Requirements

[Table 3](#) lists the packages and versions that *must* be present for List Manager to run properly:

**Table 3: Package Requirements**

Package	Version
Order Entry Results Reporting (OERR)	2.5 or greater
Kernel	6.5 or greater

### 2.3.3 Installation

Use Kernel Installation and Distribution System (KIDS) to install the List Manager.

### 2.3.4 Terminal Type Attributes for List Manager Users

[Table 4](#) lists the terminal type attributes that *must* be defined for List Manager users in order to effectively use the List Manager:

**Table 4: Terminal Type Attributes for List Manager Users**

<b>TERMINAL TYPE Field</b>	<b>Example Field Values for VT-100 Terminal</b>
Form Feed	#, \$C(27,91,50,74,27,91,72)
XY CRT	W \$C(27,91),DY+1,\$C(59),DX+1,\$C(72)
Erase to End of Page	\$C(27,91,74)
Insert Line	\$C(27,91), "1L"
Underline On	\$C(27,91,52,109)
Underline Off	\$C(27,91,109)
High Intensity	\$C(27,91,49,109)
Normal Intensity	\$C(27,91,109)
Save Cursor Position	\$C(27,55)
Restore Cursor Pos	\$C(27,56)
Set Top/Bottom Marg	\$C(27,91),+IOTM,\$C(59),+IOBM,\$C(114)
SGR Attributes Off	\$C(27,91,109)

## 3 How to Make a List Manager Application

### 3.1 Define List Template

The first step to create a List Manager application is to create the List Template for your application. A List Template is the core of a List Manager application; all of the crucial information that determines how a list works is stored in an application's List Template. The best way to set up (and maintain) a List Template is to use the [Workbench](#).

#### 3.1.1 Create a New List Template

When you invoke the [Workbench](#), it asks you for a List Template name. You can either enter an existing one or create a new one.

#### 3.1.2 Create an Outline Routine

List Templates depend on calling several subroutines to perform specific actions, including:

- Initializing your application.
- Creating the array of list items that becomes your list.

As such, creating these subroutines is central to your List Template. That is why the next question you are asked after you name your template is “Enter Routine Name: ”.

The [Workbench](#) can create an outline routine that contains subroutines to perform all of the functions List Manager requires. Entering a name is optional. However, if you enter a name for a routine, the [Workbench](#) creates an outline routine for your application with stub tags and code for the template. The created List Template is then immediately executable.

[Figure 3](#) demonstrates how you can use the [Workbench](#) to set up an outline routine for your application:

**Figure 3: Using the Workbench to Set Up an Outline Routine for Your Application—System Prompts and User Entries**

```
Select LIST TEMPLATE NAME: ZZLIST
Are you adding 'ZZLIST' as a new LIST TEMPLATE (the 14TH)? Y <Enter> (Yes)

>>> The system will create a stub routine...

>>> Enter Routine Name: ZZLIST

I am going to create a series of 'ZZLIST*' routines.
Is that OK? Yes// <Enter> (Yes)

>>> Building 'ZZLIST' stub routine
ZZLIST has been filed
```

A fully functional List Manager application (with a “dummy” list of items) has now been created; and you are placed in the [Workbench](#) with the new List Template loaded.

### 3.1.3 Edit the List Template

The [Workbench](#) lets you edit all of the fields in the List Template. It organizes the fields in a list template into six distinct categories, as shown in [Table 5](#).

**Table 5: List Template Field Categories**

Category	Description
<a href="#">Demographics</a>	Set up the list name, generic prompt, and screen title.
<a href="#">Protocol Information</a>	Set up the menus for your list.
<a href="#">List Region</a>	Set the screen region for the list.
<a href="#">Other Fields</a>	Set miscellaneous list attributes.
<a href="#">MUMPS Code Related</a>	<p>Specify the routines for:</p> <ul style="list-style-type: none"> <li>• Header</li> <li>• Entry</li> <li>• Exit</li> <li>• Expand</li> <li>• Help</li> </ul> <p>Optionally, enter the array name in which that list is kept. When List Manager creates an outline routine, it uses that routine for most of these tasks.</p>
<a href="#">Caption Line Information</a>	Define the contents of the caption line (list headings).

The [Workbench](#) also lets you perform a number of actions beyond editing the List Template. One of the actions you can perform is running the list (**Run List** action). Try running the list now as set up by default by List Manager. This gives you an idea of what a bare bones List Manager application looks like.

Later, as you add enhancements to your application, you can use the [Workbench](#) to edit a number of your List Template’s fields.

### 3.1.4 Edit the Outline Routine

The outline routine that was created contains **six** specific subroutines ([Figure 4](#)). By going through each subroutine, you see the beginning of your application.

**Figure 4: Outline Routine Subroutines**

```
ZZKYLM ; ; 08-OCT-1996
      ;; ;
EN    ; -- main entry point for ZZLIST
      D EN^VALM("ZZLIST")
      Q
      ;
HDR   ; -- header code
      S VALMHDR(1)="This is a test header for ZZLIST."
      S VALMHDR(2)="This is the second line"
      Q
      ;
INIT  ; -- init variables and list array
      F LINE=1:1:30 D SET^VALM10(LINE,LINE_"      Line number "_LINE)
      S VALMCNT=30
      Q
      ;
HELP  ; -- help code
      S X="?" D DISP^XQORM1 W !!
      Q
      ;
EXIT  ; -- exit code
      Q
      ;
EXPND ; -- expand code
      Q
      ;
```

**Table 6: Outline Routine Tags**

Outline Routine Tag	Description
<b>EN</b>	<b>Application Entry Point:</b> This section of the code in the outline routine is the line of code to independently invoke List Manager and load your List Template (and your list). If you were to make an option for your List Manager application, you would set the option's RUN ROUTINE field to this tag and routine.
<b>HDR</b>	<b>Header Code:</b> In this very simple section of the outline routine, two nodes of the <b>VALMHDR</b> array are set. These should be set to the text lines to display in the "Header Area" of the List Manager screen. List Manager calls this subroutine when initializing your list.
<b>INIT</b>	<b>List Creation:</b> In this section of the outline routine, all the work is done to create the list of items that is displayed to the user by List Manager. Setting up your list is discussed in more detail in the " <a href="#">Define List Array</a> " section.
<b>HELP</b>	<b>Help:</b> You can set up custom help in this subroutine. When a user enters a "?" at the menu prompt, your custom help would be called. This is an optional feature.
<b>EXIT</b>	<b>Exit Code:</b> Use this subroutine to clean up variables and any other exit processing your application needs to perform before exiting.
<b>EXPND</b>	<b>Expand Code:</b> This subroutine is for placing MUMPS (M) code to display a detailed inquiry-type report/screen for a specific entry in the list. This is an advanced, optional feature.

In the "[Define List Array](#)" section you edit the outline routine's **INIT** subroutine, replacing the "dummy" list of items created in the stub subroutine with your application's list items. This is the next step in your application, setting up the list of items for List Manager to display to your list user.

### 3.1.5 What Comes Next?

You have created a List Template for your application. You have created an outline routine for your application. So, what comes next?

You need to [set up the list of items](#) that your application displays to your list user. Setting up the list is the **second of four steps** in creating a List Manager application.

To add functionality to your application, you need to [create Action-type protocols](#). These are akin to menu options and are the actions available to your list users in the "Action Area" at the bottom of the List Manager screen ([Figure 1](#)). These actions let your list users select items and perform actions with the select items. Creating actions is the **third of four steps** in creating a List Manager application.

Finally, once you create some Action-type protocols, you need to [create a Menu-type protocol](#). Then, attach all of your actions to the Menu-type protocol, and designate the menu protocol as



your List Template's **Protocol Menu**. Then, run your application and test out all of your actions. Organizing your menu is the **fourth of four steps** in creating a List Manager application.

## 3.2 Define List Array

Once you have created a List Template to define your List Manager application, the next step is to set up the array (list) of items that is displayed to your list user. You set up the list array using M code in the routine specified in the List Template's [ENTRY CODE](#) field.

### 3.2.1 Routine to Create List

The routine specified in the [ENTRY CODE](#) field in the “[MUMPS Code Related Fields](#)” section of the [Workbench](#) ([Figure 2](#)) is what List Manager calls to set up your list. So, you *must* set your list array up in a routine.

If you let List Manager create an outline routine for your List Template, it sets this field in the List Template to the [INIT](#) label of the routine it creates. In the created outline routine, it sets up a “dummy” list using the [SET^VALM10](#) entry point. If you look at the code it puts in this subroutine, you can see one way to create a list. You can set up a list entirely yourself, or you can use some of List Manager's entry points. Both methods are described below.

### 3.2.2 Array to Store the List

The [ARRAY NAME](#) field in a List Template, in the “[MUMPS Code Related Fields](#)” section of the [Workbench](#), should contain the name of the array that holds your list of items to be displayed.



**NOTE:** The array name *must* be preceded by a space character. This is needed to allow global specification. VA FileMan does *not* allow “^” as the first character. The array can be either a local or global variable.

The array of list items you create needs to follow the format used in word-processing fields:

```
^TMP("SDAM",$J,line #,0)=display_string
```

There is one case in which you do *not* need to specify the array name in the [ARRAY NAME](#) field. By making calls to [SET^VALM10](#), you can have the List Manager decide where to store the list array. This method of creating a list is discussed in the “[Build the List Array Yourself](#)” section.

### 3.2.3 Build the List Array Yourself

To create a list of items yourself, do the following:

1. In the routine called by the [ENTRY CODE](#) field of the List Template, make an array of items in the list. Make sure your array is in the same format as word-processing fields, that is, `^TMP("SDAM",$J,line #,0)=display_string`. The list array should start with list item **1**, and there should be no gaps in the array line sequence.
2. It is a good idea to include the line number as the first part of the text of each display line. This aids list users when selecting items.
3. Set the [ARRAY NAME](#) field of the List Template to the name of the array.
4. Set the [VALMCNT](#) variable equal to the number of items in your list.
5. You are done!

Somewhere else, you may want to store a corresponding index of the entry number for items in your list, if your items correspond to entries in a file. Then, when you get to making actions, you are able to associate an item in the list with the entry number from which it came.

### 3.2.4 Build the List Array Using List Manager's API

List Manager provides an API, which includes entry points for creating and maintaining lists.

#### 3.2.4.1 Creating the Array with SET^VALM10

You can create the array entries in your list using the [SET^VALM10](#) entry point. When you do this, you do *not* need to set an explicit array name in the List Templates [ARRAY NAME](#) field. List Manager maintains the array itself, without you needing to know where it is stored. If you need to reference lines in the array, you can use the `@VALMAR@(<line #>,0)` syntax.

To set up and maintain your array using [SET^VALM10](#), do the following:

1. All of the code that follows should be in the routine called by the [ENTRY CODE](#) field of the List Template.
2. Keep in mind that your list array should start with list item **1**, and that there should be no gaps in the array sequence of lines.
3. To add a line to the list, make a call to [SET^VALM10](#):

```
D SET^VALM10(line_num,display_text)
```

4. It is a good idea to include the line number as the first part of the text of each display line. This aids list users when selecting items.

5. If the items in your list correspond to file entries, you may want to keep track of the internal entry number (IEN) for each list item. Simply use the optional third parameter of the [SET^VALM10](#) call to associate an internal entry number with your list item. You can then retrieve the associated internal entry number for any line with the code:

```
S Y=$O(@VALMAR@("IDX",56,""))
```

6. When you are done adding lines to the list, set the [VALMCNT](#) variable equal to the number of items in your list.
7. You are done!

### 3.2.4.2 Setting Up Text Lines with Captions and \$\$SETFLD^VALM1

To help formatting each line of text for display, you may want to consider using captions and the [\\$\\$SETFLD^VALM1](#) API. This lets you format text in a line based on any caption items you may have set up in your List Template. In the “Caption Line Information” section of the [Workbench](#), you can enter caption items. Each caption item has the following:

- Name
- Length
- Column position
- Default video attributes
- Display text fields

[\\$\\$SETFLD^VALM1](#) lets you position pieces of text in your list lines based on how you set up captions for your line in the List Template.

Supposing you have set up **four** caption items in your List Template, named:

- “LINENO”
- “NAME”
- “INIT”
- “FM ACCESS CODE”

When you create your list array, you could loop through entries in the NEW PERSON (#200) file, and format a line to display for each NEW PERSON entry as follows:

**Figure 5: Sample NEW PERSON File Entries Line Display**

```
S LINE=0,EN=.9 F S EN=$O(^VA(200,EN)) Q:'+EN D
.S LINE=LINE+1
.S ZZNODE0=$G(^VA(200,EN,0)),LINEVAR=""
.S ZZNA=$P(ZZNODE0,U,1),ZZIN=$P(ZZNODE0,U,2),ZZFM=$P(ZZNODE0,U,4)
.S LINEVAR=$$SETFLD^VALM1(LINE_".",LINEVAR,"LINENO")
.S LINEVAR=$$SETFLD^VALM1(ZZNA,LINEVAR,"NAME")
.S LINEVAR=$$SETFLD^VALM1(ZZIN,LINEVAR,"INIT")
.S LINEVAR=$$SETFLD^VALM1(ZZFM,LINEVAR,"FM ACCESS CODE")
.D SET^VALM10(LINE,LINEVAR) ; adds formatted line to list array
```

Now, your lines of text are set up according to your captions in your List Template. And if you adjust the positions of your List Template captions, your text lines are automatically adjusted too!



**NOTE:** If you have a large NEW PERSON (#200) file, and you try this example, make sure you loop only through some subset of it; lists become difficult to use once there are more than a certain number of screens in the list (**10 screens** in a list is probably a good limit).

### 3.2.4.3 Setting and Displaying Video Attributes for List Lines with FLDCTRL^VALM10

In the “Caption Line Information” section of the [Workbench](#), you can enter caption items. Each caption item has the following:

- Name
- Length
- Column position
- Default video attributes
- Display text fields

This provides a way to organize your lines of text, based on caption positions.

Using the [FLDCTRL^VALM10](#) entry point, you can set the video attributes for different portions of your line based on the default video attributes entered for every caption in the line. For example, you have a caption of length **10** starting at column **40**, with a default video attribute of **REVERSE**. If you call [FLDCTRL^VALM10](#) for a line number, all default video attributes

for the line are activated, and the region of that line from column **40** to column **49** are displayed in reverse video.

To activate the default video attributes for all lines in your array, do the following:

1. Using the [Workbench](#), set up caption items for each portion of your display line. Set the default video attributes as desired for each caption item.
2. After you add each line to the list array, make a call to [FLDCTRL^VALM10](#)(line). So, you need to call [FLDCTRL^VALM10](#) once for each line you add to the array.
3. When you run your list, each line you called [FLDCTRL^VALM10](#) is displayed with the video attributes set up in the List Template captions.

### 3.3 Define List Actions

Once you have created your List Template, and your list, the next step is to create actions for your list. Actions are what appear as menu items in the bottom of the List Manager screen. They allow you to launch any routine from a List Manager menu. Actions are stored as protocols, of TYPE **ACTION**, in the PROTOCOL (#101) file.

List Manager supplies a set of [pre-defined actions](#) that you include with your List Manager application. It is usually a good idea to make use some of these, such as [VALM DOWN A LINE](#), [VALM UP ONE LINE](#), [VALM NEXT SCREEN](#), etc. to provide the basic list functionality users expect.

In addition, you will probably want to define your own actions to add your own custom functionality to your list.

#### 3.3.1 How To Define an Action

To define an action, do the following:

1. From the [Workbench](#), choose **PE** for Protocol Edit.
2. Add a new protocol.
3. Set the new protocol's TYPE to **ACTION**.
4. Set the ITEM TEXT field to the menu item text for this action.
5. Set the ENTRY ACTION field to call a routine that performs your action(s).
6. Use the EXIT ACTION field to set List Manager status variables *before* returning control to List Manager.
7. Add your new action-type protocol to the menu-type protocol that is the main menu for your application; this makes it a menu item in List Manager.



**REF:** For information on how to do this, see the “[Define List Menu](#)” section.

The following are some more issues to consider for your actions:

- How to select items from the list in your action.
- How to determine in what Screen Mode the user is located.
- Getting control of the screen.
- What List Manager should do when your action completes.
- How to display a custom message after completing an action.

### 3.3.2 How to Select List Items

In the routine called by an action, you may want to select an item or items from the List Manager list. One easy way to do this is to make a ^DIR call in your action. Set up the **DIR(0)** input variable to ask for numbers in the range of the entire list, or only what items are displayed on the current screen, as follows ([Table 7](#)):

**Table 7: Setting DIR(0) Input Variable to Select Items**

List Item Selection	DIR(0) Input Variable
1 item from entire list	S DIR(0)="N^1:"_VALMCNT_" :0"
1 item from current screen	S DIR(0)="N^"_VALMBG_" :"_VALMLST_" :0"
Set of items from entire list	S DIR(0)="L^1:"_VALMCNT
Set of items from current screen	S DIR(0)="L^"_VALMBG_" :"_VALMLST

The interaction with the user takes place in the lower part of the screen. From the output of the ^DIR call, you have the array number(s) of the selected item(s); you can then perform whatever action you would like with the selected item(s). If the user chooses an item or set of items (as reflected in the output variables from the ^DIR call), you can either process the items immediately, or highlight them (current screen only) for further action.

Another way to select entries is to use the List Manager entry point [EN^VALM2](#). This is a generic selector that prompts the user to select list items from the current screen only.

[Figure 6](#) is a sample of the code you could call to select a single entry using [EN^VALM2](#):

**Figure 6: Sample M Code Selecting Single Entry Using EN^VALM2**

```
N ZZVALM,ZZEN
S ZZVALM="DUZ^1^ASDF^ASDF" ;?? Need to confirm how to set this up!
D EN^VALM2(ZZVALM,"O")
S ZZEN=$O(VALMY("")) ; get line number of selected entry
I '+ZZEN W !,"No Entry Selected!" H 5 Q
W !,"You selected ",@VALMAR@(ZZEN,0) H 5
Q
```

### 3.3.3 Using the Entire Screen

If your action needs control over the entire screen, make a call to [FULL^VALM1](#) at the beginning of your action's code. This call changes the scrolling region to be the full screen, and turns word-wrap on, and all user interaction is in Scrolling Mode. When you return control back to the List Manager, set **VALMBCK** to "R". This refreshes the screen and resets the scrolling region as needed by List Manager.

### 3.3.4 When Your Action Completes

When returning to the List Manager from a protocol action, make sure the [VALMBCK](#) variable is set ([Table 8](#)). This tells List Manager what to do when returning from your action.

**Table 8: VALMBCK Variable Settings When Returning to the List Manager from a Protocol Action**

VALMBCK Value	Description
R	Refresh Screen.
<NULL>	Clear bottom portion of screen and prompt for action.
Q	Exit ( <b>Quit</b> ) List Manager.

If *not* defined after an action, the List Manager acts like it was set to "Q" (**Quit**).

If you want to display a custom message in the message window after completing an action, set the [VALMSG](#) variable with the text desired. The message area allows up to **50 characters**.



**REF:** For more information, see the description of [MSG^VALM10](#).

## 3.4 Define List Menu

The final step in building a List Manager application is to create the menu for your list. This provides the set of choices at the bottom of the List Manager screen. You can create new actions to add to your menu, and/or use generic List Manager actions as well.

### 3.4.1 Steps to Set Up Your Application's Menu

To set up your application's menu, do the following:

1. From the [Workbench](#), choose **PE** for Protocol Edit.
2. Add a new protocol.
3. Set the new protocol's TYPE to **MENU**.
4. Set the new protocol's COLUMN WIDTH as follows:

**Table 9: Sample Column Width Settings**

# of Columns Desired	Column Width Setting
1	1
2	40
3	26

5. Set the new protocol's MNEMONIC WIDTH to a width that provides for the length of your longest menu item mnemonic, plus white space to separate the mnemonics from the menu text. If your longest mnemonic is **2 characters**, setting this field to **4** provides **2 characters** of white space.
6. Add any actions (either custom actions created by you, or [generic actions](#)) as ITEMS to the new protocol. You can set a mnemonic and a sequence number for each item.
7. You *must* include the following code in the HEADER field of the menu protocol:

```
D SHOW^VALM
```

The [SHOW^VALM](#) API properly displays the list of actions to the user in the “Action Area.”

8. In the MENU PROMPT field, set the text by which the users will be prompted. For example: “Select Action: ” is a good choice.
9. Once you finish editing the menu protocol, return to the [Workbench](#). Set the TYPE OF LIST to **PROTOCOL** (*not* **DISPLAY!**). This enables the list to use your new protocol, instead of the standard **VALM DISPLAY** protocol.
10. Set the PROTOCOL MENU to the name of the menu-type protocol you just created.
11. Test your new menu by choosing “**Run List**” from the [Workbench](#).



You should consider the following additional issues when setting up protocols for use by the List Manager:

- [Hidden Menu.](#)
- [Columnar Arrangement of Menu Items.](#)
- [Sub-Menus.](#)
- [Overriding the Default Action.](#)

### 3.4.2 Hidden Menu

In the [Workbench](#), you can set your list's Hidden Menu to the name of any menu protocol. This is typically used to provide some of the more basic actions like line up and line down, especially when the main menu has a lot of custom items. By default, the [Workbench](#) sets up lists to use the generic [VALM HIDDEN ACTIONS](#) protocol as the hidden menu. This provides access to all of the generic List Manager actions for negotiating the list. You can set the hidden menu to your own hidden menu, if you wish.

### 3.4.3 Columnar Arrangement of Menu Items

If the number of columns desired for your menu items is more than **one** and if you want to place each action in a particular column, you should use a **SEQUENCE** numbering scheme for the items in the menu.

List Manager displays your menu items in the minimum number of rows possible, given the number of items and the number of columns you have specified. It will place items in sequence as follows:

1	4	7
2	5	8
3	6	9

Knowing how List Manager places items, you can use sequencing to control in which column an item is placed.

If the number of items to appear in each column is *not* equal then you *must* add “blank” items and place the blank protocol in the appropriate column as described above.

A “blank” protocol is an action protocol with the ITEM TEXT and ENTRY ACTION fields left blank.

### 3.4.4 Sub-Menus

If you use a sub-menu, then the HEADER field of the (top menu) should contain a **W “”**.

### 3.4.5 Overriding the Default Action

The List Manager automatically provides a default action of “**Next Screen**” or “**Quit**”. However, you can override this default action by setting the [XQORM\(“B”\)](#) variable as part of the ENTRY ACTION code for a PROTOCOL menu.

## 3.5 Fine Tune Your Application

A number of ways that you can fine-tune a basic List Manager application are discussed in the following sections:

- Entry Selection and Light Bar Scrolling
- [Setting Video Attributes in Your List Line](#)
- [Updating Items in the List](#)
- [When the User Is In Scrolling Mode \(not Screen Mode\)](#)
- [Scroll-Locking Columns](#)
- [Browsing Word-Processing Fields](#)
- [Long Lists](#)
- [Calling List Manager and Other Programs from Actions](#)

### 3.5.1 Entry Selection and Light Bar Scrolling

List Manager does *not* support a scrolling “light bar” for entry selection. When the user presses the up and down arrow keys, there is *not* a way to hook those key presses to a scrolling light bar in the list of entries.

For entry selection, the best method is to make sure that in the text of each line, the line number is shown (preferably on the left hand side of the line). Then, you can make your own call using ^DIR, or use the [EN^VALM2](#) generic selector, to let your users choose entries. If you want to select an entry and perform an action all at once, you can do this. Another style is to have one action that selects entries. You can then use [SELECT^VALM10](#) to highlight that line of the array. This is useful if there are multiple actions a user can perform on a selected entry or entries. You can let the user select the entries, highlight them, and then have the user perform actions on the set of highlighted entries.

### 3.5.2 Setting Video Attributes in Your List Line

One enhancement you can make to your list application is setting and changing the video attributes in your list lines.

Before you load your list, for example, you can set what the video attributes (highlight, reverse video, underline, or blinking) should be for any given caption field in a line. Do this in the List Template, by editing the Default Video Attributes for your captions. Then, when you build your array list initially, you can activate these List Template attributes for each line by making calls to [FLDCTRL^VALM10](#).

Once your list is already up and displayed, you can still change the video attributes of your lines. To change video attribute based on screen position, use [CNTRL^VALM10](#). You can save ([SAVE^VALM10](#)) and restore ([RESTORE^VALM10](#)) a line's video attributes.

You can also select a line using [SELECT^VALM10](#).

### 3.5.3 Updating Items in the List

Another enhancement you can make to your list application is actively updating the lines in your list. While you *cannot* add lines to the list, you can change the contents of existing lines. This is useful, particularly if in your actions you are editing file entries, whose contents correspond to what is displayed in your list.

When a user updates an entry, you can update the corresponding list array line with a call to [FLDTEXT^VALM10](#), and then re-paint the line on the display with a call to [WRITE^VALM10](#). You can also insert text into an existing line based on caption position, using [FLDTEXT^VALM10](#).

### 3.5.4 When the User Is In Scrolling Mode (Not Screen Mode)

The [VALMCC](#) variable is always available to indicate the user's screen mode in List Manager:

- **1**—Screen Mode
- **0**—Scrolling Mode

If the user is signed on to the system using a terminal type that does *not* support the cursor control fields needed by the List Manager, List Manager automatically defaults to Scrolling Mode. This means that the list array and headers will always be totally re-painted to the screen after each action.

There may be times that the application code needs to know if the job is in Scrolling Mode. For example, if only one field in one entry is to be changed as a result of an action and the user was working totally in the “Action Area” of the screen, then the code could simply use the appropriate call to update just that field and set the [VALMBCK](#) variable to **NULL**. However, if the user is in Scrolling Mode, then you would *not* update the screen and would set the [VALMBCK](#) variable to “R”.

### 3.5.5 Scroll-Locking Columns

If your list display is going to be more than fits on a user's screen (greater than **80** or **132 columns**), you can set a Scroll Lock, so that to the left of the Scroll Lock, no scrolling occurs. This feature is based on caption fields (another good reason to set up your lines using caption fields). You can only set one caption field as the point at which no scrolling occurs. That field, and everything to the left of it, is stationary when the user scrolls the rest of the list to the right.

### 3.5.6 Browsing Word-Processing Fields

It is easy to browse word-processing fields using List Manager. Set the TYPE of your template to **DISPLAY**. This provides a menu of standard actions (line up, line down, etc.). Then, for the array, simply set the [ARRAY NAME](#) field to the global location of your word-processing field. List Manager expects the array to be in the format of a word-processing field, so at that point you are done.

You can also launch the VA FileMan Browser from within List Manager to browse a word-processing field or global array. As different mix of features is offered when browsing word-processing fields with the VA FileMan browser.

### 3.5.7 Long Lists

You should not use List Manager to display very long lists of entries. Although there is no limit other than that of system resources on the size of a list, you may find that users have difficulty if there are more than, for example, **10 screens** in the list. The exact limit on the number of screens can depend on the type of information in the list, and how willing your user is to go through such a list. At some point, performance also becomes a consideration, especially if you are building your list array.

### 3.5.8 Calling List Manager and Other Programs from Actions

From an action in your List Manager application, you can call List Manager again. It is re-entrant. You can also call other applications, for example ScreenMan, the VA FileMan Browser. You do *not* need to **NEW** any variables when calling these applications.

## 3.6 Export Your List Manager Application

Kernel 8.0's Kernel Installation and Distribution System (KIDS) made List Manager templates and protocols standard package components. This enables List Manager applications to be distributed just like any other package, using KIDS.

To export your List Manager application, you need to export your application's protocols and your application's List Template, as well as routines, options, and any other supporting components.



**REF:** For more information on KIDS, see the *Kernel 8.0 and Kernel Toolkit 7.3 Systems Management Guide* and *Kernel 8.0 and Kernel Toolkit 7.3 Developer's Guide*.

### 3.6.1 Protocols

With Kernel 8.0's Kernel Installation and Distribution System (KIDS), you can include protocols as package components in a KIDS build. You can then export your List Manager application in a KIDS build.



**REF:** For more information on KIDS, see the *Kernel 8.0 and Kernel Toolkit 7.3 Systems Management Guide* and *Kernel 8.0 and Kernel Toolkit 7.3 Developer's Guide*.

Prior to Kernel 8.0, in order to export protocols, you would have needed to use the **ORVOM** tool.



**REF:** For more information of the **ORVOM** process, see the *Order Entry/Results Reporting Developer's Guide*.

### 3.6.2 List Templates

With Kernel 8.0's Kernel Installation and Distribution System (KIDS), and with Kernel patch XU\*8.0\*2 installed, you can include List Templates as package components in a KIDS build. You can then export your List Manager application in a KIDS build.



**REF:** For more information on KIDS, see the *Kernel 8.0 and Kernel Toolkit 7.3 Systems Management Guide* and *Kernel 8.0 and Kernel Toolkit 7.3 Developer's Guide*.

### 3.6.3 Before Kernel 8.0

Prior to Kernel 8.0, in order to export List Templates, you would have needed to use the **^VALMW3 List Manager** utility.

## 3.7 Example Code

### 3.7.1 LIST TEMPLATE PROTOCOL MENU

[Figure 7](#) is an example of a protocol menu that would be attached to a List Template that has a TYPE of **PROTOCOL**.

**Figure 7: Sample Protocol Menu Attached to a List Template with TYPE of PROTOCOL**

```
NAME: SDAM MENU
ITEM TEXT: Appointment Management
TYPE: menu
PACKAGE: SCHEDULING
DESCRIPTION: This menu contains all the activities for the appointment
management option.
COLUMN WIDTH: 26
MNEMONIC WIDTH: 4

ITEM: SDAM APPT CHECK IN           MNEMONIC: CI           SEQUENCE: 11
ITEM: SDAM APPT UNSCHEDULED       MNEMONIC: UN           SEQUENCE: 12
ITEM: SDAM APPT MAKE              MNEMONIC: MA           SEQUENCE: 13
ITEM: SDAM APPT CANCEL            MNEMONIC: CA           SEQUENCE: 21
ITEM: SDAM APPT NO-SHOW           MNEMONIC: NS           SEQUENCE: 22
ITEM: SDAM LIST MENU              MNEMONIC: AL           SEQUENCE: 23
ITEM: SDAM PATIENT CHANGE         MNEMONIC: PT           SEQUENCE: 31
ITEM: SDAM CLINIC CHANGE          MNEMONIC: CL           SEQUENCE: 32
ITEM: SDAM DATE CHANGE            MNEMONIC: CD           SEQUENCE: 33

HEADER: D SHOW^VALM
MENU PROMPT: Select Action:
```

### 3.7.2 PROTOCOL Menu

The **PROTOCOL** menu is a sub-menu of the **SDAM APPOINTMENT MENU**. Please note the header ([Figure 8](#)).

**Figure 8: Sample Protocol Menu**

```
NAME: SDAM LIST MENU
ITEM TEXT: Appointment Lists
TYPE: menu
PACKAGE: SCHEDULING
COLUMN WIDTH: 40
ITEM: SDAM LIST CHECKED IN           MNEMONIC: CI
ITEM: SDAM LIST NO SHOWS             MNEMONIC: NS
ITEM: SDAM LIST ALL                   MNEMONIC: TA
ITEM: SDAM LIST NO ACTION             MNEMONIC: NA
ITEM: SDAM LIST CANCELLED             MNEMONIC: CA
ITEM: SDAM LIST FUTURE                MNEMONIC: FU
ITEM: SDAM LIST INPATIENT             MNEMONIC: IP
ITEM: SDAM LIST NON-COUNT             MNEMONIC: NC

EXIT ACTION: S:'$D(VALMBCK) VALMBCK="" D EXIT^SDAM
ENTRY ACTION: S XQORM(0)="1A"
HEADER: W ""
MENU PROMPT: Select List:
MENU DEFAULT: No Action Taken
```

### 3.7.3 PROTOCOL Action

[Figure 9](#) is sample PROTOCOL action:

**Figure 9: Sample PROTOCOL Action**

```
NAME: SDAM LIST CANCELLED
ITEM TEXT: Cancelled
TYPE: action
PACKAGE: SCHEDULING
DESCRIPTION: This list will display all the cancelled appointments for
the date range specified.
ENTRY ACTION: S X="CANCELLED" D LIST^SDAM
Appendix B - Sample List Template File Entries
PROTOCOL TYPE

NAME: SDAM APPT MGT
TYPE OF LIST: PROTOCOL
HIDDEN PROTOCOL MENU: VALM HIDDEN ACTIONS
LEFT MARGIN: 1
RIGHT MARGIN: 80
TOP MARGIN: 5
BOTTOM MARGIN: 14
RIGHT MARGIN: 80
OK TO TRANSPORT?: OK
USE CURSOR CONTROL: YES
ENTITY NAME: Appointment
PROTOCOL MENU: SDAM MENU
SCREEN TITLE: Appt Mgt Module
ALLOWABLE NUMBER OF ACTIONS: 1
DATE RANGE LIMIT: 999
ARRAY NAME: ^TMP("SDAM",$J)
ITEM NAME: NAME          COLUMN: 9   WIDTH: 22   DISPLAY TEXT: Patient or
Clinic
ITEM NAME: DATE          COLUMN: 32  WIDTH: 20   DISPLAY TEXT: Appt Date/Time
ITEM NAME: STAT          COLUMN: 53  WIDTH: 22   DISPLAY TEXT: Status
ITEM NAME: APPT#         COLUMN: 5   WIDTH: 3
ITEM NAME: TIME          COLUMN: 75  WIDTH: 5

EXPAND CODE: D EN^SDAMEP
EXIT CODE: D FNL^SDAM
HEADER CODE: D HDR^SDAM
HELP CODE: D HLP^SDAM5
ENTRY CODE: D INIT^SDAM
```



### 3.7.4 DISPLAY TYPE

[Figure 10](#) is a sample **DISPLAY** type:

**Figure 10: Sample DISPLAY Type**

```
NAME: SDAM APPT PROFILE
TYPE OF LIST: DISPLAY
HIDDEN PROTOCOL MENU: VALM HIDDEN ACTIONS
TOP MARGIN: 5
BOTTOM MARGIN: 17
RIGHT MARGIN: 80
OK TO TRANSPORT?: OK
USE CURSOR CONTROL: YES
SCREEN TITLE: Expanded Profile
ALLOWABLE NUMBER OF ACTIONS: 2
ARRAY NAME: ^TMP("SDAMEP",$J)
EXIT CODE: D FNL^SDAMEP
HEADER CODE: D HDR^SDAMEP
HELP CODE: D HLP^SDAM5
ENTRY CODE: D INIT^SDAMEP
```

### 3.7.5 Application Code Examples

[Figure 11](#) is an example of List Manager application code:

**Figure 11: Sample List Manager Application Codes**

```
SDAM ;; - main code

EN      ; -- main entry point
        K XQORS,VALMEVL D EN^VALM("SDAM APPT MGT")
        Q
        ;
INIT    ; -- set up appt man vars and list man array and other vars
        K I,X,SDBEG,SDEND,SDB,XQORNOD,SDFN,SDCLN,DA,DR,DIE,DNM,DQ
        S DIR(0)="43,213",DIR("A")="Select Patient name or Clinic name"
        D ^DIR K DIR I $D(DIRUT) S VALMQUIT="" G INITQ
        S SDY=Y
        I SDY["DPT(" S SDAMTYP="P",SDFN+=SDY D INIT^SDAM1
        I SDY["SC(" S SDAMTYP="C",SDCLN+=SDY D INIT^SDAM3
INITQ   Q
        ;
HDR     ; -- screen header set up
        N X
        I SDAMTYP="P" D HDR^SDAM10
        I SDAMTYP="C" D HDR^SDAM3
        S X=$P(SDAMLIST,"^",2)
        S VALMHDR(2)=$$SETSTR^VALM1($$FDATE^VALM1(SDBEG)_ " thru
        "_$$FDATE^SSDEND),X,59,22)
        Q
        ;
FNL     ; -- what to do upon exiting list man
        K ^TMP("SDAM",$J),^TMP("SDAMIDX",$J),^TMP("VALMIDX",$J)
        K
SDAMCNT,SDFLDD,SDACNT,VALMHCNT,SDPRD,SDFN,SDCLN,SDAMLIST,SDT,SDAT
EG,SDEND,DFN,Y,SDAMTYP,SDY,X,SDCL,Y,SDDA,VALMY
        Q

HLP     ; -- help for list
        I $D(X),X'["??" D HLPS,PAUSE^VALM1 G HLPQ
        D CLEAR^VALM1
        F I=1:1 S SDX=$P($T(HELPTXT+I),";",3,99)
        Q:SDX="$END"
        D PAUSE^VALM1:SDX="$PAUSE" Q:'Y W !,$S(SDX["$PAUSE": "",1:SDX)
        W !,"Possible actions are the following:"
        D HLPS,PAUSE^VALM1 S VALMBCK="R"
HLPQ   K SDX,Y Q
        ;
HLPS    ; -- short help
        S X="?" D DISP^XQORM1 W ! Q
        ;
HELPTXT ; -- help text
        ;;Enter actions(s) by typing the name(s), or abbreviation(s).
        ;;
        ;;ACTION PRE-SELECTION:
```

```

;; Actions may be pre-selected by separating them with ";".
;; .
;; .
;; .

SDAMEP ;; - expand code
EN      ; Selection of appointment
        K ^TMP("SDAMEP", $J)
        S VALMBCK=""
        D SEL G ENQ:'$D(SDW)!(SDERR)
        W ! D WAIT^DICD,EN^VALM("SDAM APPT PROFILE")
        S VALMBCK="R"
ENQ     Q

VALMD   ;List Manager Sample Routine; APR 2, 1992
        ;
EN      ; -- option entry point
        K XQORS,VALMEVL
        D EN^VALM("VALM DEMO APPLICATION")
ENQ     Q
        ;
        ;
INIT    ; -- build array
        W ! S DIC("A")="Select Package:",DIC="^DIC(9.4,",DIC(0)="AEMQ" D
^DIC K DIC
        I Y<0 S VALMQUIT="" G INITQ
PKG     ; -- entry pt if package known
        N VALMX,VALMCNTI,VALMPRO,VALMIFN,X,VALMPRE,Z
        S VALMPKG=+Y
        D CLEAN^VALM10
        S
        (VALMCNTI,VALMCNT)=0,(VALMPRE,VALMPRO)=$P($G(^DIC(9.4,VALMPKG,0)),U,2)
        F S VALMPRO=$O(^ORD(101,"B",VALMPRO))
        Q:$E(VALMPRO,1,$L(VALMPRE))'=VALMPRE
        S VALMIFN=0 F S VALMIFN=$O(^ORD(101,"B",VALMPRO,VALMIFN))
Q:'VALMIFN I $D(^ORD(101,VALMIFN,0)) S VALMX=^(0) D
        .S VALMCNTI=VALMCNTI+1 W:(VALMCNTI#10)=0 "."
        .S X=$$SETFLD^VALM1(VALMCNTI,"","NUMBER")
        .S X=$$SETFLD^VALM1($P(VALMX,U),X,"NAME")
        .S X=$$SETFLD^VALM1($P(VALMX,U,2),X,"TEXT") K Z S
$P(Z,$E(VALMCNTI),240)=""
        .S VALMCNT=VALMCNT+1
        .D SET^VALM10(VALMCNT,$E(X_Z,1,240),VALMCNTI) ; set text
        .S ^TMP("VALMZIDX",$J,VALMCNTI)=VALMCNT_U_VALMIFN
        .D:'(VALMCNT#9) FLDCTRL^VALM10(VALMCNT) ; defaults for all fields
        .D FLDCTRL^VALM10(VALMCNT,"NUMBER") ; default for 1 field
        .D:'(VALMCNT#5) FLDCTRL^VALM10(VALMCNT,"NAME",IOUON,IOUOFF) ;
adhoc
        D NUL:'VALMCNT
INITQ   Q
        ;
HDR     ; -- demo header
        N VALMX
        S VALMX=$G(^DIC(9.4,VALMPKG,0)),X=" Package:"_ $P(VALMX,U)
        S VALMHDR(1)=$$SETSTR^VALM1("Prefix:"_ $P(VALMX,U,2),X,63,15)
        S VALMHDR(2)="Description: "_ $E($P(VALMX,U,3),1,65)

```

```

Q
;
NUL ; -- set null message
I 'VALMCNT D
.F X=" ", " No protocols to list." S VALMCNT=VALMCNT+1 D
SET^VALM10 (VALMCNT,X)
.S ^TMP ("VALMZIDX", $J, 1)=1, ^ (2)=2
Q
;
FNL ; -- clean up
K DIE, DIC, DR, DA, DE, DQ, VALMY, VALMPKG, ^TMP ("VALMZIDX", $J)
D CLEAN^VALM10
Q
;
EXP ; -- expand action
D FULL^VALM1
N VALMI, VALMAT, VALMY
D EN^VALM2 (XQORNOD (0), "O") S VALMI=0
F S VALMI=$O (VALMY (VALMI)) Q: 'VALMI D
.S VALMAT=$G (^TMP ("VALMZIDX", $J, VALMI))
.W !!, @VALMAR@ (+VALMAT, 0), !
.S DA=+$P (VALMAT, U, 2), DIC="^ORD (101, ", DR="0"
D EN^DIQ, PAUSE^VALM1
S VALMBCK="R"
Q
;
EDIT ; -- edit action
N VALMA, VALMP, VALMI, VALMAT, VALMY
D EN^VALM2 (XQORNOD (0), "O") S VALMI=0 ; allow the user to
"O"ptionally answer
F S VALMI=$O (VALMY (VALMI)) Q: 'VALMI D
.D SELECT^VALM10 (VALMI, 1) ; -- 'select' line
.S VALMAT=$G (^TMP ("VALMZIDX", $J, VALMI))
.W !!, @VALMAR@ (+VALMAT, 0)
.S DA=+$P (VALMAT, U, 2), VALMP=$G (^ORD (101, DA, 0)), DIE=19, DR="1" D
^DIE K DIE, DR
.S VALMA=$G (^ORD (101, DA, 0))
.I $P (VALMP, U, 2) '$P (VALMA, U, 2) D
UPD ($P (VALMA, U, 2), "TEXT", .VALMAT)
.D SELECT^VALM10 (VALMI, 0) ; -- 'de-select' line
S VALMBCK=$S (VALMCC: "", 1: "R")
Q
;
DESC ; -- display description action
N VALMI, VALMY, VALMAT
D EN^VALM2 (XQORNOD (0), "OS") S VALMI=0 ; select only a "S"ingle
protocols
F S VALMI=$O (VALMY (VALMI)) Q: 'VALMI D
.S VALMAT=+$P ($G (^TMP ("VALMZIDX", $J, VALMI)), U, 2)
.I '$D (^ORD (101, VALMAT, 1)) W !!, "No Description entered." D
AUSE^VALM1 Q
.D WP^VALM ("^ORD (101, "_VALMAT_", 1)", $P ($G (^ORD (101, VALMAT, 0)), U))
S VALMBCK="R"
Q
;
UPD (TEXT, FLD, VALMAT) ; -- update data for screen
D: VALMCC FLDCTRL^VALM10 (+VALMAT, .FLD, .IOINH, .IOINORM, 1)

```

```

D FLDTEXT^VALM10 (+VALMAT, .FLD, .TEXT)
Q
;
CHG ; -- change package action
K X I $D(XQORNOD(0)) S X=$P($P(XQORNOD(0),U,4), "=",2)
I X="" R !!,"Select Package: ",X:DTIME
S DIC="^DIC(9.4,",DIC(0)="EMQ" D ^DIC K DIC G CHG:X["?"
I Y<0 D G CHGQ
.W !!,*7,"Package has not been changed."
.W ! S DIR(0)="E" D ^DIR K DIR
.S VALMBCK=""
D PKG,HDR S VALMBCK="R" S VALMBG=1
CHGQ Q

```

[Figure 12](#) is an example of a stub routine created when adding a new List Template using the [Workbench](#).

**Figure 12: Sample Stub Routines When Adding New List Templates with the Workbench**

```

ZZDEMO ;; 24-JAN-1993
;; ;
EN ; -- main entry point for DOCUMENTATION DEMO
D EN^VALM("DOCUMENTATION DEMO")
Q
;
HDR ; -- header code
S VALMHDR(1)="This is a test header for DOCUMENTATION DEMO."
S VALMHDR(2)="This is the second line"
Q
;
INIT ; -- init variables and list array
F LINE=1:1:30 D SET^VALM10(LINE,LINE_" Line number" _LINE)
S VALMCNT=30
Q
;
HELP ; -- help code
S X="?" D DISP^XQORM1 W !!
Q
;
EXIT ; -- exit code
Q
;
EXPND ; -- expand code
Q
;

```

## 4 List Template Reference

### 4.1 Fields

#### 4.1.1 Demographics Fields

The following fields are demographic fields:

- [NAME \(#.01\)](#)
- [ENTITY NAME \(#.09\)](#)
- [SCREEN TITLE \(#.11\)](#)

##### 4.1.1.1 NAME (#.01)

Name of the List Template. The template should be namespaced.

##### 4.1.1.2 ENTITY NAME (#.09) [Optional]

This field contains the term that is displayed to the user that best describes the items in the list. This field is used by the select entry point ([EN^VALM2](#)).

##### 4.1.1.3 SCREEN TITLE (#.11) [Optional but Recommended] Screen Title Field

This field contains the text that is displayed/printed in the upper, left-hand corner of the screen display.

The screen title can be changed at run time by setting the [VALM\("TITLE"\)](#) variable during [ENTRY CODE](#) or action processing. If you have one basic List Template definition that could be used for more than one application, then setting the [VALM\("TITLE"\)](#) variable allows you to re-use the template but change the title as it appears to the user.

### 4.1.2 Protocol Information Fields

The following fields are protocol information fields:

- [TYPE OF LIST \(#.02\)](#)
- [PROTOCOL MENU \(#.1\)](#)
- [PRINT PROTOCOL \(#1.01\)](#)
- [HIDDEN MENU \(#1.02\)](#)

#### 4.1.2.1 TYPE OF LIST (#.02)

Indicates the type of List Template. The type determines what actions are presented to the user:

- **PROTOCOL**—Uses the menu protocol specified in the PROTOCOL MENU field.
- **DISPLAY**—Uses the standard **VALM DISPLAY PROTOCOL** supplied by the List Manager.

#### 4.1.2.2 PROTOCOL MENU (#.1)

This field contains the name of the protocol menu that is used by the List Manager if the [TYPE OF LIST \(#.02\)](#) field is “**PROTOCOL**”. This field is *not* used for “**DISPLAY**” types.

#### 4.1.2.3 PRINT PROTOCOL (#1.01) [Optional]

This field contains the name of the protocol that is called when the user selects the generic **Print List** action. Normally, this field is blank, and the generic printing action is sufficient.

#### 4.1.2.4 HIDDEN MENU (#1.02) [Optional but *Recommended*]

This field contains the name of the protocol menu that the List Manager uses for the “hidden” actions available to the user. Normally, the application enters the [VALM HIDDEN ACTIONS](#) menu in this field. However, there may be applications that would require a different set of “hidden” actions.

If the List Template has a “hidden” menu defined the List Manager automatically displays help for the hidden menu when the user enters “??”.

### 4.1.3 List Region Fields

The following fields are list region fields:

- [TOP MARGIN \(#.05\)](#)
- [BOTTOM MARGIN \(#.06\)](#)
- [RIGHT MARGIN \(#.04\)](#)

#### 4.1.3.1 TOP MARGIN (#.05)

This field contains the number of the top row of the scrolling region where the list is displayed.

#### 4.1.3.2 BOTTOM MARGIN (#.06)

This field contains the number of the bottom row of the scrolling region where the list is displayed.

#### 4.1.3.3 RIGHT MARGIN (#.04) [Optional]

This field indicates the maximum number of characters a row can contain. If this parameter is *not* set, **80** is used as the default. Range is **80** to **240 characters**.

## 4.1.4 Other Fields

The following are other fields:

- [OK TO TRANSPORT ? \(#.07\)](#)
- [USE CURSOR CONTROL \(#.08\)](#)
- [ALLOWABLE NUMBER OF ACTIONS \(#.12\)](#)
- [DATE RANGE LIMIT \(#.13\)](#)
- [AUTOMATIC DEFAULTS \(#.14\)](#)

### 4.1.4.1 OK TO TRANSPORT ? (#.07)

This field indicates to the transport utility if this List Template should be distributed.



**NOTE:** This field is obsolete now that KIDS is used to transport List Manager applications.

**REF:** For more information on KIDS, see the *Kernel 8.0 and Kernel Toolkit 7.3 Systems Management Guide* and *Kernel 8.0 and Kernel Toolkit 7.3 Developer's Guide*.

### 4.1.4.2 USE CURSOR CONTROL (#.08)

This field indicates whether the cursor positioning and character enhancement capabilities of the device should be used. If set to “NO”, then lists are presented in Scrolling Mode.

### 4.1.4.3 ALLOWABLE NUMBER OF ACTIONS (#.12)

This field indicates the number of actions a user can select at one time.

For example, if this parameter is set to **1**, then the user can only enter one action:

- Allowed: Select Action: **NX**
- *Not* Allowed: Select Action: **NX,EP**

If this parameter is *not* entered, then the system defaults to **1**.

### 4.1.4.4 DATE RANGE LIMIT (#.13) [Optional]

This field contains the maximum number of days that can be specified by the user while entering a date range. This parameter is only used if the applications calls the List Manager's date range entry point ([RANGE^VALM1](#)).



#### 4.1.4.5 AUTOMATIC DEFAULTS (#.14) [optional]

This field indicates whether List Manager should always supply a default action at the “Select” prompt for “**PROTOCOL**” type List Templates.

If set to “**NO**”, a default is *not* provided automatically. It is your responsibility to indicate a default, if desired. This default can be indicated by setting the [XQORM\(“B”\)](#) variable as part of the **PROTOCOL** menu’s [HEADER CODE](#). For example:

```
D SHOW^VALM S XQORM("B")="Your action")
```

This parameter is only valid for “**PROTOCOL**” type List Templates.

If the parameter is set to “**YES**” or is **blank**, a default is provided by List Manager. If the current screen contains the last line in the list, then the default will be “**Quit**”; otherwise, it will be “**Next Screen**”. However, as discussed above, you can override this default by setting the [XQORM\(“B”\)](#) variable.

#### 4.1.5 MUMPS Code Related Fields

The following are MUMPS code related fields:

- [HEADER CODE \(#100\)](#)
- [ENTRY CODE \(#106\)](#)
- [EXIT CODE \(#105\)](#)
- [EXPAND CODE \(#102\)](#)
- [HELP CODE \(#103\)](#)
- [ARRAY NAME \(#107\)](#)

##### 4.1.5.1 HEADER CODE (#100)

This MUMPS field contains the code that the List Manager executes to create the application-specific screen header array. This header *must* be stored in the [VALMHDR\(\)](#) variable.

The subscripting for [VALMHDR\(\)](#) is a simple integer number. For example:

```
S VALMHDR(1) = "This is the 1st line of the header"  
S VALMHDR(2) = "This is the 2nd line of the header"
```

During action processing, if the header needs to be changed, you can **KILL** [VALMHDR](#) and then **SET** [VALMBCK](#)=“**R**”. This causes List Manager to automatically invoke this **HEADER CODE**, as part of the re-display of the screen.

#### 4.1.5.2 ENTRY CODE (#106)

This field contains MUMPS code that is executed when the List Manager is called. This code is usually used by the application to initialize variables. Any application-specific variables should also be set up here.

List Manager variables to be initialized are:

- [VALMCNT](#) [Required]—The number of lines in the list.
- [VALMBG](#) [Optional]—The number of the line you want the List Manager to start displaying from a line other than **1**. If *not* defined, it will be set to **1** by List Manager.
- [VALMQUIT](#) [Optional]—If during the building of the array, the software determines that the List Manager application *cannot* continue, this variable should be set. Setting this variable causes the List Manager to quit the current List Manager application.

The array specified in the [ARRAY NAME](#) field is also set up at this time. This array contains the list of items to display. The subscripting of the array should conform to VA FileMan word-processing format.

For example: If [ARRAY NAME](#) equals `^TMP("SDTEST",$J)`, then the list would be stored as follows:

```
^TMP("SDTEST",$J,1,0) = " 1 Valmuser,One "  
^TMP("SDTEST",$J,2,0) = " 2/2/93@0800am"
```

If you plan to use the entry selection call, [EN^VALM2](#), then the following *must* also be set:

```
^TMP("SDTEST",$J,"IDX",<line #>,<entry #>) = ""
```

The “**line #**” corresponds to the **1** and **2** shown in the above example. The “**entry #**” corresponds to an entry in your application. In the example, the **two lines** each correspond to appointment entry number. So, the “**IDX**” nodes would be set up in the following manner:

```
^TMP("SDTEST",$J,"IDX",1,1)=""  
^TMP("SDTEST",$J,"IDX",2,1)=""
```



**REF:** For more information on that List Template field, see the [ARRAY NAME](#) field.

#### 4.1.5.3 EXIT CODE (#105) [optional but recommended]

This field contains MUMPS logic that the List Manager executes when the user exits the list. This should be used to clean up variables and any other exit processing the application needs to perform.

#### 4.1.5.4 EXPAND CODE (#102) [optional]

This field contains the MUMPS code that displays a detailed inquiry-type report/screen for a specific entry in the list. If this field is filled in, then the standard “**DISPLAY**” protocol has an “**EXPANDED**” action.

The standard **VALM EXPAND** protocol uses this field to expand an entry. If the [TYPE OF LIST](#) is **PROTOCOL**, then add the **VALM EXPAND** protocol to your custom protocol menu and enter the code in this EXPAND CODE (#102) field.

A possible method for expand is to create another List Template that is a **DISPLAY** type. You need only build the display array and set this EXPAND CODE (#102) field to be another call to the List Manager, passing in the display template name.

#### 4.1.5.5 HELP CODE (#103) [optional]

This field contains the MUMPS code for custom application help. This code is executed when the user types a “?” at the “Select Action: ” prompt.

This field is optional. If this field is left blank, the normal help given by the **XQOR\*** driver takes effect.

If the List Template has a “hidden” menu defined the List Manager automatically displays help for the hidden menu when the user enters “??”.

#### 4.1.5.6 ARRAY NAME (#107) [optional]

This field contains the name of the array that holds the list of items to be displayed. The code specified in the [ENTRY CODE](#) (#106) field *must* create this array initially.



**NOTE:** The array name *must* be preceded by a space character. This is needed to allow global specification. VA FileMan does *not* allow “^” as the first character. The array can be either a local or global variable.

The array needs to follow the format used in word-processing fields (e.g., `^TMP(“SDAM”, $J, line #, 0)=string`).

Finally, you do *not* have to indicate the array in which the list will be located. By making calls to [SET^VALM10](#), you can have the List Manager decide where to store the list array. If you need to reference lines in the array, the use of the `@VALMAR@(<line #>, 0)` syntax is supported. This feature is ideal for a short list of items (e.g., <10 items).

## 4.1.6 Caption Line Information Fields

The following are caption line information fields:

- [CAPTION LINE COLUMNS \(#200\)](#)
- [ITEM NAME \(#.01\)](#)
- [COLUMN \(#.02\)](#)
- [WIDTH \(#.03\)](#)
- [DISPLAY TEXT \(#.04\)](#)
- [DEFAULT VIDEO ATTRIBUTES \(#.05\)](#)
- [SCROLL LOCK \(#.06\)](#)

### 4.1.6.1 CAPTION LINE COLUMNS (#200) [Optional]

This Multiple field contains column definitions for the data displayed in the list. Adding entries to this Multiple is optional. The column parameters are used when the List Manager writes the line indicating the top of the list's scrolling region.

### 4.1.6.2 ITEM NAME (#.01)

This field contains the reference name of the column. The [DISPLAY TEXT](#) field contains the text that is used when the caption line is written. The text in this field is used when the application refers to this column during programming.

### 4.1.6.3 COLUMN (#.02)

This field contains the column number where the data/caption starts.

### 4.1.6.4 WIDTH (#.03)

This field contains the number of characters this field uses.

### 4.1.6.5 DISPLAY TEXT (#.04) [optional]

This field contains the text that appears on the caption line for this column/field. If the text is longer than the **WIDTH** parameter, it is truncated to the **WIDTH** specification when written as part of the caption line. This field is optional and can be left **blank**.

#### 4.1.6.6 DEFAULT VIDEO ATTRIBUTES (#.05) [optional]

This parameter allows you to indicate the default video attributes that should be applied when program calls are made to the [FLDCTRL^VALM10](#) entry point.

The following is the list of attributes and abbreviations used for this parameter:

- **H**—Highlight
- **R**—Reverse video
- **U**—Underline
- **B**—Blinking

#### 4.1.6.7 SCROLL LOCK (#.06) [Optional]

If you want to lock one or more columns into place as the user scrolls horizontally through the list, you can place a “Scroll Lock” on the right most column field that should be locked in place on the screen. Only one column can have this “Scroll Lock” parameter set to “YES”. If you attempt to set more than one, the system does *not* allow it and issues a warning.

If this parameter is set to “YES”, this caption field and any other caption field, with a **COLUMN** parameter set to less than this current caption fields, the List Manage always displays it.

This parameter does *not* need to be filled in for List Templates with a [RIGHT MARGIN](#) of **80**. For those templates with a [RIGHT MARGIN](#) of over **80**, this field also does *not* need to be entered. However, the use of this field allows you to indicate the list’s identification fields for user readability.

Only one caption field can have this parameter set to “YES”.

The local array [VALMDDFQ](#) is available to you at run time. This array is subscripted by the column field’s name and contains information described above:

```
VALMDDF(<column name>)=<column name> ^ <column> ^ <width> ^ <caption> ^  
<video> ^ <scroll lock>
```

## 5 Application Programming Interfaces (APIs)

### 5.1 List Manager Variables

This section lists all of the variables within List Manager that you can either set or refer to in your List Manager application code.

Table 10: List Manager Variables

Variable	Description
<b>VALM(TITLE)</b>	The screen title can be changed at run time by setting this variable, during ENTRY CODE or action processing. If you are one basic List Template definition that could be used for more than one application, then setting <b>VALM("TITLE")</b> allows you to re-use the template but change the title as it appears to the user.
<b>VALMBCK</b>	When returning to the List Manager from a protocol action, you should set the <b>VALMBCK</b> variable. This tells List Manager what to do when returning from an action. If <i>not</i> defined after an action, List Manager acts as if it was set to "Q": <ul style="list-style-type: none"><li>• <b>R</b>—Refresh screen.</li><li>• <b>NULL</b>—Clear bottom portion of screen and prompt for action.</li><li>• <b>Q</b>—Exit (<b>Quit</b>) List Manager.</li></ul>
<b>VALMBG</b>	An optional variable you can set in the <a href="#">INIT</a> code that sets up your list. This tells List Manager what line in your list to start displaying the list in (default is line 1).  In action protocols, you can also refer to the value of this variable to find the number of the first list line currently displayed on the user's screen.
<b>VALMCC</b>	Always available to indicate the user's screen mode: <ul style="list-style-type: none"><li>• <b>1</b>—Screen Mode.</li><li>• <b>0</b>—Scrolling Mode.</li></ul>
<b>VALMCNT</b>	The number of the lines in the list. In the <a href="#">INIT</a> code that sets up the list, you <i>must</i> set <b>VALMCNT</b> equal to the number of lines in your list.
<b>VALMDDF()</b>	This array is available at runtime. It is subscripted by caption field name, so there is one node per caption field in your List Template. Each node contains the following ^-pieces: <ul style="list-style-type: none"><li>• Caption field name</li><li>• Column</li><li>• Width</li></ul>

Variable	Description
	<ul style="list-style-type: none"> <li>• Caption</li> <li>• Video (if defined)</li> <li>• Scroll Lock (if defined)</li> </ul> <p>For example:</p> <pre>VALMDDF("INIT")=INIT^37^5^Init. VALMDDF("NAME")=NAME^1^35^ Name^</pre>
<b>VALMHDR()</b>	<p>The header is stored in <b>VALMHDR()</b>. The subscripting for <b>VALMHDR()</b> is a simple integer number. For example:</p> <pre>S VALMHDR(1) = "1st line of header" S VALMHDR(2) = "2nd line of header"</pre> <p>During action processing, if the header needs to be changed, you can <b>KILL VALMHDR</b> and then <b>SET VALMBCK="R"</b>. This causes List Manager to automatically invoke what is called by the <a href="#">HEADER CODE</a> (#100) field as part of the re-display of the screen.</p>
<b>VALMLST</b>	<p>In action protocols, you can refer to the value of this variable to find the number of the last list line currently displayed on the user's screen.</p>
<b>VALMQUIT</b>	<p>If in the <a href="#">INIT</a> code, while building a list, you decide that List Manager should <i>not</i> continue, set the <b>VALMQUIT</b> variable to tell List Manager to <b>Quit</b>.</p>
<b>VALMSG</b>	<p>To display a custom message in the message window after completing an action, set this variable with the desired text (up to <b>50 characters</b>).</p>
<b>@VALMAR@(#,0)</b>	<p>If you built your array using <a href="#">SET^VALM10</a>, you can use the <b>@VALMAR@(line#,0)</b> syntax to reference text lines in the array.</p>
<b>@VALMAR@("IDX")</b>	<p>Location of entry index when you set up an array using <a href="#">SET^VALM10</a>, and pass index entries with each line. The relationship of the list line to the indexed value stored in the global referenced by <b>@VALMAR@("IDX")</b> is:</p> <pre>^... "IDX", line_num, index_num) = ""</pre> <p>To retrieve the entry number indexed for line <b>54</b> in the array, you could use:</p>

Variable	Description
	<code>S Y=\$O (@VALMAR@ ("IDX" , 56 , "" )</code>
<b>XQORM("B")</b>	List Manager automatically provides a default action of <b>"Next Screen"</b> or <b>"Quit"</b> . However, you can override this default action by setting <b>XQORM("B")</b> as part of the ENTRY ACTION code for a <b>PROTOCOL</b> menu. Set it to the text of the menu item you would like to be the new default.

## 5.2 Kernel Video Variables

[Table 11](#) lists the standard video control variables that you can use in List Manager:

Table 11: Kernel Video Variables

Attribute	Variable
Normal Video	IOINORM
High Intensity	IOINHI
Reverse Video On	IORVON
Reverse Video Off	IORVOFF
Underline On	IOUON
Underline Off	IOUOFF
Blink On	IOBON
Blink Off	IOBOFF

These variables can be used in **ON** and **OFF** parameters outlined in a number of List Manager calls. If other video attributes are needed, you need to make the appropriate call to Kernel's ENDR^%ZISS entry point to set up variables for those attributes.

The variables listed in [Table 11](#) should always remain defined and should not be **KILLED** by application code.

Finally, you can specify more than one video attribute in a single call by concatenating the variables. For example, **"D CNTRL^VALM10(1,20,30,IOINHI\_IUON,IOINORM)"** would highlight and underline **30 characters** starting at **Column 20**.



## 5.3 List Manager Generic Action Protocols

[Table 12](#) lists the generic actions in the PROTOCOL (#101) file that you can use in your List Manager application.



**NOTE:** These generic actions are all attached to the [VALM HIDDEN ACTIONS](#) protocol. This is so that you can set your list's **HIDDEN MENU** protocol to [VALM HIDDEN ACTIONS](#) and have your list automatically make all of these actions available to your list users.

**Table 12: List Manager Generic Action Protocols**

Protocol Name	Protocol Description
<b>VALM DOWN A LINE</b>	Move down a line.
<b>VALM UP ONE LINE</b>	Move up a line.
<b>VALM FIRST SCREEN</b>	This action will display the first screen.
<b>VALM LAST SCREEN</b>	The action will display the last items.
<b>VALM NEXT SCREEN</b>	This action will allow the user to view the next screen of entries, if any exist.
<b>VALM PREVIOUS SCREEN</b>	This action will allow the user to view the previous screen of entries, if any exist.
<b>VALM PRINT LIST</b>	This action allows the user to print the entire list of entries currently being displayed.
<b>VALM PRINT SCREEN</b>	This action allows the user to print the current List Manager display screen. The header and the current portion of the list are printed.
<b>VALM REFRESH</b>	This actions allows the user to re-display the current screen.
<b>VALM SEARCH LIST</b>	Finds text in list of entries.
<b>VALM TURN ON/OFF MENUS</b>	This toggles the menu of actions to be displayed/not displayed automatically.
<b>VALM GOTO PAGE</b>	This protocol will allow the user to move to any page in the list.
<b>VALM RIGHT</b>	This protocol allows the user to move the screen to the <b>right</b> if the List Template is set up for a width of more than <b>80 characters</b> .
<b>VALM LEFT</b>	This protocol allows the user to move the screen to the <b>left</b> if the List Template is set up for a width of more than <b>80 characters</b> .
<b>VALM QUIT</b>	This protocol can be used as a generic “ <b>Quit</b> ” action.

Protocol Name	Protocol Description
<b>VALM HIDDEN ACTIONS</b>	<p>This menu protocol contains all the above action protocols. You usually would specify this protocol as the “Hidden Menu” protocol in the List Template set up.</p> <p>The Workbench automatically designates this protocol as the “Hidden Menu” protocol when a List Template is initially created.</p>

## 5.4 General APIs

### 5.4.1 EN^VALM(): Load a ListMan Template/Application

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10118
<b>Description:</b>	This API invokes ListMan to load a List Manager template/application.
<b>Format:</b>	EN^VALM(template_name)
<b>Input Parameters:</b>	<b>template_name:</b> (required) This parameter contains the name of a ListMan template/application to load.
<b>Output:</b>	None.

### 5.4.2 SHOW^VALM: Display Menu to User

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10118
<b>Description:</b>	Call the SHOW^VALM API in the <a href="#">HEADER CODE (#100)</a> field of all of your menu protocols. This displays the menu to the user.
<b>Format:</b>	SHOW^VALM
<b>Input Parameters:</b>	None.
<b>Output:</b>	None.

### 5.4.3 PAUSE^VALM1: Pause the Screen

**Reference Type:** Supported  
**Category:** ListMan  
**ICR #:** 10116  
**Description:** This API pauses the screen. The call uses a ^DIR API with **DIR(0)** set to “E” for end of page. The prompt looks like:

```
Press RETURN to continue or '^' to exit:
```

**Format:** PAUSE^VALM1  
**Input Parameters:** None.  
**Output:** None.

### 5.4.4 RANGE^VALM1: Change Date Range

**Reference Type:** Supported  
**Category:** ListMan  
**ICR #:** 10116  
**Description:** This API lets the user change a date range.  
**Format:** RANGE^VALM1  
**Input Variables:** **DATE RANGE LIMIT Field:** (required) Value as stored in the LIST TEMPLATE (#409.61) file.  
**VALMB:** (optional) Default beginning date.  
**Output Variables:** **VALMBEG:** Beginning date in VA FileMan (FM) date format.  
**VALMEND:** Ending date in FM date format.

### 5.4.5 EN^VALM2(): Generic Selector

**Reference Type:** Supported  
**Category:** ListMan  
**ICR #:** 10119  
**Description:** This API is a generic selector that can be used within an action call. In order to use this API, the List Manager [ENTRY CODE \(#106\)](#) field *must* be set up the @VALMAR@("IDX") index array. This is done by

setting up the list array line-by-line with the [SET^VALM10](#) entry point and associating an IEN (Internal Entry Number) with each line created.

**Format:** EN^VALM2 (valmnode, options)

**Input Parameters:** **valmnode:** (required) String in **XQORNOD(0)** four-piece format:

1. IEN of selected item.
2. IEN of menu.
3. Menu text.
4. Text user entered to select item.

For example:

```
S VALMNOD="3^1312^Misc. Consult^3"
```

**options:** (required) Selection option flags:

- **S**—User *must* select **one and only one** number as an entry. The user *cannot* press the **Enter** key *without* entering a number; however, the user can enter a caret (^) to **QUIT** the select and exit the API.
- **SO**—User can select one number or no number, since the selection is optional (“**O**”). For example: if the option is “**SO**”, then the user is *not* required to select one number. The user can press the **Enter** key *without* a number or enter a caret (^) to **QUIT** the select and exit the API.
- **L (default)**—User can select numbers in a range, comma-listed, or a combination of both. For example: **1-10,15,20-30**. The user *cannot* press the **Enter** key *without* entering a number range or list; however, the user can enter a caret (^) to **QUIT** the select and exit the API.
- **LO**—User can select a number range, number list, combination, or no number, since the selection is optional (“**O**”). For example: if the option is “**LO**”, then the user is *not* required to select any numbers. The user can press the **Enter** key *without* a number or enter a caret (^) to **QUIT** the select and exit the API.

**Output:** VALMY(): Array with selected entries as subscripts.

### 5.4.5.1 Examples

These examples use a modified test version of the EN^VALM2 API to only test the **option** input parameter choices.

#### 5.4.5.1.1 Option “S” Example

**Figure 13: EN^ZZVALM2T API—Option “S” Example: System Prompts and User Entries**

```
>D EN^ZZVALM2T("S")  
Select Item: (1-200): <Enter>  
This is a required response. Enter '^' to exit  
With the “S” parameter, the user cannot simply press the Enter key without entering a single number.  
Select Item: (1-200): 1-5  
This response must be a number.  
With the “S” parameter, the user can only enter a single number (no range of entries).  
Select Item: (1-200): 1,5  
This response must be a number.  
With the “S” parameter, the user can only enter a single number (no list of entries).  
Select Item: (1-200): 5  
Selected Item(s) Output:  
VALMY(5)=""
```

### 5.4.5.1.2 Option “SO” Quit Examples

Figure 14: EN^ZZVALM2T API—Option “SO” Example: System Prompts and User Entries

```
>D EN^ZZVALM2T("SO")
Select Item: (1-200): <Enter>
Selected Item(s) Output:

```

With the “SO” parameter, the user can simply press the Enter key and Quit the API without error, which is unlike with the “S” only parameter (Figure 13).

```
>
>D EN^ZZVALM2T("SO")
Select Item: (1-200): ^
Selected Item(s) Output:
>
```

### 5.4.5.1.3 Option “L” Example

Figure 15: EN^ZZVALM2T API—Option “L” Example: System Prompts and User Entries

```
>D EN^ZZVALM2T("L")
Select Item: (1-200): <Enter>
This is a required response. Enter '^' to exit

```

With the “L” parameter, the user *cannot* simply press the Enter key *without* entering a range or list of numbers.

```
Select Item(s): (1-200): 1-5,150,199-200
Selected Item(s) Output:
VALMY (1)=""
VALMY (2)=""
VALMY (3)=""
VALMY (4)=""
VALMY (5)=""
VALMY (150)=""
VALMY (199)=""
VALMY (200)=""
```

#### 5.4.5.1.4 Option “LO” Quit Examples

Figure 16: EN^ZZVALM2T API—Option “LO” Quit Example: System Prompts and User Entries

```
>D EN^ZZVALM2T ("LO")  
Select Item(s): (1-200): <Enter>  
Selected Item(s) Output:  
  
>  
  
>D EN^ZZVALM2T ("LO")  
Select Item(s): (1-200): ^  
Selected Item(s) Output:  
  
>
```

With the “LO” parameter, the user can simply press the Enter key and Quit the API without error, which is unlike with the “L” only parameter ([Figure 15](#)).

## 5.5 List Line Text APIs

### 5.5.1 FLDUPD^VALM1(): Update Caption Field

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10116

**Description:** This API updates a specific caption field of a specified list line on the display screen. The field name *must* match a field defined in the [CAPTION LINE COLUMNS \(#200\)](#) Multiple field in the LIST TEMPLATE (#409.61) file.

**Format:** FLDUPD^VALM1 (text, field, entry)

**Input Parameters:**

- text:** (required) Text to insert.
- field:** (required) Caption field name.
- entry:** (required) Line number of line in the list.

**Output:** None.

### 5.5.2 \$\$SETFLD^VALM1(): Insert Text in a String

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10116

**Description:** This extrinsic function inserts text in a string based on the column position of Caption fields stored in the current List Template. Typically, this is used when you are building the lines to place in your list's array. It helps you easily place text strings in your list lines based on the position of caption headers in the active List Template. For example, if your List Template has **three** captions, you would typically make **three** calls to this function to construct your line – one call each to insert the text corresponding to each caption header.

**Format:** S X=\$\$SETFLD^VALM1 (text, string, field)

**Input Parameters:**

- text:** (required) Text to insert.
- string:** (required) String into which text is to be inserted.
- field:** (required) Caption field name in list template whose column position determines the position in string to insert text.

**Output:** **return value:** Returns string with text inserted.

### 5.5.3 \$\$SETSTR^VALM1(): Set Up String for Display

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10116

**Description:** This extrinsic function sets up a string for display. Once the string has been set up for display, you would typically set it in the [ARRAY NAME \(#107\)](#) field specified in the List Template. For example:

```
S ^TMP ("SDAM" , $J , SDLN) =X
```

**Format:** S X=\$\$SETSTR^VALM1 (text, string, column, length)



**Input Parameters:**

- text:** (required) Text to insert.
- string:** (required) String into which text is to be inserted.
- column:** (required) Column position to insert text.
- length:** (required) Number of characters to clear.

**Output:** **return value:** Returns string with text inserted.

### 5.5.3.1 Examples

```
>S X=$$SETSTR^VALM1("This","",10,4) W !,X
This
```

```
>S X=$$SETSTR^VALM1("is",X,20,2) W !,X
This is
```

```
>S X=$$SETSTR^VALM1("an",X,30,2) W !,X
This is an
```

```
>S X=$$SETSTR^VALM1("example.",X,40,8) W !,X
This is an example.
```

### 5.5.4 FLDTEXT^VALM10(): Inserts Text in a Column

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10117

**Description:** This API inserts text at the column where the specific field starts in a LINE in the list array.

The FIELD name *must* match a field defined in the [CAPTION LINE COLUMNS \(#200\)](#) Multiple field of the LIST TEMPLATE (#409.61) file.

**Format:** FLDTEXT^VALM10 (line, field, text)

**Input Parameters:**

- line:** (required) Line number in list array into which you insert text.
- field:** (required) Name of a caption field in the List Template. Text is inserted at the column position corresponding to the specified caption field.
- text:** (required) Text to insert.

**Output:** None.

### 5.5.5 SET^VALM10(): Construct Initial List Array

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10117

**Description:** This API constructs the initial list array before displaying the list to the user. It adds one line at a time to the list array.



**NOTE:** If the List Template does *not* define an [ARRAY NAME](#) (#107), then you *must* use this call to build lines in the list array.

**Format:** SET^VALM10(line,string[,ien])

**Input Parameters:**

- line:** (required) Line number in the array to set line. The list array, when completed, *must* start at line number **1**, and there *cannot* be any gaps in the line numbering sequence.
- string:** (required) Text of the line.
- ien:** (optional) Entry number to associate with the line. If passed, then the line is also indexed for use by the [EN^VALM2](#) generic list selection call.

**Output:** None.

## 5.6 List Line Video APIs

### 5.6.1 CNTRL^VALM10(): Set Video Attributes



**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10117

**Description:** This API sets the video attributes for a line in the current list.

**Format:** CNTRL^VALM10(line,column,width,on,off[,save])

<b>Input Parameters:</b>	<b>line:</b>	(required) Line number of line for which you want to set video attributes.
	<b>column:</b>	(required) Screen column position where code should be invoked.
	<b>width:</b>	(required) How many screen columns for which the code should be in effect.
	<b>on:</b>	(required) Beginning control sequence.
		 <b>REF:</b> For a set of variables you can use with this input parameter, see the “ <a href="#">Kernel Video Variables</a> ” section.
	<b>off:</b>	(required) Ending control sequence.
		 <b>REF:</b> For a set of variables you can use with this input parameter, see the “ <a href="#">Kernel Video Variables</a> ” section.
	<b>save:</b>	(optional) Possible values: <ul style="list-style-type: none"> <li>• 1—Save control sequence for later use (to be restored with <a href="#">RESTORE^VALM10</a>).</li> <li>• 0</li> </ul>

**Output:** None.

## 5.6.2 FLDCTRL^VALM10(): Activate Video Control Sequences



**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10117

**Description:** This API activates the appropriate video control sequences for a LINE in the list array based on the [DEFAULT VIDEO ATTRIBUTES \(#.05\)](#) field in the **CAPTION LINE** definition for the template.

**Format:** FLDCTRL^VALM10 (line[, field] [, on] [, off] [, save])

<b>Input Parameters:</b>	<b>line:</b>	(required) Line number in the list array for which you want to activate video attributes.
	<b>field:</b>	(optional) If passed, only the video attributes defined for text that falls within the specified caption field are activated. It <i>must</i> be the name of a caption field in the List Template.
	<b>on:</b>	(optional) If defined, then the code in this variable is used at the starting column position to turn on video attributes instead of the default.
		 <b>REF:</b> For a set of variables you can use with this input parameter, see the “ <a href="#">Kernel Video Variables</a> ” section.
	<b>off:</b>	(optional) If defined, then the code in this variable is used at the ending column position to turn off video attributes instead of the default.
		 <b>REF:</b> For a set of variables you can use with this input parameter, see the “ <a href="#">Kernel Video Variables</a> ” section.
	<b>save:</b>	(optional) Possible values: <ul style="list-style-type: none"> <li>• <b>1</b>—Save control sequence for later use (to be restored with <a href="#">RESTORE^VALM10</a>).</li> <li>• <b>0</b></li> </ul>
<b>Output:</b>		None.

### 5.6.3 RESTORE^VALM10(): Restores Video Attributes

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10117
<b>Description:</b>	This API restores the video attributes that have been saved for the indicated line. This subroutine does <i>not</i> re-write the line to the screen; use <a href="#">WRITE^VALM10</a> after restoring video attributes to actually <b>WRITE</b> the line.
<b>Format:</b>	RESTORE^VALM10 (line)
<b>Input Parameters:</b>	<b>line:</b> (required) Line number for which you want to restore video attributes.

**Output:** None.

#### 5.6.4 SAVE^VALM10(): Save Current Video Attributes

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10117

**Description:** This API saves the current video attributes for the indicated line.

**Format:** SAVE^VALM10 (line)

**Input Parameters:** **line:** (required) Line number for which you want to save the current video attributes.

**Output:** None.

#### 5.6.5 SELECT^VALM10(): Highlights/Unhighlights Line in List

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10117

**Description:** This API highlights (selects)/unhighlights (deselects) a line in the list. The API sets up or deletes the proper video controls and then “WRITES” the line to the screen.

**Format:** SELECT^VALM10 (line, mode)

**Input Parameters:** **line:** (required) Line number of line to highlight/unhighlight (select). The line *must* be one that is currently displayed on the screen.

**mode:** (required) Possible values:

- 1—Highlight (select).
- 0—Unhighlight (deselect) and restore to original state.

**Output:** None.

## 5.6.6 WRITE^VALM10(): Re-Write Line to Screen

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10117
<b>Description:</b>	This API re-writes a line to the screen.
<b>Format:</b>	WRITE^VALM10 (line)
<b>Input Parameters:</b>	<b>line:</b> (required) Number of the line in the list to re-write to the screen.
<b>Output:</b>	None.

## 5.7 Screen Control APIs

### 5.7.1 CHGCAP^VALM(): Changes Label on Caption Header

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10118
<b>Description:</b>	This API changes a label on a caption header for a field defined in the <a href="#">CAPTION LINE COLUMNS</a> (#200) Multiple field in the LIST TEMPLATE (#409.61) file.
<b>Format:</b>	CHGCAP^VALM(field, label)
<b>Input Parameters:</b>	<b>field:</b> (required) Caption field name. <b>label:</b> (required) Text for caption header.
<b>Output:</b>	None.

## 5.7.2 CLEAR^VALM1: Clean Up Screen after Error Occurs

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10116
<b>Description:</b>	Use this API in Programmer Mode during development to clean up the screen after an error occurs. It changes the screen from Screen Mode to the full scrolling region and clears the screen. Also, it turns off the following: <ul style="list-style-type: none"><li>• Underline</li><li>• High Intensity</li><li>• Reverse Video</li><li>• Blinking</li></ul>
<b>Format:</b>	CLEAR^VALM1
<b>Input Parameters:</b>	None.
<b>Output:</b>	None.

## 5.7.3 FULL^VALM1: Sets Screen to Full Scrolling Region

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10116
<b>Description:</b>	This API sets the screen to the full scrolling region.
<b>Format:</b>	FULL^VALM1
<b>Input Parameters:</b>	None.
<b>Output:</b>	None.

## 5.7.4 INSTR^VALM1(): Inserts Text on Display Screen

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10116
<b>Description:</b>	This API inserts text on the display screen at the row and column specified.
<b>Format:</b>	INSTR^VALM1 (string, column, row[, length] [, erase])
<b>Input Parameters:</b>	<b>string:</b> (required) String to insert.

**column:** (required) **X** coordinate.  
**row:** (required) **Y** coordinate.  
**length:** (optional) Number of characters to clear.  
**erase:** (optional) If a value (any value) is passed for this parameter, the screen cells from (**row,col**) to (**row,col+length**) are erased before the string is displayed.

**Output:** None.

### 5.7.5 RE^VALM4: Re-Displays List Header and List Areas

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10120

**Description:** This API re-displays the list header and list areas for the active list application. It is often used to display the results of a change an action has caused before passing control back to the List Manager.

Normally, you set [VALMBCK](#)="R" and then returns control to the List Manager.

**Format:** RE^VALM4

**Input Parameters:** None.

**Output:** None.

### 5.7.6 CLEAN^VALM10: Kills Data and Video Control Arrays

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10117

**Description:** This API **KILLS** the data and video control arrays associated with the active list. This call is commonly used to **KILL** the array related data before re-building the array.

**Format:** CLEAN^VALM10

**Input Parameters:** None.


**Output:** None.



### 5.7.7 KILL^VALM10(): Deletes Video Attributes

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10117
<b>Description:</b>	This API deletes video attributes. If the <b>line</b> input parameter is defined, then only the attributes for that line are deleted.
<b>Format:</b>	KILL^VALM10 ([line])
<b>Input Parameters:</b>	<b>line:</b> (optional) Line number for which you want to delete video attributes. If this parameter is <i>not</i> passed, then all video attributes for the current list are deleted.
<b>Output:</b>	None.

### 5.7.8 MSG^VALM10(): Post Message to “Message Window”

<b>Reference Type:</b>	Supported
<b>Category:</b>	ListMan
<b>ICR #:</b>	10117
<b>Description:</b>	This API allows you to immediately post a message to the “message window” located in the lower frame bar of the List Manager display screen.   <b>NOTE:</b> To display a custom message when List Manager re-displays the screen after an action is performed, set the <a href="#">VALMSG</a> variable to the desired message text.
<b>Format:</b>	MSG^VALM10 ([message])
<b>Input Parameters:</b>	<b>message:</b> (optional) Text up to <b>50 characters</b> . If you do <i>not</i> pass this string, any custom message currently displayed is turned off, and List Manager’s standard message is re-displayed.
<b>Output:</b>	None.

## 5.8 Conversion APIs

### 5.8.1 \$\$FDATE^VALM1(): Returns Date in “MM/DD/YY” Format

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10116

**Description:** This extrinsic function returns a date in the following format:

**MM/DD/YY**

For example: **12/12/22**.

**Format:** S X=\$\$FDATE^VALM1 (fmdate)

**Input Parameters:** **fmdate:** (required) VA FileMan formatted date.

**Output:** **return value:** Returns date in “MM/DD/YY” format.

### 5.8.2 \$\$FDTTM^VALM1(): Returns Date/Time in “MM/DD/YY@HH:MM” Format

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10116

**Description:** This extrinsic function returns a date/time in the following format:

**MM/DD/YY@HH:MM**

For example: **12/12/22@09:00**

**Format:** S X=\$\$FDTTM^VALM1 (fmdate)

**Input Parameters:** **fmdate:** (required) VA FileMan formatted date/time.

**Output:** **return value:** Returns date/time in “MM/DD/YY@HH:MM” format.

### 5.8.3 \$\$FTIME^VALM1(): Returns Date/Time in “MMM DD, YYYY@HH:MM” Format

**Reference Type:** Supported  
**Category:** ListMan  
**ICR #:** 10116  
**Description:** This extrinsic function returns a date/time in the following format:  
**MMM DD, YYYY@HH:MM**

For example: **DEC 12, 2022@09:00**

**Format:** S X=\$\$FTIME^VALM1 (fmdate)  
**Input Parameters:** **fmdate:** (required) VA FileMan formatted date/time.  
**Output:** **return value:** Returns date/time in “MMM DD, YYYY@HH:MM” format.

### 5.8.4 \$\$LOWER^VALM1(): Converts String from Uppercase to Lowercase

**Reference Type:** Supported  
**Category:** ListMan  
**ICR #:** 10116  
**Description:** This extrinsic function converts a string from uppercase to lowercase. It parses the string using a space, comma, and a “/”. It starts with the **second character** after each delimiter.

If your line of text contains many consecutive spaces, it is often faster to execute this function as you build each portion of the line, instead of after the line has been completely built.

**Format:** S X=\$\$LOWER^VALM1 (string)  
**Input Parameters:** **string:** (required) String to convert.  
**Output:** **return value:** Returns converted string.

### 5.8.4.1 Example

```
>S X="PATIENT,ONE AND/OR PATIENT,TWO"  
>S X=$$LOWER^VALM1(X)  
>W X  
Patient,One And/Or Patient,Two
```

### 5.8.5 \$\$NOW^VALM1: Returns Value of “NOW” in External Format

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10116

**Description:** This extrinsic function returns the value of “NOW” in external format.

**Format:** S X=\$\$NOW^VALM1

**Input Parameters:** None.

**Output:** **return value:** Returns value of “NOW” in [\\$\\$FTIME^VALM1](#) format (e.g., “Mar 06, 2023 11:15:29”).

### 5.8.6 \$\$UPPER^VALM1(): Converts String from Lowercase to Uppercase

**Reference Type:** Supported

**Category:** ListMan

**ICR #:** 10116

**Description:** This extrinsic function converts a string from lowercase to uppercase.

**Format:** S X=\$\$UPPER^VALM1(string)

**Input Parameters:** **string:** (required) String to convert.

**Output:** **return value:** Returns converted string.

## Index

### \$

\$\$FDATE^VALM1, 59  
\$\$FDTM^VALM1, 59  
\$\$FTIME^VALM1, 60  
\$\$LOWER^VALM1, 60  
\$\$NOW^VALM1, 61  
\$\$SETFLD^VALM1, 12, 49  
\$\$SETSTR^VALM1, 49  
\$\$UPPER^VALM1, 61

### @

@VALMAR@("IDX") Variables, 40  
@VALMAR@(#,0) Variable, 40

### A

Action Area, 2, 3, 9, 17, 20  
Actions  
    Calling List Manager and Other Programs  
        from Actions, 21  
    Creating, 14  
    Defining, 14  
    DISPLAY, 31  
    EXPANDED, 36  
    How To Define, 14  
    Overriding the Default Action, 19  
    Print List, 32  
    PROTOCOL, 25  
    PROTOCOL, 31  
    Supplied by List Manager), 42  
ALLOWABLE NUMBER OF ACTIONS  
    (#.12) Field, 33  
Anonymous Directories, xv  
API  
    ENDR^%ZISS, 41  
APIs  
    Conversion, 59  
    DIR, 44  
    General, 43  
    List Line Text, 48  
    List Line Video, 51

Screen Control, 55

ARRAY NAME (#107) Field, 36, 49, 51

ARRAY NAME Field, 10, 11, 21, 35

Arrays

    Build the List Array Using List

        Manager's API, 11

    Build the List Array Yourself, 11

    Creating, 10

    Creating the Array with SET^VALM10,  
        11

    Defining, 10

    Store the List, 10

    VALMHDR, 9

Assumptions, xiv

Attributes

    Blink Off, 41

    Blink On, 41

    High Intensity, 41

    Normal Video, 41

    Reverse Video Off, 41

    Reverse Video On, 41

    Setting and Displaying Video Attributes

        for List Lines with

        FLDCTRL^VALM10, 13

    Setting Video Attributes in Your List

        Line, 19

    Terminal Type Attributes for List

        Manager Users, 5

    Underline Off, 41

    Underline On, 41

AUTOMATIC DEFAULTS (#.14) Field, 34

### B

Blink Off Attribute, 41

Blink On Attribute, 41

BOTTOM MARGIN (#.06) Fields, 32

Browsing

    Word-Processing Fields, 21

Build the List Array Using List Manager's

    API, 11

Build the List Array Yourself, 11

## C

- Calling List Manager and Other Programs
  - from Actions, 21
- Callout Boxes, x
- Calls
  - DIR, 15, 19
- CAPTION LINE COLUMNS (#200) Field, 37, 48, 50, 55
- Caption Line Information Fields
  - CAPTION LINE COLUMNS (#200) Field, 37
  - COLUMN (#.02) Field, 37
  - DEFAULT VIDEO ATTRIBUTES (#.05) Field, 38
  - DISPLAY TEXT (#.04) Field, 37
  - ITEM NAME (#.01) Field, 37
  - SCROLL LOCK (#.06) Field, 38
  - WIDTH (#.03) Field, 37
- CHGCAP^VALM, 55
- CLEAN^VALM10, 57
- CLEAR^VALM1, 56
- CNTRL^VALM10, 51
- Code
  - Examples, 23
- COLUMN (#.02) Field, 37
- COLUMN WIDTH Field, 17
- Columnar Arrangement of Menu Items, 18
- Columns
  - Columnar Arrangement of Menu Items, 18
  - Scroll-Locking, 20
- Components
  - Major List Manager Components, 4
- Contents, iii
- Conversion APIs, 59
- Create a New List Template, 6
- Create an Outline Routine, 6
- Creating the Array with SET^VALM10, 11

## D

- Data Dictionary
  - Data Dictionary Utilities Menu, xiv
  - Listings, xiv
- DATE RANGE LIMIT (#.13) Field, 33

- DEFAULT VIDEO ATTRIBUTES (#.05) Field, 38, 52
- Define List Actions, 14
- Define List Array, 10
- Define List Menu, 17
- Define List Template, 6
- Demographics Fields
  - ENTITY NAME (#.09) Field, 31
  - Name field, 31
  - SCREEN TITLE (#.11) Field, 31
- DI DDU Menu, xiv
- DILIST Option, xiv
- DIR API, 44
- DIR Call, 15, 19
- DIR(0) Variable, 15
- Directories
  - Anonymous, xv
- Disclaimers
  - Documentation, ix
  - Software, viii
- DISPLAY Action, 31
- DISPLAY TEXT (#.04) Field, 37
- Documentation
  - Symbols, ix
- Documentation Conventions, ix
- Documentation Disclaimer, ix
- Documentation Navigation, xiii

## E

- Edit the List Template, 7
- Edit the Outline Routine, 8
- EN Outline Routine Tag, 9
- EN^VALM, 43
- EN^VALM2, 44
- ENDR^%ZISS API, 41
- ENTITY NAME (#.09) Field, 31
- ENTRY ACTION Field, 14, 19
- ENTRY CODE (#106) Field, 35, 36, 44
- ENTRY CODE Field, 10, 11
- Entry Selection, 19
- Entry Selection and Light Bar Scrolling, 19
- Examples
  - Code, 23
- EXIT ACTION Field, 14
- EXIT CODE (#105) Field, 35
- EXIT Outline Routine Tag, 9

EXPAND CODE (#102) Field, 36  
EXPANDED Action, 36  
EXPND Outline Routine Tag, 9  
Export Your List Manager Application, 21  
Exporting  
    List Manager Applications, 21

## F

### Fields

ALLOWABLE NUMBER OF ACTIONS  
    (#.12), 33  
ARRAY NAME, 10, 11, 21, 35  
ARRAY NAME (#107), 36, 49, 51  
AUTOMATIC DEFAULTS (#.14), 34  
BOTTOM MARGIN (#.06), 32  
CAPTION LINE COLUMNS (#200), 37,  
    48, 50, 55  
COLUMN (#.02), 37  
COLUMN WIDTH, 17  
DATE RANGE LIMIT (#.13), 33  
DEFAULT VIDEO ATTRIBUTES  
    (#.05), 38, 52  
DISPLAY TEXT (#.04), 37  
ENTITY NAME (#.09), 31  
ENTRY ACTION, 14, 19  
ENTRY CODE, 10, 11  
ENTRY CODE (#106), 35, 36, 44  
EXIT ACTION, 14  
EXIT CODE (#105), 35  
EXPAND CODE (#102), 36  
HEADER, 17, 18  
HEADER CODE (#100), 34, 40, 43  
HELP CODE (#103), 36  
HIDDEN MENU (#1.02), 32  
ITEM NAME (#.01), 37  
ITEM TEXT, 14  
MNEMONIC WIDTH, 17  
NAME (#.01), 31  
OK TO TRANSPORT ? (#.07), 33  
PRINT PROTOCOL (#1.01), 32  
PROTOCOL MENU, 17  
PROTOCOL MENU (#.1), 32  
RIGHT MARGIN (#.04), 32, 38  
SCREEN TITLE (#.11), 31  
SCROLL LOCK (#.06), 38  
TOP MARGIN (#.05) Field, 32

TYPE OF LIST, 17  
TYPE OF LIST (#.02), 31, 32, 36  
USE CURSOR CONTROL (#.08), 33  
WIDTH (#.03), 37  
Word-Processing  
    Browsing, 21

Figures, vi

### Files

LIST TEMPLATE (#409.61), 4, 44, 48,  
    50, 55  
NEW PERSON (#200), 13  
PROTOCOL (#101), 4, 14, 42  
Fine Tune Your Application, 19  
FLDCTRL^VALM10, 52  
FLDTEXT^VALM10, 50  
FLDUPD^VALM1, 48  
FULL^VALM1, 56

## G

General APIs, 43  
Generic Actions  
    Supplied by List Manager, 42  
Getting Started, 2

## H

HDR Outline Routine Tag, 9  
Header Area, 2, 3, 9  
HEADER CODE (#100) Field, 34, 40, 43  
HEADER Field, 17, 18  
Help  
    At Prompts, xiii  
    Online, xiii  
    Question Marks, xiii  
HELP CODE (#103) Field, 36  
HELP Outline Routine Tag, 9  
Hidden Menu, 18, 43  
HIDDEN MENU (#1.02) Field, 32  
High Intensity Attribute, 41  
History, Revisions to Documentation and  
    Patches, ii  
Home Pages  
    Adobe Website, xiv  
    ListMan Website, xiv  
    SPM Website, ix  
VA Software Document Library (VDL),  
    xiv

How to  
    Obtain Technical Information Online, xiii  
    Use this Manual, viii  
How To Define an Action, 14  
How to Make a List Manager Application, 6  
How to Select List Items, 15

## I

INIT Outline Routine Tag, 9, 39, 40  
Installation, 4, 5  
INSTR^VALM1, 56  
Intended Audience, viii  
Introduction, 1  
ITEM NAME (#.01) Field, 37  
ITEM TEXT Field, 14

## K

Kernel  
    Video Variables, 41  
Kernel Installation and Distribution System  
    (KIDS), 21, 22, 33  
KIDS  
    Kernel Installation and Distribution  
        System, 21, 22, 33  
KILL^VALM10, 58

## L

Lines  
    Updating, 20  
List Area, 2, 3  
List File Attributes Option, xiv  
List Line Text APIs, 48  
List Line Video APIs, 51  
List Manager  
    Generic Action Protocols, 42  
    Main Screen, 2  
    Workbench  
        VALMWB, 3  
List Region Fields  
    BOTTOM MARGIN (#.06) Field, 32  
    RIGHT MARGIN (#.04) Field, 32  
    TOP MARGIN (#.05) Field, 32  
List Template, 4, 6, 7, 9, 10, 11, 12, 13, 14,  
    19, 21, 23, 30, 31, 32, 33, 34, 35, 36, 39,  
    42, 43, 49, 50, 51, 53

    Creating, 6  
    Defining, 6  
    Editing, 7  
LIST TEMPLATE (#409.61) File, 4, 44, 48,  
    50, 55

ListMan  
    \$\$FDATE^VALM1, 59  
    \$\$FDTTM^VALM1, 59  
    \$\$FTIME^VALM1, 60  
    \$\$LOWER^VALM1, 60  
    \$\$NOW^VALM1, 61  
    \$\$SETFLD^VALM1, 49  
    \$\$SETSTR^VALM1, 49  
    \$\$UPPER^VALM1, 61  
    CHGCAP^VALM, 55  
    CLEAN^VALM10, 57  
    CLEAR^VALM1, 56  
    CNTRL^VALM10, 51  
    EN^VALM, 43  
    EN^VALM2, 44  
    FLDCTRL^VALM10, 52  
    FLDTEXT^VALM10, 50  
    FLDUPD^VALM1, 48  
    FULL^VALM1, 56  
    INSTR^VALM1, 56  
    KILL^VALM10, 58  
    MSG^VALM10, 58  
    PAUSE^VALM1, 44  
    RANGE^VALM1, 44  
    RE^VALM4, 57  
    RESTORE^VALM10, 53  
    SAVE^VALM10, 54  
    SELECT^VALM10, 54  
    SET^VALM10, 51  
    SHOW^VALM, 43  
    Website, xiv  
    WRITE^VALM10, 55

Lists  
    Long, 21  
Long Lists, 21

## M

Major List Manager Components, 4  
Menu  
    Creating, 17  
    Defining, 17



## Menus

- Columnar Arrangement of Menu Items, 18
- Data Dictionary Utilities, xiv
- DI DDU, xiv
- Hidden, 18
- PROTOCOL, 24
- SDAM APPOINTMENT MENU, 24
- Steps to Set Up Your Application's Menu, 17
- Sub-Menus, 18
- Message Window, 16, 40, 58
- MNEMONIC WIDTH Field, 17
- Modes
  - Screen, 20, 56
  - Scrolling, 20
- MSG^VALM10, 58
- MUMPS Code Related Fields
  - ARRAY NAME (#107) Field, 36
  - ENTRY CODE (#106) Field, 35
  - EXIT CODE (#105) Field, 35
  - EXPAND CODE (#102) Field, 36
  - HEADER CODE (#100) Field, 34
  - HELP CODE (#103) Field, 36

## N

- NAME (#.01) Field, 31
- Network File System (NFS), xv
- NEW PERSON (#200) File, 13
- Normal Video Attribute, 41

## O

- Obtaining
  - Data Dictionary Listings, xiv
- OK TO TRANSPORT ? (#.07) Field, 33
- Online
  - Documentation, xiii
  - Technical Information, How to Obtain, xiii
- Options
  - Data Dictionary Utilities, xiv
  - DI DDU, xiv
  - DILIST, xiv
  - List File Attributes, xiv
- Orientation, viii
- Other Fields

- ALLOWABLE NUMBER OF ACTIONS (#.12) Field, 33
- AUTOMATIC DEFAULTS (#.14) Field, 34
- DATE RANGE LIMIT (#.13) Field, 33
- OK TO TRANSPORT ? (#.07) Field, 33
- USE CURSOR CONTROL (#.08) Field, 33

## Outline Routine, 6, 7, 8, 9, 10

- Tag
  - EN, 9
  - EXIT, 9
  - EXPND, 9
  - HDR, 9
  - HELP, 9
  - INIT, 9, 39, 40

## Overriding the Default Action, 19

## P

- Package Requirements, 4
- Patches
  - History, ii
- PAUSE^VALM1, 44
- Print List Action, 32
- PRINT PROTOCOL (#1.01) Field, 32
- PROTOCOL (#101) File, 4, 14, 42
- PROTOCOL Action, 25, 31
- Protocol Information Fields
  - HIDDEN MENU (#1.02) Field, 32
  - PRINT PROTOCOL (#1.01) Field, 32
  - PROTOCOL MENU (#.1) Field, 32
  - TYPE OF LIST (#.02) Field, 31
- PROTOCOL Menu, 24
- PROTOCOL MENU (#.1) Fields, 32
- PROTOCOL MENU Field, 17
- Protocols
  - Supplied by List Manager, 42
  - VALM DOWN A LINE, 42
  - VALM FIRST SCREEN, 42
  - VALM GOTO PAGE, 42
  - VALM HIDDEN ACTIONS, 43
  - VALM LAST SCREEN, 42
  - VALM LEFT, 42
  - VALM NEXT SCREEN, 42
  - VALM PREVIOUS SCREEN, 42
  - VALM PRINT LIST, 42

VALM PRINT SCREEN, 42  
VALM QUIT, 42  
VALM REFRESH, 42  
VALM RIGHT, 42  
VALM SEARCH LIST, 42  
VALM TURN ON/OFF MENUS, 42  
VALM UP ONE LINE, 42

## Q

Question Mark Help, xiii

## R

RANGE^VALM1, 44  
RE^VALM4, 57  
Reference Materials, xiv  
Reference Type  
Supported  
    \$\$FDATE^VALM1, 59  
    \$\$FDTM^VALM1, 59  
    \$\$FTIME^VALM1, 60  
    \$\$LOWER^VALM1, 60  
    \$\$NOW^VALM1, 61  
    \$\$SETFLD^VALM1, 49  
    \$\$SETSTR^VALM1, 49  
    \$\$UPPER^VALM1, 61  
    CHGCAP^VALM, 55  
    CLEAN^VALM10, 57  
    CLEAR^VALM1, 56  
    CNTRL^VALM10, 51  
    EN^VALM, 43  
    EN^VALM2, 44  
    FLDCTRL^VALM10, 52  
    FLDTEXT^VALM10, 50  
    FLDUPD^VALM1, 48  
    FULL^VALM1, 56  
    INSTR^VALM1, 56  
    KILL^VALM10, 58  
    MSG^VALM10, 58  
    PAUSE^VALM1, 44  
    RANGE^VALM1, 44  
    RE^VALM4, 57  
    RESTORE^VALM10, 53  
    SAVE^VALM10, 54  
    SELECT^VALM10, 54  
    SET^VALM10, 51  
    SHOW^VALM, 43

WRITE^VALM10, 55

Requirements  
    Package, 4  
RESTORE^VALM10, 53  
Reverse Video Off Attribute, 41  
Reverse Video On Attribute, 41  
Revision History, ii  
    Patches, ii  
RIGHT MARGIN (#.04) Field, 32, 38  
Routine to Create List, 10  
Routines  
    Outline, 7, 8, 9, 10  
    VALMWB, 3

## S

SAVE^VALM10, 54  
Screen  
    Main, 2  
Screen Control APIs, 55  
Screen Mode, 20, 56  
SCREEN TITLE (#.11) Field, 31  
Screens  
    Using the Entire Screen, 16  
SCROLL LOCK (#.06) Field, 38  
Scrolling Mode, 20  
Scroll-Locking Columns, 20  
SDAM APPOINTMENT MENU, 24  
SELECT^VALM10, 54  
Selecting  
    Entry Selection and Light Bar Scrolling,  
        19  
    Items, 19  
    List Items, 15  
SET^VALM10, 51  
Setting and Displaying Video Attributes for  
    List Lines with FLDCTRL^VALM10, 13  
Setting Up Text Lines with Captions and  
    \$\$SETFLD^VALM1, 12  
Setting Video Attributes in Your List Line,  
    19  
Setup, 4  
SHOW^VALM, 43  
Software Disclaimer, viii  
Steps to Set Up Your Application's Menu,  
    17  
Sub-Menus, 18

## Symbols

Found in the Documentation, ix

## T

Table of Contents, iii

Tables, vi

Terminal Type Attributes for List Manager Users, 5

TOP MARGIN (#.05) Field, 32

TYPE OF LIST (#.02) Field, 31, 32, 36

TYPE OF LIST Field, 17, 31

## U

Underline Off Attribute, 41

Underline On Attribute, 41

Updating

List Lines, 20

Updating Items in the List, 20

URLs

Adobe Website, xiv

ListMan Website, xiv

SPM Website, ix

VA Software Document Library (VDL), xiv

USE CURSOR CONTROL (#.08) Field, 33

Use this Manual, How to, viii

Using the Entire Screen, 16

## V

VA FileMan, xiii, xiv, 10, 21, 35, 36, 44, 59, 60

Browser, 21

VA Software Document Library (VDL)

Website, xiv

VALM

CHGCAP^VALM, 55

EN^VALM, 43

SHOW^VALM, 43

VALM DOWN A LINE Protocol, 42

VALM FIRST SCREEN Protocol, 42

VALM GOTO PAGE Protocol, 42

VALM HIDDEN ACTIONS Protocol, 43

VALM LAST SCREEN Protocol, 42

VALM LEFT Protocol, 42

VALM NEXT SCREEN Protocol, 42

VALM PREVIOUS SCREEN Protocol, 42

VALM PRINT LIST Protocol, 42

VALM PRINT SCREEN Protocol, 42

VALM QUIT Protocol, 42

VALM REFRESH Protocol, 42

VALM RIGHT Protocol, 42

VALM SEARCH LIST Protocol, 42

VALM TURN ON/OFF MENUS Protocol, 42

VALM UP ONE LINE Protocol, 42

VALM(TITLE) Variable, 39

VALM1

\$\$FDATE^VALM1, 59

\$\$FDTTM^VALM1, 59

\$\$FTIME^VALM1, 60

\$\$LOWER^VALM1, 60

\$\$NOW^VALM1, 61

\$\$SETFLD^VALM1, 49

\$\$SETSTR^VALM1, 49

\$\$UPPER^VALM1, 61

CLEAR^VALM1, 56

FLDUPD^VALM1, 48

FULL^VALM1, 56

INSTR^VALM1, 56

PAUSE^VALM1, 44

RANGE^VALM1, 44

VALM10

CLEAN^VALM10, 57

CNTRL^VALM10, 51

FLDCTRL^VALM10, 52

FLDTEXT^VALM10, 50

KILL^VALM10, 58

MSG^VALM10, 58

RESTORE^VALM10, 53

SAVE^VALM10, 54

SELECT^VALM10, 54

SET^VALM10, 51

WRITE^VALM10, 55

VALM2

EN^VALM2, 44

VALM4

RE^VALM4, 57

VALMBCK Variables, 16, 20, 34, 39, 40, 57

VALMBG Variables, 35, 39

VALMCC Variable, 20

VALMCC Variables, 39  
 VALMCNT Variable, 11, 12  
 VALMCNT Variables, 35, 39  
 VALMDDF() Variable, 38, 39  
 VALMHDR Array, 9  
 VALMHDR() Variable, 34, 40  
 VALMQUIT Variable, 40  
 VALMQUIT Variables, 35, 40  
 VALMSG Variable, 16, 58  
 VALMSG Variables, 40  
 VALMST Variables, 40  
 VALMWB Routine, 3  
 Variables  
   @VALMAR@("IDX"), 40  
   @VALMAR@(#,0), 40  
   DIR(0), 15  
   Kernel Video Variables, 41  
   VALM(TITLE), 39  
   VALMBCK, 16, 20, 39, 40  
   VALMBCK Variables, 34  
   VALMBG, 35, 39  
   VALMCC, 20, 39  
   VALMCNT, 11, 12, 35, 39  
   VALMDDF(), 38, 39  
   VALMHDR(), 34, 40  
   VALMQUIT, 35, 40  
   VALMQUIT variable, 40  
   VALMSG, 16, 40, 58

VALMST, 40  
 XQORM("B"), 41  
 XQORM("B"), 34  
 Variables, List Manager, 39  
 VariablesVALMBCK, 57

## W

Websites  
   Adobe Website, xiv  
   ListMan, xiv  
   SPM, ix  
   VA Software Document Library (VDL),  
     xiv  
 What Comes Next?, 9  
 When the User Is In Scrolling Mode (Not  
   Screen Mode), 20  
 WIDTH (#.03) Field, 37  
 Word-Processing Fields  
   Browsing, 21  
 Workbench, 3, 4, 6, 7, 10, 12, 13, 14, 17, 18,  
   30, 43  
 WRITE^VALM10, 55

## X

XQORM("B") Variable, 41  
 XQORM("B") Variable, 34