

**VA FileMan 22.2**  
**Advanced User Manual**



**July 2025**

**Department of Veterans Affairs (VA)**  
**Office of Information and Technology (OIT)**  
**Product Delivery Service (PDS)**

## Revision History



**REF:** For the archived document revision history, see “[Appendix A—Revision History Archive](#).”

Date	Revision	Description	Author
07/21/2025	1.6	Updates: <ul style="list-style-type: none"> <li>• Updated all styles and formatting throughout.</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>• Updated all references throughout to Kernel manuals to the current, correct title:               <ul style="list-style-type: none"> <li>○ <i>Kernel 8.0 Systems Management: &lt;Functional Division&gt; User Guide.</i></li> <li>○ <i>Kernel 8.0 and Kernel Toolkit 7.3 Developer’s Guide.</i></li> </ul> </li> <li>• Changed references from “Software Product Management (SPM)” to “Product Delivery Service (PDS)”.</li> </ul>	VistA Application Shared Services (VASS) Development Team



**REF:** For the current patch history related to this software, see the Patch Module (i.e., **Patch User Menu** [A1AE USER]) on FORUM.

# Table of Contents

Revision History .....	ii
List of Figures.....	xi
List of Tables .....	xxiii
Orientation .....	xxvi
<b>1 Introduction .....</b>	<b>1</b>
<b>1.1 Menus and Options.....</b>	<b>1</b>
1.1.1 VA FileMan Main Menu .....	1
1.1.2 Utility Functions Menu .....	2
1.1.3 Data Dictionary Utilities Functions Menu.....	2
1.1.4 Other Options Menu.....	3
<b>2 Import and Export Tools.....</b>	<b>4</b>
<b>2.1 What Applications Can You Exchange Data With? .....</b>	<b>4</b>
<b>2.2 How Data is Moved between Applications.....</b>	<b>5</b>
<b>2.3 Dependency on Correct Data Communication.....</b>	<b>5</b>
2.3.1 Data Formats.....	6
2.3.2 How to Export Data.....	8
2.3.3 How to Import Data .....	22
2.3.4 Foreign Formats.....	31
<b>3 Relational Navigation .....</b>	<b>45</b>
<b>3.1 Simple Extended Pointer.....</b>	<b>46</b>
3.1.1 Simple Extended Pointer Syntax (Short form) .....	47
3.1.2 Simple Extended Pointer Syntax (Long Form) .....	48
3.1.3 Examples.....	48
3.1.4 How to Navigate With a Variable Pointer Field.....	49
<b>3.2 Relational Jumps across Files .....</b>	<b>50</b>
<b>3.3 Backward Extended Pointer.....</b>	<b>52</b>
<b>3.4 Join Extended Pointer .....</b>	<b>55</b>
3.4.1 Limitations.....	55
3.4.2 Example.....	56
<b>3.5 Multiline Return Values.....</b>	<b>56</b>
3.5.1 WORD-PROCESSING Field.....	57
3.5.2 Multiples.....	57
3.5.3 Backward Pointer.....	58

<b>4</b>	<b>Advanced Edit Techniques.....</b>	<b>60</b>
<b>4.1</b>	<b>Field Value Stuffing .....</b>	<b>60</b>
4.1.1	Set Field Default (2 //) .....	60
4.1.2	Stuff/Delete Field Value (3///) .....	60
4.1.3	Unvalidated Stuffs: (4////) .....	61
4.1.4	Variable Stuffs.....	62
4.1.5	WORD-PROCESSING Field Stuffing .....	63
4.1.6	Looping (^LOOP) .....	63
<b>4.2</b>	<b>INPUT Templates .....</b>	<b>66</b>
4.2.1	Overview .....	66
4.2.2	Branching within INPUT Templates .....	67
<b>4.3</b>	<b>Edit Qualifiers .....</b>	<b>71</b>
4.3.1	Edit Qualifiers and Customizing Data Editing.....	71
4.3.2	Forcing Special Prompts.....	71
4.3.3	Duplicating Input Values .....	72
4.3.4	Forcing Required Input .....	72
<b>4.4</b>	<b>Text Formatting in Word-Processing Fields .....</b>	<b>73</b>
4.4.1	Word Wrapping .....	73
4.4.2	Tabs .....	73
4.4.3	Formatting Text with Word-processing Windows (Frames)     .....	74
4.4.4	Text Formatting Expressions in Word-Processing Windows.....	75
<b>5</b>	<b>Computed Expressions.....</b>	<b>78</b>
<b>5.1</b>	<b>Syntax.....</b>	<b>78</b>
5.1.1	Elements of Computed Expressions .....	78
5.1.2	Operators in Computed Expressions.....	79
5.1.3	Data Types in Computed Expressions .....	82
5.1.4	Using Functions as Elements in Computed Expressions.....	83
<b>5.2</b>	<b>Where to Use .....</b>	<b>84</b>
5.2.1	Using Computed Expressions in COMPUTED Fields.....	84
5.2.2	Where to Use Computed Expressions "On-the-Fly" .....	87
<b>6</b>	<b>VA FileMan Functions .....</b>	<b>92</b>
<b>6.1</b>	<b>How to Use VA FileMan Functions.....</b>	<b>92</b>
<b>6.2</b>	<b>Documentation Conventions for VA FileMan Functions.....</b>	<b>93</b>
<b>6.3</b>	<b>VA FileMan Function Categories .....</b>	<b>94</b>
6.3.1	Date/Time Functions.....	96

6.3.2	Environmental Functions .....	105
6.3.3	File and File Data Functions.....	107
6.3.4	Mathematical Functions.....	115
6.3.5	Printing Related Functions.....	118
6.3.6	String Functions.....	119
6.3.7	Temporary Data Storage Functions.....	123
6.3.8	M-Related Functions.....	125
<b>7</b>	<b>Statistics .....</b>	<b>135</b>
<b>7.1</b>	<b>How to Generate Statistics from Reports .....</b>	<b>135</b>
<b>7.2</b>	<b>Descriptive Statistics.....</b>	<b>136</b>
7.2.1	Initial Print.....	136
7.2.2	Generating Descriptive Statistics.....	137
<b>7.3</b>	<b>Scattergram .....</b>	<b>138</b>
7.3.1	Initial Print.....	138
7.3.2	Generating the Scattergram.....	139
<b>7.4</b>	<b>Histogram .....</b>	<b>140</b>
7.4.1	Initial Print.....	141
7.4.2	Generating the Histogram .....	142
<b>8</b>	<b>System Management.....</b>	<b>143</b>
<b>8.1</b>	<b>Setup.....</b>	<b>143</b>
8.1.1	Initialization .....	143
8.1.2	Security.....	143
<b>8.2</b>	<b>Standalone VA FileMan .....</b>	<b>144</b>
8.2.1	Device Handling for Standalone VA FileMan.....	144
8.2.2	NEW PERSON File for Standalone VA FileMan.....	150
<b>8.3</b>	<b>^%ZOSF Nodes.....</b>	<b>152</b>
8.3.1	Manually Setting ^%ZOSF Nodes.....	152
<b>8.4</b>	<b>Alternate Editors .....</b>	<b>153</b>
8.4.1	Setting Up Alternate Editors.....	153
<b>8.5</b>	<b>COMPILED ROUTINE File.....</b>	<b>156</b>
8.5.1	COMPILED ROUTINE File Cleanup: ENRLS^DIOZ().....	156
<b>8.6</b>	<b>Compare Data and Data Dictionaries across Environments .....</b>	<b>157</b>
8.6.1	Compare Data Dictionaries.....	157
8.6.2	Compare File Entries .....	159

<b>9</b>	<b>List File Attributes .....</b>	<b>161</b>
<b>9.1</b>	<b>List File Attributes Option.....</b>	<b>161</b>
9.1.1	Brief Data Dictionary .....	162
9.1.2	Condensed Data Dictionary.....	165
9.1.3	Standard and Modified Standard Data Dictionaries .....	168
9.1.4	Custom-Tailored Data Dictionary.....	171
9.1.5	Templates Only Format.....	174
9.1.6	Global Map.....	174
9.1.7	Indexes and Cross-References Only.....	175
9.1.8	Keys Only.....	176
<b>9.2</b>	<b>Map Pointer Relations Option.....</b>	<b>176</b>
<b>9.3</b>	<b>Check/Fix DD Structure Option.....</b>	<b>178</b>
<b>9.4</b>	<b>Find Pointers Into a File Option .....</b>	<b>180</b>
<b>9.5</b>	<b>Meta Data Dictionary .....</b>	<b>182</b>
<b>10</b>	<b>Creating Files and Fields .....</b>	<b>185</b>
<b>10.1</b>	<b>Creating a File.....</b>	<b>185</b>
10.1.1	Naming a New File.....	185
<b>10.2</b>	<b>Creating Fields.....</b>	<b>186</b>
10.2.1	Screen Mode Field Editing .....	187
10.2.2	Field Data Types.....	189
10.2.3	DATE/TIME Data Type .....	190
10.2.4	NUMERIC Data Type .....	192
10.2.5	SET OF CODES Data Type.....	193
10.2.6	FREE TEXT Data Type .....	194
10.2.7	WORD-PROCESSING Data Type .....	195
10.2.8	COMPUTED Data Type.....	197
10.2.9	POINTER TO A FILE Data Type.....	198
10.2.10	VARIABLE-POINTER Data Type .....	200
10.2.11	MUMPS Data Type.....	202
10.2.12	BOOLEAN Data Type.....	203
10.2.13	LABEL REFERENCE Data Type.....	203
10.2.14	TIME Data Type.....	203
10.2.15	YEAR Data Type .....	204
10.2.16	UNIVERSAL TIME Data Type.....	204
10.2.17	FT POINTER Data Type.....	204

10.2.18	FT DATE Data Type .....	204
10.2.19	RATIO Data Type .....	205
<b>10.3</b>	<b>Multiple-Valued Field (Multiples) .....</b>	<b>205</b>
<b>10.4</b>	<b>Making a Field Mandatory .....</b>	<b>206</b>
<b>10.5</b>	<b>Field Number Sequences .....</b>	<b>207</b>
<b>10.6</b>	<b>NUMBER (.001) Field .....</b>	<b>207</b>
10.6.1	Forced Lookups Using Numbers .....	208
<b>10.7</b>	<b>Changing and Deleting Fields .....</b>	<b>209</b>
10.7.1	Changing Field Attributes .....	209
10.7.2	Changing a Field's DATA TYPE Value .....	212
10.7.3	Deleting an Existing Field .....	213
<b>10.8</b>	<b>Examples of File and Field Creation .....</b>	<b>213</b>
10.8.1	File Creation .....	214
10.8.2	DATE/TIME Fields .....	215
10.8.3	SET OF CODES Field .....	217
10.8.4	FREE TEXT Field .....	218
10.8.5	WORD-PROCESSING Field .....	220
10.8.6	COMPUTED Field .....	221
10.8.7	POINTER TO A FILE Field .....	223
10.8.8	VARIABLE-POINTER Field .....	224
10.8.9	BOOLEAN Field .....	225
10.8.10	LABEL REFERENCE Field .....	226
10.8.11	TIME Field .....	227
10.8.12	YEAR Field .....	228
10.8.13	UNIVERSAL TIME Field .....	229
10.8.14	FT POINTER Field .....	230
10.8.15	FT DATE Field .....	230
10.8.16	RATIO Field .....	232
10.8.17	Creating a Multiple .....	233
<b>11</b>	<b>File Utilities .....</b>	<b>237</b>
<b>11.1</b>	<b>Verify Fields .....</b>	<b>237</b>
<b>11.2</b>	<b>Cross-Reference a Field or File .....</b>	<b>238</b>
11.2.1	Types of Traditional Cross-references .....	239
11.2.2	Edit a Traditional Cross-reference .....	240
11.2.3	Create a Traditional Cross-reference .....	241

11.2.4	Delete a Traditional Cross-reference.....	242
11.2.5	New-Style Cross-References .....	242
11.2.6	Edit a New-Style Cross-reference.....	245
11.2.7	Create a New-Style Cross-Reference .....	247
11.2.8	Delete a New-Style Cross-Reference .....	249
<b>11.3</b>	<b>Identifier .....</b>	<b>250</b>
<b>11.4</b>	<b>Re-Index File Option.....</b>	<b>252</b>
11.4.1	Limits on Reindexing Files.....	252
<b>11.5</b>	<b>INPUT Transform (Syntax).....</b>	<b>253</b>
<b>11.6</b>	<b>Edit File.....</b>	<b>254</b>
<b>11.7</b>	<b>OUTPUT Transform.....</b>	<b>258</b>
<b>11.8</b>	<b>Template Edit .....</b>	<b>259</b>
<b>11.9</b>	<b>Uneditable Data .....</b>	<b>263</b>
<b>11.10</b>	<b>Mandatory/Required Field Check .....</b>	<b>264</b>
<b>11.11</b>	<b>Key Definition.....</b>	<b>264</b>
11.11.1	Create a Key .....	265
11.11.2	Edit a Key.....	267
11.11.3	Delete a Key .....	268
11.11.4	Verify a Key.....	269
<b>12</b>	<b>Auditing.....</b>	<b>270</b>
<b>12.1</b>	<b>Auditing a Data Field.....</b>	<b>271</b>
12.1.1	Overview.....	271
12.1.2	Setting a Data Field Audit .....	272
12.1.3	Turning Data Field Audit On/Off .....	274
12.1.4	Reviewing the Data Field Audit Trail .....	275
12.1.5	Tracking Data Field Audits .....	278
12.1.6	Purging a Data Field Audit Trail .....	278
<b>12.2</b>	<b>Auditing a Data Dictionary .....</b>	<b>281</b>
12.2.1	Setting Automatic Data Dictionary Auditing.....	281
12.2.2	Reviewing the Data Dictionary Audit Trail .....	282
12.2.3	Purging a Data Dictionary Audit Trail .....	285
12.2.4	Auditable Word Processing Fields .....	286
12.2.5	Word-Processing Fields Can be Made Uneditable .....	287
12.2.6	Reviewing a User's Data Access .....	287

<b>13</b>	<b>Data Security</b>	<b>289</b>
<b>13.1</b>	<b>Security at the File Level</b>	<b>289</b>
13.1.1	Access Code Security on Files	289
13.1.2	File Access Security (Formerly Part 3 of Kernel)	291
<b>13.2</b>	<b>Protection for Fields in a File</b>	<b>292</b>
<b>13.3</b>	<b>Protection for Templates</b>	<b>293</b>
<b>14</b>	<b>Transferring File Entries</b>	<b>294</b>
<b>14.1</b>	<b>Transfer File Entries Option</b>	<b>294</b>
14.1.1	Transferring Data within the Same File	295
14.1.2	Transferring Entries between Files	297
14.1.3	Transferring Entries into a New File	299
<b>14.2</b>	<b>Compare/Merge File Entries Option</b>	<b>300</b>
14.2.1	Comparing Entries	300
14.2.2	Merging Entries	302
<b>15</b>	<b>Extract Tool</b>	<b>308</b>
<b>15.1</b>	<b>Extract Overview</b>	<b>308</b>
<b>15.2</b>	<b>Important Items to Note</b>	<b>308</b>
15.2.1	Source File	309
15.2.2	Destination File	309
<b>15.3</b>	<b>Mapping Information</b>	<b>311</b>
<b>15.4</b>	<b>ARCHIVAL ACTIVITY File</b>	<b>313</b>
<b>15.5</b>	<b>Extract Steps</b>	<b>314</b>
15.5.1	Select Entries to Extract Option (1 of 9)	315
15.5.2	Add/Delete Selected Entries Option (2 of 9)	317
15.5.3	Print Selected Entries Option (3 of 9)	318
15.5.4	Modify Destination File Option (4 of 9)	319
15.5.5	Create Extract Template Option (5 of 9)	320
15.5.6	Update Destination File Option (6 of 9)	323
15.5.7	Purge Extracted Entries Option (7 of 9)	325
15.5.8	Cancel Extract Selection Option (8 of 9)	326
15.5.9	Validate Extract Template Option (9 of 9)	327
<b>16</b>	<b>Filegrams</b>	<b>328</b>
<b>16.1</b>	<b>FILEGRAM-Type Templates</b>	<b>329</b>
<b>16.2</b>	<b>Filegram and Archiving Relationship</b>	<b>329</b>
<b>16.3</b>	<b>Filegram Menu Options</b>	<b>329</b>

<b>16.4</b>	<b>Using Filegrams</b> .....	<b>330</b>
<b>16.5</b>	<b>Filegram Steps</b> .....	<b>331</b>
16.5.1	Create/Edit Filegram Template Option .....	331
16.5.2	Display Filegram Template Option .....	333
16.5.3	Specifiers Option .....	334
16.5.4	Generate Filegram Option .....	335
16.5.5	Receiving Filegrams with MailMan .....	336
16.5.6	View Filegram Option .....	337
16.5.7	Install/Verify Filegram Option.....	338
16.5.8	Deleting a Filegram .....	338
<b>17</b>	<b>Archiving</b> .....	<b>339</b>
<b>17.1</b>	<b>Considerations before Archiving</b> .....	<b>340</b>
<b>17.2</b>	<b>Archiving Process, including Archiving Options (1-9)</b> .....	<b>341</b>
17.2.1	Select Entries to Archive .....	342
17.2.2	Add/Delete Selected Entries.....	344
17.2.3	Print Selected Entries .....	346
17.2.4	Create Filegram Archiving Template.....	347
17.2.5	Write Entries to Temporary Storage.....	348
17.2.6	Move Archived Data to Permanent Storage .....	349
17.2.7	Purge Stored Entries.....	351
17.2.8	Cancel Archival Selection.....	352
17.2.9	Find Archived Entries .....	353
17.2.10	ARCHIVAL ACTIVITY File.....	355
<b>18</b>	<b>Meta Data Dictionary</b> .....	<b>356</b>
<b>18.1</b>	<b>Overview</b> .....	<b>356</b>
<b>18.2</b>	<b>^DDD: Initial Creation</b> .....	<b>357</b>
<b>18.3</b>	<b>FILELIST^DDD: File List Partial Update</b> .....	<b>357</b>
<b>18.4</b>	<b>PARTIAL1^DDD: Partial Update using ^DIC(DDD,“%MSC”) .....</b>	<b>358</b>
<b>18.5</b>	<b>PARTIAL2^DDD: Partial Update using ^DD(FILE,FIELD,“DT”) .....</b>	<b>358</b>
<b>19</b>	<b>Data Synchronization</b> .....	<b>359</b>
<b>19.1</b>	<b>Overview</b> .....	<b>359</b>
<b>19.2</b>	<b>Input JSON Object</b> .....	<b>359</b>
<b>19.3</b>	<b>Input JSON Data File</b> .....	<b>360</b>
<b>20</b>	<b>Appendix A—Revision History Archive</b> .....	<b>362</b>
	Glossary .....	365

## List of Figures

Figure 1: VA FileMan [DIUSER] Main Menu .....	1
Figure 2: VA FileMan—Utility Functions Menu Options .....	2
Figure 3: VA FileMan—Data Dictionary Utilities Menu Options .....	2
Figure 4: VA FileMan—Other Options Menu Options.....	3
Figure 5: Import and Export Tools—Example of a Record Delimited by a Comma.....	6
Figure 6: Import and Export Tools—Example of a File with Records Delimited by a Comma .....	6
Figure 7: Import and Export Tools—Example of a Record Where the Delimiter between Quotes is Ignored .....	7
Figure 8: Import and Export Tools—Example of a Fixed-Length Record .....	7
Figure 9: Import and Export Tools—Example of a File with Fixed-Length Records.....	7
Figure 10: Import and Export Tools—Data Export Options.....	8
Figure 11: Import and Export Tools—Creating the SELECTED EXPORTED FIELDS Template	12
Figure 12: Import and Export Tools—Creating the EXPORT Template .....	14
Figure 13: Import and Export Tools—Identifying the FOREIGN FORMAT and EXPORT Templates .....	15
Figure 14: Import and Export Tools—Entering DATA TYPE Field Values in an EXPORT Template .....	15
Figure 15: Import and Export Tools—Searching for Entries to be Exported .....	17
Figure 16: Import and Export Tools—Choosing a Device to Send Exported Data.....	18
Figure 17: Import and Export Tools—Example of Exported Data .....	20
Figure 18: Import and Export Tools—Example of Data Flattening When Exporting Data from Multiples.....	21
Figure 19: Import and Export Tools—Example of a File Structure .....	21
Figure 20: Import and Export Tools—Import Data Option.....	22
Figure 21: Import and Export Tools—Example of a Completed Data Import Form .....	24
Figure 22: Import and Export Tools—Example of Fields Selected for Import.....	25
Figure 23: Import and Export Tools—Exiting the Template Form and Performing the Import .....	27
Figure 24: Import and Export Tools—Example of an Import Results Report.....	28
Figure 25: Import and Export Tools—Example of Fields Selected for Import to a Multiple	29
Figure 26: Import and Export Tools—Example of Data <i>Not</i> Flattened When Importing Data to a Multiple.....	30

Figure 27: Import and Export Tools—Verifying the Maximum Record Length on a VMS System.....	30
Figure 28: Import and Export Tools—Using VA FileMan Functions When Exporting Data.....	37
Figure 29: Import and Export Tools—Print Format Documentation Option.....	39
Figure 30: Import and Export Tools—Listing FOREIGN FORMAT File Entries Using the Print Format Documentation Option .....	40
Figure 31: Import and Export Tools—Define Foreign File Format Option.....	41
Figure 32: Import and Export Tools—Choosing the Define Foreign Format Option .....	41
Figure 33: Import and Export Tools—Selecting an Existing FOREIGN FORMAT File Entry.....	42
Figure 34: Import and Export Tools—Viewing the Contents of a FOREIGN FORMAT File Entry.....	42
Figure 35: Import and Export Tools—Creating a New FOREIGN FORMAT File Entry .....	43
Figure 36: Import and Export Tools—ScreenMan Form for Editing Foreign Formats .....	43
Figure 37: Import and Export Tools—Second Page of a Multiple’s “Popup” Window Opened.....	44
Figure 38: Relational Navigation—Example Illustrating Relational Navigation.....	45
Figure 39: Relational Navigation—Example of a Simple Extended Pointer.....	47
Figure 40: Relational Navigation—Example of a Relational Query.....	48
Figure 41: Relational Navigation—Example of the Short Form Extended Pointer Syntax.....	49
Figure 42: Relational Navigation—Entering Print Specifications and Including Fields in Pointed-To Files.....	50
Figure 43: Relational Navigation—Example of Output that Includes Fields from Pointed-To Files.....	50
Figure 44: Relational Navigation—Using Relational Jumps with the Enter or Edit File Entries Option .....	51
Figure 45: Relational Navigation—Example Illustrating a File with Pointers to another File.....	52
Figure 46: Relational Navigation—Example Using a Backward Extended Pointer .....	53
Figure 47: Relational Navigation—Example of the Output Produced after Using a Backward Extended Pointer .....	54
Figure 48: Relational Navigation—Using a Value from One File to Do a Lookup in a Second File .....	55
Figure 49: Relational Navigation—Example of Matching Entries in Two Files Using the SORT BY Field.....	56
Figure 50: Relational Navigation—Example of Using a WORD-PROCESSING Field in an Extended Pointer Expression .....	57
Figure 51: Relational Navigation—Example of Using the Simple Pointer Syntax to Get Data from a Multiple .....	57

Figure 52: Relational Navigation—Example Using a Cross-Referenced Backward Pointer to Yield a Multiline Response: Stored in an INPUT Template.....	58
Figure 53: Relational Navigation—Example Using an INPUT Template with a Cross-Referenced Backward Pointer to Yield a Multiline Response.....	59
Figure 54: Advanced Edit Techniques—Setting a Default Value for a Field.....	60
Figure 55: Advanced Edit Techniques—“Stuffing” a Value into a Field in the Database..	60
Figure 56: Advanced Edit Techniques—Deleting a Value from a Field in the Database...	60
Figure 57: Advanced Edit Techniques—Warning Message When Deleting a Value from a Field in the Database.....	61
Figure 58: Advanced Edit Techniques—“Stuffing” Default Value into a Field in the Database—Bypassing INPUT Transform .....	61
Figure 59: Advanced Edit Techniques—Example of “Stuffing” a Variable Default Value into a Field in the Database .....	62
Figure 60: Advanced Edit Techniques—Appending Text on to a WORD-PROCESSING Field Value.....	63
Figure 61: Advanced Edit Techniques—Example of “Looping” through Entries in a File .	64
Figure 62: Advanced Edit Techniques—Example of Loading Data into a Newly Created Field for Select Records .....	65
Figure 63: Advanced Edit Techniques—Example of Deleting Data from a Newly Created Field for Select Records .....	65
Figure 64: Advanced Edit Techniques—Storing a List of Edit Fields in an INPUT Template	66
Figure 65: Advanced Edit Techniques—Creating a Special INPUT Template.....	67
Figure 66: Advanced Edit Techniques—Defining INPUT Template to Branch to Different Field Based on another Field’s Value (1 of 2) .....	68
Figure 67: Advanced Edit Techniques—Defining INPUT Template to Branch to Different Field Based on another Field’s Value (2 of 2) .....	69
Figure 68: Advanced Edit Techniques—Example Verifying Automatic Branching to Other Fields Based on User’s Entry (1 of 2).....	69
Figure 69: Advanced Edit Techniques—Example Verifying Automatic Branching to Other Fields Based on User’s Entry (2 of 2).....	70
Figure 70: Advanced Edit Techniques—Example Using the Title Edit Qualifier .....	72
Figure 71: Advanced Edit Techniques—Example Using the Duplicate Edit Qualifier.....	72
Figure 72: Advanced Edit Techniques—Example Using the Required Edit Qualifier.....	72
Figure 73: Computed Expressions—Example Using the Print File Entries Option to Identify a Caption .....	82
Figure 74: Computed Expressions—Defining a DATA TYPE Field as COMPUTED.....	84

Figure 75: Computed Expressions—Entering the Computed Expression into a DATA TYPE Field of COMPUTED .....	84
Figure 76: Computed Expressions—Sample Dialog Encountered with a COMPUTED Field with Expected Numeric Result: Selecting Number-Valued .....	85
Figure 77: Computed Expressions—Sample Dialog Encountered with a COMPUTED Field with Expected Numeric Result: Setting Rounding Value.....	85
Figure 78: Computed Expressions—Sample Dialog Encountered with a COMPUTED Field with Expected Numeric Result: Setting Totaling Value.....	85
Figure 79: Computed Expressions—Dialog Encountered When Defining a COMPUTED Field .....	86
Figure 80: Computed Expressions—Entering a Computed Expression at a “PRINT FIELD” Prompt.....	87
Figure 81: Computed Expressions—Entering a Computed Expression at a “SORT BY” Prompt.....	88
Figure 82: Computed Expressions—Entering a Computed Expression at the “Start with...” and “Go to...” Prompt.....	89
Figure 83: Computed Expressions—“Stuffing” a Value in a Field via a Computed Expression.....	89
Figure 84: Computed Expressions—Entering a Computed Expression in an OUTPUT Transform.....	90
Figure 85: Computed Expressions—Entering a Computed Expression in an OUTPUT Transform Attached to a Field .....	90
Figure 86: Computed Expressions—Example of the Result of an OUTPUT Transform with a Computed Expression.....	90
Figure 87: Computed Expressions—A  Window  with a Computed Expression .....	91
Figure 88: Computed Expressions—Example of the Result of a  Window  with a Computed Expression .....	91
Figure 89: Statistics—Initial Print Dialog with Descriptive Statistics.....	136
Figure 90: Statistics—Generating Descriptive Statistics .....	137
Figure 91: Statistics—Initial Print Dialog for a Scattergram .....	138
Figure 92: Statistics—Generating Dialog and Sample Output of a Scattergram.....	139
Figure 93: Statistics—Initial Print Dialog for a Count Histogram .....	141
Figure 94: Statistics—Generating the Count Histogram Diagram .....	142
Figure 95: System Management—Example of Creating an ALTERNATE EDITOR File Entry	154
Figure 96: System Management—Example Where the User is Prompted to Choose an Alternate Editor .....	156

Figure 97: System Management—Example Where the User Selects to Compare Data Dictionaries .....	157
Figure 98: System Management—Example Namespace Compare File Entries.....	159
Figure 99: List File Attributes—File Attribute Listing Format Choices.....	162
Figure 100: List File Attributes—Choosing to Display the Brief Listing .....	162
Figure 101: List File Attributes—Example of a Brief Data Dictionary Listing .....	163
Figure 102: List File Attributes—Example of a Condensed Data Dictionary Listing.....	165
Figure 103: List File Attributes—Example of a Standard Data Dictionary Listing .....	168
Figure 104: List File Attributes—Choosing the Modified Standard Data Dictionary Listing	170
Figure 105: List File Attributes—Choosing the Custom-Tailored Data Dictionary Listing	171
Figure 106: List File Attributes—Choosing from a List of Field Attributes .....	172
Figure 107: List File Attributes—Help on Print Formatting in the Custom-Tailored Data Dictionary Listing .....	173
Figure 108: List File Attributes—Selecting the Field Attributes to Print.....	173
Figure 109: List File Attributes—Example of a Custom-Tailored Data Dictionary Listing	173
Figure 110: List File Attributes—Example of a Global Map Data Dictionary Listing.....	174
Figure 111: List File Attributes—Example of an Indexes and Cross-References Only Data Dictionary Listing .....	175
Figure 112: List File Attributes—Example of a Keys Only Data Dictionary Listing .....	176
Figure 113: List File Attributes—Example of the Dialog Encountered When Using the Map Pointer Relations Option .....	177
Figure 114: List File Attributes—Example of the Output Produced with the Map Pointer Relations Option .....	178
Figure 115: List File Attributes—Example of Dialog and Output Encountered When Using Check/Fix DD Structure Option .....	179
Figure 116: Data Dictionary Utilities—Example of Dialog and Output Encountered When Using the Find Pointers Into a File Option .....	181
Figure 117: List File Attributes—Example Setting Up the Meta Data Dictionary.....	182
Figure 118: List File Attributes—Example Meta Data Dictionary .....	183
Figure 119: Creating Files and Fields—Choosing Screen Mode When Using the Modify File Attributes Option.....	187
Figure 120: Creating Files and Fields—Example Using the Modify File Attributes Option in Screen Mode.....	187
Figure 121: Creating Files and Fields—Defining a DATA TYPE Field Value as DATE/TIME in Scrolling Mode (1 of 2) .....	190
Figure 122: Creating Files and Fields—Defining a DATA TYPE Field Value as DATE/TIME in Scrolling Mode (2 of 2) .....	191

Figure 123: Creating Files and Fields—Defining a DATA TYPE Field Value as NUMERIC in Scrolling Mode (1 of 2) .....	192
Figure 124: Creating Files and Fields—Defining a DATA TYPE Field Value as NUMERIC in Scrolling Mode (2 of 2) .....	192
Figure 125: Creating Files and Fields—Defining a DATA TYPE Field Value as SET OF CODES in Scrolling Mode .....	193
Figure 126: Creating Files and Fields—Defining a DATA TYPE Field Value as FREE TEXT in Scrolling Mode (1 of 3) .....	194
Figure 127: Creating Files and Fields—Defining a DATA TYPE Field Value as FREE TEXT in Scrolling Mode (2 of 3) .....	194
Figure 128: Creating Files and Fields—Defining a DATA TYPE Field Value as FREE TEXT in Scrolling Mode (3 of 3) .....	195
Figure 129: Creating Files and Fields—Defining a DATA TYPE Field Value as WORD-PROCESSING in Scrolling Mode .....	196
Figure 130: Creating Files and Fields—Defining a DATA TYPE Field Value as COMPUTED in Scrolling Mode (1 of 2) .....	197
Figure 131: Creating Files and Fields—Defining a DATA TYPE Field Value as COMPUTED in Scrolling Mode (2 of 2) .....	198
Figure 132: Creating Files and Fields—Defining a DATA TYPE Field Value as POINTER TO A FILE in Scrolling Mode (1 of 3) .....	198
Figure 133: Creating Files and Fields—Defining a DATA TYPE Field Value as POINTER TO A FILE in Scrolling Mode (2 of 3) .....	199
Figure 134: Creating Files and Fields—Defining a DATA TYPE Field Value as POINTER TO A FILE in Scrolling Mode (3 of 3) .....	199
Figure 135: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (1 of 5) .....	200
Figure 136: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (2 of 5) .....	200
Figure 137: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (3 of 5) .....	201
Figure 138: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (4 of 5) .....	201
Figure 139: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (5 of 5) .....	201
Figure 140: Creating Files and Fields—Example of Help Associated with a VARIABLE-POINTER Field .....	202
Figure 141: Creating Files and Fields—Example of “Sequencing” a Field .....	207
Figure 142: Creating Files and Fields—Creating a NUMBER (#.001) Field .....	207

Figure 143: Creating Files and Fields—Example of Creating a New File Entry with a .001 Field Defined .....	208
Figure 144: Creating Files and Fields—Looking Up an Entry in a File Using the IEN .....	208
Figure 145: Creating Files and Fields—Looking Up an Entry in a File Using the FMPATIENT,FIVE's IEN.....	208
Figure 146: Creating Files and Fields—Looking Up an Entry in a File Using the FMPATIENT,ONE's IEN .....	209
Figure 147: Editing a Field—LABEL, TITLE, and AUDIT Attributes .....	210
Figure 148: Editing a Field—ACCESS Privileges Attributes .....	210
Figure 149: Editing a Field—SOURCE, DESTINATION, GROUP Attributes.....	210
Figure 150: Editing a Field—DESCRIPTION Attributes .....	211
Figure 151: Editing a Field—DATA TYPE, LENGTH, PATTERN MATCH, MANDATORY 'HELP' PROMPT Attributes.....	211
Figure 152: Editing a Field—Adding Fields to a GROUP (1 of 2) .....	211
Figure 153: Editing a Field—Adding Fields to a GROUP (2 of 2) .....	212
Figure 154: Editing a Field—Deleting a Field and its Definition .....	213
Figure 155: Modify File Attributes Option—Creating a File .....	214
Figure 156: Modify File Attributes Option—Defining the NAME (#.01) Field in Screen Mode .....	215
Figure 157: Modify File Attributes Option—Editing a DATE/TIME Field in Screen Mode	215
Figure 158: Modify File Attributes Option—Defining a DATA TYPE Field as DATE/TIME in Screen Mode .....	216
Figure 159: Modify File Attributes Option—Editing a SET OF CODES Field in Screen Mode.....	217
Figure 160: Modify File Attributes Option—Defining a DATA TYPE Field as SET OF CODES in Screen Mode .....	217
Figure 161: Modify File Attributes Option—Editing a FREE TEXT Field in Screen Mode	218
Figure 162: Modify File Attributes Option—Defining a Data Type as FREE TEXT in Screen Mode.....	218
Figure 163: Modify File Attributes Option—Caret (^) in a FREE TEXT Field: Piece Position	219
Figure 164: Modify File Attributes Option—Editing a WORD-PROCESSING Field in Screen Mode.....	220
Figure 165: Modify File Attributes Option—Defining a DATA TYPE Field as WORD-PROCESSING in Screen Mode.....	220
Figure 166: Modify File Attributes Option—Editing a COMPUTED Field in Screen Mode	221
Figure 167: Modify File Attributes Option—Defining a DATA TYPE Field as COMPUTED in Screen Mode .....	222

Figure 168: Modify File Attributes Option—Editing a POINTER TO A FILE Field in Screen Mode .....	223
Figure 169: Modify File Attributes Option—Defining a DATA TYPE Field as POINTER TO A FILE in Screen Mode .....	223
Figure 170: Modify File Attributes Option—Editing a VARIABLE-POINTER Field in Screen Mode .....	224
Figure 171: Modify File Attributes Option—Defining a DATA TYPE Field as VARIABLE-POINTER in Screen Mode .....	224
Figure 172: Addition/Editing of a Field of Data Type BOOLEAN .....	225
Figure 173: Addition/Editing of a Field of Data Type LABEL REFERENCE.....	226
Figure 174: Addition/Editing of a Field of Data Type TIME .....	227
Figure 175: Addition/Editing of a Field of Data Type YEAR.....	228
Figure 176: Addition/Editing of a Field of Data Type UNIVERSAL TIME.....	229
Figure 177: Addition/Editing of a Field of Data Type FT POINTER .....	230
Figure 178: Addition/Editing of a Field of Data Type FT DATE.....	231
Figure 179: Addition/Editing of a Field of Data Type RATIO.....	232
Figure 180: Modify File Attributes Option—Creating a Multiple in Screen Mode .....	233
Figure 181: Modify File Attributes Option—Defining a DATA TYPE Field as a NUMERIC Multiple in Screen Mode.....	233
Figure 182: Modify File Attributes Option—Editing a Multiple’s Subfield in Screen Mode.....	234
Figure 183: Modify File Attributes Option—Reviewing/Editing the Properties of a Multiple Data Type Field in Screen Mode.....	234
Figure 184: Modify File Attributes Option—Example of a .01 Subfield of a Multiple.....	235
Figure 185: Modify File Attributes Option—Defining a Data Type Field as a NUMERIC Subfield in Screen Mode .....	235
Figure 186: File Utilities—Editing a Traditional Cross-Reference (1 of 2).....	240
Figure 187: File Utilities—Editing a Traditional Cross-Reference (2 of 2).....	241
Figure 188: File Utilities—Creating a Traditional Cross-Reference .....	241
Figure 189: File Utilities—Deleting a Traditional Cross-Reference .....	242
Figure 190: File Utilities—Editing a New-Style Cross-Reference .....	245
Figure 191: File Utilities—Editing a New-Style Cross-Reference in Screen Mode.....	246
Figure 192: File Utilities—Creating a New-Style Cross-Reference .....	247
Figure 193: File Utilities—Creating a New-Style Cross-Reference in Screen Mode.....	248
Figure 194: File Utilities—Deleting a New-Style Cross-Reference .....	249
Figure 195: File Utilities—Example of Setting a Field as an Identifier.....	250
Figure 196: File Utilities—Example of an Identifier Field Displayed When Doing a Lookup.....	251

Figure 197: File Utilities—Example of a Subfield as an Identifier .....	251
Figure 198: File Utilities—Deleting an Identifier Field .....	251
Figure 199: File Utilities—Sample Dialog When Re-Indexing a File .....	252
Figure 200: File Utilities—Choosing the Edit File Option.....	254
Figure 201: File Utilities—Using the Edit File Option in Screen Mode .....	255
Figure 202: File Utilities—Example of Creating an OUTPUT Transform .....	258
Figure 203: File Utilities—Example of the <i>First</i> Screen of a PRINT Template .....	260
Figure 204: File Utilities—Editing a PRINT Template’s Properties in Screen Mode ( <i>First</i> Screen) .....	260
Figure 205: File Utilities—Editing a PRINT Template’s Properties in Screen Mode ( <i>Second</i> Screen) .....	261
Figure 206: File Utilities—Example of a SORT Template ( <i>First</i> Screen).....	262
Figure 207: File Utilities—Editing a SORT Template’s Properties in Screen Mode ( <i>First</i> Screen) .....	262
Figure 208: File Utilities—Editing a SORT Template’s Properties in Screen Mode ( <i>Second</i> Screen) .....	263
Figure 209: File Utilities—Mandatory/Required Field Check Report.....	264
Figure 210: File Utilities—Creating a Key .....	265
Figure 211: File Utilities—Creating a Key in Screen Mode.....	266
Figure 212: File Utilities—Creating the Uniqueness Index Automatically .....	266
Figure 213: File Utilities—Resolving a Conflict with the Key Fields and Uniqueness Index	267
Figure 214: File Utilities—Editing a Key .....	267
Figure 215: File Utilities—Deleting a Key .....	268
Figure 216: File Utilities—Verifying a Key.....	269
Figure 217: Auditing—Audit Options .....	270
Figure 218: Auditing—Example of a Data Field Audit.....	272
Figure 219: Auditing—Turning a Data Audit On .....	274
Figure 220: Auditing—Turning a Data Audit Off.....	275
Figure 221: Auditing—CAPTIONED Output with Audit Trail.....	275
Figure 222: Auditing—AUDIT File: Query.....	276
Figure 223: Auditing—AUDIT File: Output.....	277
Figure 224: Auditing—Sample Listing Showing Fields Flagged for Auditing .....	278
Figure 225: Auditing—Choosing to Purge Only <i>Selected</i> Data Audit Records .....	279
Figure 226: Auditing—Listing Internal Entry Numbers for Data Audit Fields for Possible Purging .....	280
Figure 227: Auditing—Purging <i>Selected</i> Audit Records from a File .....	281

Figure 228: Auditing—Purging <i>All</i> Audit Records from a File .....	281
Figure 229: Auditing—Choosing to Review a Data Dictionary Audit.....	282
Figure 230: Auditing—Specifying a Data Dictionary Audit.....	282
Figure 231: Auditing—Reviewing a Data Dictionary Audit.....	283
Figure 232: Auditing—Reviewing DD Changes for Time Period.....	284
Figure 233: Auditing—Purging Selected Data Dictionary Audit Records.....	285
Figure 234: Auditing—Purging All Data Dictionary Audit Records .....	286
Figure 235: Auditing—Auditable Word Processing Fields .....	286
Figure 236: Auditing—Uneditable Data.....	287
Figure 237: Auditing—Sample User Access Report.....	288
Figure 238: Transferring File Entries—Transferring Data within a File .....	295
Figure 239: Transferring File Entries—Example Displaying Two Records in a File <i>Prior</i> to a Transfer.....	296
Figure 240: Transferring File Entries—Initiating a Transfer of File Entries.....	296
Figure 241: Transferring File Entries—Results <i>after</i> a Transfer of File Entries .....	297
Figure 242: Transferring File Entries—Transferring Entries from one File to Another .....	298
Figure 243: Transferring File Entries—Selecting Specific Entries for Transfer .....	299
Figure 244: Transferring File Entries—Using the Transfer File Entries Option to Create a New File.....	299
Figure 245: Transferring File Entries—Selecting Entries to Compare in a File (1 of 2).....	300
Figure 246: Transferring File Entries—Selecting Entries to Compare in a File (2 of 2).....	301
Figure 247: Transferring File Entries—Comparison Output.....	301
Figure 248: Transferring File Entries—Merging Entries in a File .....	302
Figure 249: Transferring File Entries—Choosing which File Entry will Serve as the Default Entry.....	302
Figure 250: Transferring File Entries—Deleting the “Merged From” File Entry .....	303
Figure 251: Transferring File Entries—Setting Up the Merge Output.....	303
Figure 252: Transferring File Entries—Merge Output (1 of 2) .....	304
Figure 253: Transferring File Entries—Merge Output (2 of 2) .....	305
Figure 254: Transferring File Entries—Merge Options .....	305
Figure 255: Transferring File Entries—Merge PROCEED Option.....	306
Figure 256: Transferring File Entries—Merge SUMMARIZE Option.....	307
Figure 257: Extract Tool—Extract Data To Fileman File [DIAX EXTRACT MENU] Menu Options.....	314
Figure 258: Extract Tool—Search, Sort, and Print Options When Selecting Entries to Extract .....	316

Figure 259: Extract Tool—Select Entries to Extract Output.....	316
Figure 260: Extract Tool—Example of a Notice Regarding an Outstanding Extract Activity.....	316
Figure 261: Extract Tool—Using the ADD/DELETE SELECTED ENTRIES Option .....	317
Figure 262: Extract Tool—Using the PRINT SELECTED ENTRIES Option.....	318
Figure 263: Extract Tool—PRINT SELECTED ENTRIES Option Output.....	319
Figure 264: Extract Tool—Using the MODIFY DESTINATION FILE Option (1 of 2).....	319
Figure 265: Extract Tool—Using the MODIFY DESTINATION FILE Option (2 of 2).....	320
Figure 266: Extract Tool—Using the CREATE EXTRACT TEMPLATE Option .....	321
Figure 267: Extract Tool—Example of a Notice Regarding a Discrepancy.....	322
Figure 268: Extract Tool—Example of the Warning Message When the Validation Check Fails .....	322
Figure 269: Extract Tool—Using the UPDATE DESTINATION FILE Option .....	323
Figure 270: Extract Tool—Exception Report.....	324
Figure 271: Extract Tool—Example of a Notice from VA FileMan When Extract Activity Lacks the Status UPDATED DESTINATION FILE .....	325
Figure 272: Extract Tool—Using the PURGE EXTRACTED ENTRIES Option (1 of 2) .....	325
Figure 273: Extract Tool—Using the PURGE EXTRACTED ENTRIES Option (2 of 2) .....	325
Figure 274: Extract Tool—Using the CANCEL EXTRACT SELECTION Option.....	326
Figure 275: Extract Tool—Using the VALIDATE EXTRACT TEMPLATE Option.....	327
Figure 276: Filegrams [DIFG] Menu Options.....	329
Figure 277: Filegrams—Creating a FILEGRAM Template (1 of 3) .....	331
Figure 278: Filegrams—Creating a FILEGRAM Template (2 of 3) .....	332
Figure 279: Filegrams—Creating a FILEGRAM Template (3 of 3) .....	332
Figure 280: Filegrams—FILEGRAM Template Output.....	333
Figure 281: Filegrams—Example of Creating a Specifier (1 of 2) .....	334
Figure 282: Filegrams—Example of Creating a Specifier (2 of 2) .....	334
Figure 283: Filegrams—Deleting a Specifier .....	334
Figure 284: Filegrams—Example of Generating a Filegram.....	335
Figure 285: Filegrams—Example of a Filegram Received and Forwarded .....	336
Figure 286: Filegrams—Example of a Simple Filegram ( <i>without</i> Pointers).....	337
Figure 287: Filegrams—Deleting a Filegram .....	338
Figure 288: Archiving—Options.....	341
Figure 289: Archiving—Example of Selecting Entries to Archive.....	343
Figure 290: Archiving—Example of a Notice Regarding an Outstanding Archiving Activity.....	344
Figure 291: Archiving—Example of Adding an Entry to the Archival Activity.....	345
Figure 292: Archiving—Printing an Archival Activity in a Regular Format .....	346

Figure 293: Archiving—Printing an Archival Activity in a Filegram Format.....	346
Figure 294: Archiving—Example of Creating a FILEGRAM ARCHIVING Template .....	348
Figure 295: Archiving—Example of Writing Entries to <i>Temporary Storage</i> .....	349
Figure 296: Archiving—Example of Moving Archived Data to <i>Permanent Storage</i> .....	350
Figure 297: Archiving—Example of an Archive Activity Report .....	351
Figure 298: Archiving—Example of a Notice from VA FileMan When Purging without Archiving Data .....	352
Figure 299: Archiving—Example of Purging Permanently Archived Data.....	352
Figure 300: Archiving—VA FileMan Notifies You of the Number of Entries Purged.....	352
Figure 301: Archiving—Canceling an Archival Activity.....	353
Figure 302: Archiving—Example of Finding Archived Entries.....	354
Figure 303: Meta Data Dictionary—Sample Entry for File #200, Field #.01 .....	357
Figure 304: FILELIST^DDD API—Example .....	357
Figure 305: Example of the JSON Object Entry Parameter .....	359
Figure 306: Example of a JSON Object Formatted Update Data File .....	361

## List of Tables

Table 1: Documentation Symbol Descriptions .....	xxx
Table 2: Import and Export Tools—Foreign Format Field Prompts .....	13
Table 3: Import and Export Tools—Allowable Sort Qualifiers When Exporting Data .....	16
Table 4: Relational Navigation—Relational Jumps that Correspond to Extended Pointer Syntax.....	51
Table 5: Advanced Edit Techniques—Edit Qualifiers.....	71
Table 6: Advanced Edit Techniques—Text Formatting Expressions in Word-Processing Windows .....	75
Table 7: Computed Expressions—Unary Operators .....	79
Table 8: Computed Expressions—Binary Operators.....	80
Table 9: Computed Expressions—Boolean Operators.....	80
Table 10: Computed Expressions—Example Indicating Possible Results of Computed Expression Based on Different Entries to “Totaling” Prompt.....	86
Table 11: VA FileMan Functions—Documentation Conventions.....	93
Table 12: VA FileMan Functions—By Category .....	94
Table 13: VA FileMan Functions—Date/Time Function: BETWEEN.....	96
Table 14: VA FileMan Functions—Date/Time Function: DATE.....	97
Table 15: VA FileMan Functions—Date/Time Function: DAYOFWEEK.....	97
Table 16: VA FileMan Functions—Date/Time Function: MID.....	98
Table 17: VA FileMan Functions—Date/Time Function: MINUTES .....	98
Table 18: VA FileMan Functions—Date/Time Function: MONTH.....	99
Table 19: VA FileMan Functions—Date/Time Function: MONTHNAME .....	99
Table 20: VA FileMan Functions—Date/Time Function: NOON.....	100
Table 21: VA FileMan Functions—Date/Time Function: NOW .....	100
Table 22: VA FileMan Functions—Date/Time Function: NUMDATE.....	101
Table 23: VA FileMan Functions—Date/Time Function: NUMDATE4.....	101
Table 24: VA FileMan Functions—Date/Time Function: NUMDAY .....	101
Table 25: VA FileMan Functions—Date/Time Function: NUMMONTH .....	102
Table 26: VA FileMan Functions—Date/Time Function: NUMYEAR .....	102
Table 27: VA FileMan Functions—Date/Time Function: NUMYEAR4.....	102
Table 28: VA FileMan Functions—Date/Time Function: RANGEDATE .....	103
Table 29: VA FileMan Functions—Date/Time Function: TIME .....	103
Table 30: VA FileMan Functions—Date/Time Function: TODAY .....	104
Table 31: VA FileMan Functions—Date/Time Function: YEAR.....	104

Table 32: VA FileMan Functions—Environmental Function: BREAKABLE .....	105
Table 33: VA FileMan Functions—Environmental Function: CLOSE.....	105
Table 34: VA FileMan Functions—Environmental Function: SITENUMBER.....	106
Table 35: VA FileMan Functions—Environmental Function: USER.....	106
Table 36: VA FileMan Functions—File and File Data Function: COUNT .....	107
Table 37: VA FileMan Functions—File and File Data Function: DUPLICATED .....	108
Table 38: VA FileMan Functions—File and File Data Function: FILE .....	109
Table 39: VA FileMan Functions—File and File Data Function: INTERNAL.....	109
Table 40: VA FileMan Functions—File and File Data Function: LAST .....	110
Table 41: VA FileMan Functions—File and File Data Function: MAXIMUM.....	111
Table 42: VA FileMan Functions—File and File Data Function: MINIMUM.....	112
Table 43: VA FileMan Functions—File and File Data Function: nTH.....	113
Table 44: VA FileMan Functions—File and File Data Function: NEXT .....	114
Table 45: VA FileMan Functions—File and File Data Function: PREVIOUS.....	114
Table 46: VA FileMan Functions—File and File Data Function: TOTAL.....	115
Table 47: VA FileMan Functions—Mathematical Function: ABS .....	115
Table 48: VA FileMan Functions—Mathematical Function: BETWEEN.....	116
Table 49: VA FileMan Functions—Mathematical Function: MAX .....	116
Table 50: VA FileMan Functions—Mathematical Function: MIN .....	117
Table 51: VA FileMan Functions—Mathematical Function: MODULO.....	117
Table 52: VA FileMan Functions—Mathematical Function: SQUAREROOT .....	118
Table 53: VA FileMan Functions—Printing Related Function: IOM.....	118
Table 54: VA FileMan Functions—Printing Related Function: PAGE .....	119
Table 55: VA FileMan Functions—String Function: DUP.....	119
Table 56: VA FileMan Functions—String Function: LOWERCASE .....	120
Table 57: VA FileMan Functions—String Function: PADRIGHT .....	120
Table 58: VA FileMan Functions—String Function: REPLACE.....	121
Table 59: VA FileMan Functions—String Function: REVERSE.....	121
Table 60: VA FileMan Functions—String Function: STRIPBLANKS.....	122
Table 61: VA FileMan Functions—String Function: TRANSLATE.....	122
Table 62: VA FileMan Functions—String Function: UPPERCASE.....	123
Table 63: VA FileMan Functions—Temporary Data Storage Function: PARAM.....	123
Table 64: VA FileMan Functions—Temporary Data Storage Function: SETPARAM.....	124
Table 65: VA FileMan Functions—Temporary Data Storage Function: VAR.....	124
Table 66: VA FileMan Functions—Temporary Data Storage Function: SET .....	125
Table 67: VA FileMan Functions—M-Related Function: \$A[SCII].....	125

Table 68: VA FileMan Functions—M-Related Function: \$C[HAR].....	126
Table 69: VA FileMan Functions—M-Related Function: \$E[XTRACT] .....	126
Table 70: VA FileMan Functions—M-Related Function: \$F[IND] .....	127
Table 71: VA FileMan Functions—M-Related Function: \$H[OROLOG].....	127
Table 72: VA FileMan Functions—M-Related Function: \$I[O].....	128
Table 73: VA FileMan Functions—M-Related Function: \$J[OB].....	128
Table 74: VA FileMan Functions—M-Related Function: \$J[USTIFY].....	129
Table 75: VA FileMan Functions—M-Related Function: \$L[ENGTH].....	130
Table 76: VA FileMan Functions—M-Related Function: \$P[IECE] .....	131
Table 77: VA FileMan Functions—M-Related Function: \$R[ANDOM] .....	132
Table 78: VA FileMan Functions—M-Related Function: \$S[ELECT].....	132
Table 79: VA FileMan Functions—M-Related Function: \$S[TORAGE] .....	133
Table 80: VA FileMan Functions—M-Related Function: \$X .....	133
Table 81: VA FileMan Functions—M-Related Function: \$Y .....	134
Table 82: Statistics—Descriptive Statistics Qualifiers .....	136
Table 83: Statistics—Histogram Qualifiers .....	140
Table 84: System Management—%ZIS Variables Returned .....	145
Table 85: System Management—%ZISS Variables Returned.....	146
Table 86: System Management—Optimal Procedures for Screen-Oriented Utilities: Based on Terminal Type .....	149
Table 87: System Management—NEW PERSON (#200) File Fields that Enhance Standalone VA FileMan .....	150
Table 88: System Management—NEW PERSON (#200) File Fields to Define Key Variables in VA FileMan .....	151
Table 89: System Management—Description of the ^%ZOSF Nodes .....	152
Table 90: List File Attributes—Condensed Data Dictionary Codes .....	166
Table 91: Creating Files and Fields—Data Types.....	189
Table 92: File Utilities—Traditional Cross-references.....	239
Table 93: File Utilities—X, X1, and X2 Arrays.....	244
Table 94: Auditing—“AUDIT” Prompt Response .....	272
Table 95: Data Security—File Access Codes.....	290
Table 96: Data Security—Field Access Codes .....	292
Table 97: Extract Tool—DATA TYPE Field Value Recommendations.....	311
Table 98: META DATA DICTIONARY (#.9) File Fields.....	356

## Orientation

### What is VA FileMan?

VA FileMan is the database management system for the Veterans Health Information Systems and Technology Architecture user (VistA) environment. VA FileMan creates and maintains a database management system that includes features such as:

- Report writer
- Data dictionary manager
- Scrolling and screen-oriented data entry
- Text editors
- Programming utilities
- Tools for sending data to other systems
- File archiving

VA FileMan can be used as a:

- Standalone database
- Set of interactive or "silent" routines
- Set of application utilities, in all modes

It is used to define, enter, and retrieve information from a set of computer-stored files, each of which is described by a data dictionary.

VA FileMan is a public domain software package that is developed and maintained by the Department of Veterans Affairs (VA). It is widely used by VA medical centers and in clinical, administrative, and business settings in this country and abroad.



**CAUTION: Programmer access in VistA is defined as DUZ(0)="@" . It grants the privilege to become a developer in VistA. Programmer access allows you to work outside many of the security controls enforced by VA FileMan, enables access to all VA FileMan files, access to modify data dictionaries, etc. It is important to *proceed with caution* when having access to the system in this way.**

## How to Use this Manual

The VA FileMan User Manual is comprised of two separate documents that describe the VA FileMan functionality of VistA's database management system:

- The *VA FileMan Advanced User Manual* (this manual) shows how to use the features of VA FileMan that are likely to be used by experienced VistA users. It introduces advanced VA FileMan concepts and shows you how to use VA FileMan's advanced tools. It describes features that are more likely to be used by:
  - Automated Data Processing Application Coordinators (ADPACs)
  - System Administrators
  - Other technical users
- The *VA FileMan Advanced User Manual* introduces basic VA FileMan concepts. It shows you how to use:
  - VA FileMan's basic tools for displaying and editing data.
  - VA FileMan features that are used in most VistA applications, which are used by all VistA users.



**NOTE:** These documents are available in Microsoft Word (.docx), Adobe Acrobat Portable Document Format (PDF), and Hypertext Markup Language (HTML) format (see the "[HTML Manuals](#)" section).

In this manual, the following major features of VA FileMan are introduced along with a description on how to use them:

- [Menus and Options.](#)
- [Import and Export Tools.](#)
- [Relational Navigation.](#)
- [Advanced Edit Techniques.](#)
- [Computed Expressions.](#)
- [VA FileMan Functions.](#)
- [Statistics.](#)
- [System Management.](#)

- [List File Attributes.](#)
- [Creating Files and Fields.](#)
- [File Utilities.](#)
- [Auditing.](#)
- [Data Security.](#)
- [Transferring File Entries.](#)
- [Extract Tool.](#)
- [Filegrams.](#)
- [Archiving.](#)
- [Meta Data Dictionary.](#)
- [Data Synchronization.](#)



**REF:** For VA FileMan installation instructions in the VistA environment see the *VA FileMan Installation Guide* and any national patch description of the patch being released.

## HTML Manuals

Why produce an HTML (Hypertext Markup Language) edition of the VA FileMan User Manual?

- The HTML versions of the VA FileMan manuals are useful as online documentation support as you use VA FileMan. HTML manuals allow you to instantly jump (link) to specific topics or references online.
- The VA FileMan HTML manuals are “living” documents that are continuously updated with the most current VA FileMan information (unlike paper or printed documentation). They are updated based on new versions, patches, or enhancements to VA FileMan.
- Presenting manuals in an HTML format on a Web server also gives new opportunities, such as accessing embedded multimedia training material (e.g., movies) directly in the manuals themselves.
- Manuals are accessible over the VA Intranet network.

## Intended Audience

The intended audience of this manual is all key stakeholders. The stakeholders include the following:

- Automated Data Processing Application Coordinators (ADPACs)
- System Administrators—System administrators at Department of Veterans Affairs (VA) sites who are responsible for computer management and system security on the VistA M Servers.
- Product Delivery Service (PDS)—VistA legacy development teams.
- 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 and/or modified freely provided that any derivative works bear some notice that they are derived from it, and any modified versions bear some notice that they have been 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 VA FileMan, and the functionality contained in VA FileMan 22.0; however, no attempt is made to explain how the overall VistA programming system is integrated and maintained. Such methods and procedures are documented elsewhere. We suggest you look at the various VA Internet and Intranet Websites for a general orientation to VistA. For example, visit the Office of Information and Technology (OIT) VistA Development 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 inform the reader to take special notice of critical information.
	<b>TIP:</b> Used to inform the reader of helpful tips or tricks they can use when working with VA FileMan.

- 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 Kernel (**DI** or **FM**) test patient and user names would be documented as follows:

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

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

- "Snapshots" of computer online displays (i.e., screen captures/dialogs) and computer source code, if any, are shown in a *non*-proportional font and enclosed within a box.
  - User's responses to online prompts are **bold** typeface and highlighted in yellow (e.g., **<Enter>**).
  - Emphasis within a dialog box is **bold** typeface and highlighted in blue (e.g., **STANDARD LISTENER: RUNNING**).
  - Some software code reserved/key words are **bold** typeface 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.

- 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., **DIEXTRACT**).



**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).

## Internal Word Navigation Links Setup Steps

This document uses Microsoft® Word's built-in navigation for internal hyperlinks. To add **Back** and **Forward** navigation buttons to your toolbar for all Word documents, see the *Tech Writer Tips: Internal Word Navigation Links Setup* 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 topic.

**REF:** For further information, see the *VA FileMan Technical Manual*.

## 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 the 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.

## Assumptions

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

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

## Reference Materials

Readers who wish to learn more about VA FileMan should consult the following documents:

- *VA FileMan Release Notes*
- *VA FileMan Installation Guide*
- *VA FileMan Technical Manual*
- *VA FileMan User Manual* (PDF and HTML format)
- *VA FileMan Advanced User Manual* (this manual; PDF and HTML format)
- *VA FileMan Developer's Guide* (PDF and HTML format)



**REF:** Zip files of the VA FileMan documentation in HTML format are available upon request.

Using a Web browser, open the **HTML** documents "table of contents" page

(i.e., **index.aspx**). The *VA FileMan User Manual*, the *VA FileMan Advanced User Manual*, and the *VA FileMan Developer's Guide* are all linked together.

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](#).

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



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

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

# 1 Introduction

Department of Veterans Affairs (VA) FileMan is the Veterans Health Information Systems and Technology Architecture (VistA) MUMPS (M)-based Database Management System (DBMS). The central component that defines the way standard VistA files are structured and manipulated. Most of the Veterans Health Administration (VHA) clinical data is stored in VA FileMan files and is retrieved and accessed through VA FileMan application programming interfaces (APIs) and user interfaces. VA FileMan runs in any American National Standards Institute (ANSI) M environment.

This is the *VA FileMan Advanced User Manual*. It includes additional VA FileMan functionality not described in the *VA FileMan User Manual*.

## 1.1 Menus and Options

### 1.1.1 VA FileMan Main Menu

The **VA FileMan** [DIUSER] main menu contains the following submenus and options:

**Figure 1: VA FileMan [DIUSER] Main Menu**

```
SELECT VA FILEMAN <TEST ACCOUNT> OPTION: ?? <ENTER>

ENTER OR EDIT FILE ENTRIES           [DIEDIT]
PRINT FILE ENTRIES                    [DIPRINT]
SEARCH FILE ENTRIES                   [DISEARCH]
MODIFY FILE ATTRIBUTES                 [DIMODIFY]
INQUIRE TO FILE ENTRIES               [DIINQUIRE]
UTILITY FUNCTIONS ...                  [DIUTILITY]
DATA DICTIONARY UTILITIES ...          [DI DDU]
TRANSFER ENTRIES                       [DITRANSFER]
OTHER OPTIONS ...                       [DIOOTHER]
```

## 1.1.2 Utility Functions Menu

Figure 2: VA FileMan—Utility Functions Menu Options

```
SELECT UTILITY FUNCTIONS <TEST ACCOUNT> OPTION: ?? <ENTER>

  VERIFY FIELDS                                [DIVERIFY]
  CROSS-REFERENCE A FIELD                       [DIXREF]
  IDENTIFIER                                   [DIIDENT]
  RE-INDEX FILE                               [DIRDEX]
  INPUT TRANSFORM (SYNTAX)                   [DIITRAN]
  EDIT FILE                                   [DIEDFILE]
  OUTPUT TRANSFORM                           [DIOTRAN]
  TEMPLATE EDIT                               [DITEMP]
  UNEDITABLE DATA                           [DIUNEDIT]
  MANDATORY/REQUIRED FIELD CHECK            [DIFIELD CHECK]
  KEY DEFINITION                             [DIKEY]
```

## 1.1.3 Data Dictionary Utilities Functions Menu

Figure 3: VA FileMan—Data Dictionary Utilities Menu Options

```
SELECT DATA DICTIONARY UTILITIES <TEST ACCOUNT> OPTION: ?? <ENTER>

  LIST FILE ATTRIBUTES                        [DILIST]
  MAP POINTER RELATIONS                      [DI DDMAP]
  CHECK/FIX DD STRUCTURE                    [DI DDUCHK]
  FIND POINTERS INTO A FILE                 [DDU FIND POINTERS INTO A FILE]
  UPDATE THE META DATA DICTIONARY         [DDU UPDATE META DD]
```

## 1.1.4 Other Options Menu

Figure 4: VA FileMan—Other Options Menu Options

```
SELECT OTHER OPTIONS <TEST ACCOUNT> OPTION: ?? <ENTER>

FILEGRAMS ... [DIFG]
  **> LOCKED WITH XUFILEGRAM
AUDIT MENU ... [DIAUDIT]
  **> LOCKED WITH XUAUDITING
SCREENMAN ... [DDS SCREEN MENU]
  **> LOCKED WITH XUSCREENMAN
STATISTICS [DISTATISTICS]
VA FILEMAN MANAGEMENT ... [DI MGMT MENU]
  **> LOCKED WITH XUMGR
DATA EXPORT TO FOREIGN FORMAT ... [DDXP EXPORT MENU]
EXTRACT DATA TO FILEMAN FILE ... [DIAX EXTRACT MENU]
  **> LOCKED WITH DIEXTRACT
IMPORT DATA [DDMP IMPORT]
BROWSER [DDBROWSER]
DATA ACCESS CONTROL ... [DIACCESS]
DATA MAPPING ... [DDE ENTITY MAPPING]
```

## 2 Import and Export Tools

If you want to use an application like Microsoft® Excel to manipulate data stored in a VA FileMan file, you need some way to exchange that data between VA FileMan and your application. VA FileMan provides the Import and Export Tools for this purpose.

Suppose, for example, that you want to use Microsoft® Word's Print Merge utility to print a form letter to a list of recipients that is maintained in a VA FileMan file. You can use VA FileMan's **Export Tool** to export the list of recipients from the VA FileMan file to Microsoft® Word. Once you have done this, you can use Word to generate your form letters based on the exported list.

### 2.1 What Applications Can You Exchange Data With?

In theory, you can exchange data with any application that supports delimited or fixed-length ASCII data exchange. Many applications do, using a variety of formats. Typically, you can expect the ability to import and export data with the following types of applications:

- Databases
- Spreadsheets
- Statistical and Analysis Programs (SAS, SPSS, etc.)
- Vertical Applications
- Word Processor (data records, *not* word-processing text)



**NOTE:** You can export data records to a word-processor, which often uses data records for functions such as print merges. You *cannot* use the Import or Export Tools to exchange **WORD-PROCESSING** fields from VA FileMan files, however.

## 2.2 How Data is Moved between Applications

Movement of data between applications that do *not* “speak the same language” is a complex process, because it involves coordinating activities in different computer applications and often in multiple computing environments.

VA FileMan’s **Import** and **Export Tools** use ASCII data exchange. It is the oldest and most widely supported way of exchanging data between applications. Data for a particular record or group of records can be transported in one of two standard formats:

- Delimited
- Fixed-length

To **export** data from a VA FileMan file, use the **Export Tool** to create an ASCII data file containing exported records. The exported data is formatted in such a way that it can be recognized by a particular foreign application. The ASCII data file can then be imported into the foreign application.

To **import** data to a VA FileMan file, use your foreign application to generate an ASCII data file containing records in either delimited or fixed-length formats. Then use the **Import Tool** to load those records into the VA FileMan file you specify.

## 2.3 Dependency on Correct Data Communication

For import or export of data to succeed, the data *must* be passed correctly on all communication pathways between VA FileMan and the foreign application. A glitch in the communication of data can cause data exchange to fail.

For example, suppose the foreign application expects the fields in records you are exporting to be separated (or “delimited”) by the **Tab** character (<**TAB**>). The Export Tool can output a <**TAB**> between each field’s data value. However, if you use a communication program’s screen capture facility to create a file of the exported data and if that communication program automatically changes <**TAB**>s into a certain number of spaces to align text, the exported data is corrupted, and the import fails.

You should be familiar with your importing or exporting application and with any communications programs that you are using. Knowledge of all the applications involved, starting with VA FileMan and its Import and Export Tools, increases the likelihood of a successful transfer of data.

## 2.3.1 Data Formats

### 2.3.1.1 Delimited Data Format

Suppose you have a record with the following data:

- LASTNAME = "FMPATIENT"
- FIRSTNAME = "ONE"
- AGE = "36"

In delimited data format, you choose a delimiter character to place between field values. For this example, use a comma (,) as the delimiter character.

A comma (,) is then inserted between each of a record's fields, to "delimit" them. The resulting record, exported in comma-delimited format, would look like [Figure 5](#):

**Figure 5: Import and Export Tools—Example of a Record Delimited by a Comma**

```
FMPATIENT,ONE,36
```

Groups of records are exported line-by-line, one line after another. A file of records in comma-delimited format might look like [Figure 6](#):

**Figure 6: Import and Export Tools—Example of a File with Records Delimited by a Comma**

```
FMPATIENT,TWO,1 GREEN LANE, , ,AMHERST,NH,99999  
FMPATIENT,THREE,0 PLAZA COURT, , ,SAN FRANCISCO,CA,99999  
FMPATIENT,FOUR,0 123RD ST. , , ,SAN FRANCISCO,CA,99999
```

To use delimited data format, both applications (the exporting application and the importing application) *must* be able to recognize the format.

### 2.3.1.1.1 Quoted Fields in Delimited Format

Now, suppose in the previous example that instead of two separate fields for LASTNAME and FIRSTNAME, there is only a single NAME field for both. Suppose that incoming data you want to place in the single NAME field comes in the form FMPATIENT,FOUR, but you still want to use commas as your delimiter. You can use the “Fields Quoted” setting in the Import form (or the Quote Non-Numeric Fields setting in a Foreign Format) to ignore the delimiter if it is between quotes in the incoming data. Thus, if you set “Fields Quoted” to **YES** in your import form, and you pass in a record that looks like [Figure 7](#):

**Figure 7: Import and Export Tools—Example of a Record Where the Delimiter between Quotes is Ignored**

□FMPATIENT,FOUR□,0 123RD ST.,,,SAN FRANCISCO,CA,99999
---

For quoted fields, like “**FMPATIENT,FOUR**”, the Import Tool ignores the comma delimiter between the quotes and treats “**FMPATIENT,FOUR**” as a single field value.

### 2.3.1.2 Fixed-Length Data Format

In fixed-length data format, a standard width is expected for each field in the record. For example, suppose you have a record with the following data:

- LASTNAME = “**FMPATIENT**”; **25 characters** for LASTNAME.
- FIRSTNAME = “**ONE**”; **20 characters** for FIRSTNAME.
- AGE = “**36**”; **3 characters** for AGE.

The resulting record, exported in fixed-length format, would look like [Figure 8](#):

**Figure 8: Import and Export Tools—Example of a Fixed-Length Record**

FMPATIENT	ONE	36
-----------	-----	----

Groups of records are exported line-by-line, one line after another. A file of records in fixed-length format might look like [Figure 9](#):

**Figure 9: Import and Export Tools—Example of a File with Fixed-Length Records**

FMPATIENT	TWO	29
FMPATIENT	THREE	47
FMPATIENT	FOUR	38

To use fixed-length data format, both applications (the exporting application and the importing application) *must* be able to recognize the format.

### 2.3.2 How to Export Data

The menu in [Figure 10](#) shows the export data options, which are located under the **Other Options** [DIOTHER] menu:

**Figure 10: Import and Export Tools—Data Export Options**

VA FILEMAN ...	[DIUSER]
OTHER OPTIONS ...	[DIOTHER]
DATA EXPORT TO FOREIGN FORMAT ...	[DDXP EXPORT MENU]
DEFINE FOREIGN FILE FORMAT	[DDXP DEFINE FORMAT]
**> LOCKED WITH DDXP-DEFINE	
SELECT FIELDS FOR EXPORT	[DDXP SELECT EXPORT FIELDS]
CREATE EXPORT TEMPLATE	[DDXP CREATE EXPORT TEMPLATE]
EXPORT DATA	[DDXP EXPORT DATA]
PRINT FORMAT DOCUMENTATION	[DDXP FORMAT DOCUMENTATION]



**NOTE:** The **Export Data** option [DDXP EXPORT DATA] is used to stream data to external devices or files. It is *not* designed to print clearly to the screen.

If you know how to print file entries, you already know most of the procedures to export file entries. The Export Tool is based on the standard VA FileMan **Print File Entries** [DIPRINT] option.



**REF:** For more information on the **Print File Entries** [DIPRINT] option, see the “Print: How to Print Reports from Files” section in the *VA FileMan User Manual*.

The Export Tool creates a specially formatted print output. Some limitations apply to data exports that do *not* apply to setting up a regular print (e.g., **WORD-PROCESSING**-type fields *cannot* be exported). Some capabilities are available when exporting that are *not* when you are printing (e.g., the records you export can be longer than **245 characters**, if you are using a delimited format; see the description of the Maximum Output Length FOREIGN FORMAT attribute below). These differences are discussed below.

The steps to export data are:

1. [Make Sure a FOREIGN FORMAT File Entry is Available](#)—Make sure there is a FOREIGN FORMAT (#.44) file entry available to export your data in the format expected by the receiving application.
2. [Select Fields for Export Option](#) [DDXP SELECT EXPORT FIELDS]—Use this option to select the fields you want to export. This creates a **SELECTED EXPORT FIELDS** template.
3. [Create Export Template Option](#) [DDXP CREATE EXPORT TEMPLATE]—Use this option to create an **EXPORT** template. This is where you combine the **SELECTED EXPORT FIELDS** template with a desired FOREIGN FORMAT.
4. [Choose Entries/Export Data](#)—Use the Export Data option [DDXP EXPORT DATA]. This is where you select which entries to export and perform the export.

### 2.3.2.1 Make Sure a FOREIGN FORMAT File Entry is Available

First, you need to determine an ASCII data format (some form of delimited or fixed-length) that your foreign application recognizes. This is the format you need the Export Tool to generate.

This data format *must* be set up in advance, as an entry in the FOREIGN FORMAT (#.44) file. The following are the major format parameters stored in a FOREIGN FORMAT (#.44) file entry:

- What delimiters are used between fields?
- Does the export use fixed length fields?
- What headers to output before the body of the data, and what footers after the data
- Any special formatting for specific DATA TYPE field values (e.g., dates and numbers)?

Some formats are already set up in advance in the FOREIGN FORMAT (#.44) file, targeted towards specific foreign applications. These include:

- Word Data File (Comma)
- Excel (Comma)
- Excel (Tab)
- 1-2-3 Import Numbers

- 1-2-3 Data Parse
- Oracle (Delimited)

Keep in mind that applications are often updated. A format that worked for one version may *not* work for a different version, or a more efficient, simpler format might be possible for a different version.



**REF:** The full details of the export parameters that can be set up for exporting are described in the "[FOREIGN FORMAT File Attributes Reference](#)" section.

In many cases, you can use an existing FOREIGN FORMAT (#.44) file entry for your export. If you need to create a *new* FOREIGN FORMAT (#.44) file entry (rather than using an existing entry), set up the new entry with the **Define Foreign File Format** [DDXP DEFINE FORMAT] option.

### 2.3.2.2 Select Fields for Export Option

With the [Define Foreign File Format Option](#) [DDXP DEFINE FORMAT], you determined the data format for your export and made sure there was a corresponding FOREIGN FORMAT (#.44) file entry. The next step is to choose what file and field data to export. Do this using the **Select Fields for Export** [DDXP SELECT EXPORT FIELDS] option; this creates a **SELECTED EXPORT FIELDS** template.

The process of creating a **SELECTED EXPORT FIELDS** template is very similar to the way you choose fields for printing with the **Print File Entries** [DIPRINT] option.



**REF:** For details on selecting fields, see the "Choosing Print Fields" section in the "Print: How to Print Reports from Files" section in the *VA FileMan User Manual*.

First, you *must* identify the file from which you are exporting data. This is the primary file. Then you choose from which fields to export data.

In addition to fields from that file and its Multiples, you can export data from other files by using the extended pointer syntax.



**REF:** For more information on pointer syntax, see the "[Relational Navigation](#)" section.

Also, you can put other computed expressions at the "EXPORT FIELD:" prompt to make use of VA FileMan functions or M code.

There are several kinds of specifications that are valid at the "PRINT FIELD:" prompt that are *not* allowed at the "EXPORT FIELD:" prompt. They are:

- **WORD-PROCESSING**-type fields.
- "**ALL**" signifying all the fields in a file.
- Print qualifiers following the field designation (e.g., "**X**" or "**C22**").
- Statistical print qualifiers preceding the field (e.g., **#** or **&**).
- Backward extended pointers.
- Relational jumps to other files (i.e., use of a terminating colon); instead, use the full extended pointer syntax to obtain data from other files.
- Specifications that return more than one value (e.g., a Multiple in a pointed-to file); you can specify Multiples in the primary file.

After you enter a set of field specifications, you are *immediately* prompted for a template in which to store the selected fields. You *must* store your field specifications in a template to proceed with the next step in the data export. After you specify a template name for the **SELECTED EXPORT FIELDS** template, you have completed this step.

Figure 11 is an example of the “EXPORT FIELD:” dialog. The example uses the sample PATIENT (#2) file. Several unacceptable responses are shown; the error messages are the ones you would receive to these responses:

**Figure 11: Import and Export Tools—Creating the SELECTED EXPORTED FIELDS Template**

```

SELECT VA FILEMAN OPTION: OTHER OPTIONS
SELECT OTHER OPTION: DATA EXPORT TO FOREIGN FORMAT

SELECT DATA EXPORT TO FOREIGN FORMAT OPTION: SELECT FIELDS FOR EXPORT

OUTPUT FROM WHAT FILE: PATIENT

FIRST EXPORT FIELD: NAME;S

SORRY. YOU CANNOT ADD ;S TO THE EXPORT FIELD SPECIFICATIONS.

FIRST EXPORT FIELD: NAME
THEN EXPORT FIELD: INTERNAL (SEX)
THEN EXPORT FIELD: RELIGION:

SORRY. YOU CANNOT JUMP TO ANOTHER FILE WHEN SELECTING FIELDS
FOR EXPORT.

THEN EXPORT FIELD: $E (RELIGION:CLASSIFICATION,1,5)
THEN EXPORT FIELD: DIAGNOSIS <ENTER> (MULTIPLE)
  THEN EXPORT DIAGNOSIS SUB-FIELD: DIAGNOSIS
  THEN EXPORT DIAGNOSIS SUB-FIELD: HISTORY <ENTER> (WORD-PROCESSING)

SORRY. YOU CANNOT CHOOSE A WORD PROCESSING FIELD FOR EXPORT.

  THEN EXPORT DIAGNOSIS SUB-FIELD: AGE AT ONSET
  THEN EXPORT DIAGNOSIS SUB-FIELD: <ENTER>
THEN EXPORT FIELD: <ENTER>
STORE EXPORT LOGIC IN TEMPLATE: PATIENT TEST
  ARE YOU ADDING PATIENT TEST AS A NEW PRINT TEMPLATE? NO// Y <ENTER> (YES)

SELECT DATA EXPORT TO FOREIGN FORMAT OPTION:

```

**SELECTED EXPORT FIELDS** templates are sometimes referred to as **PRINT** templates in the user dialog. This is because they are stored in the PRINT TEMPLATE (#.4) file.



**NOTE:** Even though you *cannot* “jump” to the RELIGION (#13) file using the RELIGION field, which is a pointer to the RELIGION (#13) file, you can retrieve data from that file by using extended pointer syntax.

**REF:** For more information on pointer syntax, see the “[Relational Navigation](#)” section.

You can edit a **SELECTED EXPORT FIELDS** template. The editing *must* occur in the **Export Data** [DDXP EXPORT DATA] option, *not* in the standard **Print File Entries** [DIPRINT] option. To edit one, enter the template name at the "FIRST EXPORT FIELD:" prompt preceded by a left bracket ( [ ).

If an **EXPORT** template (see Section 2.3.2.3, "[Create Export Template Option](#)") has been created based on the **SELECTED EXPORT FIELDS** template that you edit, the **SELECTED EXPORT FIELDS** template are *not* updated to reflect the changes. You *must* create a new **SELECTED EXPORT FIELDS** template to make use of the changes.

### 2.3.2.3 Create Export Template Option

The next step to export data is to create an **EXPORT** template with the **Create Export Template** [DDXP CREATE EXPORT TEMPLATE] option. The **EXPORT** template combines the **SELECTED EXPORT FIELDS** template (created in Step 2; Section 2.3.2.2, "[Select Fields for Export Option](#)") with a FOREIGN FORMAT (#.44) file (see Step 1; Section 2.3.2.1, "[Make Sure a FOREIGN FORMAT File Entry is Available](#)").

Besides choosing a **SELECTED EXPORT FIELDS** template and a FOREIGN FORMAT, you are asked for any additional information that is needed to fully define the export. If you do *not* supply the requested information, the **EXPORT** template *cannot* be created. Values in the FOREIGN FORMAT entry you choose determine whether you are prompted for more information.

[Table 2](#) indicates which values for which FOREIGN FORMAT fields result in prompts:

**Table 2: Import and Export Tools—Foreign Format Field Prompts**

Foreign Format Field	Value	Information Required
FIELD DELIMITER	"ASK"	The character or characters to separate fields.
RECORD DELIMITER	"ASK"	The character or characters to separate records.
RECORD LENGTH FIXED?	"1" or "YES"	The number of characters in each field to be exported.
NEED FOREIGN FIELD NAMES?	"1" or "YES"	The name of each field recognized by the importing application.
MAXIMUM OUTPUT LENGTH	"Ø"	The maximum number of characters on each line of output, usually the longest possible exported record.
PROMPT FOR DATA	"1" or	The DATA TYPE field value of each exported field; possible

Foreign Format Field	Value	Information Required
TYPE?	"YES"	choices are: <ul style="list-style-type: none"> <li>• <b>FREE TEXT</b></li> <li>• <b>NUMERIC</b></li> <li>• <b>DATE/TIME</b></li> </ul>

In the example in [Figure 12](#), the file and field specifications in the **SELECTED EXPORT FIELDS** template example ([Figure 11](#)) are combined with the 123 Import Numbers FOREIGN FORMAT:

**Figure 12: Import and Export Tools—Creating the EXPORT Template**

```

SELECT DATA EXPORT TO FOREIGN FORMAT OPTION: CREATE EXPORT TEMPLATE

OUTPUT FROM WHAT FILE: PATIENT <ENTER> (10 ENTRIES)

ENTER SELECTED EXPORT FIELDS TEMPLATE: PATIENT TEST
**SELECTED EXPORT FIELDS** (OCT 30, 1992@11:32) USER #7 FILE #99002

DO YOU WANT TO SEE THE FIELDS STORED IN THE PATIENT TEST TEMPLATE?
ENTER YES OR NO: NO// YES

FIRST PRINT FIELD: NAME// <ENTER>
THEN PRINT FIELD: INTERNAL(SEX)// <ENTER>
THEN PRINT FIELD: $(RELIGION:CLASSIFICATION,1,5)// <ENTER>
THEN PRINT FIELD: DIAGNOSIS// <ENTER>
    THEN PRINT DIAGNOSIS SUB-FIELD: DIAGNOSIS// <ENTER>
    THEN PRINT DIAGNOSIS SUB-FIELD: AGE AT ONSET// <ENTER>
    THEN PRINT DIAGNOSIS SUB-FIELD: // <ENTER>
THEN PRINT FIELD: // <ENTER>

DO YOU WANT TO USE THIS TEMPLATE?
ENTER YES OR NO: YES// <ENTER>

DO YOU WANT TO DELETE THE PATIENT TEST TEMPLATE
AFTER THE EXPORT TEMPLATE IS CREATED?
ENTER YES OR NO: NO// <ENTER>

```

When asked if you want the **SELECTED EXPORT FIELDS** template deleted, answer **YES** only if you know you do *not* need the template again. If an **EXPORT** template is *not* successfully created, the **SELECTED EXPORT FIELDS** template is *not* deleted.

Next, identify the FOREIGN FORMAT to use, and name the **EXPORT** template that you are creating. You *cannot* overwrite an existing **PRINT** template:

**Figure 13: Import and Export Tools—Identifying the FOREIGN FORMAT and EXPORT Templates**

```
SELECT FOREIGN FORMAT: 123 IMPORT NUMBERS <ENTER> **DISTRIBUTED BY VA FILEMAN**
ENTER NAME FOR EXPORT TEMPLATE: PATIENT TO 123
ARE YOU ADDING PATIENT TO 123 AS
A NEW PRINT TEMPLATE (THE 197TH)? NO// Y <ENTER> (YES)
```

After you choose the **EXPORT** template name, you are prompted for any additional information needed. In this example, the format does require additional information: the DATA TYPE field value for each field (in this situation the defaults derived by the Export Tool are correct) and the maximum length of each record:

**Figure 14: Import and Export Tools—Entering DATA TYPE Field Values in an EXPORT Template**

```
ENTER THE DATA TYPES OF THE FIELDS BEING EXPORTED BELOW.
DO YOU WANT TO CONTINUE?
ENTER YES OR NO: YES// <ENTER>
NAME: FREE TEXT// <ENTER> FREE TEXT
INTERNAL(SEX): FREE TEXT// <ENTER> FREE TEXT
$(RELIGION:CLASSIFICATION,1,5): FREE TEXT// <ENTER> FREE TEXT
DIAGNOSIS IN DIAGNOSIS SUBFILE: FREE TEXT// <ENTER> FREE TEXT
AGE AT ONSET IN DIAGNOSIS SUBFILE: NUMERIC// <ENTER> NUMERIC
ENTER THE MAXIMUM LENGTH OF A PHYSICAL RECORD THAT CAN BE EXPORTED.
ENTER ^ TO STOP THE CREATION OF AN EXPORT TEMPLATE.
MAXIMUM OUTPUT LENGTH: 100
EXPORT TEMPLATE CREATED.
```

The Export Tool checks to make sure that your **SELECTED EXPORT FIELDS** template does *not* contain fields from Subfiles (Multiples) that are *not* descendent from each other.



**REF:** For more information on Subfiles (Multiples), see the “[Exporting Data from Multiples](#)” section.

If you have *not* followed that restriction, you receive an error message. The **SELECTED EXPORT FIELDS** template would have to be modified.

### 2.3.2.4 Choose Entries/Export Data

In the final step to export data, use the **Export Data** [DDXP EXPORT DATA] option to select which entries from the file to export, and then perform the export.

First, choose which entries to export with a "SEARCH" dialog; then choose the order of the exported entries with a "SORT BY" dialog (you are *not* given the "SORT BY" dialog, if you are exporting fields from Subfiles.) Finally, specify the device to send the exported data.

During either the Search or Sort process, you can use previously created **SEARCH** and **SORT** templates. Those templates need *not* have been originally made during a data export; however, **SORT** templates that contain unacceptable qualifiers should *not* be used. At the "SORT BY:" prompt, you can only use the subset of sort qualifiers shown in [Table 3](#):

**Table 3: Import and Export Tools—Allowable Sort Qualifiers When Exporting Data**

Sort Qualifier	Description
,	To <i>not</i> sort. Used when you want to use the "FROM ... TO" dialog to restrict the entries to be exported.
-	To sort in reverse order.
;Ln	To sort on the first <i>n</i> -characters only.
;TXT	To sort following strict ASCII sorting sequence.



**REF:** For more detailed information about searching and sorting, see the "Print: How to Print Reports from Files" and "Search" sections in the *VA FileMan User Manual*.

### 2.3.2.4.1 Export Example

Figure 15 is an example of an export using the "PATIENT TO 123" **EXPORT** template created in the previous section (Figure 13 and Figure 14). You begin by identifying the file and the **EXPORT** template that you want to use for the export. Do *not* enclose the template's name with brackets. Again, you can delete the **EXPORT** template after a successful export.

Because there is a Multiple involved, you are told that you do *not* have the opportunity to sort. Then, you are given the opportunity to search the file for entries to export.

**Figure 15: Import and Export Tools—Searching for Entries to be Exported**

```
SELECT DATA EXPORT TO FOREIGN FORMAT OPTION: EXPORT DATA
OUTPUT FROM WHAT FILE: PATIENT// <ENTER>
CHOOSE AN EXPORT TEMPLATE: PATIENT TO 123 <ENTER> **EXPORT**
(OCT 30, 1992@15:08) USER #7 FILE #99002

DO YOU WANT TO DELETE THE PATIENT TO 123 TEMPLATE
AFTER THE DATA EXPORT IS COMPLETE?
ENTER YES OR NO: NO// <ENTER>

SINCE YOU ARE EXPORTING FIELDS FROM MULTIPLES,
A SORT WILL BE DONE AUTOMATICALLY.
YOU WILL NOT HAVE THE OPPORTUNITY TO SORT THE DATA BEFORE EXPORT.

DO YOU WANT TO SEARCH FOR ENTRIES TO BE EXPORTED? NO// YES

-A- SEARCH FOR PATIENT FIELD: DATE OF BIRTH
-A- CONDITION: < <ENTER> LESS THAN
-A- LESS THAN DATE: 1980 <ENTER> (1980)

-B- SEARCH FOR PATIENT FIELD: <ENTER>

IF: A// <ENTER> DATE OF BIRTH LESS THAN 1980 (1980)

STORE RESULTS OF SEARCH IN TEMPLATE: <ENTER>
```

If Multiples had *not* been involved, you would now be able to respond to the SORT BY dialog. You can do the same things with sort here that you can do when using the **Print File Entries** [DIPRINT] option.

### 2.3.2.4.2 What Device to Send Export Data To

After you complete the sort dialog, you are immediately given the "DEVICE:" prompt. Choose to which device the exported data should be sent:

**Figure 16: Import and Export Tools—Choosing a Device to Send Exported Data**

```
DEVICE: <ENTER>
```

If you press the **Enter** key at the "DEVICE:" prompt, the export output is displayed on your *screen*. Sending the formatted export data to the screen allows you to use a PC-based screen capture to put the data into a file. This file would be a readable ASCII file on that computer. This method of transferring the data into a file is a simple one that is often successful and convenient, especially if the importing application is on the same PC.

When using a screen capture to create a file from the exported data, you *must* consider the peculiarities of your communication and terminal emulation software. Your communication application, for example, can intercept certain control characters (like the **<TAB>**, ASCII 9) and convert them into something else. This can cause the import to fail. Also, your terminal emulation can *automatically* "break" lines at **80 characters** by inserting an unwanted carriage return or line feed. When emulating **VT-100** and other ANSI terminals, you can avoid this last problem by turning wraparound mode **off**.



**CAUTION: When exporting data to your terminal's screen, there are no page breaks. Therefore, there is no graceful way to interrupt the export once it has begun.**

### 2.3.2.4.3 Sending Export Data to a Host File

Having data printed on-screen is of little use if you are using a terminal with no screen-capture capabilities. An alternative is to send the data to a file on the host system, for example, to a VMS file if you are using DSM. Another advantage to sending data to a Host file is that only the exported data is in the file. (Often, screen captures unavoidably contain extraneous parts of the user's dialog prior to or after the export.) To export your data to a file, at the "DEVICE:" prompt, send your export output to an HFS-type device.

The system administrators should be able to help you, if you are *not* sure how to use HFS devices.



**REF:** The *Kernel 8.0 Systems Management: Device Handler User Guide* also describes how to send output to Host files, including how to set up and use HFS-type devices.

When a Host file is created, you *must* move that ASCII file to the computer on which the importing application resides. A file transfer protocol (e.g., KERMIT or XMODEM) can be used to move this file.

The export can be queued, if it is *not* sent to the screen. Queuing the export is *recommended* for large files and for complex sorts of the data.



**NOTE: On HFS Device Setup on OpenVMS Systems:** DSM for OpenVMS requires that you add a command parameter to the **OPEN** command, if you export records longer than **512 characters** to a Host file. The parameter is **RECORDSIZE=nnnn**, where "**nnnn**" is greater than the longest record that you are exporting. If you are using Kernel's DEVICE (#3.5) file, the OPEN PARAMETER field for the HFS device you are using should be edited to look like **"(NEW:RECORDSIZE=nnnn)"**.

#### 2.3.2.4.4 Sample Output

The data in [Figure 17](#) has been prepared for import by Lotus 1-2-3, so it need *not* be easily read by people. However, you can see that text fields are surrounded by quotes; empty text fields consist just of two quotes (""). A space is in between each field's value. Numeric values have no quotes. If a field defined as **NUMERIC** in the VA FileMan data dictionary has no value, a **Zero (0)** is output, because this format has SUBSTITUTE FOR NULL set to **0 (Zero)**.

**Figure 17: Import and Export Tools—Example of Exported Data**

```

[FMPATIENT,FIVE [M] [PROTE [GANGRENE 45
[FMPATIENT,SIX [F [CATHO [SLEEPING SICKNESS 28
[FMPATIENT,SEVEN [M] [PROTE [CIRRHOISIS 25
[FMPATIENT,EIGHT [F [OTHER [FLU 34
[FMPATIENT,NINE [M] [ [BLOOD POISONING 44
[FMPATIENT,FIVE [M] [PROTE [GUN SHOT 50
[FMPATIENT,EIGHT [F [OTHER [FLU 37
[FMPATIENT,NINE [M] [ [FLU 0
[FMPATIENT,EIGHT [F [OTHER [FLU 46
[FMPATIENT,EIGHT [F [OTHER [APPENDICITIS 39
```

#### 2.3.2.5 Special Considerations: Exporting Numbers

If a number comes from a field in your primary file that is defined as **NUMERIC** or **COMPUTED**, that number is exported with all leading spaces or trailing insignificant **Zeros** removed. This is different from the way that the regular VA FileMan **Print File Entries** [DIPRINT] option works. If the field had a value of **Zero (0)** is exported. If the value of a numeric field in the primary file is **NULL**, the exported value depends on the contents of the SUBSTITUTE FOR NULL field for the format being used.

If a number comes from a source other than a DATA TYPE field of **NUMERIC** or **COMPUTED** in the primary file, it can be output with leading spaces or trailing insignificant **Zeros**. Such a number might originate from a field in a pointed-to file reached by the relational syntax, a VA FileMan function, or other computed expression. In these cases, the value of the SUBSTITUTE FOR NULL field usually has no effect on what is exported.



**NOTE:** Whether exported numbers have leading spaces or trailing insignificant **Zeros** and whether **NULLs** produce special output is controlled by how the field is defined in the VA FileMan data dictionary. The DATA TYPE field input by the user when the PROMPT FOR DATA TYPE? field contains **YES** does *not* affect these characteristics of the export.

## 2.3.2.6 Special Considerations: Multiples

### 2.3.2.6.1 Exporting Data from Multiples

#### 2.3.2.6.1.1 Data Flattening

Data exported from Multiples is “flattened” (i.e., data at upper levels is repeated for each subentry). For example, take the comma-delimited export for a top-level file’s #.01 NAME field and a Subfile’s #.01 DATE and #1 TYPE fields. The output for an entry with four subentries would look like [Figure 18](#):

**Figure 18: Import and Export Tools—Example of Data Flattening When Exporting Data from Multiples**

```
FMPATIENT,01-JAN-95,SC
FMPATIENT,24-JUN-95,NSC
FMPATIENT,14-AUG-95,SC
FMPATIENT,21-JUL-96,NSC
```

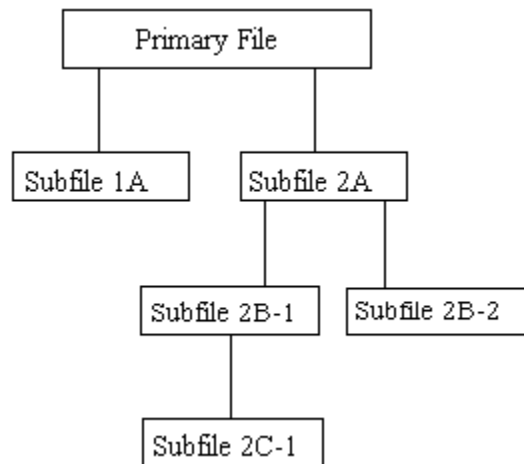


**NOTE:** The top-level .01 field is repeated for each Subfile entry.

#### 2.3.2.6.2 No More Than One Multiple at Any One File Level

You *cannot* export *more than one* Multiple at *any one file level*. You can export data from one Multiple and from Subfiles directly descendent from that Multiple (if you never export more than one Subfile at the same level). Suppose you are exporting data from a file with the structure shown in [Figure 19](#):

**Figure 19: Import and Export Tools—Example of a File Structure**



In addition to fields in the Primary file, you can export from Subfile 1A *or* Subfile 2A, but *not* from both. Also, you can export from Subfile 2A, Subfile 2B-1, and Subfile 2C-1, but you could *not* additionally choose fields in Subfile 2B-2. If you need data from Subfiles that are *not* directly descendent from each other, you can do multiple exports and “join” the data together in the importing application.

### 2.3.2.6.3 Sorting with Multiples

A special, automated sort is done to the data when Multiples are exported; you *cannot* perform your own sort. When Subfiles are involved, the Export Tool performs a special sort to format the data. Since the Export Tool *must* do this customized sort, you *cannot* sort the data yourself. If you need the data in a particular sequence, sort it in the importing application. You can perform any search on the data that is necessary to choose entries for export.

### 2.3.2.7 About EXPORT Templates

The Export Tool uses two types of templates:

- **EXPORT FIELDS** template (created in Step 2)
- **EXPORT** template (created in Step 3).

These templates are variations on standard **PRINT** templates. They are stored in the PRINT TEMPLATE (#.4) file and are sometimes referred to as **PRINT** templates in the user dialog. Although like **PRINT** templates, they do differ in important respects. For example, you *cannot compile* either of the Export Tool’s templates.

You can delete these templates as soon as they are used if you wish. Also, both kinds of templates can be deleted using the **Template Edit** [DITEMP] option on the **Utility Functions** [DIUTILITY] menu. In addition, you can delete an **EXPORT FIELDS** template by choosing the template within the **Select Fields for Export** [DDXP SELECT EXPORT FIELDS] option, editing it, and putting an at-sign (@) at the “NAME:” prompt. Do *not* delete an **EXPORT** template *before* a queued export has been completed.

## 2.3.3 How to Import Data

The menu in [Figure 20](#) shows the **Import Data** [DDMP IMPORT] option used to import data, which is located under the **Other Options** [DIOOTHER] menu:

**Figure 20: Import and Export Tools—Import Data Option**

VA FILEMAN ...	[DIUSER]
OTHER OPTIONS ...	[DIOOTHER]
IMPORT DATA	[DDMP IMPORT]

The Import Tool lets you import records stored in an ASCII data file into a VA FileMan file.

The Import Tool imports records from an ASCII data file by *adding* them as new records to the VA FileMan file in question. Existing records in the destination VA FileMan file are *never* edited or updated, and the Import Tool does *not* prevent duplicate records from being added.

Importing data records from an ASCII file is a four-step process, as described below.

### 2.3.3.1 Generate ASCII Source File

Generate your source file (from your non-VA FileMan application), containing the records to be imported. Generate the file with one record per line, with the fields in each record being set off using either the delimited or fixed-length method. The last record in the file *must* be terminated with the appropriate EOL (End-of-Line) characters for your operating system.

Once you generate your ASCII source file, you need to move it to a disk that is accessible from the computer system running VA FileMan. System administrators should be able to assist you with this.

### 2.3.3.2 Specify Data Format, Source File, and Destination File

Invoke VA FileMan's **Import Data** [DDMP IMPORT] option. It loads a two-page ScreenMan form. On page one of the form, you need to specify the: data format, source file, and destination file for your import.

- **DATA FORMAT—INTERNAL or EXTERNAL:** Specify if the incoming data is in external form (the way VA FileMan would display it) or internal form (the way VA FileMan would store it). Unless you are knowledgeable about how VA FileMan stores data, you should choose EXTERNAL. Also, the incoming data is only validated by VA FileMan if you choose EXTERNAL (validation prevents you from putting invalid data into the file).
- **FOREIGN FORMAT:** Choose a Foreign Format entry whose settings match the ASCII format for the incoming records. The only settings used from the Foreign Format entry are
  - Record Delimiter
  - Record Length Fixed?
  - Quote Non-numeric Fields?

Make sure the settings in the Foreign Format match the format of your incoming data. Because some foreign applications export data in a different format than they import it, a Foreign Format that works for export may *not* have the appropriate settings for import.

As an alternative to specifying a Foreign Format entry, you can manually specify the settings for your incoming data in the three provided fields:

1. Is the data fixed length?
  2. If not, what is the field delimiter?
  3. Are fields quoted?
- **SOURCE FILE:** Enter the path and name of your source file (the file containing the records to import).
  - **VA FILEMAN FILE:** Specify the destination file for the imported records.
  - **FIELD SELECTION PAGE/IMPORT TEMPLATE:** This is where you match the fields in the incoming records to the fields in the destination file. If you do *not* have an existing **IMPORT** template that matches incoming to destination fields, go to the Field Selection page and specify those fields individually (see the [“Match Source to Destination Fields”](#) section).

A completed page one of the form might look like [Figure 21](#):

**Figure 21: Import and Export Tools—Example of a Completed Data Import Form**

DATA IMPORT	PAGE 1
=====	
DATA FORMAT □□□□□□□□□□- <u>INTERNAL OR EXTERNAL</u> : EXTERNAL  <u>FOREIGN FORMAT</u> : EXCEL (COMMA) OR <u>DATA FIXED LENGTH?</u> <u>FIELD DELIMITER</u> : <u>FIELDS QUOTED?</u>	SOURCE FILE □□□□□□□□□□- <u>FULL PATH</u> : USER\$:[FMPATIENT] <u>HOST FILE NAME</u> : IMPORT.DAT  VA FILEMAN FILE □□□□□□□□□□□□□□- <u>PRIMARY FILE</u> : NEW PERSON  <u>FIELD SELECTION PAGE...</u> OR <u>IMPORT TEMPLATE</u> :
COMMAND: <span style="float: right;">PRESS &lt;PF1&gt;H FOR HELP <span style="border: 1px solid black; padding: 2px;">INSERT</span></span>	

### 2.3.3.3 Match Source to Destination Fields

For your import, you need to match each field in the incoming record to a field in the destination VA FileMan file.

Fields in the incoming record are imported in order, from left to right. Thus, for each field in the incoming record, you specify the corresponding destination field in the VA FileMan file, in the same order. The first VA FileMan field you specify is the destination for the first field in the incoming record, the second matches the second field in the incoming record, and so forth.

**Figure 22: Import and Export Tools—Example of Fields Selected for Import**

```
FIELD SELECTION FOR IMPORT                                PAGE 2
=====
CHOOSE A FIELD FROM
NEW PERSON
FIELD:
          DELETE LAST FIELD SELECTED?

THESE ARE THE FIELDS SELECTED SO FAR:
1  NAME
2  STREET ADDRESS 1
3  STREET ADDRESS 2
4  STREET ADDRESS 3
5  CITY
6  STATE
7  ZIP CODE

COMMAND:                                PRESS <PF1>H FOR HELP    INSERT
```

Remember that you *must* include the **.01** field, and any fields that are required identifiers for the top-level of the file. The same is true for any Subfiles (Multiples).

If you specified a fixed-length (as opposed to delimited) data format for the incoming records, you *must* enter *not* only the destination VA FileMan field, but also the length for each corresponding incoming field.

Each time you enter a field at the "Field:" prompt, it's added to the bottom of the list of fields displayed on the form. This shows you the destination fields you have selected, and their order. If you make a mistake, you can delete fields from the bottom of the list, one-by-one, by entering **YES** at the "Delete last field selected?" prompt. To insert a field, delete back to the insertion point, enter the new field, and then re-enter the deleted fields.



**REF:** There are special issues when importing data into fields in Multiples; see the [“Special Considerations: Multiples”](#) section.

You can save the information you specify on the Field Selection page in an **IMPORT** template. This lets you reuse the field matching criteria you have entered for subsequent imports that use the same file and fields, without having to re-enter it. To save your field specifications as an **IMPORT** template, enter **YES** at the “Do you want to store the selected fields in an Import Template?” prompt, which you are asked after you exit the Import form (see the [“Run the Import”](#) section). Then, for future imports, simply enter the name of the **IMPORT** template on **Page 1** of the Import form. You can use any **IMPORT** template to which your VA FileMan Access Code gives you access.

#### 2.3.3.4 Run the Import

Once you have set up your data format, source file, and destination file, and matched source to destination fields, exit the Import form (press **<PF1>E**). After you exit the form, you are asked a series of questions:

1. Do you want to store the selected fields in an Import Template?
2. Do you want to proceed with the import?
3. Device for Import Results Report

Storing your file and field specifications in an **IMPORT** template lets you do subsequent imports *without* having to re-enter all field information.

If you proceed with the import, enter a device to which the Import Results report should print. You can run the Import directly or queue it.

As the import proceeds, if an error occurs updating a field in a particular record, the record is *not* added, and an error message is added to the Import Report saying what the problem was.

An example of the dialog after exiting the Import form is shown in [Figure 23](#):

**Figure 23: Import and Export Tools—Exiting the Template Form and Performing the Import**

DO YOU WANT TO STORE THE SELECTED FIELDS IN AN IMPORT TEMPLATE? <b>YES</b>
NAME OF IMPORT TEMPLATE: <b>ZZIMPORT</b>
ARE YOU ADDING <input type="checkbox"/> ZZIMPORT <input type="checkbox"/> AS A NEW IMPORT TEMPLATE? <b>YES</b>
DO YOU WANT TO PROCEED WITH THE IMPORT? <b>YES</b>
DEVICE FOR IMPORT RESULTS REPORT: HOME// <b>&lt;ENTER&gt;</b> SYSTEM

Once the import finishes, you can review the Import Results report. It lists:

- The criteria you chose for your import.
- Any records for which the import failed.
- The internal entry numbers of the first and last records imported.

Figure 24 is a sample Import Results report:

**Figure 24: Import and Export Tools—Example of an Import Results Report**

```
LOG FOR VA FILEMAN DATA IMPORT                                PAGE 1
=====

IMPORT INITIATED BY: 10 FMPATIENT

SOURCE FILE: USER$:[FMPATIENT1]IMPORT.DAT
FIXED LENGTH: NO
DELIMITED BY: ,
TEXT VALUES QUOTED: NO
VALUES ARE: EXTERNAL

PRIMARY FILEMAN DESTINATION FILE: NEW PERSON

SEQ  LEN  FIELD NAME                                SUBFILE NAME (IF APPLICABLE)
---  ---  -
1    N/A  NAME
2    N/A  STREET ADDRESS 1
3    N/A  STREET ADDRESS 2
4    N/A  STREET ADDRESS 3
5    N/A  CITY
6    N/A  STATE
7    N/A  ZIP CODE

ERROR REPORT
-----

RECORD #4 REJECTED:
THE VALUE ILLINOIS FOR FIELD STATE IN FILE NEW PERSON IS NOT VALID.

SUMMARY OF IMPORT
-----

TOTAL RECORDS READ: 7
TOTAL RECORDS FILED: 6
TOTAL RECORDS REJECTED: 1

IEN OF FIRST RECORD FILED: 209
IEN OF LAST RECORD FILED: 214

IMPORT FILING STARTED: JUL 16, 1996@08:24:36
IMPORT FILING COMPLETED: JUL 16, 1996@08:24:38
TIME OF IMPORT FILING: 0:00:02
```

In this example (Figure 24), six records were added, and one record was *not* added. The record that was *not* added was the fourth record in the source file. It failed due to the misspelled value “**Illinois**” being rejected by the STATE field in the NEW PERSON (#200) file.

### 2.3.3.5 Special Considerations: Multiples

#### 2.3.3.5.1 Importing Data into Multiples



**CAUTION: Incoming Data Should *not* be flattened.**

The Import Tool expects that any data bound for a Multiple be contained in the same import record (line of data) as the data for the top file level. This is different from the output of the Export Tool, which “flattens” exported data from Multiples into separate lines of output.

For example, consider a comma-delimited import of records, each including a name plus four subentries. Each subentry contains a DATE and a TYPE. The records are imported into a file with a top-level NAME (#.01) field and a Multiple with DATE (#.01) field and TYPE (#1) field. For this import, you would choose the destination fields as shown in [Figure 25](#):

**Figure 25: Import and Export Tools—Example of Fields Selected for Import to a Multiple**

```
FIELD SELECTION FOR IMPORT                                PAGE 2
=====
CHOOSE A FIELD FROM
PATIENT : DATE      SUBFILE
FIELD:

                                DELETE LAST FIELD SELECTED?

THESE ARE THE FIELDS SELECTED SO FAR:
1  NAME
2  DATE:DATE
3  DATE:TYPE
4  DATE:DATE
5  DATE:TYPE
6  DATE:DATE
7  DATE:TYPE
8  DATE:DATE
9  DATE:TYPE

-----
EXIT      SAVE      NEXT PAGE      REFRESH

ENTER A COMMAND OR ^ FOLLOWED BY A CAPTION TO JUMP TO A SPECIFIC FIELD.

COMMAND:  NEXT                                PRESS <PF1>H FOR HELP                                INSERT
```

A corresponding line of data to be imported for a record, containing data for both the top-level record and its subentries, would look like [Figure 26](#):

**Figure 26: Import and Export Tools—Example of Data *Not* Flattened When Importing Data to a Multiple**

```
FMPATIENT,01-JAN-95,SC,24-JUN-95,NSC,14-AUG-95,SC,21-JUL-96,NSC
```



**NOTE:** You *must* file the same number of subentries in each record you import.

### 2.3.3.5.2 Completeness of Subfile Entries

New subentries need to be added to every Subfile on a path to the lowest level Subfiles. Your data *must* include values for the **.01** field and all the required identifiers for every Subfile (as well as for the top-level of the file). You can add more than one subentry in a particular Subfile. However, you are restricted to the same set of fields for every entry in each Subfile.

### 2.3.3.6 Importing from VMS Files

When importing from a data file that's been transferred to a VMS-based computer system, a problem can occur if, once transferred, the data file does *not* get a maximum record length stored in its file header. This can happen when a DOS file is moved to a VMS system by some protocols. When the maximum record length is unknown, VMS uses a default maximum size of 510. If the length of a data record in the source file is larger than the maximum size, an error results.

The solution is to run the VMS CONVERT utility on the Host file. This utility adds the maximum record information to the file header and everything works just fine!

You can see if the maximum record length is stored in a file's header on a VMS system, by using DCL command in [Figure 27](#):

**Figure 27: Import and Export Tools—Verifying the Maximum Record Length on a VMS System**

```
DIR FILENAME /FULL
```

## 2.3.4 Foreign Formats

### 2.3.4.1 FOREIGN FORMAT File Attributes Reference

The following fields in the FOREIGN FORMAT (#.44) File correspond to attributes of the formatted data that you wish to export or import:

- FIELD DELIMITER
- QUOTE NON-NUMERIC FIELDS?
- SEND LAST FIELD DELIMITER?
- PROMPT FOR DATA TYPE?
- RECORD DELIMITER
- SUBSTITUTE FOR NULL
- RECORD LENGTH FIXED?
- DATE FORMAT
- MAXIMUM OUTPUT LENGTH
- FILE HEADER
- NEED FOREIGN FIELD NAMES?
- FILE TRAILER

When *exporting* records, all fields in this file are used in the export process. When *importing* records, only three fields are used in the import process:

- FIELD DELIMITER
- RECORD LENGTH FIXED?
- QUOTE NON-NUMERIC FIELDS?

In this section, each format characteristic is described. Some combinations of characteristics are unacceptable; these situations are mentioned.

Also, some of the fields allow you to enter M code.



**REF:** Export-specific variables you can use in this M code are described in the [“Variables Available for Developer Use”](#) section.

To set up a FOREIGN FORMAT (#.44) file entry, use the **Define Foreign File Format** [DDXP DEFINE FORMAT] option to print out a format, use the **Print Format Documentation** [DDXP FORMAT DOCUMENTATION] option.

### 2.3.4.1.1 FIELD DELIMITER

Many applications can import and export data if the values of fields in each record are separated by a known character or sequence of characters. The application puts (or expects) data before the first delimiter into its first field, between the first and second delimiter into the second field, and so on. Therefore, the ability to specify and recognize these field delimiters is a crucial aspect of many data exchanges.

The Import and Export Tools' FIELD DELIMITER fields allow you to specify up to **15 characters** to be placed between each field. You can directly enter any string of characters except ones that begin with a number or consist of characters that have special meaning when editing VA FileMan data (e.g., ^ or @).

If your field delimiter begins with one of these restricted characters or consists of an unprintable control character (like <**TAB**>), you can enter the ASCII-value of the delimiter. When entering the ASCII values, always use three digits. Thus, <**TAB**> (**ASCII 9**) becomes "009" and @ (**ASCII 64**) becomes "064". You can enter up to four ASCII values. If more than one is needed, separate the values with commas (e.g., "048,094").

If you want the user to be prompted for a field delimiter at the time the **EXPORT** template is being created, enter "**ASK**" in this field.



**CAUTION:** Using unprintable control characters (ASCII values less than 32) as delimiters may not have the effect you want. During either export or import, often control characters are intercepted by terminal software, communication programs, or network links; they may not be passed through unaltered as regular printable characters usually are. For example, ASCII value 5 is interpreted by many terminals as a request for their Answerback Message. Thus, putting "005" in the FIELD DELIMITER field might cause an Answerback Message to be returned by your terminal instead of the ASCII value 5 being inserted between field values.



**NOTE:** The importing application will find the delimiting character, if it occurs in the data. This causes an incorrect determination of the boundary between fields. For example, if a comma (,) is the field delimiter and the data for a field was **FMPATIENT,10**, the importing application would put **FMPATIENT** into the first field and **10** into the second field. You can avoid this problem by specifying that data in *non*-numeric fields be surrounded by quotes (e.g., "**FMPATIENT,10**").

Most importing applications ignore delimiters, if they occur within a quoted string.

**REF:** For more information on *non*-numeric fields, see the "[QUOTE NON-NUMERIC FIELDS?](#)" section.

#### **2.3.4.1.2 SEND LAST FIELD DELIMITER?**

Some importing applications expect a field delimiter following every field, including the final field in a record. Other applications only expect delimiters between fields; nothing follows the final field. This field allows you to specify whether a field delimiter should be exported after the last field. A **YES** answer sends the delimiter, a **NO** answer does *not*.

The contents of this field does *not* affect whether a delimiter is sent after each *record*.

#### **2.3.4.1.3 RECORD DELIMITER**

Applications that import delimited fields need to know when one record ends and a new one begins. In most cases, records are separated by a carriage return (or by a line feed and a carriage return). This is the same as pressing **Enter** at the end of a line. The Export Tool *automatically* puts this separator after each record; every record begins on a new line of output. You do *not* need to put the ASCII values for carriage return and line feed in this field.

Some applications may also require that additional characters be placed after each record. If this is the case, put those characters into the RECORD DELIMITER field. The requirements for coding the field are the same as for the FIELD DELIMITER field.

#### **2.3.4.1.4 RECORD LENGTH FIXED?**

A second common way to import and export data (in addition to using delimited data) is with fixed length records. In a fixed length record, each field has a predetermined, constant data length. For example, a name field might be **30 characters** long. The name "**FMPATNT,10**" is only **10 characters** long; thus, **20 spaces** would be added to the field value to fill the required **30 characters**. The next field's value would begin in the thirty-first column.

If you want to import or export fixed length records, answer **YES** to this field. At the time that the **EXPORT** template is created (or an import is done), the user is prompted for the length of each field in the target or source file.

During export, in most cases data is truncated when the length of a field is reached. Thus, if a field contains **32 characters** but the user-defined length is **30**, the last **2 characters** are *not* exported. However, **DATE/TIME**-valued fields are always exported in

their entirety. For dates, the user *must* indicate a data length at least as long as the exported date, which is **11 characters** for standard VA FileMan dates.



**NOTE:** Fixed record lengths *cannot* be used in conjunction with field delimited data. Also, the **maximum record size for exports for a fixed length format is 255 characters**. There is **no limit on record length during import**, however.



**CAUTION:** Fixed length exports succeed only if all fields are exported on the same physical line. Therefore, the total of all the field lengths *must not* be more than the value stored in the **MAXIMUM OUTPUT LENGTH** field.

#### 2.3.4.1.5 MAXIMUM OUTPUT LENGTH

In many cases, data import is much easier if an entire record is contained on a single “line” of output; there are no carriage returns within a single record, only between records. (This is a requirement for a successful fixed length export.)

In a regular VA FileMan print, the amount of data printed before a carriage return is dependent on the type of device being used for output (i.e., a CRT screen would normally have **80 characters** on a line, a printer **80** or **132**). For data export, however, the physical characteristics of the output device are *not* controlling. Rather, the capabilities of the application importing data are overriding. Therefore, you can use the **MAXIMUM OUTPUT LENGTH** field to specify the length of a physical record. For field delimited (as opposed to fixed length) exports, this record length can be larger than the traditional M data limit of **255 characters**.

Put a number from **0** through **9999** into this field. The default record length is **80**. If you want the user to be prompted for a record length at the time that an **EXPORT** template is being created, put **0 (Zero)** into this field.

Regardless of the length of the maximum record, a carriage return is written after each record is output.



**NOTE:** The length of a record *cannot* exceed **255 characters** when using a fixed length format.



**CAUTION:** When sending exports to a Host file on a DSM for OpenVMS (e.g., VAX) system, you *must* add a parameter to the **OPEN** command, if any of

**your exported records are longer than 512 characters. See the “[How to Export Data](#)” section for details.**

#### 2.3.4.1.6 NEED FOREIGN FIELD NAMES?

If this field is answered **YES**, the user is prompted for a field name for each exported field when the **EXPORT** template is created. The field names are stored in the NAME OF FOREIGN FIELD field in the EXPORT FIELD Multiple in the PRINT TEMPLATE (#.4) file.



**REF:** For one way to use this information, see the discussion in the “[FILE HEADER](#)” section.

#### 2.3.4.1.7 QUOTE NON-NUMERIC FIELDS?

When **importing** data, VA FileMan ignores the field delimiter in a quoted string when this field is set to **YES**.

When **exporting** data, if you want all values that do *not* belong to a DATA TYPE field of **NUMERIC** to be surrounded by quotation marks, answer **YES** to this field.

Many importing applications treat data within quotation marks (“”) in a special way. Sometimes such data is automatically considered to be text, as opposed to numbers. Also, the importer may ignore the field delimiter character, if it falls within a quoted string. Quoting a **NULL** value from a *non*-numeric field results in two double quotes (“”) being exported.

During export, the DATA TYPE field value is automatically determined for fields in the primary file and its Multiples. DATA TYPE fields of **NUMERIC** are considered **NUMERIC**. There may be other fields that you want treated as **NUMERIC**. For example:

- **COMPUTED**-type fields with numeric results.
- Fields referenced by the extended pointer syntax.
- Replies to the “EXPORT FIELD:” prompt that are computed expressions with numeric results.

By default, these fields are assigned a **FREE TEXT** DATA TYPE. If you want the user to choose the DATA TYPE when the **EXPORT** template is created, answer **YES** to the PROMPT FOR DATA TYPE? field.

If the Export Tool assigns a *non*-numeric to a DATA TYPE field or if the user chooses one of those DATA TYPE field values, the field's values is surrounded by quotes when this field contains **YES**.



**NOTE:** Do *not* set this field to **YES** if a fixed length record is being exported or imported.

#### 2.3.4.1.8 PROMPT FOR DATA TYPE?

The Export Tool determines the DATA TYPE field value for fields in the primary file and its Multiples based on their definition in the data dictionary. Other fields are automatically assigned a DATA TYPE of **FREE TEXT**. If you want the user to choose the DATA TYPE of each field when creating an **EXPORT** template, answer **YES** to this field. The only DATA TYPE field values recognized by the Export Tool are the following:

- **FREE TEXT**
- **NUMERIC**
- **DATE/TIME**

The DATA TYPE field value entered by the user controls whether the values from that field are surrounded by quotes if the QUOTE NON-NUMERIC FIELDS? field is set to **YES**. The user supplied DATA TYPE field value does *not* affect how numbers are exported; numeric export is controlled by the DATA TYPE field value in the data dictionary only.

#### 2.3.4.1.8.1 SUBSTITUTE FOR NULL

**NUMERIC** fields with no data (a **NULL** value) results by default in nothing being exported for that field. For fixed record length exports, this should *not* be a problem. However, if your importing application uses spaces as a delimiter, you may need a printable character to be exported for **NULL**-valued **NUMERIC** fields. If you want a character or characters (such as "0" or ".") substituted for numeric **NULLs**, put them into this field. **NULL** values for DATA TYPE field values of **NUMERIC** in the primary file (including its Multiples) have this character exported. If you want quotes (") in your substitute string, enter two quote marks (""") for each quote you want.



**NOTE:** Do *not* put anything in this field when defining a fixed length format.



**CAUTION:** There are no substitutions for NULL values if the field being exported is *not* in the primary file, if it is reached using relational navigation.

### 2.3.4.1.9 DATE FORMAT

The native, or default, format for dates varies from application to application. VA FileMan uses two formats:

- Internal or Storage format:

*YYMMDD*

Where *YYY* is the year minus 1700.

- External or Default display format:

*MON DD,YYYY*

When data from a DATA TYPE field of **DATE/TIME** is exported, it is in the external format.

Since the importing application may recognize a different format, you can change the exported value by placing M code in this field (only those with **Programmer** access can enter code in this field.) When this M code is executed, the local variable **X** contains the date in VA FileMan *internal* format. Your M code should result in the local variable **Y** containing the date in the format you want exported.

If your format is used with Kernel, it is *recommended* that you make use of the date extrinsic functions provided by Kernel, if possible.



**REF:** For more information on Kernel date extrinsic functions, see the *Kernel 8.0 and Kernel Toolkit 7.3 Developer's Guide*.

Data from fields with DATA TYPE field values of **DATE/TIME** in the primary file, its Multiples and pointed-to files are altered by the code in this field; date values from other sources are *not*. There is another way to change the exported output; you can use a VA FileMan function when selecting fields for export:

**Figure 28: Import and Export Tools—Using VA FileMan Functions When Exporting Data**

THEN EXPORT FIELD: **NUMDATE (DATE OF BIRTH)**

The DATE FORMAT field has no effect on that output.

#### 2.3.4.1.10 FILE HEADER

Some applications require special information to process the data in the file that is imported. For example, field names might be needed. Also, you can put some special data into the file for identification or documentation purposes.

The FILE HEADER field allows you to output information before the stream of exported data. This field can contain either a literal string surrounded by quotation marks (e.g., "Data for Lotus 1-2-3") or M code that, when executed, writes the desired output.

You can put M code here only if you have **Programmer** access. The local **DDXPXTNO** variable, which equals the internal entry number in the PRINT TEMPLATE (#.4) file of the **EXPORT** template being used for data output, is defined when the code is executed.

You can use this variable to access information about the export. The data type, length, and foreign field name are stored in the EXPORT FIELD (#100) Multiple field.



**REF:** For additional information, see the data dictionary for the PRINT TEMPLATE (#.4) file.

#### 2.3.4.1.11 FILE TRAILER

You can use this field like the FILE HEADER field. The literal or M code is output *after* the exported data.

#### 2.3.4.1.12 Variables Available for Developer Use

Some of the fields in the FOREIGN FORMAT (#.44) file allow you to enter M code, if you have **Programmer** access. You can use data stored in the **EXPORT** template entry at the time the export is performed. You can also access information in the FOREIGN FORMAT (#.44) file entry used for the export.

Two variables are available for use in the M code entered in the FOREIGN FORMAT (#.44) file fields:

- **DDXPXTNO**—Internal entry number of the **EXPORT** template in the PRINT TEMPLATE (#.4) file.
- **DDXPFFNO**—Internal entry number of the Foreign Format in the FOREIGN FORMAT (#.44) file.

Consult the data dictionaries of the two files for fields that can contain useful information about either the format or the specific export itself. The EXPORT FIELD

(#100) Multiple field in the PRINT TEMPLATE (#.4) file might be of particular interest. This Multiple contains information about each field being exported.

### 2.3.4.2 Print Format Documentation Option

**Figure 29: Import and Export Tools—Print Format Documentation Option**

VA <u>FILEMAN</u> ...	[DIUSER]
<u>OTHER OPTIONS</u> ...	[DIOTHER]
<u>DATA EXPORT TO FOREIGN FORMAT</u> ...	[DDXP EXPORT MENU]
<u>PRINT FORMAT DOCUMENTATION</u>	[DDXP FORMAT DOCUMENTATION]

Use the **Print Format Documentation** [DDXP FORMAT DOCUMENTATION] option to list the available FOREIGN FORMAT (#.44) file entries on the system. When you use this option, you are given the choice of specifying individual formats or of printing all formats on your system. Since your system can contain many formats, try to select individual ones.

Figure 30 shows a typical dialog for choosing formats and the resulting output:

**Figure 30: Import and Export Tools—Listing FOREIGN FORMAT File Entries Using the Print Format Documentation Option**

```
SELECT DATA EXPORT TO FOREIGN FORMAT OPTION: PRINT FORMAT DOCUMENTATION

SELECT ONE OF THE FOLLOWING:
1 ONLY PRINT SELECTED FOREIGN FORMATS
2 PRINT ALL FOREIGN FORMATS

ENTER RESPONSE: 1 <ENTER> ONLY PRINT SELECTED FOREIGN FORMATS

SELECT FOREIGN FORMAT: 123 IMPORT NUMBERS
SELECT FOREIGN FORMAT: EXCEL - COMMA
SELECT FOREIGN FORMAT: <ENTER>
DEVICE: <ENTER>

AVAILABLE FOREIGN FORMATS NOV 2,1992 15:34 PAGE 1
-----
NAME: 123 IMPORT NUMBERS

DESCRIPTION: THIS FORMAT EXPORTS DATA FOR USE WITH LOTUS 1-2-3 SPREADSHEETS.
NON-NUMERIC FIELDS WILL BE IN QUOTES.
EACH FIELD WILL BE SEPARATED BY A SPACE.

USAGE NOTE: TO IMPORT INTO 1-2-3, CHOOSE FILE->IMPORT->NUMBERS.

OTHER NAME: LOTUS 123 (NUMBERS)

DESCRIPTION:

NAME: EXCEL - COMMA

DESCRIPTION: USE THIS FORMAT TO EXPORT DATA TO THE EXCEL SPREADSHEET ON THE
MACINTOSH. THE EXPORTED DATA WILL HAVE A COMMA BETWEEN EACH FIELD'S VALUE. THE
USER WILL BE ASKED TO SPECIFY THE DATA TYPE OF EACH EXPORTED FIELD. THOSE FIELDS
THAT ARE NOT NUMERIC WILL BE SURROUNDED BY QUOTES ("). COMMAS ARE ALLOWED IN THE
NON-NUMERIC DATA, BUT QUOTES (") ARE NOT.

USAGE NOTE:

OTHER NAME: COMMA DELIMITED

DESCRIPTION: EXPORTED DATA IS DELIMITED BY COMMAS. NON-NUMERIC DATA IS SURROUNDED
BY QUOTES.

OTHER NAME: CSV

DESCRIPTION: COMMA SEPARATED VALUES.
```

### 2.3.4.3 Define Foreign File Format Option

**Figure 31: Import and Export Tools—Define Foreign File Format Option**

```
VA FILEMAN ... [DIUSER]
OTHER OPTIONS ... [DIOTHER]
  DATA EXPORT TO FOREIGN FORMAT ... [DDXP EXPORT MENU]
    DEFINE FOREIGN FILE FORMAT [DDXP DEFINE FORMAT]
      **> LOCKED WITH DDXP-DEFINE
```

All exports depend on a Foreign Format. In addition, you can use Foreign Formats for imports as well. Usually, you can use an existing format to properly format your data for export or import.



**REF:** To find out what formats exist on your system, see the [“Print Format Documentation Option”](#) section.

If no existing format meets your needs, use the **Define Foreign File Format** [DDXP DEFINE FORMAT] option to create a new one. You can use the **Define Foreign File Format** [DDXP DEFINE FORMAT] option to:

- Define a new Foreign Format from scratch.
- Modify a Foreign Format that has *not* been used to create an **EXPORT** template.
- Copy an existing format to create a similar, modified one.

If you are using the Export Tool through Kernel’s menu system, you need the **DDXP-DEFINE** security key to use the **Define Foreign File Format** [DDXP DEFINE FORMAT] option.

[Figure 32](#) is an example of making a new format from an existing one.

The **Define Foreign File Format** [DDXP DEFINE FORMAT] option is the first option on the **Data Export to Foreign Format** [DDXP EXPORT MENU] menu, which is located under the **Other Options** [DIOTHER] menu:

**Figure 32: Import and Export Tools—Choosing the Define Foreign Format Option**

```
SELECT OPTION: OTHER OPTIONS
SELECT OTHER OPTION: DATA EXPORT TO FOREIGN FORMAT

SELECT DATA EXPORT TO FOREIGN FORMAT OPTION: DEFINE FOREIGN FILE FORMAT
```

You are first asked for the name of a format. If you want to create a new format from scratch, enter a new name. You are presented with the ScreenMan form used to define a Foreign Format (see [Figure 33](#)).



**NOTE:** Whenever you are asked to choose a FOREIGN FORMAT, you can reply with either the format's NAME or one of its OTHER NAMES.

In [Figure 33](#), an existing format's name is given:

**Figure 33: Import and Export Tools—Selecting an Existing FOREIGN FORMAT File Entry**

```
SELECT FOREIGN FORMAT: 123 IMPORT NUMBERS
123 IMPORT NUMBERS FOREIGN FORMAT HAS BEEN USED TO CREATE AN EXPORT TEMPLATE.
THEREFORE, ITS DEFINITION CANNOT BE CHANGED.
```

This format has already been used to create an **EXPORT** template. Since that template relies on the information in the FOREIGN FORMAT file's (#.44) entry at the time the template was created, you *cannot* modify this format. Instead, you are given the option of seeing what is in the format:

**Figure 34: Import and Export Tools—Viewing the Contents of a FOREIGN FORMAT File Entry**

```
DO YOU WANT TO SEE THE CONTENTS OF 123 IMPORT NUMBERS FORMAT? NO// YES
NAME: 123 IMPORT NUMBERS                FIELD DELIMITER: 032
MAXIMUM OUTPUT LENGTH: 0                 FORMAT USED?: YES
QUOTE NON-NUMERIC FIELDS?: YES          PROMPT FOR DATA TYPE?: YES
SEND LAST FIELD DELIMITER?: YES         SUBSTITUTE FOR NULL: 0
DESCRIPTION: THIS FORMAT EXPORTS DATA FOR USE WITH LOTUS 1-2-3
SPREADSHEETS. NON-NUMERIC FIELDS WILL BE IN QUOTES. EACH FIELD
WILL BE SEPARATED BY A SPACE. A 0 WILL BE EXPORTED FOR NULL-
VALUED NUMERIC FIELDS IN THE PRIMARY FILE.
USAGE NOTES: TO IMPORT INTO 1-2-3, CHOOSE FILE->IMPORT->NUMBERS.
```

As this example shows ([Figure 34](#)), the FORMAT USED? field is **YES**. This indicates that the format has been used to create an **EXPORT** template.

Whether you ask to see the contents of the format or not, you are next given the chance to make a copy of the format to modify it. You enter a name for the new format that does *not* yet exist in the FOREIGN FORMAT (#.44) file:

**Figure 35: Import and Export Tools—Creating a New FOREIGN FORMAT File Entry**

```

DO YOU WANT TO USE 123 IMPORT NUMBERS AS THE BASIS
FOR A NEW FORMAT? NO// YES <ENTER> (YES)

NAME FOR NEW FOREIGN FORMAT: CLONE 123 IMPORT NUMBERS

ARE YOU ADDING CLONE 123 IMPORT NUMBERS AS
A NEW FOREIGN FORMAT (THE 22ND)? NO// Y <ENTER> (YES)

```

When the new format has been created, you are given the opportunity to modify it. The ScreenMan form in [Figure 36](#) is used for editing Foreign Formats:

**Figure 36: Import and Export Tools—ScreenMan Form for Editing Foreign Formats**

```

FOREIGN FILE FORMAT: CLONE 123 IMPORT NUMBERS PAGE 1
=====

FIELD DELIMITER: 032 RECORD LENGTH FIXED?
SEND LAST DELIMITER? YES MAXIMUM OUTPUT LENGTH: 0
RECORD DELIMITER: NEED FOREIGN FIELD NAMES?

FILE HEADER:
FILE TRAILER:
DATE FORMAT:

SUBSTITUTE FOR NULL: 0
QUOTE NON-NUMERIC? YES
PROMPT FOR DATA TYPE? YES

GO TO NEXT PAGE TO DOCUMENT FORMAT.

-----
COMMAND: PRESS <PF1>H FOR HELP INSERT

```



**REF:** The meaning of the fields on this page of the form is described in the [“FOREIGN FORMAT File Attributes Reference”](#) section.

You are presented with the same form whether you are modifying an existing format or creating one from scratch.



**TIP:** It is important to always create and edit formats using the Data Export options, because validity checks on the relationships between the various fields are built into the ScreenMan form. If you enter inconsistent data, you are alerted when you try to exit the form.

There is a second page of the form that contains documenting information about the format. The second page allows you to enter a description and usage notes for the format. You can also enter other names for the format (in a Multiple); these other names can then be used to reference the format anywhere in the Export or Import Tools.

[Figure 37](#) is what the second page looks like with the Multiple's "popup" window opened:

**Figure 37: Import and Export Tools—Second Page of a Multiple's "Popup" Window Opened**

```
FOREIGN FILE FORMAT: CLONE 123 IMPORT NUMBERS          PAGE 2
=====
DESCRIPTION (WP):
USAGE NOTES (WP):

SELECT OTHER NAME FOR FORMAT: LOTUS 123 (NUMBERS)

  OTHER NAME: LOTUS 123 (NUMB
  DESCRIPTION (WP):

COMMAND:                                PRESS <PF1>H FOR HELP      INSERT
```

After you have completed and filed the ScreenMan forms, you are returned to the Data Export submenu. You can now use the new format to create an **EXPORT** template or do an import.

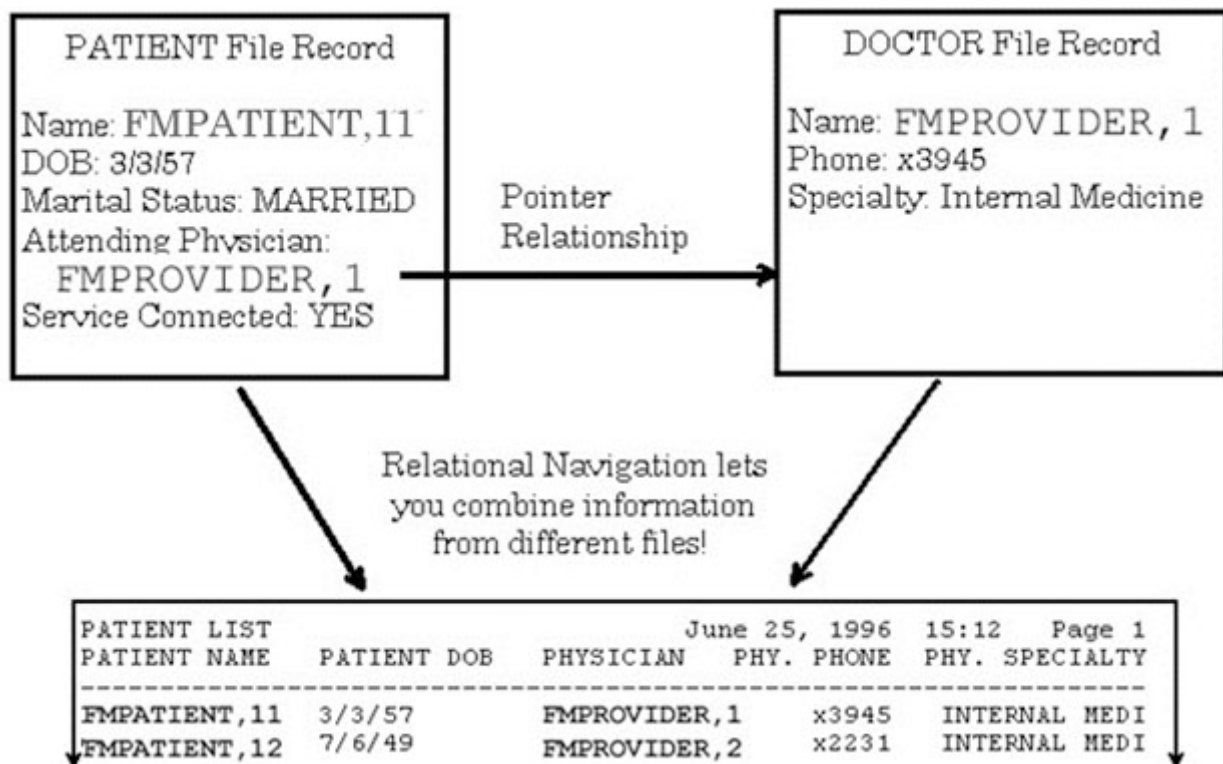
### 3 Relational Navigation

Relational navigation gives you a way to reach beyond the current file to reference fields within other files.

Suppose, for example, you are doing a printout from the PATIENT (#2) file. In the PATIENT (#2) file, there is a pointer to the (fictitious) DOCTOR file. This links a given patient to a given doctor. But the only information about the doctor available from the point of view of the PATIENT (#2) file is the doctor's name. What if, in your printout, you want to print the doctor's name, phone number, and specialty (where phone number and specialty are fields in the [fictitious] DOCTOR file)?

The answer is to use relational navigation. By using the pointer relationship between the PATIENT and the (fictitious) DOCTOR file, you can start from the PATIENT (#2) file, and for each record in the PATIENT (#2) file, retrieve not only the name of the doctor for that patient, but also additional information about the doctor from the (fictitious) DOCTOR file.

**Figure 38: Relational Navigation—Example Illustrating Relational Navigation**



You can use relational navigation in many places in VA FileMan to *move beyond the current file* and retrieve or edit information in related files' records, including:

- Reports (Print Fields, Sort Criteria, Search Criteria)
- Editing Records (edit information in related files, *not* just current file)
- Computed Expressions
- **COMPUTED** Fields
- Within word-processing **|Windows|**

The syntax to perform relational navigation, called Extended Pointer syntax, is discussed throughout this section.

Several types of pointer relationships between files can be exploited to combine information:

- [Simple Extended Pointer](#) (most common)
- [Backward Extended Pointer](#)
- [Join Extended Pointer](#)

A special form of relational navigation, called relational jumping, uses these pointer relationships to let you "jump" from one file to another. This makes it easier to specify a group of fields from another file when specifying what fields to edit, search, print, or sort by in interactive VA FileMan.

### 3.1 Simple Extended Pointer

The most common form of relational navigation uses *simple extended pointers*. This type of navigation requires a pointer field to exist from the current file to another file. Using a pointer field from an entry in the current file, you can easily retrieve information from the pointed-to entry in another file.

For example, suppose you are printing a report from the PATIENT (#2) file. Further suppose that the PATIENT (#2) file has a pointer field called ATTENDING PHYSICIAN field to the (fictitious) DOCTOR file. Now, what if you wanted to include the phone number of the attending physician for each patient in your report from the PATIENT file? The attending physician's phone number is stored in the (fictitious) DOCTOR file, *not* the PATIENT file.

You can include the attending physician's phone number for each patient in your report, by using a simple extended pointer at the "PRINT FIELD:" prompt:

**Figure 39: Relational Navigation—Example of a Simple Extended Pointer**

```
PRINT FIELD: ATTENDING PHYSICIAN:PHONE NUMBER
```

You can use simple extended pointers in many places in VA FileMan, including:

- Reports (Print Fields, Sort Criteria, Search Criteria)
- Editing Records (edit information in related files, *not* just current file)
- Computed Expressions
- **COMPUTED** Fields
- Within word-processing **|Windows|**

The syntax for simple extended pointers is described below.

### 3.1.1 Simple Extended Pointer Syntax (Short form)

With simple extended pointers, there *must* be an existing relationship based on a pointer field from the current file to the file you are interested in. In this case, you can reference a field in a pointed-to entry by using the following syntax:

**PFIELD:ELEMENT**

- "**pfield**" is the name (or number, preceded by #) of a pointer field in the current file.
- "**element**" is an element that exists in the field to which **pfield** points.

This is called the short form of extended pointer syntax.

For example, since ATTENDING PHYSICIAN is a pointer field in the current file to the (fictitious) DOCTOR file, the short form of extended pointer syntax to reference the PHONE NUMBER field in the (fictitious) DOCTOR file would be:

**ATTENDING PHYSICIAN:PHONE NUMBER**

### 3.1.2 Simple Extended Pointer Syntax (Long Form)

The most complete or general form of extended pointer syntax (also called long form) is shown below:

EXPR:FILE:ELEMENT

OR

EXPR IN FILE FILE:ELEMENT

“**Expr**” is any expression that applies to the file that is your current context. “**File**” is the name of any file. “**Element**” is any element (field) in the file named by “File”.

For example, since ATTENDING PHYSICIAN is a pointer field in the current file to the (fictitious) DOCTOR file, the long form of extended pointer syntax to reference the PHONE NUMBER field in the (fictitious) DOCTOR file would be:

ATTENDING PHYSICIAN:DOCTOR:PHONE NUMBER

OR

ATTENDING PHYSICIAN IN FILE DOCTOR:PHONE NUMBER

### 3.1.3 Examples

#### 3.1.3.1 Relational Query Example

You can use simple extended pointers to make relational queries. For example, suppose you want to print all patients who are older than their attending physicians. A field in the PATIENT file called ATTENDING PHYSICIAN points to the (fictitious) DOCTOR file. Given a field PT AGE in the PATIENT file and a field DR AGE in the (fictitious) DOCTOR file, you can use the **Print File Entries** [DIPRINT] option and then enter the information shown in [Figure 40](#):

**Figure 40: Relational Navigation—Example of a Relational Query**

```
OUTPUT FROM WHAT FILE: PATIENT
SORT BY: NAME// PT AGE> (ATTENDING PHYSICIAN:DR AGE)
WITHIN PT AGE>(ATTENDING PHYSICIAN:DR AGE), SORT BY: <ENTER>

FIRST PRINT FIELD: NAME
```

Here, the simple extended pointer (ATTENDING PHYSICIAN:DR AGE) is used to make a comparison between values in fields in two different files.

### 3.1.3.2 COMPUTED Field Example

Suppose the PATIENT file has an ATTENDING PHYSICIAN field that points to the (fictitious) DOCTOR file. The (fictitious) DOCTOR file, in turn, has a field called SPECIALTY. If you want to create a **COMPUTED** field within the PATIENT (#2) file data dictionary that is equivalent to the SPECIALTY field in the (fictitious) DOCTOR file, you can define a **COMPUTED** field as:

**Figure 41: Relational Navigation—Example of the Short Form Extended Pointer Syntax**

```
□COMPUTED-FIELD□ EXPRESSION: ATTENDING PHYSICIAN:SPECIALTY
```

The file does *not* have to be specified in this case, since there is a direct link between the two files through the pointer field. This is an example of the **short form** of the simple extended pointer syntax.

An equivalent computed expression that explicitly identifies the file is: ATTENDING PHYSICIAN IN DOCTOR FILE:SPECIALTY. This is the **long form** of the syntax. It is “long” because the file name is included.

### 3.1.4 How to Navigate With a Variable Pointer Field

If the pointing field is a **VARIABLE POINTER**, the long form of the extended pointer syntax *must* be used so that VA FileMan knows which of the pointed-to files to search. Here is the syntax:

VPFIELD IN FILE FILE:ELEMENT

OR

VPFIELD:FILE:ELEMENT

“**Vpfield**” is the **VARIABLE POINTER** field in the current file, “**file**” is one of the possible pointed-to files, and “**element**” applies to that pointed-to file.

[Figure 42](#) is an example from the PATIENT (#2) file where the PROVIDER field is a **VARIABLE POINTER** to either the (fictitious) PHYSICIAN file or the (fictitious) PERSON file, and PHONE is a field in the (fictitious) PERSON file. You could enter the print specifications shown in [Figure 42](#):

**Figure 42: Relational Navigation—Entering Print Specifications and Including Fields in Pointed-To Files**

```
FIRST PRINT FIELD: NAME
THEN PRINT FIELD: PROVIDER
THEN PRINT FIELD: FILE (PROVIDER)
THEN PRINT FIELD: PROVIDER: PERSON: PHONE
THEN PRINT FIELD: <ENTER>
```

You receive the output shown in [Figure 43](#):

**Figure 43: Relational Navigation—Example of Output that Includes Fields from Pointed-To Files**

NAME	PROVIDER	FILE (PROVIDER)	PROVIDER: PERSON: PHONE
-----	-----	-----	-----
FMPATIENT, 13	FMPROVIDER, 3	PHYSICIAN	
FMPATIENT, 14	FMPROVIDER, 4	PERSON	555-3332

The long form simple pointer asked for the PHONE field from the PERSON file. Only the **VARIABLE POINTER** from the FMPATIENT,14 entry pointed to the (fictitious) PERSON file. Thus, only his phone number is displayed.

## 3.2 Relational Jumps across Files

In interactive VA FileMan mode, you can use the following syntax:

FILE:

This syntax changes your context to the file you specify; you “jump” to the specified file. You can then select fields from the file to which you have jumped. You can only do this in four places in VA FileMan:

- “EDIT WHICH FIELD:” prompt (**Enter or Edit File Entries** [DIEDIT] option)
- “SEARCH FOR FIELD:” prompt (**Search File Entries** [DISEARCH] option)
- “SORT BY:” prompt (**Print File Entries** [DIPRINT] and **Search File Entries** [DISEARCH] options)
- “PRINT FIELD:” prompt (**Print File Entries** [DIPRINT] and **Search File Entries** [DISEARCH] options)

Relational jumping is mainly a convenience to make it easier to select more than one field from another file. By letting you temporarily “jump” to the other file, it’s easier to pick all the fields you want directly, rather than having to use extended pointer syntax to specify each field.

**i** **NOTE:** When sorting, printing, searching, or editing, if you want to reference several fields from another file, it is more efficient to jump to the file and specify the needed fields than it is to use the extended pointer syntax to reference the fields one at a time. Multiple uses of the extended pointer cause multiple relational jumps.

[Table 4](#) lists the four types of relational jumps that correspond to the four extended pointer syntax:

**Table 4: Relational Navigation—Relational Jumps that Correspond to Extended Pointer Syntax**

Type	Example
<b>Simple (short form)</b>	ATTENDING PHYSICIAN:
<b>Simple (long form)</b>	PROVIDER IN PERSON FILE:
<b>Backward</b>	RADIOLOGY EXAM:
<b>Join</b>	PAYSCALE IN FACTOR FILE:

Within the **Enter or Edit File Entries** [DIEDIT] option, for example, you can respond to the prompts as depicted in the dialog that follows:

**Figure 44: Relational Navigation—Using Relational Jumps with the Enter or Edit File Entries Option**

```

INPUT TO WHAT FILE: PATIENT
EDIT WHICH FIELD: ALL// NAME
THEN EDIT FIELD: ATTENDING PHYSICIAN:

```

**Relational Jump!**

```

EDIT WHICH DOCTOR FIELD: ALL// NAME; [PHYSICIAN NAME]
THEN EDIT DOCTOR FIELD: NICKNAME
THEN EDIT DOCTOR FIELD: <ENTER>
THEN EDIT FIELD: <ENTER>

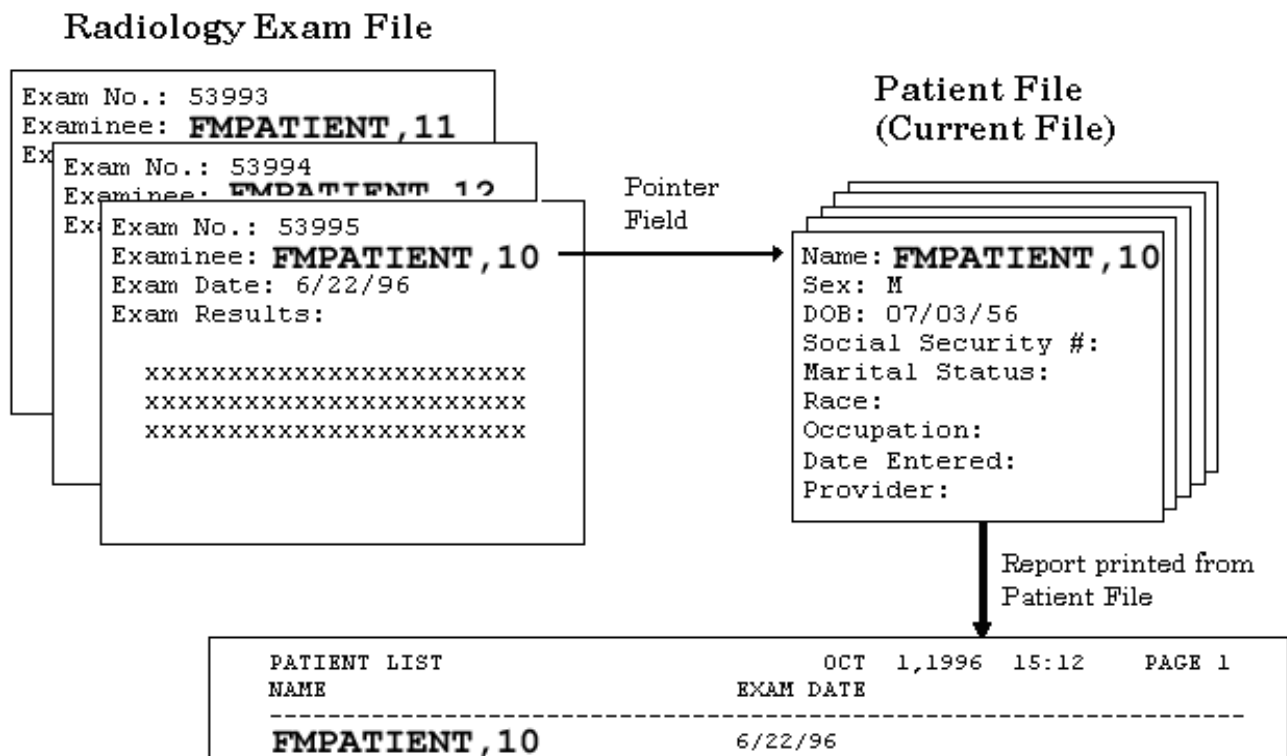
```

Because of a pointer linkage between the ATTENDING PHYSICIAN field in the PATIENT (#2) file and the (fictitious) DOCTOR file, you can use the simple, short form of the extended pointer to navigate to the (fictitious) DOCTOR file. Then, during an interactive editing session you can specify the fields you want to edit for each patient. In this case, after you edit the patient's name, you can edit that patient's physician's name and nickname.

### 3.3 Backward Extended Pointer

Simple extended pointers let you retrieve information from an entry in another file that the current entry explicitly points to through a POINTER TO A FILE field. What if you wanted to go the other way—retrieve information from an entry in another file that points to (*not from*) the current entry?

**Figure 45: Relational Navigation—Example Illustrating a File with Pointers to another File**



Suppose you have selected the PATIENT (#2) file, and you want to list dates of radiology exams for certain patients. If the pointer is from the RADIOLOGY EXAM file to the PATIENT file (*not from*), you can list the radiology exam dates using a **Backward Extended Pointer**.

In the file that contains the **POINTER TO A FILE** field, one of the following three conditions *must* be true:

1. Either a New-Style or Traditional cross-reference on the field exists. If the **POINTER TO A FILE** field is in a subfile Multiple, the whole file *must* be cross-referenced. Compound cross-references can be used if the first subscript in the cross-reference is the pointer value with no transforms. The use of a compound cross-reference can result in “navigation” to only a subset of the pointing entries. Even though a record can have a valid **POINTER TO A FILE** field, unless all the other fields that make up subscripts on the compound index are also *non-NULL*, there is no entry in the index for that record.
2. The **.001** field of the file is the pointing field.
3. The **.01** field of the pointing file is the pointing field, and there is a “**DINUM**” condition on the field.

To use a `,` you *must* make a relational jump from the current file to the file in question (enter the name of the file pointing to the current file, followed by a colon). Once you make the relational jump to the backwards-pointer-linked file, specify which fields/elements to access in that file.

Returning to the situation mentioned above, within the RADIOLOGY EXAM file there is a field called EXAMINEE pointing back to the PATIENT (#2) file. That EXAMINEE pointer field is cross-referenced. You want to list the EXAM DATE field from the RADIOLOGY EXAM file entries that point back to a patient. From the PATIENT (#2) file, enter:

**Figure 46: Relational Navigation—Example Using a Backward Extended Pointer**

```
FIRST PRINT FIELD: NAME;N;S1
THEN PRINT FIELD: RADIOLOGY EXAM:

Relational Jump!

BY [RADIOLOGY EXAM], DO YOU MEAN THE RADIOLOGY EXAM FILE,
  POINTING VIA ITS [EXAMINEE] FIELD? YES// <ENTER> (YES)
THEN PRINT RADIOLOGY EXAM FIELD: EXAM DATE
THEN PRINT RADIOLOGY EXAM FIELD: <ENTER>
THEN PRINT FIELD: <ENTER>
```

As indicated by this example, you did *not* have to specify the EXAMINEE field. That field was identified because it is a field in the RADIOLOGY EXAM file that points back to the current file.

[Figure 47](#) is the output produced by these print specifications:

**Figure 47: Relational Navigation—Example of the Output Produced after Using a Backward Extended Pointer**

PATIENT LIST	OCT 1,1996 15:12	PAGE 1
NAME	EXAM DATE	
FMPATIENT,13	DEC 22,1995	
FMPATIENT,14		
FMPATIENT,15	1995 1993	
FMPATIENT,10	SEP 29,1995 JUN 22,1996	

The resulting output is a two-column report containing names from the PATIENT file and corresponding examination dates from the RADIOLOGY EXAM file. Since there may be several RADIOLOGY EXAM file entries for a given patient, this report is an example of a Multiple-valued (Multiline) result being returned.



**REF:** For more information on Multiline results being returned, see the "[Multiline Return Values](#)" section.

You can use Backwards Extended Pointers in the following places in VA FileMan:

- "EDIT WHICH FIELD:" prompt (**Enter or Edit File Entries** [DIEDIT] option)
- "SEARCH FOR FIELD:" prompt (**Search File Entries** [DISEARCH] option)
- "SORT BY:" prompt (**Print File Entries** [DIPRINT] and **Search File Entries** [DISEARCH] options)
- "PRINT FIELD:" prompt (**Print File Entries** [DIPRINT] and **Search File Entries** [DISEARCH] options)

## 3.4 Join Extended Pointer

You can establish an extended pointer link even if there is no pre-existing pointer relationship between the two files. You use a value from one file to do a lookup in a second file.

Suppose you store in the (fictitious) PAY FACTOR file a list of factors for calculating taxes. Each entry in this file corresponds to a different pay scale. In the (fictitious) PERSONNEL file, you have a field called PAYSCALE. You want to retrieve the value of a field DEDUCTION in the PAY FACTOR entry that equals the PAYSCALE field for each entry in the (fictitious) PERSONNEL file. You can create a **COMPUTED** field expression in the (fictitious) PERSONNEL file:

**Figure 48: Relational Navigation—Using a Value from One File to Do a Lookup in a Second File**

```
□COMPUTED-FIELD□ EXPRESSION: PAYSCALE IN PAY FACTOR FILE:DEDUCTION
```



**NOTE:** PAYSCALE was *not* defined as pointing to the (fictitious) PAY FACTOR file. The link to that file is made by the **COMPUTED** field definition. PAYSCALE could itself be a **COMPUTED** field. In this situation, the value of the PAYSCALE field in the (fictitious) PERSONNEL file is used to do a normal lookup in the (fictitious) PAY FACTOR file using all lookup type cross-references.

In database terminology, this extended pointer capability is like a **JOIN** operation, because you can specify at any time a new relationship between two formerly unrelated files. Therefore, this type of pointing is called the Join Extended Pointer.

### 3.4.1 Limitations

If the join expression matches more than one entry in the file being joined, the first matching entry (by internal entry number) is returned as the result of the join. Thus, if your join expression is likely to match more than one entry, be aware that only the *first* matching entry is returned.

### 3.4.2 Example

You could find out if any entries in the PERSONNEL file could be matched against the NAME field in the PATIENT file just by specifying the sort shown in [Figure 49](#):

**Figure 49: Relational Navigation—Example of Matching Entries in Two Files Using the SORT BY Field**

OUTPUT FROM WHAT FILE: <b>PATIENT</b>
SORT BY: <b>NAME IN PERSONNEL FILE</b>

The expression at the “SORT BY:” prompt selects entries in the PERSONNEL file where the value of the NAME field in the PATIENT file matches the PERSONNEL file’s **.01** field. The PATIENT file’s NAME field is being used as a lookup in the PERSONNEL file. Since you are evaluating the **.01** field of the PERSONNEL file, the “:**element**” part of the extended pointer syntax is unnecessary.

### 3.5 Multiline Return Values

When you use extended pointer syntax, a lookup is performed in the navigated-to file. This lookup usually evaluates to a single value. However, in some situations, extended pointer syntax can end up returning a Multiple-valued or “Multiline” result. Multiline responses can be generated by:

- Simple Pointer to a **WORD-PROCESSING** Field
- Simple Pointer to a Multiple
- Backward Pointer

You *cannot* use extended pointer syntax that can evaluate to a Multiline value at VA FileMan’s “SORT BY:” and “SEARCH FOR FIELD:” prompts. Some of the ways in which you *can* use extended pointers that evaluate to a Multiline value are:

- As the definition of a **COMPUTED** field.
- Within word-processing **[Windows]** (so one document can call another document to print inside it).
- For input to word-processing data elements (so you can use the **Enter or Edit File Entries** [DIEDIT] option to stuff one document into another).
- As the name of a transfer document in the **Line Editor’s Transfer** option.
- As a Print Field: specification in the **Print File Entries** [DIPRINT] option.
- In an **INPUT** template when a multi-valued field is being edited.

### 3.5.1 WORD-PROCESSING Field

**WORD-PROCESSING** field names (or field numbers preceded with a #) are allowed as elements in extended pointer expressions. For example, in the PATIENT (#2) file the HISTORY field is in the DIAGNOSIS Multiple. You can define this computed expression:

**Figure 50: Relational Navigation—Example of Using a WORD-PROCESSING Field in an Extended Pointer Expression**

```
[B-12 DEFICIENCY] IN DIAGNOSIS FILE:HISTORY
```

This Multiline computed expression would signify the WORD-PROCESSING HISTORY field text associated with a patient’s B-12 Deficiency DIAGNOSIS. A lookup is done on the DIAGNOSIS Multiple using “**B-12 Deficiency**” as the lookup value. If the patient does *not* have that DIAGNOSIS (or no HISTORY is associated with it), the value of this extended pointer expression would be **NULL**.

### 3.5.2 Multiples

You can use the simple pointer syntax to get data from Multiples of files pointed to by other files. The RADIOLOGY EXAM file described above points to the PATIENT (#2) file by way of the EXAMINEE field. In the PATIENT (#2) file there is a DIAGNOSIS Multiple. You could obtain a list of diagnoses associated with RADIOLOGY EXAM file entries by doing what is shown in [Figure 51](#):

**Figure 51: Relational Navigation—Example of Using the Simple Pointer Syntax to Get Data from a Multiple**

```
SELECT OPTION: PRINT FILE ENTRIES  
  
OUTPUT FROM WHAT FILE: RADIOLOGY EXAM// <ENTER>  
SORT BY: NAME// <ENTER>  
START WITH NAME: FIRST// <ENTER>  
FIRST PRINT FIELD: TEST NUMBER  
THEN PRINT FIELD: EXAMINEE:DIAGNOSIS  
THEN PRINT FIELD: <ENTER>  
HEADING: RADIOLOGY EXAM LIST// <ENTER>  
STORE PRINT LOGIC IN TEMPLATE: EXAM DIAGNOSES
```

For each entry in the RADIOLOGY EXAM file, EXAMINEE points to an entry in the PATIENT (#2) file. The diagnoses associated with that patient are returned as the Multiline output of the expression EXAMINEE:DIAGNOSIS.

### 3.5.3 Backward Pointer

Figure 52 shows how you can use the cross-referenced Backward Pointer that yields a Multiline response in an **INPUT** template:

**Figure 52: Relational Navigation—Example Using a Cross-Referenced Backward Pointer to Yield a Multiline Response: Stored in an INPUT Template**

```
INPUT TO WHAT FILE: PATIENT
EDIT WHICH FIELD: ALL// NAME
THEN EDIT FIELD: RADIOLOGY EXAM:
  BY  RADIOLOGY EXAM, DO YOU MEAN THE RADIOLOGY EXAM FILE,
  POINTING VIA ITS  EXAMINEE FIELD? YES// <ENTER> (YES)
WILL TERMINAL USER BE ALLOWED TO SELECT PROPER ENTRY IN  RADIOLOGY EXAM FILE?
YES// <ENTER> (YES)
DO YOU WANT TO PERMIT ADDING A NEW  RADIOLOGY EXAM ENTRY? NO// <ENTER>
  EDIT WHICH RADIOLOGY EXAM FIELD: DATE OF EXAM
  THEN EDIT WHICH RADIOLOGY EXAM FIELD: RESULTS
  THEN EDIT WHICH RADIOLOGY EXAM FIELD: <ENTER>
THEN EDIT FIELD: ATTENDING PHYSICIAN
THEN EDIT FIELD: <ENTER>
STORE THESE FIELDS IN TEMPLATE: PATIENT-EXAM
```

To use this template, you do the following:

1. Specify the patient's name to edit.
2. Select one of the RADIOLOGY EXAM file's entries that point back to that patient.
3. Edit data within that selected entry in the RADIOLOGY EXAM file.
4. Return to edit another field in the PATIENT (#2) file.

A sample editing session using this **INPUT** template looks like this:

**Figure 53: Relational Navigation—Example Using an INPUT Template with a Cross-Referenced Backward Pointer to Yield a Multiline Response**

```
INPUT TO WHAT FILE: PATIENT
EDIT WHICH FIELD: ALL// [PATIENT-EXAM
SELECT PATIENT NAME: FMPATIENT,11
NAME: FMPATIENT,11// <ENTER>
SELECT RADIOLOGY EXAM: ?
CHOOSE FROM:
1. DEC 4, 1984
2. OCT 1, 1985
CHOOSE 1-2: 2
DATE OF EXAM: OCT 1, 1985// <ENTER>
RESULTS: NORMAL
ATTENDING PHYSICIAN: FMPATIENT// <ENTER>
```

As indicated by this example, the only RADIOLOGY EXAM file entries you were allowed to choose were the two that pointed back to the selected patient (FMPATIENT,11).

Each file, for the purpose of this editing sequence, is considered a subfile of the original, so that when no more fields within the second file are specified, the dialog falls back to the original file. Having navigated over to a second file, you can use another extended pointer to move to still a third file.

You *cannot* cross file boundaries on input unless you have **WRITE** access to the file to which you move. This restriction applies to the individual who created this Patient-Exam **INPUT** template.

## 4 Advanced Edit Techniques

### 4.1 Field Value Stuffing

You can make the editing process quicker, easier, and more accurate by “stuffing” field values, when appropriate. The amount of data that needs to be entered from the keyboard can be reduced by providing responses that can be verified by pressing the **Enter** key or that are automatically put into the file.

#### 4.1.1 Set Field Default (2 //)

You can require a particular field to default to a certain data value by answering the “EDIT WHICH FIELD:” prompt with the name of the field followed with two slashes (//) and the default value.

For example, if you enter:

**Figure 54: Advanced Edit Techniques—Setting a Default Value for a Field**

```
EDIT WHICH FIELD: SEX//MALE
```

In this example, every time you get to the SEX field prompt for an entry in which sex has *not* yet been recorded, MALE is prompted as the default value of the SEX field.

#### 4.1.2 Stuff/Delete Field Value (3///)

VA FileMan offers a way to force a value to be inserted into the database (i.e., “stuff”), even if a different value is already on file. You simply use three slashes (“///”) instead of two:

**Figure 55: Advanced Edit Techniques—“Stuffing” a Value into a Field in the Database**

```
EDIT WHICH FIELD: SEX///MALE
```

No terminal dialog occurs when such mandatory defaults are inserted.

If you want to force the value of SEX to be deleted, you should respond as follows:

**Figure 56: Advanced Edit Techniques—Deleting a Value from a Field in the Database**

```
EDIT WHICH FIELD: SEX///@
```

After entering the at-sign (@), you would see the message shown in [Figure 57](#):

**Figure 57: Advanced Edit Techniques—Warning Message When Deleting a Value from a Field in the Database**

WARNING: THIS MEANS AUTOMATIC DELETION!!

The three-slash default's value *must* contain the external value of the field. The value is validated (using the **INPUT** transform) just as a user-supplied response is validated.

### 4.1.3 Unvalidated Stuffs: (4////)

If you have **Programmer** access, you can define a default that does *not* go through the **INPUT** transform by using four slashes ("////"). If you use this kind of default, you *must* show the internally stored value of the field.

For example, the SEX field has a DATA TYPE field value of **SET OF CODES**, where "m" stands for MALE and "f" stands for FEMALE; you could define a four-slash stuff like this:

**Figure 58: Advanced Edit Techniques—"Stuffing" Default Value into a Field in the Database—Bypassing INPUT Transform**

EDIT WHICH FIELD: SEX////M

## 4.1.4 Variable Stuffs

An even more powerful kind of default is the variable default. In this mode, you specify, *not* a literal value like the word **MALE**, but rather a field name from which to calculate the default value for each entry being edited.

One example of the usefulness of this kind of default is a case where you are editing two fields that usually have the same value. Suppose that, for a set of patients, you want to enter a NEXT OF KIN field, followed by a BENEFICIARY field. Once you have typed a patient's NEXT OF KIN, you want to see that answer as the default value of **BENEFICIARY**.

The process would look like [Figure 59](#):

**Figure 59: Advanced Edit Techniques—Example of “Stuffing” a Variable Default Value into a Field in the Database**

```
INPUT TO WHAT FILE: PATIENT
EDIT WHICH FIELD: ALL// NEXT OF KIN
THEN EDIT FIELD: BENEFICIARY//NEXT OF KIN
DO YOU MEAN NEXT OF KIN AS A VARIABLE? YES// <ENTER>
THEN EDIT FIELD: <ENTER>

SELECT PATIENT NAME: FMPATIENT,11
NEXT OF KIN: MRS CLOSERELATIVE FMPATIENT
BENEFICIARY: MRS CLOSERELATIVE FMPATIENT// <ENTER>

SELECT PATIENT NAME: FMPATIENT,14
NEXT OF KIN: MR CLOSERELATIVE FMPATIENT
BENEFICIARY: MR CLOSERELATIVE FMPATIENT// MISS CLOSERELATIVE_2 FMPATIENT
```

Here, Mrs. CLOSERELATIVE FMPATIENT ends up as both the NEXT OF KIN and BENEFICIARY for 11 FMPATIENT, while 14 FMPATIENT's NEXT OF KIN and BENEFICIARY are two distinct people.

A variable default value can be any computed expression—such as **LAST VISIT DATE+365**.



**REF:** For more information on computed expressions, see the “[Computed Expressions](#)” section.

### 4.1.5 WORD-PROCESSING Field Stuffing

The effect of stuffing values in a DATA TYPE field of **WORD-PROCESSING** is like defaults for other fields: the default value becomes the first line of the word-processing text. Also, you can stuff many lines of text into a DATA TYPE field of **WORD-PROCESSING** by use of a computed expression that has a Multiline value (e.g., another **WORD-PROCESSING**-type field).

Alternatively, you can automatically append data to a DATA TYPE field of **WORD-PROCESSING** by following the `//` or `//` with a `+` sign. This means add on the text shown in [Figure 60](#) to whatever may already be on file. Taking the example of the **WORD-PROCESSING**-type HISTORY field data in the PATIENT (#2) file:

**Figure 60: Advanced Edit Techniques—Appending Text on to a WORD-PROCESSING Field Value**

```
EDIT WHICH FIELD: DIAGNOSIS  
EDIT WHICH DIAGNOSIS SUB-FIELD: HISTORY//+ THIS CASE IS ESSENTIALLY NORMAL
```

The text string following the `//+` is appended automatically to any HISTORY field text that already exists for the chosen patient and diagnosis. If no HISTORY field text existed, the string would become Line 1 of the HISTORY field text.

When editing the entry, you see the text with the addition and can edit it in the usual way. If you use three slashes (`///`) instead of two, the addition is made, and you are *not* presented with the text to edit.

### 4.1.6 Looping (^LOOP)

The **Enter or Edit File Entries** [DIEDIT] option allows you to loop through a group of entries, without having to select each entry individually. After choosing the fields to edit, enter the entire word **^LOOP** in upper- or lowercase. Then, you can choose which entries to loop through by responding to the “EDIT ENTRIES BY:” and “START WITH ... GO TO” prompts. Answer these prompts in the same way that you respond to the “SORT BY:” and “START WITH ... GO TO” prompts in the **Print File Entries** [DIPRINT] option.



**REF:** For more details, see the “Specifying SORT BY Fields” topic in the “Print: How to Print Reports from Files” section in the *VA FileMan User Manual*.

In [Figure 61](#), all entries would be looped through:

**Figure 61: Advanced Edit Techniques—Example of “Looping” through Entries in a File**

```
EDIT WHICH FIELD: NAME
THEN EDIT FIELD: DATE OF BIRTH
THEN EDIT FIELD: <ENTER>

SELECT PATIENT NAME: ^LOOP
EDIT ENTRIES BY: NAME// <ENTER>
START WITH NAME: FIRST// <ENTER>

FMPATIENT,15
NAME: FMPATIENT,15// FMPATIENT,16
DATE OF BIRTH: APR 1, 1923// <ENTER>

FMPATIENT,11
NAME: FMPATIENT,11// <ENTER>
DATE OF BIRTH: FEB 27, 1939// JAN 27, 1939
:
```



**NOTE:** You can enter a **Sort** template at the “EDIT ENTRIES BY:” prompt.

This **^LOOP** feature, in combination with the **///-stuff** convention, makes it easy to load data values into newly created fields.



**CAUTION:** Use caution with the automatic loading and automatic deleting features of VA FileMan, since these features loop through entries and make changes *without stopping* for verification.

For example, suppose you have a patient database to which a new field called FOLLOW-UP DATE has been added. You want to create values for this field for all patients who have LAST VISIT DATEs earlier than **1977** on file. For all such patients, you want FOLLOW-UP DATE set equal to **JUNE 1, 1982**. Use the **Enter or Edit File Entries** [DIEDIT] option as shown in [Figure 62](#):

**Figure 62: Advanced Edit Techniques—Example of Loading Data into a Newly Created Field for Select Records**

```

EDIT WHICH FIELD: FOLLOW-UP DATE///JUNE 1, 1982
THEN EDIT FIELD: <ENTER>

SELECT PATIENT NAME: ^LOOP

EDIT ENTRIES BY: NAME// LAST VISIT DATE
START WITH LAST VISIT DATE: FIRST// 1900
GO TO LAST VISIT DATE: LAST// DEC 31, 1976
    WITHIN LAST VISIT DATE, EDIT ENTRIES BY: <ENTER>

...HOLD ON, PLEASE...

    FMPATIENT,17
    FMPATIENT,18
    :
```

Now, without keyboard input, the system automatically loads the **June 1, 1982**, data value into each entry's new FOLLOW-UP DATE field while looping through LAST VISIT DATEs up to **1977**.

Suppose you wanted to undo the work done in the previous example; you want to delete all these FOLLOW-UP DATEs:

**Figure 63: Advanced Edit Techniques—Example of Deleting Data from a Newly Created Field for Select Records**

```

EDIT WHICH FIELD: FOLLOW-UP DATE/// @
WARNING-THIS MEANS AUTOMATIC DELETION!
THEN EDIT FIELD: <ENTER>

SELECT PATIENT NAME: ^LOOP
EDIT ENTRIES BY: NAME// LAST VISIT DATE
START WITH LAST VISIT DATE: FIRST// 1900
GO TO LAST VISIT DATE: LAST// 12 31 76
    WITHIN LAST VISIT DATE, EDIT ENTRIES BY: <ENTER>

...JUST A MOMENT, PLEASE...

    FMPATIENT, 17
    FMPATIENT, 18
    :
```

## 4.2 INPUT Templates

### 4.2.1 Overview

Just as you can store complex output specifications in a **PRINT** or a **SORT** template for later use, you can store a long list of edit fields in an **INPUT** template. If you answer the “EDIT WHICH FIELD:” prompt at least five different times, or if you answer it with a right bracket ( ] ), you are prompted for a template name, as shown in [Figure 64](#):

**Figure 64: Advanced Edit Techniques—Storing a List of Edit Fields in an INPUT Template**

```
SELECT OPTION: ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: PATIENT
EDIT WHICH FIELD: ALL// NAME
THEN EDIT FIELD: DATE OF BIRTH
THEN EDIT FIELD: ]
THEN EDIT FIELD: <ENTER>
STORE THESE FIELDS IN TEMPLATE: UPDATE
```

In this example ([Figure 64](#)), **UPDATE** is the name of the template. You notice that brackets were *not* included.

When stored in a template, the input specifications can be easily recalled in the future without retyping them. The template name *must* be from **2 to 30 characters** in length; do *not* begin the template name with a bracket. Any field numbers (with their defaults and other qualifications, if you have specified any) are stored. When you return to this option, you can edit the same fields again in the same way by answering the “EDIT WHICH FIELD:” prompt with the name of the template enclosed in brackets, (e.g., [UPDATE]).

When you return to use an **INPUT** template in this way, you are asked to edit its field specifications. If you answer **YES**, you first see the template name, which you can then edit. Entering an at-sign (@) at the “NAME:” prompt deletes the entire **INPUT** template.

You can then edit the security codes for **READ** and **WRITE** access, and then the original answers to the “EDIT WHICH FIELD:” prompts.

If your previous answer is less than **20 characters**, it is followed by two slashes (//), after which you can re-enter the line. Longer answers are followed by “Replace” and are edited with the “**Replace...With**” syntax. Deleting with the at-sign (@) works in either case.



**REF:** The “**Replace...With**” syntax is described in the “Longer Default Responses and the ‘Replace ... With’ Editor” topic in the “VA FileMan Prompts” section in the *VA FileMan User Manual*.

To insert a new field ahead of the field being displayed, precede your line with a caret (^). When you have finished, you can save your edited **INPUT** template under the same name (use <Spacebar> <Enter>) or a new one.

You can create a special **INPUT** template by entering the right bracket ( ] ) at the "EDIT WHICH FIELD: ALL/" prompt. This template contains all the fields currently in the file and updates the template when new fields are added to the file.

**Figure 65: Advanced Edit Techniques—Creating a Special INPUT Template**

```
EDIT WHICH FIELD:  ALL// ]  
EDIT WHICH FIELD:  ALL// <ENTER>  
STORE THESE FIELDS IN TEMPLATE: EVERY FIELD
```

### 4.2.2 Branching within INPUT Templates

Sometimes, you want to dynamically control editing based on the responses given for a particular entry or on other aspects of the editing session. By using a technique called branching, the designer of an **INPUT** template can make the presentation of certain fields conditional based on the values of other fields. You *must* have **Programmer** access to set up branching. With **Programmer** access, any executable M code can be put into an **INPUT** template.

You can branch either to a field prompt elsewhere in the template or to a predefined place holder. The place holder is identified by @n, where "n" is an integer (e.g., @1).

To branch within an **INPUT** template, you enter M code at one of the "EDIT FIELD:" prompts. You set the variable **Y** to the branch destination. The **Y** variable can be given the value of a field label, a field number, or a place holder:

- If **Y** is set to **Zero** and editing is being done at the top-level of a file, the template is exited.
- If **Y** is set to **Zero** and a Multiple is being edited, the Multiple is exited.

The variable **X** contains the *updated, internal* value of the field edited at the previous prompt. Thus, you can check **X** to determine if you want to set **Y** to branch or not. For example, suppose you had a file called **ADMISSIONS**. Some of the fields are concerned only with the discharge of a patient. You want to branch around those fields if the **DATE OF DISCHARGE** is empty in the database and no date is given in the current editing session. Your template could be defined as shown in [Figure 66](#):

**Figure 66: Advanced Edit Techniques—Defining INPUT Template to Branch to Different Field Based on another Field’s Value (1 of 2)**

```

SELECT OPTION: ENTER OR EDIT FILE ENTRIES

INPUT TO WHAT FILE: ADMISSIONS
EDIT WHICH FIELD: ALL// NAME
THEN EDIT FIELD: DIAGNOSIS
THEN EDIT FIELD: ADMITTING PHYSICIAN
THEN EDIT FIELD: DATE OF DISCHARGE
THEN EDIT FIELD: S:X=  Y=@1
THEN EDIT FIELD: DISCHARGING PHYSICIAN
THEN EDIT FIELD: FOLLOW-UP DATE
THEN EDIT FIELD: @1
THEN EDIT FIELD: BILLING METHOD
THEN EDIT FIELD: <ENTER>
STORE THESE FIELDS IN TEMPLATE: EDIT ADMISSION
ARE YOU ADDING [EDIT ADMISSION] AS A NEW INPUT TEMPLATE? Y <ENTER> (YES)

```

This template branches around the discharge-related questions, if the **DATE OF DISCHARGE** is **NULL**.

If you wanted to further enhance the template to ask for MEDICARE NUMBER only if BILLING METHOD is **M** (for Medicare), you could change the template, as shown in [Figure 67](#):

**Figure 67: Advanced Edit Techniques—Defining INPUT Template to Branch to Different Field Based on another Field's Value (2 of 2)**

```

INPUT TO WHAT FILE: ADMISSIONS// <ENTER>
EDIT WHICH FIELD: ALL// [EDIT ADMISSION] <ENTER> (OCT 31, 1991@14:17)
  USER #2 FILE #16155
WANT TO EDIT EDIT ADMISSION  INPUT TEMPLATE? NO// Y <ENTER> (YES)
NAME: EDIT ADMISSION// <ENTER>
READ ACCESS: @// <ENTER>
WRITE ACCESS: @// <ENTER>
EDIT WHICH FIELD: .01// <ENTER> NAME
THEN EDIT FIELD: 1// <ENTER> DIAGNOSIS
THEN EDIT FIELD: 2// <ENTER> ADMITTING PHYSICIAN
THEN EDIT FIELD: 3// <ENTER> DATE OF DISCHARGE
THEN EDIT FIELD: S:X= Y=@1// <ENTER>
THEN EDIT FIELD: 4// <ENTER> DISCHARGING PHYSICIAN
THEN EDIT FIELD: 5// <ENTER> FOLLOW-UP DATE
THEN EDIT FIELD: @1// <ENTER>
THEN EDIT FIELD: 6// <ENTER> BILLING METHOD
THEN EDIT FIELD: 7// S:X=M Y=MEDICARE NUMBER
THEN EDIT FIELD: S Y=0
THEN EDIT FIELD: MEDICARE NUMBER
THEN EDIT FIELD: <ENTER>
STORE THESE FIELDS IN TEMPLATE: <SPACEBAR><ENTER> EDIT ADMISSION
(OCT 31, 1991@14:17)      USER #2 FILE #16155
EDIT ADMISSION TEMPLATE ALREADY EXISTS... OK TO REPLACE? Y <ENTER> (YES)

```

After the BILLING METHOD field is edited, a test is made of its contents. It is a DATA TYPE field of **SET OF CODES**; thus, the test is for the letter **M** alone (the internal value of the field). If it is equal to **M**, the template branches to the MEDICARE NUMBER field. If it is *not* equal to **M**, the template proceeds to the next prompt where **Y** is set unconditionally to **Zero**. The template is exited here so that the “MEDICARE NUMBER” prompt is *not* shown when it is *not* needed.

An editing session using this template to add a new admission might look like [Figure 68](#):

**Figure 68: Advanced Edit Techniques—Example Verifying Automatic Branching to Other Fields Based on User's Entry (1 of 2)**

```

SELECT ADMISSIONS NAME: FMPATIENT,19
ARE YOU ADDING FMPATIENT,19 AS A NEW ADMISSIONS (THE 4TH)? NO// Y <ENTER>
(YES)
DIAGNOSIS: MEASLES
ADMITTING PHYSICIAN: FM PROVIDER,4
DATE OF DISCHARGE: <ENTER>
BILLING METHOD: M <ENTER> MEDICARE
MEDICARE NUMBER: 3093-0393

```

The discharge related questions were skipped, and the “MEDICARE NUMBER:” prompt was given. A future editing of this record upon patient discharge could look like [Figure 69](#):

**Figure 69: Advanced Edit Techniques—Example Verifying Automatic Branching to Other Fields Based on User’s Entry (2 of 2)**

```
SELECT ADMISSIONS NAME: FMPATIENT,19
...OK? YES// <ENTER> (YES)

NAME: FMPATIENT,19// ^DATE OF DISCHARGE
DATE OF DISCHARGE: 5/9/90 <ENTER> (MAY 09, 1990)
DISCHARGING PHYSICIAN: FM PROVIDER,4
FOLLOW-UP DATE: 6/1/90 <ENTER> (JUN 01, 1990)
BILLING METHOD: MEDICARE// <ENTER>
MEDICARE NUMBER: 3093-0393// <ENTER>
```

There is a potential hazard in using branching. In this example, suppose the BILLING METHOD were changed to “P” (for private insurance). The simple branching logic used would *not* show you the MEDICARE NUMBER field to edit or delete. You *must* ensure that your template can handle this kind of situation. In this example, if you have **Programmer** access to do so, you might add M code to delete the MEDICARE NUMBER, if BILLING METHOD were *not* equal to “M”.

## 4.3 Edit Qualifiers

### 4.3.1 Edit Qualifiers and Customizing Data Editing

When creating an **INPUT** template, there are several ways you can control the editing session to display customized prompts, to enable the duplication of data by pressing the **Spacebar** and the **Enter** keys (<**Spacebar**> <**Enter**>), and to make a field required.

[Table 5](#) summarizes the edit qualifiers you can use to accomplish these results. They are described in more detail in the next three sections. Enter these qualifiers in conjunction with fields at the "EDIT FIELD:" prompt.

**Table 5: Advanced Edit Techniques—Edit Qualifiers**

Qualifier	Action
<b>field; "xxx"</b>	Replace the field's label with a literal string during an editing session (see the <a href="#">"Forcing Special Prompts"</a> section).
<b>field;T</b>	Replace a field's label with its title during an editing session (see the <a href="#">"Forcing Special Prompts"</a> section).
<b>field;DUP</b>	Save responses for later use with < <b>Spacebar</b> > < <b>Enter</b> > and allow their recall (see the <a href="#">"Duplicating Input Values"</a> section).
<b>field;REQ</b>	Require a response to a field that is usually <i>not</i> required (see the <a href="#">"Forcing Special Prompts"</a> section).

You can combine specifiers if you separate them with semicolons (e.g., **DATE OF BIRTH;T;REQ**).

### 4.3.2 Forcing Special Prompts

Normally, the standard label or name of a field is used to ask the user for the input value of that field. You can customize the prompt for a field by answering the "EDIT WHICH FIELD:" prompt with the label, followed by a semi-colon (;) and the desired prompt in quotation marks. Thus:

```
EDIT WHICH FIELD: DATE OF BIRTH; "DOB"
```

Causes the DATE OF BIRTH field to be presented in the form:

```
DOB:
```

Or, in the form:

DOB: APR 1, 1923//

To use the field's title instead of its label as the input prompt, follow the field name (or number) with ;**T**. Thus, when editing the PATIENT (#2) file, you can enter:

**Figure 70: Advanced Edit Techniques—Example Using the Title Edit Qualifier**

```
EDIT WHICH FIELD: .01 <ENTER> NAME
THEN EDIT FIELD: SSN;T
THEN EDIT FIELD: <ENTER>
```

If you enter these specifications *and* if this field's title is defined as "Social Security Number," the user encounters the "Social Security Number:" prompt instead of the "SSN:" prompt.

### 4.3.3 Duplicating Input Values

Sometimes many entries need the same data value input for a particular field. If you follow a field label with ;**DUP** when selecting the field for editing, VA FileMan uses the data value that was just input for the prior entry, if you enter a single space character (<**Spacebar**> <**Enter**>) at the field prompt. For example:

**Figure 71: Advanced Edit Techniques—Example Using the Duplicate Edit Qualifier**

```
EDIT WHICH FIELD: SEX;DUP
```



**NOTE:** If all entries have the same data value, you can instead use the ^**LOOP** facility described earlier in this section.

### 4.3.4 Forcing Required Input

When creating an **INPUT** template, VA FileMan allows you to designate fields as required. Designating a field as required means that the user *must* enter data in that field. To do this, follow the field name with ;**REQ**. The required specification looks like [Figure 72](#):

**Figure 72: Advanced Edit Techniques—Example Using the Required Edit Qualifier**

```
EDIT WHICH FIELD: NAME;REQ
```

Adding **;REQ** does *not permanently* affect the definition of the field. It is only effective for the current input session or for the specific **INPUT** template. To *permanently* make a field mandatory, use the **Modify File Attributes** [DIMODIFY] option.

## 4.4 Text Formatting in Word-Processing Fields

### 4.4.1 Word Wrapping

Word wrapping is performed when a **WORD-PROCESSING**-type field is printed. Two functions occur as part of word wrapping during prints:

- Lines are “filled” to the right margin.
- Lines are “broken” only at word breaks.

If word wrap is on (a data dictionary setting for the **WORD-PROCESSING**-type field in question), you can *override* the word wrapping function and force a line to be printed as it appears in the editor by doing one of the following with the line:

- Starting the line with a space.
- Pressing the **Tab** key at the end of the line while using the Line Editor, or type **|Tab|** at the end of the line while using the Screen Editor.
- Turning wrap off by using the **|NOWRAP|** function described below.

Lines that contain only punctuation are always printed as is. Thus, if you put a single space on a line, the previous line is *not* filled and the subsequent line begins in column one.



**NOTE:** The editor’s line numbers are meaningful only when editing. Since word-processing data is usually printed in a wraparound mode, what is internally line three might be printed as lines five and six.

### 4.4.2 Tabs

Tabs can be meaningful wherever they occur in a line.



**NOTE:** If you insert a tab by typing the special **Tab** key on the keyboard (or **<Ctrl-I>** on terminals without a **Tab** key), a **|Tab|** is inserted in the text instead. When editing, a tab is recognized as **|Tab|**, *not* as five blank spaces.

### 4.4.3 Formatting Text with Word-processing Windows (Frames) | |

Expressions framed by vertical bars (“| |”) are known as word-processing *windows* or *frames* and are evaluated as computed expression at print-time and are printed as evaluated. (MailMan does *not typically* evaluate expressions within vertical bars, neither does the **Inquire to File Entries** [DIINQUIRE] option or the **CAPTIONED PRINT** template.) For example, **|TODAY+1|** prints out tomorrow’s date.

You can use word-processing windows to insert one of the following into the text of a **WORD-PROCESSING**-type field when that **WORD-PROCESSING**-type field is printed:

- A Field Name.
- A Computed Expression.
- Text Formatting Expression.




**REF:** For details of how to compose and use computed expressions, see the [“Computed Expressions”](#) section.

#### 4.4.4 Text Formatting Expressions in Word-Processing Windows

[Table 6](#) lists the recognized special text formatting functions that you can use within word-processing windows. Most of these functions can be used in other contexts—for example, at the “PRINT FIELD:” prompt.

**Table 6: Advanced Edit Techniques—Text Formatting Expressions in Word-Processing Windows**

Text Formatting Expression	Description
<b> RIGHT-JUSTIFY </b>	Causes the text that follows it to be padded with spaces between words, so the right margin is even.
<b> DOUBLE-SPACE </b>	Causes the text that follows it to be printed with blank lines inserted every other line.
<b> SINGLE-SPACE </b>	Turns off double-spacing for the text that follows it.
<b> TOP </b>	Causes a page break to occur at this point.
<b> NOBLANKLINE </b>	If nothing is printed on the line, this causes the line to be suppressed so that a blank line is <i>not</i> output. It is useful if the line contains only a computed expression that might evaluate to <b>NULL</b> .
<b> PAGEFEED (arg) </b>	Causes page breaks to occur in the text that follows it, whenever fewer than <b>arg</b> number lines remain on the current page.
<b> PAGESTART (arg) </b>	Causes the text on the following pages to begin at line # <b>arg</b> of the page.
<b> SETPAGE (arg) </b>	Resets page numbering, so that the page number that follows it is <b>arg + 1</b> .
<b> BLANK (arg) </b>	Causes arg number of blank lines to be inserted at this point in the text.
<b> INDENT (arg) </b>	Causes the text that follows it to be indented <b>arg</b> number of spaces from the left margin.

Text Formatting Expression	Description
SETTAB (arg1,arg2,arg3..)	Sets tab positions for the text that follows it. In subsequent lines, the first <b> TAB </b> encountered causes indentation to column position <b>arg1</b> characters from the left margin. The second <b> TAB </b> encountered causes indentation to column position <b>arg2</b> , and so on. If any <b>SETTAB arg</b> is negative, the text following the corresponding <b> TAB </b> is right justified so that the rightmost column of that text falls in the column number that is the absolute value of the <b>SETTAB arg</b> . If a <b>SETTAB arg</b> is the literal <b>C</b> (i.e., <b> SETTAB("C") </b> ), the text following the corresponding tab setting is centered.
CENTER (arg)	Causes the <b>arg</b> to be centered.
TAB	Causes the text to start printing at predetermined indents. The default column settings are 5,10,15,20, ..., which can be reset with <b>SETTAB</b> . <b> TAB </b> at the end of a line causes that line to be printed as is (no word wrapping).
TAB n	Overrides any <b>SETTAB</b> specification for the text that follows it and causes tabbing to the <b>n<sup>th</sup></b> column over from the left margin. Output is right justified on the <b>n<sup>th</sup></b> column, if " <b>n</b> " is negative. For example, the text following <b> TAB 12 </b> begins at column 12; the text following <b> TAB "C" </b> is centered.
WIDTH (arg)	<p>Specifies that the text that follows it is always printed in a column <b>arg</b> characters wide. (<b>Arg</b>, in other words, is the difference between the left margin position and the right margin position, plus one.)</p> <p> <b>NOTE:</b> In the absence of a <b>WIDTH</b> specification, the output column width is determined by the user (or defaulted by the system) at print time.</p>

Text Formatting Expression	Description
NOWRAP	Causes the text that follows it to be printed line-for-line (without wraparound). This eliminates the need to end each line with a tab or start the line with a space to force the line to be printed as it stands.
WRAP	Causes the text that follows it to be printed in wraparound mode. This is the default setting.
UNDERLINE (arg)	Causes the <b>arg</b> to be underlined.
_	Starts underlining. Underlining continues until a second  _  is encountered. This only works on printers that underline.



**REF:** For additional information about functions, see the [“Computed Expressions”](#) section.



**NOTE:** To print a “|” character, you *must* enter it as “||”. Likewise, to print “||” enter “||||”.

## 5 Computed Expressions

You can use computed expressions in several places within VA FileMan to obtain, manipulate, modify, and format data. Computed expressions consist of one or more elements linked together with operators. Most computed expressions return a value after performing the actions you have requested. The way this result is used or displayed depends on where you have used the computed expression.

### 5.1 Syntax

#### 5.1.1 Elements of Computed Expressions

You can use any of the following elements in constructing a computed expression:

- A **field name within the current file** (e.g., RELIGION). The field name can be partially spelled (e.g., REL), if the partial spelling is unambiguous.
- A **field number**, preceded with # (e.g., #3).
- A **literal number**. When used as part of a computed expression, do *not* use quotes (e.g., AGE AT ONSET+20). However, you *must* use quotes if the number will stand alone as a constant (e.g., "3.14159265").
- A **literal text string**, in quotes (e.g., "HELLO").
- A validly formatted **date**, such as **20 JULY 1969**, which is punctuated only by spaces.



**NOTE:** Dashes in a computed expression are interpreted as minus signs. For example, **7-20-1969** would indicate subtraction and be evaluated as **-1982**.

- The word **NUMBER** (or the name of the file followed by the word NUMBER, such as, PATIENT NUMBER). NUMBER returns the internal entry number of the entry in the file or subfile in question.
- The **name of a file followed by the name of a field in that file** (e.g., PATIENT NAME). Like PATIENT NUMBER, this syntax is helpful when it is unclear to which file or subfile an expression is referring. However, this syntax *cannot* obtain data from **another** file; NAME and PATIENT NAME returns the same data. To obtain data from another file, the extended pointer syntax *must* be used.

- A **VA FileMan function**—e.g., [TODAY or MONTH(DATE OF BIRTH)].



**REF:** Functions are discussed in the "[VA FileMan Functions](#)" section.

- An **extended pointer reference** to fields in another file.



**REF:** Extended pointers and relational jumping are described in the "[Relational Navigation](#)" section.

## 5.1.2 Operators in Computed Expressions

Computed expressions can consist of a single element. However, often several elements are joined together using operators. Operators are characters that perform some action on elements.

- [Unary Operators](#)
- [Binary Operators](#)
- [BOOLEAN Data Type](#)
- [Parentheses in Expressions](#)
- [Example of Compound Expression](#)

### 5.1.2.1 Unary Operators

The simplest operators are the unary operators. They force a numeric interpretation of the element that follows. They can also affect the sign of the resulting number. [Table 7](#) lists the unary operators:

**Table 7: Computed Expressions—Unary Operators**

Operator	Description
+	Positive numeric interpretation (sign unchanged)
-	Negative numeric interpretation (sign changed)

### 5.1.2.2 Binary Operators

Another set of operators takes two elements, manipulates them, and returns a result. These are called binary operators. You can use the binary operators listed in [Table 8](#) in computed expressions:

**Table 8: Computed Expressions—Binary Operators**

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
\	Integer (truncated) division (e.g., $13 \setminus 2 = 6$ )
_	Concatenation (e.g., "AB"_"CDE" = ABCDE)

### 5.1.2.3 Boolean Operators

A third set of operators makes a comparison between two elements and returns a true or false value. These are known as Boolean operators. If the outcome of a Boolean operation is:

- **True—One (1)** is returned.
- **False—Zero (0)** is returned.

You can use the Boolean operators listed in [Table 9](#) in computed expressions:

**Table 9: Computed Expressions—Boolean Operators**

Operator	Description
>	Greater than
<	Less than
=	Equal to
]	Follows (in alphabetical order)

Operator	Description
[	Contains (e.g., "AB"["A" is true; "A"["AB" is false)
!	Or, either element is true [e.g., (2=3)!(5<10) is true]
&	And, both elements are true [e.g., (2=3)&(5<10) is false]

An apostrophe (') means negation or *not*. It can precede any of the Boolean operators. Thus, 6'>8 is read six is *not* greater than eight, which is true (a one is returned).

**5.1.2.4 Parentheses in Expressions**

In the absence of parentheses, the expression is evaluated strictly left to right. One operator is *not* given precedence over another. Use parentheses to control the order in which the operations of a computed expression are performed. Expressions within parentheses are evaluated first. Thus, 3+4/2 is 3.5, whereas 3+(4/2) is 5.

You can also use parentheses to ensure that the enclosed material is treated as an expression when there might be some ambiguity. For example, suppose you want to force a numeric interpretation of the SSN field. You need to use the + unary operator. However, the following does *not* yield the desired result:

```
SORT BY: +SSN
```

Is the + the unary operator or the sort specifier (meaning that you want to subtotal results by SSN)? In this case, it is interpreted as the sort specifier. However, if you put the expression in parentheses, the + is interpreted as an operator:

```
SORT BY: (+SSN)
```

**5.1.2.5 Example of Compound Expression**

The following is an example of a computed expression containing several elements and operators:

```
BEDS OCCUPIED:  (NUMBER OF BEDS*OCCUPANCY PERCENTAGE/100)
```

First, the part within the parentheses is evaluated. NUMBER OF BEDS and OCCUPANCY PERCENTAGE are field names. Their contents are multiplied, and the result is divided by 100. That result is concatenated with the literal string "Beds occupied: " giving a result like:

```
BEDS OCCUPIED: 484
```

### 5.1.3 Data Types in Computed Expressions

When you are working with file data in computed expressions, you *must* consider the appropriateness of the DATA TYPE field value for the operation or function you are using. The following are some notes regarding data types and computed expressions:

- [SET OF CODES, POINTER TO A FILE, and VARIABLE-POINTER Data Types](#)
- [DATE/TIME Data Type](#)
- [WORD-PROCESSING Data Type](#)

#### 5.1.3.1 SET OF CODES, POINTER TO A FILE, and VARIABLE-POINTER Data Types

These data types are manipulated using the external representations, *not* the internal ones. The internal value can be accessed using the **INTERNAL** function.

#### 5.1.3.2 DATE/TIME Data Type

The DATA TYPE field value of **DATE/TIME** usually yields results based on the internal value of the field when used in a computed expression. For example, the computed expression "DATE OF BIRTH: " \_DOB, where DOB is a field with a DATA TYPE field value of **DATE/TIME**, yields "DATE OF BIRTH: 2910713", where 2910713 is the internal representation of the date.

Often, you do *not* want the internal representation of the date to be used for output. There are alternatives. Continuing with concatenation as an example, you can concatenate a caption with the output of a function (e.g., "DATE OF BIRTH: " \_NUMDATE(DOB) yields "DATE OF BIRTH: 07/13/91"). When using the **Print File Entries** [DIPRINT] option, you can separately identify the caption as shown in [Figure 73](#):

**Figure 73: Computed Expressions—Example Using the Print File Entries Option to Identify a Caption**

```
FIRST PRINT FIELD: [DATE OF BIRTH: ]
THEN PRINT FIELD: DOB;X
```

Since DOB was *not* entered as part of a computed expression, it produces output in VA FileMan's external date format: "**DATE OF BIRTH: JUL 13, 1991**".

You can perform certain arithmetic operations with DATA TYPE field values of **DATE/TIME** that directly yield useful results:

- If you subtract a DATA TYPE field value of **DATE/TIME** from another **DATE/TIME**-valued field, the result is the number of days the two differ.
- If you add a number to or subtract a number from a **DATE/TIME**-valued field, the result is a new date. For example, if the DOB field has the value **JUL 20, 1969**, then the value of the computed expression **DOB+30** is **AUG 19, 1969**.

### 5.1.3.3 WORD-PROCESSING Data Type

DATA TYPE fields with a value of **WORD-PROCESSING** can be manipulated only with the contains (left bracket, [ ) operator (e.g., a valid computed expression within the DIAGNOSIS Multiple of the sample PATIENT [#2] file is **HISTORY["poverty"]**). This Boolean expression is true, if the DIAGNOSIS in question has HISTORY text that contains the string "poverty".

Also, you *cannot concatenate* **WORD-PROCESSING**-type fields with other values using the concatenation (underscore, \_ ) operator.

### 5.1.4 Using Functions as Elements in Computed Expressions

You can use recognized functions as an element in any **COMPUTED** field expression. A function performs an operation that returns a value. These functions are available to all users. Functions can also be added by making entries in the FUNCTION (#.5) file. If you examine this file, you will know all functions available to you.



**REF:** For a description on how to add functions, see the "VA FileMan Functions (Creating)" section in the *VA FileMan Developer's Guide*.

Some functions require an argument or arguments; others are "argumentless." The arguments of the function can be any element, including field name, field number (preceded with the #), quoted literal, or even other functions. The **SQUAREROOT** function, for example, would take an argument of **64** and return **8**. Thus, if the AGE field of a patient has the value **64**, the expression **SQUAREROOT(AGE)** would equal **8**.



**REF:** For information on the syntax and description of the functions exported with VA FileMan, see the "[VA FileMan Functions](#)" section.

## 5.2 Where to Use

### 5.2.1 Using Computed Expressions in COMPUTED Fields

One important place where you can use a computed expression is in a field that is computed. The DATA TYPE field value of **COMPUTED** allows a computed expression to be stored in the data dictionary.

To define a field as **COMPUTED**, use the **Modify File Attributes** [DIMODIFY] option and reply to the "DATA TYPE:" prompt with "COMPUTED."

**Figure 74: Computed Expressions—Defining a DATA TYPE Field as COMPUTED**

```
SELECT OPTION: MODIFY FILE ATTRIBUTES
DO YOU WANT TO USER THE SCREEN-MODE VERSION? YES// NO <ENTER> (NO)

MODIFY WHAT FILE: PATIENT

SELECT FIELD: AGE
ARE YOU ADDING AGE AS A NEW FIELD (THE 13TH)? Y <ENTER> (YES)
FIELD NUMBER: 13// <ENTER>

DATA TYPE OF AGE: COMPUTED
```

You now enter the computed expression that is stored in the AGE field. In this case, a function (TODAY), a field name (DATE OF BIRTH), and a numeric literal are combined with two arithmetic binary operators to give a numeric result.

**Figure 75: Computed Expressions—Entering the Computed Expression into a DATA TYPE Field of COMPUTED**

```
COMPUTED-FIELD EXPRESSION: TODAY-DATE OF BIRTH\365.25

TRANSLATES TO THE FOLLOWING CODE:
S Y(16033,13,1)=$S($D(^DIZ(16033,D0,0)):^(0),1:),X=DT S
X=X,X1=X,X2=$P(Y(16033,13,1),U,3),X= D:X2 ^%DTC:X1 S X=X\365.25
```



**NOTE:** You only see the generated code if you have **Programmer** access.

When creating a **COMPUTED** field that might have a numeric result, the dialog in [Figure 76](#) is presented:

**Figure 76: Computed Expressions—Sample Dialog Encountered with a COMPUTED Field with Expected Numeric Result: Selecting Number-Valued**

NUMBER OF FRACTIONAL DIGITS TO OUTPUT (ONLY ANSWER IF NUMBER-VALUED): **0**

Pressing the **Enter** key without an entry at this prompt means that the field is *not* numeric; it is left justified on output. If you do answer ([Figure 76](#)), you indicate that the field is numeric and that you want the computed value rounded to a certain number of decimal places when it is printed. In this case, the number is rounded to a whole number.

**Figure 77: Computed Expressions—Sample Dialog Encountered with a COMPUTED Field with Expected Numeric Result: Setting Rounding Value**

SHOULD VALUE ALWAYS BE INTERNALLY ROUNDED TO 0 DECIMAL PLACES?  
NO// **<ENTER>** (NO)

Since the value of a **COMPUTED** field can be used in other calculations, you need to indicate when rounding should occur. If you accept the default (i.e., “**No**”), rounding is *not* done when the **COMPUTED** field is used in other calculations. A **YES** answer to this prompt means that you do want the rounded value used in calculations. Usually, you do *not* want values rounded at interim steps in a series of calculations. Thus, usually, you accept the “**No**” default.

When a **COMPUTED** field is printed, the value is always rounded to the number of decimal places you specify.

**Figure 78: Computed Expressions—Sample Dialog Encountered with a COMPUTED Field with Expected Numeric Result: Setting Totaling Value**

WHEN TOTALLING THIS FIELD, SHOULD THE SUM BE COMPUTED FROM THE SUMS OF THE COMPONENT FIELDS? NO// **<ENTER>** (NO)

If your computed expression involves division or multiplication, you are asked how the field should be totaled ([Figure 78](#)):

- A **NO** answer to this prompt means that the **COMPUTED** field’s expression is evaluated for each entry and those results are added.
- A **YES** answer means that values of each of the fields in the **COMPUTED** field’s expression is added first and then the **COMPUTED** field’s expression is applied to those totals.

You can total the values of a field in the **Print File Entries** [DIPRINT] option.

For example, suppose **A** and **B** are the names of two fields and **A/B** is a computed expression. [Table 10](#) shows the results of printing **A**, **B**, and **A/B** with different answers to the “WHEN TOTALLING THIS FIELD, ...” question:

**Table 10: Computed Expressions—Example Indicating Possible Results of Computed Expression Based on Different Entries to “Totaling” Prompt**

	<b>A</b>	<b>B</b>	<b>A/B</b> (YES: Total from totals of component fields)	<b>A/B</b> (NO: Total from results for each entry)
	10	5	2	2
	100	50	2	2
	2	1	2	2
<b>Total</b>	<b>112</b>	<b>56</b>	[112/56=] 2	[2+2+2=] 6

To summarize, if you want the total to be the ratio or product of the total of the component fields, then answer this question **YES**. Otherwise, a **NO** answer is appropriate.



**NOTE:** The answer to this prompt only affects the Total produced by the **Print File Entries** [DIPRINT] option].

When defining a **COMPUTED** field, you are also asked:

**Figure 79: Computed Expressions—Dialog Encountered When Defining a COMPUTED Field**

LENGTH OF FIELD: 8 // <b>&lt;ENTER&gt;</b>
--

Here you can enter the maximum number of character positions that the field should occupy in output. The default value is eight, even if the **COMPUTED** field involves **FREE TEXT**-type fields. Be sure to allocate enough space to accommodate the results. If the **COMPUTED** field’s value is numeric, the entire result is displayed regardless of the requested length.

The **COMPUTED**-type field can be a very useful tool. Having set up such a field, you can then search or sort by it and include it in the definition of other **COMPUTED**-type fields. In the latter case, independence is preserved. Thus, for example, if you define **COMPUTED** Field #2 in terms of **COMPUTED** Field #1 and then decide to redefine Field #1, Field #2 automatically uses the new Field #1 calculation. If you try to delete a field that is referenced by a **COMPUTED**-type field, you are warned.

## 5.2.2 Where to Use Computed Expressions “On-the-Fly”

### 5.2.2.1 “On-the-Fly” Computed Expressions

In addition to permanently storing a computed expression in a data dictionary, there are several places within VA FileMan’s dialog where you can use a computed expression “on-the-fly”:

- [“PRINT FIELD:” Prompt](#)
- [“SEARCH FOR FIELD:” Prompt](#)
- [“SORT BY:” Prompt](#)
- ["Start with" and "Go to" SORT Values](#)
- [Field Value Stuffing](#)
- [OUTPUT Transforms](#)
- [Word-Processing Windows \(| |\)](#)

### 5.2.2.2 “PRINT FIELD:” Prompt

Whenever, you are within the **Print File Entries** [DIPRINT] or **Search File Entries** [DISEARCH] options, you are asked:

FIRST PRINT FIELD:

OR

THEN PRINT FIELD:

You can answer with a computed expression. For example:

**Figure 80: Computed Expressions—Entering a Computed Expression at a “PRINT FIELD” Prompt**

FIRST PRINT FIELD: SEX_00 00_RELIGION;0000;L33
--

This sample computed expression returns the contents of the SEX and RELIGION fields separated by a space. You can follow the computed expression with print qualifiers: ;””

to suppress the column heading and ;L33 to indicate that the **COMPUTED** field length can be **33 characters** long.



**NOTE:** If the computed expression begins with a quoted string, then the column heading will be suppressed as though the print qualifier ;"" had been specified.

A user with **Programmer** access can also enter M code at this prompt. The M code *must* have a **WRITE** statement for anything that is to be written to the report.

### 5.2.2.3 “SEARCH FOR FIELD:” Prompt

In the **Search File Entries** [DISEARCH] option, you can answer the following prompt with a computed expression:

SEARCH FOR FIELD:

If the expression is Boolean (i.e., its value is either **true** or **false**), you are *not* asked the condition of the search, because the computed expression itself specifies that condition.

A user with **Programmer** access can also enter M code at this prompt. The M code *must* set the variable **X** to whatever is compared against the search value.

### 5.2.2.4 “SORT BY:” Prompt

In the **Print File Entries** [DIPRINT] or **Search File Entries** [DISEARCH] options, you can answer the “SORT BY:” prompt with a computed expression.

If the expression is Boolean (i.e., its value is either **true** or **false**), you are *not* asked the condition of the search, because the computed expression itself specifies that condition.

A user with **Programmer** access can also enter M code at this prompt. The M code *must* set variable **X** to the sort value.

For example, if you want to print a list of the names of all patients who are Baptists, you could enter:

**Figure 81: Computed Expressions—Entering a Computed Expression at a “SORT BY” Prompt**

```
SELECT OPTION: PRINT FILE ENTRIES
OUTPUT FROM WHAT FILE: PATIENT
SORT BY: RELIGION=BAPTIST
  WITHIN RELIGION=BAPTIST, SORT BY: <ENTER>
FIRST PRINT FIELD: NAME
```

This is a common way to select certain records for printing.

### 5.2.2.5 "Start with" and "Go to" SORT Values

In the **Print File Entries** [DIPRINT] or **Search File Entries** [DISEARCH] options, if the "SORT BY:" prompt is answered, the user is asked for the range of values by which to sort. Sometimes it is convenient to have the values of these answers be computed at run time. In such cases, if the answers are preceded with @, they can be understood as computed expressions.

For example, if you wanted the range of a certain date-valued field to be calculated for the week up to the day the output is run, do the following:

**Figure 82: Computed Expressions—Entering a Computed Expression at the "Start with..." and "Go to..." Prompt**

```
SELECT OPTION: PRINT FILE ENTRIES

OUTPUT FROM WHAT FILE: PTF
SORT BY: NUMBER// DISCHARGE DATE
START WITH DISCHARGE DATE: FIRST// @TODAY-6
DO YOU MEAN 'TODAY-6' AS A VARIABLE? YES// <ENTER>
(YES)
GO TO DISCHARGE DATE: LAST// @TODAY
DO YOU MEAN 'TODAY' AS A VARIABLE? YES// <ENTER>
(YES)
WITHIN DISCHARGE DATE, SORT BY:
....
```

### 5.2.2.6 Field Value Stuffing

In the **Enter or Edit File Entries** [DIEDIT] option, you can follow the // or /// specifiers with computed expressions. The expression is evaluated for the entry you are inputting and used as a variable stuff value.

Suppose you want to put the current contents of a patient's NEXT OF KIN field into the BENEFICIARY field, with a notation that this value is **UNVERIFIED**, for all patients who do *not* have a value in the BENEFICIARY field. The dialog would look like [Figure 83](#):

**Figure 83: Computed Expressions—"Stuffing" a Value in a Field via a Computed Expression**

```
SELECT OPTION: ENTER OR EDIT FILE ENTRIES
INPUT TO WHAT FILE: PATIENT
EDIT WHICH FIELD: BENEFICIARY///NEXT OF KIN (UNVERIFIED)
THEN EDIT FIELD: <ENTER>

SELECT PATIENT NAME: ^LOOP
EDIT ENTRIES BY: BENEFICIARY=
WITHIN BENEFICIARY=, EDIT ENTRIES BY: <ENTER>
```

This example uses two “on-the-fly” expressions:

- Answer to the “EDIT ENTRIES BY:” prompt (which is essentially a SORT BY for looping).
- Forced default value for the BENEFICIARY input field.

BENEFICIARY= "" is a Boolean (True/False) computed expression that means “The BENEFICIARY value equals **NULL**.”

After the previous dialog, the names of such patients would be printed out, and their BENEFICIARY value would automatically be set equal to their NEXT OF KIN field value, concatenated with a space followed by “**(UNVERIFIED)**.”

### 5.2.2.7 OUTPUT Transforms

**OUTPUT** transforms change the way a field is displayed when printed. Frequently, the **OUTPUT** transform contains a computed expression that alters the data stored internally in the field. A simple **OUTPUT** transform that converts the internally stored date into **MM/DD/YY** format is shown in [Figure 84](#):

**Figure 84: Computed Expressions—Entering a Computed Expression in an OUTPUT Transform**

```
DATE OF BIRTH OUTPUT TRANSFORM: NUMDATE (DATE OF BIRTH)
```

If an **OUTPUT** transform is applied to a field, the result of the transform is used if that field is used in another computed expression. For example, if DATE OF BIRTH is used in a **PRINT** template, the “transformed” value is output, as shown in [Figure 85](#):

**Figure 85: Computed Expressions—Entering a Computed Expression in an OUTPUT Transform Attached to a Field**

```
THEN PRINT FIELD: NAME [ ]S BIRTHDAY: [ ]_DATE OF BIRTH
```

The result of this computed expression would be like [Figure 86](#):

**Figure 86: Computed Expressions—Example of the Result of an OUTPUT Transform with a Computed Expression**

```
ONE FMPATIENT[S BIRTHDAY: 03/07/42
```

### 5.2.2.8 Word-Processing Windows (| |)

When entering text into a DATA TYPE field with a value of **WORD-PROCESSING**, you can insert a computed expression within a **|Window|**. This expression is evaluated at the time the **WORD-PROCESSING**-type field is printed. If the expression is meaningful, its value replaces the **|Window|** in the printed output.

For example, you could embed within the text of the HISTORY WORD-PROCESSING-type field a **|Window|** containing a **COMPUTED** field expression:

**Figure 87: Computed Expressions—A |Window| with a Computed Expression**

```
HISTORY:  
1> PATIENT IS A |SEX_| |RELIGION| WHO HAS NO  
2> APPARENT PROBLEMS.
```

When this field is printed for a patient who has a SEX value of **MALE** and a RELIGION value of **CATHOLIC**, the output would look like [Figure 88](#):

**Figure 88: Computed Expressions—Example of the Result of a |Window| with a Computed Expression**

```
PATIENT IS A MALE CATHOLIC WHO HAS NO  
APPARENT PROBLEMS.
```

## 6 VA FileMan Functions

### 6.1 How to Use VA FileMan Functions

This section lists each VA FileMan function, including syntax and simple examples of their use. You can use them in any computed expression.



**REF:** For more information on computed expressions, see the "[Computed Expressions](#)" section.

A function performs an operation that returns a value. Many functions are included with VA FileMan; you can also add functions by making entries in the FUNCTION (#.5) file.



**REF:** For a description on how to add functions, see the "VA FileMan Functions (Creating)" section in the *VA FileMan Developer's Guide*.

Some functions require an argument or arguments; others are "argumentless." The arguments of the function can be any element, including field name, field number (preceded with the #), quoted literal, or even other functions. The **SQUAREROOT** function, for example, would take an argument of **64** and return **8**. Thus, if the AGE field of a patient has the value **64**, the expression **SQUAREROOT(AGE)** would return **8**.




**NOTE:** If there is an **OUTPUT** transform on a field, the function code is applied to the field after it has been transformed. In most cases, if a field has an **OUTPUT** transform, you should therefore use the syntax **FUNCTION\_NAME(INTERNAL(FIELD\_NAME))**, rather than **FUNCTION\_NAME(FIELD\_NAME)**.

## 6.2 Documentation Conventions for VA FileMan Functions

While studying this section's functions, syntax, and examples, you encounter the conventions listed in [Table 11](#):

**Table 11: VA FileMan Functions—Documentation Conventions**

Convention	Description
"	In the format arguments: Indicates mandatory quotation marks.  <b>NOTE:</b> If you enter a literal string as an argument, quotation marks are also necessary.
=>	In examples: Indicates the output of the function.
[ ]	In examples: Indicates information about the outcome of the function.
<b>boldface type</b>	Indicates specific reference to an argument.

**FUNCTION(argument, . . .)** is the general format. You *must* enter the function's name in uppercase; the case of the arguments depends on the circumstances. Arguments are *always* surrounded by parentheses.

## 6.3 VA FileMan Function Categories

[Table 12](#) lists the VA FileMan functions by category.



**REF:** Each of these functions is described in the sections that follow.

**Table 12: VA FileMan Functions—By Category**

Category	Function
Date/Time	<a href="#">BETWEEN</a> <a href="#">DATE</a> <a href="#">DAYOFWEEK</a> <a href="#">MID</a> <a href="#">MINUTES</a> <a href="#">MONTH</a> <a href="#">MONTHNAME</a> <a href="#">NOON</a> <a href="#">NOW</a> <a href="#">NUMDATE</a> <a href="#">NUMDATE4</a> <a href="#">NUMDAY</a> <a href="#">NUMMONTH</a> <a href="#">NUMYEAR</a> <a href="#">NUMYEAR4</a> <a href="#">RANGEDATE</a> <a href="#">TIME</a> <a href="#">TODAY</a> <a href="#">YEAR</a>
Environmental	<a href="#">BREAKABLE</a> <a href="#">CLOSE</a> <a href="#">SITENUMBER</a> <a href="#">USER</a>

Category	Function
File and File Data	<a href="#">COUNT</a> <a href="#">DUPLICATED</a> <a href="#">FILE</a> <a href="#">INTERNAL</a> <a href="#">LAST</a> <a href="#">MAXIMUM</a> <a href="#">MINIMUM</a> <a href="#">nTH</a> <a href="#">NEXT</a> <a href="#">PREVIOUS</a> <a href="#">TOTAL</a>
Mathematical	<a href="#">ABS</a> <a href="#">BETWEEN</a> <a href="#">MAX</a> <a href="#">MIN</a> <a href="#">MODULO</a> <a href="#">SQUAREROOT</a>
Printing Related Functions	<a href="#">IOM</a> <a href="#">PAGE</a>
String	<a href="#">DUP</a> <a href="#">LOWERCASE</a> <a href="#">PADRIGHT</a> <a href="#">REPLACE</a> <a href="#">REVERSE</a> <a href="#">STRIPBLANKS</a> <a href="#">TRANSLATE</a> <a href="#">UPPERCASE</a>
Temporary Data Storage	<a href="#">PARAM</a> <a href="#">SETPARAM</a> <a href="#">VAR</a> <a href="#">SET</a>

Category	Function
M-Related Functions	<a href="#">\$A[SCII]</a> <a href="#">\$C[HAR]</a> <a href="#">\$E[XTRACT]</a> <a href="#">\$F[IND]</a> <a href="#">\$H[OROLOG]</a> <a href="#">\$I[O]</a> <a href="#">\$J[OB]</a> <a href="#">\$J[USTIFY]</a> <a href="#">\$L[ENGTH]</a> <a href="#">\$P[IECE]</a> <a href="#">\$R[ANDOM]</a> <a href="#">\$S[ELECT]</a> <a href="#">\$S[TORAGE]</a> <a href="#">\$X</a> <a href="#">\$Y</a>

## 6.3.1 Date/Time Functions

### 6.3.1.1 BETWEEN

Table 13: VA FileMan Functions—Date/Time Function: BETWEEN

Item	Description
<b>Format:</b>	<b>BETWEEN(d1,d2,d3)</b>
<b>Parameters:</b>	<p>The <b>d1</b>, <b>d2</b>, and <b>d3</b> are dates or date expressions:</p> <ul style="list-style-type: none"> <li>• <b>d1</b> is the date being tested.</li> <li>• <b>d2</b> is one limit for the test.</li> <li>• <b>d3</b> is the other limit for the test.</li> </ul>
<b>Use:</b>	This Boolean function determines if <b>d1</b> is within the limits defined by <b>d2</b> and <b>d3</b> . If <b>d1</b> is within this range, a value of 1 (true) is returned; otherwise, 0 (false) is returned. If <b>d1</b> equals <b>d2</b> or <b>d3</b> , 1 (true) is returned.
<b>Examples:</b>	<pre> SELECT OPTION: SEARCH FILE ENTRIES OUTPUT FROM WHAT FILE: BUILD// &lt;ENTER&gt; -A- SEARCH FOR BUILD FIELD: BETWEEN( DATE DISTRIBUTED, 1JAN2000, 1JAN2001) -B- SEARCH FOR BUILD FIELD: IF: A// &lt;ENTER&gt; BETWEEN( DATE DISTRIBUTED, 1JAN2000, 1JAN2001) </pre>

### 6.3.1.2 DATE

Table 14: VA FileMan Functions—Date/Time Function: DATE

Item	Description
<b>Format:</b>	<b>DATE(datexp)</b>
<b>Parameters:</b>	<b>datexp</b> is an expression with a date/time value.
<b>Use:</b>	This date function returns the date portion of a date/time expression.
<b>Example:</b>	DATE(NOW) => AUG 21,1991



**REF:** For tips on displaying date-valued elements such as this function in computed expressions (e.g., printing), see the [“Data Types in Computed Expressions”](#) section.

### 6.3.1.3 DAYOFWEEK

Table 15: VA FileMan Functions—Date/Time Function: DAYOFWEEK

Item	Description
<b>Format:</b>	<b>DAYOFWEEK(datexp)</b>
<b>Parameters:</b>	<b>datexp</b> is an expression with date/time value.
<b>Use:</b>	This function returns the day of the week of the date in <b>datexp</b> .
<b>Example:</b>	DAYOFWEEK(DATE OF BIRTH) => TUESDAY

### 6.3.1.4 MID

Table 16: VA FileMan Functions—Date/Time Function: MID

Item	Description
Format:	MID
Parameters:	(none)
Use:	This argumentless function returns the current date with a <b>24:00</b> time stamp. It represents tonight at midnight.
Example:	MID => AUG 23,1991 24:00



**REF:** For tips on displaying date-valued elements such as this function in computed expressions (e.g., printing), see the "[Data Types in Computed Expressions](#)" section.

### 6.3.1.5 MINUTES

Table 17: VA FileMan Functions—Date/Time Function: MINUTES

Item	Description
Format:	MINUTES(datexp1,datexp2)
Parameters:	<b>datexp1</b> and <b>datexp2</b> are date/time expressions. Time stamps are <i>not</i> necessary.
Use:	This function returns the number of minutes that <b>datexp1</b> is <i>after</i> <b>datexp2</b> . If no time is associated with a date/time expression, <b>DATE@12:00 A.M.</b> is used.
Examples:	MINUTES(MID,NOW) => 832 MINUTES(MID,TODAY) => 1440

### 6.3.1.6 MONTH

Table 18: VA FileMan Functions—Date/Time Function: MONTH

Item	Description
<b>Format:</b>	<b>MONTH(datexp)</b>
<b>Parameters:</b>	<b>datexp</b> is a date/time expression.
<b>Use:</b>	This function returns the month and year from a date/time valued expression.
<b>Example:</b>	MONTH(DATE OF BIRTH) => AUG 1943

### 6.3.1.7 MONTHNAME

Table 19: VA FileMan Functions—Date/Time Function: MONTHNAME

Item	Description
<b>Format:</b>	<b>MONTHNAME(<i>n</i>)</b>
<b>Parameters:</b>	The <b><i>n</i></b> is an expression that evaluates to an integer from <b>1 through 12</b> .
<b>Use:</b>	This function returns the full name of the month corresponding to <b><i>n</i></b> .
<b>Examples:</b>	MONTHNAME(4) => APRIL MONTHNAME(+\$E(DATE OF BIRTH,4,5)) => APRIL [Function \$E extracts the 4th and 5th digits from a date stored in FileMan internal format: YYYYMMDD.]

### 6.3.1.8 NOON

Table 20: VA FileMan Functions—Date/Time Function: NOON

Item	Description
<b>Format:</b>	<b>NOON</b>
<b>Parameters:</b>	(none)
<b>Use:</b>	This argumentless function returns today's date with a time stamp of <b>12:00</b> .
<b>Example:</b>	NOON => AUG 23,1991 12:00



**REF:** For tips on displaying date-valued elements such as this function in computed expressions (e.g., printing), see the "[Data Types in Computed Expressions](#)" section.

### 6.3.1.9 NOW

Table 21: VA FileMan Functions—Date/Time Function: NOW

Item	Description
<b>Format:</b>	<b>NOW</b>
<b>Parameters:</b>	(none)
<b>Use:</b>	This argumentless function returns the current date and time.
<b>Example:</b>	NOW => AUG 23,1991 11:23



**REF:** For tips on displaying date-valued elements such as this function in computed expressions (e.g., printing), see the "[Data Types in Computed Expressions](#)" section.

### 6.3.1.10 NUMDATE

Table 22: VA FileMan Functions—Date/Time Function: NUMDATE

Item	Description
Format:	NUMDATE(datexp)
Parameters:	<b>datexp</b> is an expression with a date/time value.
Use:	This function returns the date in <b>datexp</b> in <i>MM/DD/YY</i> format.
Example:	NUMDATE(DATE OF BIRTH) => 03/07/49

### 6.3.1.11 NUMDATE4

Table 23: VA FileMan Functions—Date/Time Function: NUMDATE4

Item	Description
Format:	NUMDATE4(datexp)
Parameters:	<b>datexp</b> is an expression with a date/time value.
Use:	This function returns the date in <b>datexp</b> in <i>MM/DD/YYYY</i> format.
Example:	NUMDATE4(DATE OF BIRTH) => 03/07/1949

### 6.3.1.12 NUMDAY

Table 24: VA FileMan Functions—Date/Time Function: NUMDAY

Item	Description
Format:	NUMDAY(datexp)
Parameters:	<b>datexp</b> is an expression with a date/time value.
Use:	This function returns the day of the month in <b>datexp</b> as a number.
Example:	NUMDAY(DATE OF BIRTH) => 7 [DATE OF BIRTH = March 7, 1949]

### 6.3.1.13 NUMMONTH

Table 25: VA FileMan Functions—Date/Time Function: NUMMONTH

Item	Description
Format:	NUMMONTH(datexp)
Parameters:	<b>datexp</b> is an expression with a date/time value.
Use:	This function returns the month in <b>datexp</b> as a number.
Example:	NUMMONTH(DATE OF BIRTH) => 3 [DATE OF BIRTH = March 7, 1949]

### 6.3.1.14 NUMYEAR

Table 26: VA FileMan Functions—Date/Time Function: NUMYEAR

Item	Description
Format:	NUMYEAR(datexp)
Parameters:	<b>datexp</b> is an expression with a date/time value.
Use:	This function returns the last two digits of the year in <b>datexp</b> as a number.
Example:	NUMYEAR(DATE OF BIRTH) => 49 [DATE OF BIRTH = March 7, 1949]

### 6.3.1.15 NUMYEAR4

Table 27: VA FileMan Functions—Date/Time Function: NUMYEAR4

Item	Description
Format:	NUMYEAR4(datexp)
Parameters:	<b>datexp</b> is an expression with a date/time value.
Use:	This function returns the <b>four-digit year</b> in <b>datexp</b> as a number.
Example:	NUMYEAR4(DATE OF BIRTH) => 1949 [DATE OF BIRTH = March 7, 1949]

### 6.3.1.16 RANGEDATE

Table 28: VA FileMan Functions—Date/Time Function: RANGEDATE

Item	Description
<b>Format:</b>	<b>RANGEDATE(datexp1,datexp2,datexp3,datexp4)</b>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>datexp1</b> is a date valued expression beginning the first range of dates.</li> <li>• <b>datexp2</b> is a date valued expression ending the first range of dates.</li> <li>• <b>datexp3</b> is a date valued expression beginning the second range of dates.</li> <li>• <b>datexp4</b> is a date valued expression ending the second range of dates.</li> </ul>
<b>Use:</b>	This function returns the number of days that the two ranges of dates overlap.
<b>Example:</b>	RANGEDATE(DATE OF BIRTH,NOW,20 JUL 1981,20 JUL 1982) => 366

### 6.3.1.17 TIME

Table 29: VA FileMan Functions—Date/Time Function: TIME

Item	Description
<b>Format:</b>	<b>TIME(datexp)</b>
<b>Parameters:</b>	<b>datexp</b> is an expression with a date/time value.
<b>Use:</b>	This function returns time from <b>datexp</b> in <b>12-hour</b> format with <b>AM/PM</b> .
<b>Example:</b>	TIME(NOW) => 1:15 PM

### 6.3.1.18 TODAY

Table 30: VA FileMan Functions—Date/Time Function: TODAY

Item	Description
Format:	TODAY
Parameters:	(none)
Use:	This argumentless function returns today's date.
Example:	TODAY => AUG 26,1991



**REF:** For tips on displaying date-valued elements such as this function in computed expressions (e.g., printing), see the "[Data Types in Computed Expressions](#)" section.

### 6.3.1.19 YEAR

Table 31: VA FileMan Functions—Date/Time Function: YEAR

Item	Description
Format:	YEAR(datexp)
Parameters:	<b>datexp</b> is an expression with a date/time value.
Use:	This function returns the year from <b>datexp</b> .
Example:	YEAR(DATE OF BIRTH) => 1949

## 6.3.2 Environmental Functions

### 6.3.2.1 BREAKABLE

Table 32: VA FileMan Functions—Environmental Function: BREAKABLE

Item	Description
<b>Format:</b>	<b>BREAKABLE(<i>n</i>)</b>
<b>Parameters:</b>	The <i>n</i> is a number or numeric expression with a value of 1 or 0.
<b>Use:</b>	This function returns nothing. When used within a <b>PRINT</b> template, this function determines whether <b>&lt;Ctrl&gt;C</b> can be used to break out of a report print. If <i>n</i> = 1, <b>&lt;Ctrl&gt;C</b> breaks out; if <i>n</i> = 0, it does <i>not</i> . Under default conditions, <b>&lt;Ctrl&gt;C</b> breaks you out. The value of <i>n</i> is returned.
<b>Example:</b>	BREAKABLE(0) =>0; [ <b>&lt;Ctrl&gt;C</b> is disabled]

### 6.3.2.2 CLOSE

Table 33: VA FileMan Functions—Environmental Function: CLOSE

Item	Description
<b>Format:</b>	<b>CLOSE(device)</b>
<b>Parameters:</b>	<b>device</b> is an open device, in the form of a valid argument for an M <b>Close</b> command.
<b>Use:</b>	This function should only be used within VA FileMan code when Kernel is unavailable. It closes the specified device.



### 6.3.2.3 SITENUMBER

Table 34: VA FileMan Functions—Environmental Function: SITENUMBER

Item	Description
<b>Format:</b>	<b>SITENUMBER</b>
<b>Parameters:</b>	(none)
<b>Use:</b>	This argumentless function returns your site's identifying number that was entered during VA FileMan initialization and stored in <b>^DD("SITE",1)</b> . (Do <i>not</i> use this function to retrieve a VA Institution Station Number.)
<b>Example:</b>	SITENUMBER => 99

### 6.3.2.4 USER

Table 35: VA FileMan Functions—Environmental Function: USER

Item	Description
<b>Format:</b>	<b>USER("attribute")</b>
<b>Parameters:</b>	<p><b>attribute</b> is one of these codes:</p> <ul style="list-style-type: none"> <li>• <b>#</b>—User's <b>DUZ</b> value (the user's number)</li> <li>• <b>N</b>—User's name</li> <li>• <b>I</b>—User's initials</li> <li>• <b>T</b>—User's title</li> <li>• <b>NN</b>—User's nickname</li> </ul> <p> <b>NOTE:</b> These codes <i>must</i> be surrounded by quotes within the function.</p>
<b>Use:</b>	<p>This function returns information about the currently logged on user. The information comes from the NEW PERSON (#200) file.</p> <p> <b>NOTE:</b> This function does <i>not</i> work if you are using VA FileMan <i>without</i> a NEW PERSON file in <b>^VA(200,</b></p>
<b>Example:</b>	USER("#") => 160

## 6.3.3 File and File Data Functions

### 6.3.3.1 COUNT

Table 36: VA FileMan Functions—File and File Data Function: COUNT

Item	Description
<b>Format:</b>	<b>COUNT(fname)</b> <b>COUNT(fname:field)</b>
<b>Parameters:</b>	<ol style="list-style-type: none"><li>1. In the first format, <b>fname</b> is the name of a file or of a Multiple in your current file.</li><li>2. In the second format:<ul style="list-style-type: none"><li>• <b>fname</b> is the name of your current file or Multiple.</li><li>• <b>field</b> is the name of a field (or a field number preceded by #) in <b>fname</b>.</li></ul></li></ol>
<b>Use:</b>	This function counts the number of entries in a file or in a Multiple. You can count the lines in a <b>WORD-PROCESSING</b> field by using the first format with the <b>WORD-PROCESSING</b> field name as the <b>fname</b> . If the second format is used, the number of entries with <i>non-NULL</i> values in <b>field</b> is returned.
<b>Examples:</b>	<ol style="list-style-type: none"><li>1. <b>COUNT(PATIENT) -&gt; 1349</b> [the number of entries in the PATIENT file]</li><li>2. <b>COUNT(PATIENT:PROVIDER) =&gt; 1288</b> [number of patients with providers recorded]</li></ol>

### 6.3.3.2 DUPLICATED

Table 37: VA FileMan Functions—File and File Data Function: DUPLICATED

Item	Description
<b>Format:</b>	<b>DUPLICATED(field)</b>
<b>Parameters:</b>	<b>field</b> is the name of a field (or a field number preceded by #). The field <i>must</i> be a cross-referenced field.
<b>Use:</b>	<p>This function, when used on any cross-referenced field, finds all duplicates within a given file or determines whether a specific entry is duplicated. Returns one of the possible Boolean values:</p> <ul style="list-style-type: none"> <li>• <b>1</b> = field value is duplicated in another entry.</li> <li>• <b>""</b> = field value is unique.</li> </ul>
<b>Examples:</b>	<p>Example using the <b>Search File Entries</b> [DISEARCH] option to perform a search on the example file named ZZINDIVIDUAL:</p> <pre style="border: 1px solid black; padding: 10px;"> SELECT OPTION: <b>SEARCH FILE ENTRIES</b>  OUTPUT FROM WHAT FILE: ZZINDIVIDUAL// <b>&lt;ENTER&gt;</b>  -A- SEARCH FOR ZZINDIVIDUAL FIELD: <b>DUPLICATED(NAME)</b>  -B- SEARCH FOR ZZINDIVIDUAL FIELD:  IF: A// <b>&lt;ENTER&gt;</b> DUPLICATED(NAME)  STORE RESULTS OF SEARCH IN TEMPLATE: <b>&lt;ENTER&gt;</b>  SORT BY: NAME// <b>&lt;ENTER&gt;</b> START WITH NAME: FIRST// <b>&lt;ENTER&gt;</b> FIRST PRINT FIELD: <b>NUMBER</b> THEN PRINT FIELD: <b>NAME</b> THEN PRINT FIELD: <b>&lt;ENTER&gt;</b> HEADING (S/C): ZZINDIVIDUAL SEARCH// <b>&lt;ENTER&gt;</b> DEVICE: <b>&lt;ENTER&gt;</b> TELNET TERMINAL RIGHT MARGIN: 80// <b>&lt;ENTER&gt;</b> ZZINDIVIDUAL SEARCH MAR 18,2008 14:44 PAGE 1 NUMBER NAME ----- 5 FMPATIENT,ONE 15 FMPATIENT,ONE  2 MATCHES FOUND.</pre> <p>Another example for using DUPLICATED, this time using the <b>Print File Entries</b> [DIPRINT] option, would be if you wanted to print the name with</p>

Item	Description
	three asterisks in front of it if it were a duplicated name: FIRST PRINT FIELD: \$(DUPLICATED(NAME):***,1:*)_NAME

### 6.3.3.3 FILE

**Table 38: VA FileMan Functions—File and File Data Function: FILE**

Item	Description
<b>Format:</b>	<b>FILE(vpointer)</b>
<b>Parameters:</b>	<b>vpointer</b> is the label or field number of a <b>VARIABLE POINTER</b> field.
<b>Use:</b>	This function returns the name of the file to which a <b>VARIABLE POINTER</b> points to a particular entry.
<b>Example:</b>	<b>FILE(PROVIDER) =&gt; STAFF PROVIDERS</b>

### 6.3.3.4 INTERNAL

**Table 39: VA FileMan Functions—File and File Data Function: INTERNAL**

Item	Description
<b>Format:</b>	<b>INTERNAL(field)</b>
<b>Parameters:</b>	<b>field</b> is the label of a field, or a field number preceded by #.
<b>Use:</b>	This function returns the internally stored value of the field for a particular entry. It is useful in obtaining the internally stored (instead of displayed) DATA TYPE field value of any of the following: <ul style="list-style-type: none"> <li>• <b>POINTER TO A FILE</b></li> <li>• <b>VARIABLE POINTER</b></li> <li>• <b>DATE/TIME</b></li> <li>• <b>SET OF CODES</b></li> </ul>
<b>Examples:</b>	<b>INTERNAL(PROVIDER) =&gt; 136;VA(200,</b> <b>INTERNAL(SEX) =&gt; m</b>

### 6.3.3.5 LAST

Table 40: VA FileMan Functions—File and File Data Function: LAST

Item	Description
<b>Format:</b>	<b>LAST(fname)</b> <b>LAST(fname:field)</b>
<b>Parameters:</b>	<p>In the first format, <b>fname</b> is the name of a file or of a Multiple-valued field in your current file.</p> <p>In the second format:</p> <ul style="list-style-type: none"> <li>• <b>fname</b> is the name of your current file or Multiple.</li> <li>• <b>field</b> is the name of a field (or a field number preceded by #) in <b>fname</b>.</li> </ul>
<b>Use:</b>	<p>This function returns the last entry in a file or in a Multiple identified by <b>fname</b>. If the second format is used, the last entry with a <i>non-NULL</i> value in <b>field</b> is returned. The last entry is the one with the highest internal entry number; the function does <i>not</i> analyze the values of the entries.</p>
<b>Examples:</b>	<b>LAST(DIAGNOSIS) =&gt; Sepsis</b> [last entry in this Multiple field] <b>LAST(DIAGNOSIS:OCCURRENCES) =&gt; 3</b>

### 6.3.3.6 MAXIMUM

Table 41: VA FileMan Functions—File and File Data Function: MAXIMUM

Item	Description
<b>Format:</b>	<ol style="list-style-type: none"> <li>1. <b>MAXIMUM(fname)</b></li> <li>2. <b>MAXIMUM(fname:field)</b></li> </ol>
<b>Parameters:</b>	<p>In the first format, <b>fname</b> is the name of a file or of a Multiple in your current file.</p> <p>In the second format:</p> <ul style="list-style-type: none"> <li>• <b>fname</b> is the name of your current file or Multiple.</li> <li>• <b>field</b> is the name of a field (or a field number preceded by #) in <b>fname</b>.</li> </ul>
<b>Use:</b>	<p>With the first format, this function returns the largest value from the <b>.01</b> field of the file or Multiple identified by <b>fname</b>. The second format returns the largest value from <b>field</b>. The function works only if the internally stored values of the entries are numeric. Thus, you can use numeric or date valued fields. Also, <b>FREE TEXT</b> fields work if the stored values are numbers. <b>COMPUTED</b> fields with numeric results can be used. Pointer fields return the value from the pointed-to file.</p>
<b>Examples:</b>	<p><b>MAXIMUM(APPOINTMENT) =&gt; FEB 25,1991</b> [APPOINTMENT is a Multiple-valued DATE/TIME field]</p> <p><b>MAXIMUM(PATIENT:AGE) =&gt; 93</b> [AGE is a field in the current file, PATIENT]</p>

### 6.3.3.7 MINIMUM

Table 42: VA FileMan Functions—File and File Data Function: MINIMUM

Item	Description
<b>Format:</b>	<ol style="list-style-type: none"> <li>1. <b>MINIMUM(fname)</b></li> <li>2. <b>MINIMUM(fname:field)</b></li> </ol>
<b>Parameters:</b>	<p>In the first format, <b>fname</b> is the name of a file or of a Multiple-valued field in your current file.</p> <p>In the second format:</p> <ul style="list-style-type: none"> <li>• <b>fname</b> is the name of your current file or Multiple.</li> <li>• <b>field</b> is the name of a field (or a field number preceded by #) in <b>fname</b>.</li> </ul>
<b>Use:</b>	<p>This function returns the smallest value from the file's <b>.01</b> field or from the Multiple identified by <b>fname</b>. The second format returns the smallest value from <b>field</b>. (See <a href="#">MAXIMUM</a> for limits of use.)</p>
<b>Examples:</b>	<p><b>MINIMUM(APPOINTMENT) =&gt; MAR 1,1979</b> [APPOINTMENT is a Multiple-valued <b>DATE/TIME</b> field]</p> <p><b>MINIMUM(PATIENT:AGE) =&gt; 18</b></p>

### 6.3.3.8 nTH

**Table 43: VA FileMan Functions—File and File Data Function: nTH**

Item	Description
<b>Format:</b>	<p>The syntax of this function is different, because the function's name is defined by the user. The name is a number followed by an ordinal number suffix.</p> <ol style="list-style-type: none"> <li>1. <b>nTH(fname)</b></li> <li>2. <b>nTH(fname:field)</b></li> </ol>
<b>Parameters:</b>	<p>In the first format, <b>fname</b> is the name of a file or of a Multiple in your current file.</p> <p>In the second format:</p> <ul style="list-style-type: none"> <li>• <b>fname</b> is the name of your current file or Multiple.</li> <li>• <b>field</b> is the name of a field (or a field number preceded by #) in <b>fname</b>.</li> </ul>
<b>Use:</b>	<p>This function returns the <b>n<sup>th</sup></b> entry in a file or in a Multiple identified by <b>fname</b>. If the second format is used, the value of the specified field associated with the <b>n<sup>th</sup></b> entry in <b>fname</b> is returned. The <b>n<sup>th</sup></b> entry is determined by the internal entry number; the function does <i>not</i> analyze the values of the entries. When used with the second format, the <b>n<sup>th</sup></b> subentry with a <i>non-NULL</i> value is returned.</p>
<b>Examples:</b>	<p><b>2ND(DIAGNOSIS) =&gt; Angina Pectoris</b> [the second entry in the DIAGNOSIS Multiple]</p> <p><b>10TH(ADMISSION:ADMISSION DATE) =&gt; JAN 2,1990</b> [ADMISSION DATE associated with the tenth ADMISSION]</p>

### 6.3.3.9 NEXT

Table 44: VA FileMan Functions—File and File Data Function: NEXT

Item	Description
<b>Format:</b>	<b>NEXT(field)</b>
<b>Parameters:</b>	<b>field</b> is a field's number preceded by a # or a field's label from the current file or Multiple.
<b>Use:</b>	This function returns the value for the field identified by <b>field</b> in the next entry. The next entry is determined by internal entry number. No analysis of the value of entries is done. If there are no more entries, the function returns <b>NULL</b> .
<b>Example:</b>	<b>NEXT(AGE AT ONSET) =&gt; 56</b> [the value of AGE AT ONSET for the next entry in the Subfile]

### 6.3.3.10 PREVIOUS

Table 45: VA FileMan Functions—File and File Data Function: PREVIOUS

Item	Description
<b>Format:</b>	<b>PREVIOUS(field)</b>
<b>Parameters:</b>	<b>field</b> is a field's number preceded by a # or a field's label from the current file or Multiple.
<b>Use:</b>	This function returns the value for the field identified by <b>field</b> in the previous entry. The previous entry is determined by internal entry number. No analysis of the value of entries is done. If there is no prior entry, the function returns <b>NULL</b> .
<b>Example:</b>	<b>PREVIOUS(AGE AT ONSET) =&gt; 29</b> [the value of AGE AT ONSET for the prior entry in the Subfile]

### 6.3.3.11 TOTAL

Table 46: VA FileMan Functions—File and File Data Function: TOTAL

Item	Description
<b>Format:</b>	<ol style="list-style-type: none"> <li>1. <b>TOTAL(fname)</b></li> <li>2. <b>TOTAL(fname:field)</b></li> </ol>
<b>Parameters:</b>	<p>In the first format, <b>fname</b> is the name of a file or of a Multiple-valued field in your current file.</p> <p>In the second format:</p> <ul style="list-style-type: none"> <li>• <b>fname</b> is the name of your current file or Multiple.</li> <li>• <b>field</b> is the name of a field (or a field number preceded by #) in <b>fname</b>.</li> </ul>
<b>Use:</b>	<p>With the first format, this function totals the values of the <b>.01</b> field of a Multiple or file identified by <b>fname</b>. The second format totals the values in field. The field being totaled <i>must</i> have numeric values.</p>
<b>Example:</b>	<p>"\$_TOTAL(VISIT COST) =&gt; \$569.32 [VISIT COST is a Multiple]</p>

## 6.3.4 Mathematical Functions

### 6.3.4.1 ABS

Table 47: VA FileMan Functions—Mathematical Function: ABS

Item	Description
<b>Format:</b>	<b>ABS(<i>n</i>)</b>
<b>Parameters:</b>	The <b><i>n</i></b> is a number or an expression with a numeric value.
<b>Use:</b>	This mathematical function returns the value of <b><i>n</i></b> <i>without</i> a sign; it gives the absolute value of <b><i>n</i></b> .
<b>Example:</b>	ABS(-23.87) => 23.87

### 6.3.4.2 BETWEEN

Table 48: VA FileMan Functions—Mathematical Function: BETWEEN

Item	Description
<b>Format:</b>	<b>BETWEEN(<i>n1</i>,<i>n2</i>,<i>n3</i>)</b>
<b>Parameters:</b>	The <b><i>n1</i></b> , <b><i>n2</i></b> , and <b><i>n3</i></b> are numbers or numeric expressions: <ul style="list-style-type: none"> <li>• <b><i>n1</i></b> is the number being tested.</li> <li>• <b><i>n2</i></b> is one limit for the test.</li> <li>• <b><i>n3</i></b> is the other limit for the test.</li> </ul>
<b>Use:</b>	This Boolean function determines if <b><i>n1</i></b> is within the limits defined by <b><i>n2</i></b> and <b><i>n3</i></b> : <ul style="list-style-type: none"> <li>• If <b><i>n1</i></b> is within this range, <b>1 (true)</b> is returned.</li> <li>• If <b><i>n1</i></b> is not within this range, <b>0 (false)</b> is returned.</li> <li>• If <b><i>n1</i></b> equals <b><i>n2</i></b> or <b><i>n3</i></b>, <b>1 (true)</b> is returned.</li> </ul>
<b>Examples:</b>	BETWEEN(OCCURRENCES,5,10) => 0 [OCCURRENCES is a field with value = 3] BETWEEN(-3,-10,0) => 1

### 6.3.4.3 MAX

Table 49: VA FileMan Functions—Mathematical Function: MAX

Item	Description
<b>Format:</b>	<b>MAX(<i>n1</i>,<i>n2</i>)</b>
<b>Parameters:</b>	The <b><i>n1</i></b> and <b><i>n2</i></b> are numbers or numeric expressions.
<b>Use:</b>	This function returns the larger of <b><i>n1</i></b> and <b><i>n2</i></b> . <b>DATE/TIME</b> field values can be used resulting in the most recent date/time being returned.
<b>Examples:</b>	MAX(54,23) => 54 MAX(DATE OF BIRTH,TODAY) => AUG 23,1991

#### 6.3.4.4 MIN

Table 50: VA FileMan Functions—Mathematical Function: MIN

Item	Description
<b>Format:</b>	<b>MIN(<i>n1</i>,<i>n2</i>)</b>
<b>Parameters:</b>	The <b><i>n1</i></b> and <b><i>n2</i></b> are numbers or numeric expressions.
<b>Use:</b>	This function returns the smaller of <b><i>n1</i></b> and <b><i>n2</i></b> . <b>DATE/TIME</b> field values can be used resulting in the earliest date/time being returned.
<b>Examples:</b>	MIN(54,23) => 23 MIN(DATE OF BIRTH,TODAY) => NOV 1,1938

#### 6.3.4.5 MODULO

Table 51: VA FileMan Functions—Mathematical Function: MODULO

Item	Description
<b>Format:</b>	<b>Format: MODULO(<i>n1</i>,<i>n2</i>)</b>
<b>Parameters:</b>	The <b><i>n1</i></b> , <b><i>n2</i></b> are numbers or numeric expressions: <ul style="list-style-type: none"><li>• <b><i>n1</i></b> is the dividend.</li><li>• <b><i>n2</i></b> is the divisor.</li></ul>
<b>Use:</b>	This mathematical function returns the remainder when <b><i>n2</i></b> is divided into <b><i>n1</i></b> ; it performs modulo division.
<b>Example:</b>	MODULO(54,5) => 4

### 6.3.4.6 SQUAREROOT

Table 52: VA FileMan Functions—Mathematical Function: SQUAREROOT

Item	Description
Format:	<b>SQUAREROOT(<i>n</i>)</b>
Parameters:	The <i>n</i> is a numeric expression <b>greater than 0</b> .
Use:	This mathematical function returns the square root of <i>n</i> .
Example:	SQUAREROOT(9) => 3

## 6.3.5 Printing Related Functions

### 6.3.5.1 IOM

Table 53: VA FileMan Functions—Printing Related Function: IOM

Item	Description
Format:	<b>IOM</b>
Parameters:	(none)
Use:	This argumentless function returns the number of columns for the present output device.
Example:	IOM/2 => 0

### 6.3.5.2 PAGE

Table 54: VA FileMan Functions—Printing Related Function: PAGE

Item	Description
<b>Format:</b>	<b>PAGE</b>
<b>Parameters:</b>	(none)
<b>Use:</b>	This argumentless function returns the current page number when output is being printed.
<b>Example:</b>	"Page "_PAGE => Page 23 [the 23rd page of output]

## 6.3.6 String Functions

### 6.3.6.1 DUP

Table 55: VA FileMan Functions—String Function: DUP

Item	Description
<b>Format:</b>	<b>DUP(string,n)</b>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b> is any string of characters or an expression yielding a string of characters.</li> <li>• <b>n</b> is a positive integer or a numeric expression.</li> </ul>
<b>Use:</b>	This function returns a string of characters <b>n</b> characters long. If <b>string</b> is less than <b>n</b> characters long, the characters in <b>string</b> are repeated until the output string is <b>n</b> characters in length.
<b>Example:</b>	DUP(DIAGNOSIS,3) => Ang [value of DIAGNOSIS = Angina Pectoris] DUP("_",IOM) => _____ [line drawn has a length equal to the value of IOM]

### 6.3.6.2 LOWERCASE

Table 56: VA FileMan Functions—String Function: LOWERCASE

Item	Description
<b>Format:</b>	<b>LOWERCASE(string)</b>
<b>Parameters:</b>	<b>string</b> is an expression yielding alphabetic characters.
<b>Use:</b>	This function changes uppercase characters in <b>string</b> to lowercase except for the first character and the first character after a punctuation mark: <ul style="list-style-type: none"> <li>• A space is a punctuation mark; thus, the first letter of a word is <i>not</i> changed.</li> <li>• <b>String cannot</b> be a <b>WORD-PROCESSING</b> field; the contents of a <b>WORD-PROCESSING</b> field are unaffected.</li> </ul>
<b>Example:</b>	LOWERCASE("FMPATIENT,20") => Fmpatient,20

### 6.3.6.3 PADRIGHT

Table 57: VA FileMan Functions—String Function: PADRIGHT

Item	Description
<b>Format:</b>	<b>PADRIGHT(string,n)</b>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b> is a string or string expression to be printed.</li> <li>• <b>n</b> is the total size of the output string.</li> </ul>
<b>Use:</b>	This function pads <b>string</b> on the right with spaces to make a string <b>n</b> characters long. If <b>string</b> is longer than <b>n</b> characters, the entire string is returned; this function does <i>not</i> truncate.
<b>Examples:</b>	PADRIGHT("Peter",10) => Peter [five spaces after the 'r'] PADRIGHT(CITY,15) => San Juan Capistrano

### 6.3.6.4 REPLACE

Table 58: VA FileMan Functions—String Function: REPLACE

Item	Description
<b>Format:</b>	<b>REPLACE(string,oldstring,newstring)</b>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b>—String expression that is changed.</li> <li>• <b>oldstring</b>—String expression containing the characters in <b>string</b> that are replaced.</li> <li>• <b>newstring</b>—String expression containing the characters that replace those in <b>oldstring</b>.</li> </ul>
<b>Use:</b>	This function returns the input <b>string</b> with all occurrences of the <b>oldstring</b> changed to the <b>newstring</b> . The <b>oldstring</b> and <b>newstring</b> can be any length. They do <i>not</i> have to be equal in length.
<b>Examples:</b>	REPLACE("abracadabra","ab","*") => *racad*ra REPLACE("Name is: XXX","XXX",NAME) => Name is: FMPATIENT,21

### 6.3.6.5 REVERSE

Table 59: VA FileMan Functions—String Function: REVERSE

Item	Description
<b>Format:</b>	<b>REVERSE(string)</b>
<b>Parameters:</b>	<b>string</b> is a string expression.
<b>Use:</b>	This function returns the characters in <b>string</b> in reverse order.
<b>Example:</b>	REVERSE(NAME) => neB,nilknarF

### 6.3.6.6 STRIPBLANKS

Table 60: VA FileMan Functions—String Function: STRIPBLANKS

Item	Description
<b>Format:</b>	<b>STRIPBLANKS(string)</b>
<b>Parameters:</b>	<b>string</b> is a string expression.
<b>Use:</b>	This function removes leading and trailing spaces from <b>string</b> .
<b>Example:</b>	STRIPBLANKS(" Waste no space ") => Waste no space [no leading or trailing spaces]

### 6.3.6.7 TRANSLATE

Table 61: VA FileMan Functions—String Function: TRANSLATE

Item	Description
<b>Format:</b>	<b>TRANSLATE(string,"oldchar","newchar")</b>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b>—String expression to be changed.</li> <li>• <b>oldchar</b>—Characters to be translated.</li> <li>• <b>newchar</b>—Characters to replace the <b>oldchar</b>.</li> </ul>
<b>Use:</b>	This function alters <b>string</b> by changing each character in <b>oldchar</b> into the character in the corresponding position in <b>newchar</b> . The translation is one character for one character.
<b>Examples:</b>	TRANSLATE("08261991","123","ABC") => 08B6A99A TRANSLATE(NAME,"F","f") => fMPATIENT,fORTY-ONE

### 6.3.6.8 UPPERCASE

Table 62: VA FileMan Functions—String Function: UPPERCASE

Item	Description
Format:	UPPERCASE(string)
Parameters:	<b>string</b> is an expression with alphabetic characters.
Use:	This function changes lowercase characters in <b>string</b> to uppercase. <b>String</b> <i>cannot</i> be a <b>WORD-PROCESSING</b> field; the contents of a <b>WORD-PROCESSING</b> field are unaffected.
Example:	UPPERCASE("vista") => VISTA

## 6.3.7 Temporary Data Storage Functions

### 6.3.7.1 PARAM

Table 63: VA FileMan Functions—Temporary Data Storage Function: PARAM

Item	Description
Format:	PARAM("parameter")
Parameters:	<b>parameter</b> has been assigned a value by the <b>SETPARAM</b> function.
Use:	This function works with the <b>SETPARAM</b> function. It returns the value that has been given to <b>parameter</b> by use of the <b>SETPARAM</b> function.
Example:	PARAM("AGE") => 45

### 6.3.7.2 SETPARAM

Table 64: VA FileMan Functions—Temporary Data Storage Function: SETPARAM

Item	Description
<b>Format:</b>	<b>SETPARAM(value,"parameter")</b>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>value</b> is an expression to be evaluated.</li> <li>• <b>parameter</b> is a string <b>1 to 30 characters</b> long identifying a storage location to hold value.</li> </ul>
<b>Use:</b>	This function works with the PARAM function. It returns nothing. <b>Value</b> is stored in <b>parameter</b> for later reference.
<b>Example:</b>	SETPARAM(TODAY-DATE OF BIRTH\365,"AGE") => [no output; result of the expression put into "AGE"]

### 6.3.7.3 VAR

Table 65: VA FileMan Functions—Temporary Data Storage Function: VAR

Item	Description
<b>Format:</b>	<b>VAR("variable")</b>
<b>Parameters:</b>	<b>variable</b> is a variable in the local symbol table.
<b>Use:</b>	This function returns the value of <b>variable</b> . The <b>variable</b> can be one that you set using the <b>SET</b> function.
<b>Examples:</b>	<b>VAR("COUNT") =&gt; 1</b> [1 is the current value of <b>COUNT</b> ] <b>VAR("DUZ") =&gt; 160</b>

### 6.3.7.4 SET

Table 66: VA FileMan Functions—Temporary Data Storage Function: SET

Item	Description
<b>Format:</b>	<b>SET(value,"variable")</b>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>value</b> is an expression to be evaluated.</li> <li>• <b>variable</b> is a local variable name used to hold the value of value.</li> </ul>
<b>Use:</b>	This function returns <b>value</b> 's value. In addition, the value is placed in a local variable. <b>Variable</b> should be namespaced to avoid conflict with other local variables. You can use this function only if you have <b>Programmer</b> access.
<b>Example:</b>	<b>SET(1,"COUNT") =&gt; 1</b> [this would put <b>1</b> into the <b>COUNT</b> variable]

## 6.3.8 M-Related Functions

### 6.3.8.1 \$A[SCII]

Table 67: VA FileMan Functions—M-Related Function: \$A[SCII]

Item	Description
<b>Format:</b>	<b>\$A(string,n)</b>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b> is a string of characters or an expression yielding a string.</li> <li>• <b>n</b> is an integer or expression yielding an integer.</li> </ul>
<b>Use:</b>	The function returns the numeric ASCII value of the character in position <b>n</b> within <b>string</b> . If <b>n</b> is <i>not</i> specified, the value of the first character is returned.
<b>Examples:</b>	<b>\$A(NAME,4) =&gt; 77</b> [NAME is SHAM,SAM THE] <b>\$A("Get the value") =&gt; 71</b>

### 6.3.8.2 \$C[HAR]

Table 68: VA FileMan Functions—M-Related Function: \$C[HAR]

Item	Description
<b>Format:</b>	$\$C(n, \dots)$
<b>Parameters:</b>	The <i>n</i> is an integer or an expression yielding an integer.
<b>Use:</b>	This function returns the character corresponding to the ASCII value of <i>n</i> . If more than one <i>n</i> is specified in the argument, a string of characters is returned.
<b>Examples:</b>	$\$C(100) \Rightarrow d$ $\$C(99,100,101) \Rightarrow cde$

### 6.3.8.3 \$E[XTRACT]

Table 69: VA FileMan Functions—M-Related Function: \$E[XTRACT]

Item	Description
<b>Format:</b>	<ol style="list-style-type: none"> <li>1. <math>\\$E(\text{string}, n1, n2)</math></li> <li>2. <math>\\$E(\text{string}, n)</math></li> <li>3. <math>\\$E(\text{string})</math></li> </ol>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b> is a string expression.</li> <li>• <b>n, n1, and n2</b> are positive integers or expressions yielding positive integers.</li> </ul>
<b>Use:</b>	<p>This function returns a substring from <b>string</b>:</p> <ul style="list-style-type: none"> <li>• If you use only <b>string</b> as an argument, the first character is returned.</li> <li>• If you specify one <b>n</b>, the character in that position in the string is returned.</li> <li>• If you specify <b>n1</b> and <b>n2</b>, a string starting at <b>n1</b> and ending at <b>n2</b> is returned.</li> </ul>
<b>Examples:</b>	$\$E(\text{NAME}, 3, 7) \Rightarrow \text{patie}$ [NAME is FMPATIENT, 21] $\$E(\text{NAME}, 2) \Rightarrow m$ $\$E(\text{NAME}) \Rightarrow S$

### 6.3.8.4 \$F[IND]

Table 70: VA FileMan Functions—M-Related Function: \$F[IND]

Item	Description
<b>Format:</b>	<ol style="list-style-type: none"> <li>1. \$F(string,target)</li> <li>2. \$F(string,target,n)</li> </ol>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b>—String expression.</li> <li>• <b>target</b>—Characters or an expression yielding the characters to be searched.</li> <li>• <b>n</b>—Positive integer or an expression yielding a positive integer.</li> </ul>
<b>Use:</b>	<p>This function returns the character position in <b>string</b> following the <b>target</b>:</p> <ul style="list-style-type: none"> <li>• If <b>n</b> is specified as a third argument, the search for <b>target</b> is begun after character position <b>n</b>.</li> <li>• If <b>target</b> is <i>not</i> found, <b>0</b> is returned.</li> </ul>
<b>Examples:</b>	<p>\$F("FMPATIENT,21",",") =&gt; 7            \$F(NAME,",",7) =&gt; 0 [NAME has value of FMPATIENT,21]</p>

### 6.3.8.5 \$H[OROLOG]

Table 71: VA FileMan Functions—M-Related Function: \$H[OROLOG]

Item	Description
<b>Format:</b>	\$H
<b>Parameters:</b>	(none)
<b>Use:</b>	<p>This system variable returns the date and time in internal M format. The format is number of days since <b>December 31, 1840</b>, followed by a comma, followed by the number of seconds from midnight.</p>
<b>Example:</b>	\$H => 55032,48780

### 6.3.8.6 \$I[O]

Table 72: VA FileMan Functions—M-Related Function: \$I[O]

Item	Description
<b>Format:</b>	<b>\$I</b>
<b>Parameters:</b>	(none)
<b>Use:</b>	This system variable returns the current device. It can return the operating system's designation of the current device.
<b>Example:</b>	<b>\$I =&gt; _LTA9239</b>

### 6.3.8.7 \$J[OB]

Table 73: VA FileMan Functions—M-Related Function: \$J[OB]

Item	Description
<b>Format:</b>	<b>\$J</b>
<b>Parameters:</b>	(none)
<b>Use:</b>	This system variable returns your current job number.
<b>Example:</b>	<b>\$J =&gt; 666172581</b>

### 6.3.8.8 \$J[USTIFY]

Table 74: VA FileMan Functions—M-Related Function: \$J[USTIFY]

Item	Description
Format:	<ol style="list-style-type: none"> <li>1. \$J(string,n)</li> <li>2. \$J(n1,n2,n3)</li> </ol>
Parameters:	<ol style="list-style-type: none"> <li>1. In the first format, string is a string expression; <b>n</b> is an integer representing width of field.</li> <li>2. In the second format: <ul style="list-style-type: none"> <li>• <b>n1</b> is a numeric expression.</li> <li>• <b>n2</b> is an integer representing the width of field.</li> <li>• <b>n3</b> is the number of decimal places to output with the number.</li> </ul> </li> </ol>
Use:	<ol style="list-style-type: none"> <li>1. In the first format, the function returns string right justified within a field that has a width of <b>n</b>. If string is longer than <b>n</b>, there is no truncation.</li> <li>2. In the second format, the function returns <b>n1</b> right justified in a field that has a width of <b>n2</b>. There are <b>n3</b> decimal places to the right of the decimal point.</li> </ol>
Example:	<p><b>\$J(NAME,20) =&gt;</b>                    <b>FMPATIENT,21</b>  [12 spaces preceding the "F"]  <b>"\$"_\$J(PRESCRIPTION COST,8,2) =&gt;</b>  <b>\$ 25.88</b>  [3 spaces preceding the "2"]</p>

### 6.3.8.9 \$L[ENGTH]

Table 75: VA FileMan Functions—M-Related Function: \$L[ENGTH]

Item	Description
<b>Format:</b>	<ol style="list-style-type: none"> <li>1. \$L(string)</li> <li>2. \$L(string,delimiter)</li> </ol>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b> is a string expression.</li> <li>• <b>delimiter</b> is a character (or characters) or an expression yielding a character (or characters) that divides the string into pieces.</li> </ul>
<b>Use:</b>	<ol style="list-style-type: none"> <li>1. In the first format, the function returns the number of characters in <b>string</b>.</li> <li>2. In the second format, the function returns the number of pieces into which <b>delimiter</b> divides the string. If <b>delimiter</b> does <i>not</i> exist within <b>string</b>, <b>1</b> is returned.</li> </ol>
<b>Examples:</b>	<p>\$L(PROVIDER) =&gt; 11 [PROVIDER is FM PROVIDER,5]            \$L(PROVIDER,",") =&gt; 2 [same PROVIDER]</p>

### 6.3.8.10 \$P[IECE]

Table 76: VA FileMan Functions—M-Related Function: \$P[IECE]

Item	Description
<b>Format:</b>	<ol style="list-style-type: none"> <li>1. <b>\$P(string,"delimiter",n)</b></li> <li>2. <b>\$P(string,"delimiter",n1,n2)</b></li> <li>3. <b>\$P(string,"delimiter")</b></li> </ol>
<b>Parameters:</b>	<ul style="list-style-type: none"> <li>• <b>string</b> is a string expression.</li> <li>• <b>delimiter</b> is a character (or characters) or an expression yielding a character (or characters) that divides the string into pieces.</li> <li>• <b>n, n1, and n2</b> are positive integers or expressions evaluating to positive integers.</li> </ul>
<b>Use:</b>	<p>The function returns a part of <b>string</b>. <b>String</b> is divided into substrings by delimiter:</p> <ul style="list-style-type: none"> <li>• In the first format, the <b>n<sup>th</sup></b> substring is returned.</li> <li>• In the second format, the substrings starting with <b>n1</b> and ending with <b>n2</b> are returned. The delimiters between those substrings are also returned.</li> <li>• In the third format, the first substring (i.e., the one preceding the first occurrence of delimiter) is returned.</li> </ul>
<b>Examples:</b>	<p><b>\$P("FMPATIENT,22",",",2) =&gt; 22</b>  <b>\$P(PHONE,"-",2,3) =&gt; 943-2109</b>  <b>\$P(PHONE,"-") =&gt; 510</b></p>

### 6.3.8.11 \$R[ANDOM]

Table 77: VA FileMan Functions—M-Related Function: \$R[ANDOM]

Item	Description
<b>Format:</b>	\$R( <i>n</i> )
<b>Parameters:</b>	The <i>n</i> is a positive integer or an expression evaluating to a positive integer.
<b>Use:</b>	This function returns a randomly generated integer from the range of <b>0</b> through <i>n-1</i> .
<b>Example:</b>	\$R(5000) => 1076

### 6.3.8.12 \$\$[ELECT]

Table 78: VA FileMan Functions—M-Related Function: \$\$[ELECT]

Item	Description
<b>Format:</b>	\$\$( <b>test:value,test:value,...</b> )
<b>Parameters:</b>	<ul style="list-style-type: none"> <li><b>expression</b> is an expression that can be evaluated as <b>True</b> or <b>False</b> (<i>not Zero</i> or <b>Zero</b>).</li> <li><b>value</b> is any expression that can yield a value.</li> </ul>
<b>Use:</b>	Each <b>value</b> is associated with the <b>test</b> from which it is separated by a colon. The function returns the evaluation of the <b>value</b> associated with the first <b>test</b> that evaluates as true (i.e., <i>not</i> equal to <b>Zero</b> ). Any number of <b>test:value</b> pairs can be used; however, one of the tests <i>must</i> evaluate as <b>true</b> . To assure that one test always evaluates as <b>true</b> , the last test is usually the literal <b>1</b> .
<b>Examples:</b>	<p>\$\$("SIX FMPROVIDER, Ph.D."["M.D.":"He is a medical doctor.",1:"He is not a medical doctor."]) =&gt; He is not a medical doctor.</p> <p>\$\$(OCCURRENCES&gt;3:"Chronic Condition",OCCUR-RENCES&gt;0: "Non-Chronic Condition",1:"No Occurrences Recorded") =&gt; Chronic Condition [Here the contents of the OCCURRENCES field is being tested. If the first test (&gt;3) is true (as in this example), the result of the second test (&gt;0) is <i>not</i> relevant.]</p>

### 6.3.8.13 \$S[TORAGE]

Table 79: VA FileMan Functions—M-Related Function: \$S[TORAGE]

Item	Description
Format:	\$S
Parameters:	(none)
Use:	This system variable returns the number of bytes of free space available for use. Its meaning varies with the M implementation.
Example:	\$S => 52672

### 6.3.8.14 \$X

Table 80: VA FileMan Functions—M-Related Function: \$X

Item	Description
Format:	\$X
Parameters:	(none)
Use:	This system variable returns the current <b>X</b> coordinate (column) location of the cursor or print head. If the application that moved the cursor did <i>not</i> update the value of <b>\$X</b> , the value of <b>\$X</b> is <i>not</i> reliable.
Example:	\$X => 43

### 6.3.8.15 \$Y

**Table 81: VA FileMan Functions—M-Related Function: \$Y**

Item	Description
<b>Format:</b>	\$Y
<b>Parameters:</b>	(none)
<b>Use:</b>	This system variable returns the current <b>Y</b> coordinate (row) location of the cursor. Like <b>\$X</b> , its reliability depends on the controlling application.
<b>Example:</b>	\$Y => 6

## 7 Statistics

### 7.1 How to Generate Statistics from Reports

VA FileMan currently offers three types of statistical processing:

- [Descriptive Statistics](#)
- [Scattergram](#)
- [Histogram](#)

In each case, to generate statistics from reports, you use a two-step process:

1. Use the **Print File Entries** [DIPRINT] or **Search File Entries** [DISEARCH] options to generate a VA FileMan report. Do *not* queue the report. The entries you select in your report are the ones on which statistics are generated; the way you use sort and print qualifiers in the report affects the way statistics are generated, as discussed later in this section.
2. Immediately after the report finishes, use the **Statistics** [DISTATISTICS] option, which is located on the **Other Options** [DIOOTHER] menu under the **VA FileMan** [DIUSER] menu, to generate statistics.

The two-step process for each type of statistical output is described below.



**NOTE:** If you have statistical software on a personal computer, you might want to consider using VA FileMan's Export Tool as an alternative to VA FileMan's statistics options, especially if the statistics options described in this section do *not* provide the statistical analysis you need. With the Export Tool, you can export your data into a format your personal computer statistical software can read and use all of that software's capabilities to perform statistical analyses on VA FileMan data.

## 7.2 Descriptive Statistics

The **Descriptive Statistics** routine creates a summary report of the numeric information produced by the preceding print. The number of cases is always shown.

To get descriptive statistics for fields printed out in a report, you *must* associate one of the qualifiers listed in [Table 82](#) with fields in the print:

**Table 82: Statistics—Descriptive Statistics Qualifiers**

Qualifier	Description
#	Count, mean, standard deviation, minimum, and maximum
+	Count and mean

To obtain descriptive statistics:

1. Print a report and use the # or + print qualifiers on one or more fields.
2. Immediately after the report completes, generate the Descriptive Statistics based on the report.

### 7.2.1 Initial Print

**Figure 89: Statistics—Initial Print Dialog with Descriptive Statistics**

Using the # print qualifier prints total, count, mean, standard deviation, minimum, and maximum for these two fields.

```
FIRST PRINT FIELD: #BUDGET
THEN PRINT FIELD: #COST
THEN PRINT FIELD: <ENTER>
HEADING (S/C): PATIENT STATISTICS// <ENTER>
DEVICE: <ENTER> SSH VIRTUAL TERMINAL RIGHT MARGIN: 80// <ENTER>
...SORRY, JUST A MOMENT PLEASE...
Output is generated here.
.
.
.
```

## 7.2.2 Generating Descriptive Statistics

Figure 90: Statistics—Generating Descriptive Statistics

```
SELECT OPTION: OTHER OPTIONS
SELECT OTHER OPTION: STATISTICS
SELECT STATISTICAL ROUTINE: DES <ENTER> CRIPTIVE STATISTICS
      USER: FMUSER,TWO          2:51 PM 02/15/96
```

DESCRIPTIVE STATISTICS

	N OF CASES	MEAN	STANDARD DEVIATION	MINIMUM	MAXIMUM
BUDGET	27	45845.1481	25685.8582	2589.0000	95200.0000
COST	27	45914.1111	25796.2936	259.0000	96000.0000

## 7.3 Scattergram

If you subtotal by two fields (i.e., sub-subtotal) in a sort, you can create Scattergrams for fields that were counted with **!**, **+**, or **#** in the corresponding print.

Only numeric values are charted. The Scattergram is scaled to fit your output device's row and column dimensions. Occurrences of more than nine points in a single print position are marked by an asterisk (\*).

### 7.3.1 Initial Print

Figure 91: Statistics—Initial Print Dialog for a Scattergram

```
SELECT VA FILEMAN OPTION: PRINT <ENTER> FILE ENTRIES
OUTPUT FROM WHAT FILE: PATIENT// <ENTER>
Subtotal by two fields ("+xxxxxxxx").
SORT BY: NAME// +WARD LOCATION
START WITH WARD LOCATION: FIRST// <ENTER>
WITHIN WARD LOCATION, SORT BY: +ROOM-BED
START WITH ROOM-BED: FIRST// <ENTER>
WITHIN ROOM-BED, SORT BY: <ENTER>
Use print qualifiers for these two fields ("!xxxxxxxx"), which is
used for the scattergram.
FIRST PRINT FIELD: !WARD LOCATION
THEN PRINT FIELD: !ROOM-BED
THEN PRINT FIELD: <ENTER>
HEADING (S/C): PATIENT STATISTICS// <ENTER>
DEVICE: <ENTER> SSH VIRTUAL TERMINAL RIGHT MARGIN: 80// <ENTER>
...SORRY, JUST A MOMENT PLEASE...
Output is generated here.
.
.
.
```



## 7.4 Histogram

If you subtotal by one or more fields in a sort, you can get Histograms for the fields that are preceded by #, !, &, or + qualifiers in the corresponding print. The Histograms that you can produce depend on which print qualifier is used, as listed in [Table 83](#):

**Table 83: Statistics—Histogram Qualifiers**

Qualifier	Description
!	Produces a Count Histogram
&	Produces a Sum Histogram
+	Produces Count, Sum, and Mean Histograms
#	Produces Count, Sum, and Mean Histograms

## 7.4.1 Initial Print

Figure 93 is an example of using a subtotal in a print, and then producing a Count Histogram:

**Figure 93: Statistics—Initial Print Dialog for a Count Histogram**

```
SELECT VA FILEMAN OPTION: PRINT <ENTER> FILE ENTRIES
OUTPUT FROM WHAT FILE: PATIENT// SIGN-ON LOG <ENTER> (159963 ENTRIES)
Subtotal on a field in the sort.
SORT BY: DATE/TIME// +NODE NAME
START WITH NODE NAME: FIRST// <ENTER>
WITHIN NODE NAME, SORT BY: <ENTER>
FIRST PRINT FIELD: DATE/TIME
Use the ! print qualifier, so you can create a Count Histogram on the NODE NAME field.
THEN PRINT FIELD: !NODE NAME
THEN PRINT FIELD: <ENTER>
HEADING (S/C): SIGN-ON LOG STATISTICS REPLACE <ENTER>
DEVICE: <ENTER> SSH VIRTUAL TERMINAL RIGHT MARGIN: 80// <ENTER>
...HMMM, JUST A MOMENT PLEASE...
Output is generated here.
.
.
.
```

## 7.4.2 Generating the Histogram

Figure 94: Statistics—Generating the Count Histogram Diagram

```
SELECT VA FILEMAN OPTION: OTHER <ENTER> OPTIONS
      FILEGRAMS ...
      AUDIT MENU ...
      SCREENMAN ...
      STATISTICS
      VA FILEMAN MANAGEMENT ...
      DATA EXPORT TO FOREIGN FORMAT ...
      EXTRACT DATA TO FILEMAN FILE ...
      IMPORT DATA
      BROWSER

SELECT OTHER OPTIONS OPTION: STAT <ENTER> ISTICS
SELECT STATISTICAL ROUTINE: HISTOGRAM

DEVICE: HOME// <ENTER> SSH VIRTUAL TERMINAL RIGHT MARGIN: 80// <ENTER>

                COUNT, NODE NAME, BY NODE NAME

XXXXY1 |*****
XXXXY2 |*****
XXXXY3 |*****
XXXXY4 |*****
      +-----+-----+-----+-----+-----+-----+
                8      16      24      31      39      47      55      63
```

## 8 System Management

VA FileMan is designed to be used either with Kernel or as a standalone application running under a variety of implementations of ANSI standard M. If VA FileMan is used without Kernel, the basic DBMS features of VA FileMan all work as described in the manuals. However, there are some features (e.g., bulletin-type cross-references, print queuing, and Filegrams) that do *not* work without portions of Kernel. Whenever Kernel is needed to support a particular VA FileMan feature, that fact is mentioned in the manuals.

The installation of VA FileMan 22.2 is *not* integrated with the installation of Kernel. The *VA FileMan Installation Guide* contains instructions on how to install VA FileMan, both for standalone sites and for sites running Kernel.

### 8.1 Setup

#### 8.1.1 Initialization

VA FileMan 22.2 is installed and initialized with a Kernel Installation and Distribution System (KIDS) build. That build installs all the VA FileMan routines, updates all VA FileMan files, populates those files with necessary data, and installs other necessary VA FileMan components (e.g., Kernel options). In the past, the VA FileMan **DINIT** routine was run to initialize the VA FileMan files. Now, **DINIT** is run automatically from the KIDS install. It should not be run independently after the install.



**REF:** For more information on installing VA FileMan 22.2, see the *VA FileMan 22.2 Installation Guide*.

#### 8.1.2 Security

VA FileMan provides tools for application packages to protect their data. VA FileMan identifies users by a number in the local variable **DUZ** and user security by codes stored in the local variable **DUZ(0)**.



**REF:** For a description of the use and setup of file, field, and template security, see the "[Data Security](#)" section.

## 8.2 Standalone VA FileMan

### 8.2.1 Device Handling for Standalone VA FileMan

VA FileMan requires a **%ZIS** routine. Kernel supplies a **%ZIS** as the device selection gateway to its device handler component. In addition to writing your own device selection routine and saving it as **%ZIS**, standalone users have two possibilities:

1. You can have a device selection routine supplied by your M vendor. You can use the M vendor's routine if it returns the variables expected by VA FileMan from such a device selection program. These variables are listed in [Table 84](#). In this case, you simply create a **%ZIS** routine that calls the vendor-supplied routine.

However, even if you use a vendor's (or your own) routine for device selection, you *must* have **%ZISS**, as described below, if you want to use VA FileMan's screen-oriented utilities.

2. If you do *not* have another device selection routine, refile the **DIIS** and **DISS** routines as described in the *VA FileMan Installation Guide*. This results in all VA FileMan output going to the terminal that requests it. If you do *not* modify the **%ZIS** and **%ZISS** routines as described below, your terminal is treated as a **VT100** (ANSI) terminal.

VA FileMan controls terminals by using terminal characteristics stored in **IO** variables. In addition, certain operating system dependent actions are controlled by executing code stored in **%ZOSF** nodes. Together the **IO** variables and **%ZOSF** nodes allow full use of VA FileMan in both scrolling and screen-oriented modes. The instructions below describe how to modify the **%ZIS** and **%ZISS** routines to set the necessary **IO** variables, and how to set the necessary **%ZOSF** nodes.

### 8.2.1.1 Setting IO variables: %ZIS and %ZISS

**%ZIS** sets the **IO** variables required for terminal output that is *not* screen oriented. The **DIIS** routine supplied with VA FileMan sets the **IO** variables to the values specific to the **VT100** terminal type. If you are using or emulating a **VT100** terminal, you can rename the **DIIS** routine unmodified as **%ZIS**. If you are using or emulating a **VT220** or **VT320** terminal, you *must* modify **%ZIS** to set **IOST** equal to "**C-VT220**" or "**C-VT320**".

[Table 84](#) lists the variables returned by **%ZIS**:

**Table 84: System Management—%ZIS Variables Returned**

Variable	Description
<b>IO</b>	The device <b>\$I</b> . If <b>IO</b> is <b>NULL</b> (""), no input or output occurs.
<b>IOM</b>	The margin width (e.g., <b>80</b> ).
<b>ION</b>	The device name consists of <b>1 to 30 alphanumeric characters</b> .
<b>IOSL</b>	The screen length (e.g., <b>24</b> ).
<b>IOF</b>	The indirect argument of a <b>WRITE</b> statement to generate a top-of-page (e.g., <b>#</b> ).
<b>IOST</b>	The output device type (e.g., <b>CRT</b> ): <ul style="list-style-type: none"> <li>• If <b>IOST</b> begins with the letter <b>C</b>, the Inquire, search, and print output programs wait until the user presses the <b>Enter</b> key after each screen's worth of display.</li> <li>• If <b>IOST</b> begins with a <b>P</b>, output terminates with a page feed.</li> <li>• If <b>IOST</b> contains <b>SINGLE</b>, output stops after each page feed and waits for the <b>Enter</b> key to be pressed (&lt;<b>Enter</b>&gt;).</li> <li>• If the output terminal is other than the terminal requesting the output, and <b>IOST</b> does <i>not</i> contain <b>K</b>, the &lt;<b>Enter</b>&gt; is read from the requesting terminal.</li> </ul>
<b>IOPAR</b>	The parameter that should follow the first colon in the argument of the <b>OPEN</b> command. For most devices, this string should be <b>NULL</b> .
<b>IOT</b>	Equal to a string naming the device type (e.g., <b>IOT=TRM</b> ). This variable <i>must</i> be returned equal to the string <b>SDP</b> , so VA FileMan's multiple-copies feature is called.
<b>IOXY</b>	The executable M code that performs cursor positioning given the input variables <b>DX</b> and <b>DY</b> . <b>DX</b> and <b>DY</b> contain the column and row positions, respectively, to which to move the cursor.

The **%ZISS** routine sets the **IO** variables required by VA FileMan’s screen-oriented utilities. The **DISS** routine supplied by VA FileMan sets the **IO** variables according to the value of **IOST**. **DISS** recognizes **IOST** values of **C-VT220** and **C-VT320**. If **IOST** equals anything else, the **IO** variables are set to values specific to the **VT100** terminal type.

Not all variables returned by **%ZISS** are required by VA FileMan’s screen-oriented utilities.

[Table 85](#) lists the variables returned by **%ZISS**:

**Table 85: System Management—%ZISS Variables Returned**

<b>Variable</b>	<b>Description</b>
<b>IOAWM0</b>	Auto wrap mode off
<b>IOAWM1</b>	Auto wrap mode on
<b>IOCOMMA</b>	Keypad’s comma key
<b>IOCUB</b>	Cursor backward
<b>IOCUD</b>	Cursor down
<b>IOCUF</b>	Cursor forward
<b>IOCUU</b>	Cursor up
<b>IODCH</b>	Delete character
<b>IODL</b>	Delete line
<b>IODO</b>	Do key
<b>IOEDALL</b>	Erase in display entire page
<b>IOEDEOP</b>	Erase in display from cursor to end of page
<b>IOELALL</b>	Erase in line entire line
<b>IOELEOL</b>	Erase in line from cursor to end of line
<b>IOENTER</b>	Keypad’s enter key
<b>IOFIND</b>	Find key

<b>Variable</b>	<b>Description</b>
<b>IOHELP</b>	Help key
<b>IOICH</b>	Insert character
<b>IOIL</b>	Insert line
<b>IOINHI</b>	High intensity
<b>IOINLOW</b>	Low intensity
<b>IOINORM</b>	Normal intensity
<b>IOINSERT</b>	Insert key
<b>IOIRM0</b>	Replace mode
<b>IOIRM1</b>	Insert mode
<b>IOKP0</b>	Keypad 0 key
<b>IOKP1</b>	Keypad 1 key
<b>IOKP2</b>	Keypad 2 key
<b>IOKP3</b>	Keypad 3 key
<b>IOKP4</b>	Keypad 4 key
<b>IOKP5</b>	Keypad 5 key
<b>IOKP6</b>	Keypad 6 key
<b>IOKP7</b>	Keypad 7 key
<b>IOKP8</b>	Keypad 8 key
<b>IOKP9</b>	Keypad 9 key
<b>IOKPAM</b>	Keypad application mode on
<b>IOKPNM</b>	Keypad numeric mode on
<b>IOMINUS</b>	Keypad's minus key

<b>Variable</b>	<b>Description</b>
<b>IONEXTSC</b>	Next screen key
<b>IOPERIOD</b>	Keypad's period key
<b>IOPF1</b>	Function key 1
<b>IOPF2</b>	Function key 2
<b>IOPF3</b>	Function key 3
<b>IOPF4</b>	Function key 4
<b>IOPREVSC</b>	Previous screen key
<b>IOREMOVE</b>	Keypad's remove key
<b>IORI</b>	Reverse index
<b>IORVOFF</b>	Reverse video off
<b>IORVON</b>	Reverse video on
<b>IOSGR0</b>	Turn off select graphic rendition attributes
<b>IOSELECT</b>	Select key
<b>IOSTBM</b>	Set top and bottom margins
<b>IOUOFF</b>	Underline off
<b>IOUON</b>	Underline on
<b>IOBLC</b>	Bottom left corner
<b>IOBRC</b>	Bottom right corner
<b>IOBT</b>	Bottom "T"
<b>IOG0</b>	Graphics off
<b>IOG1</b>	Graphics on
<b>IOHL</b>	Horizontal line

Variable	Description
<b>IOLT</b>	Left "T"
<b>IOMT</b>	Middle "T", or cross hair (+)
<b>IORT</b>	Right "T"
<b>IOTLC</b>	Top left corner
<b>IOTRC</b>	Top right corner
<b>IOTT</b>	Top "T"
<b>IOVL</b>	Vertical line

After you save **DIISS** as **%ZISS**, you can modify **%ZISS** to use ScreenMan on terminal types other than those supported in **DIISS**. The routine **DIISS** itself contains more information on how to modify the routine.



**NOTE: IO variables and DataTree MUMPS:** If your version of DataTree MUMPS supports **VT220** emulation, set up **%ZIS** and **%ZISS** for **VT220** and use the emulation.

### 8.2.1.2 Summary of IO Setups

Depending on your terminal type, you need to take the appropriate action listed in [Table 86](#) to make full use of VA FileMan's screen-oriented utilities.

**Table 86: System Management—Optimal Procedures for Screen-Oriented Utilities: Based on Terminal Type**

Terminal Type	Description
VT100	No action needed; the default settings are acceptable.
VT220, VT320	Edit the <b>%ZIS</b> routine to set <b>IOST</b> variable to " <b>C-VT220</b> " or " <b>C-VT320</b> ".
Other terminal types	Edit <b>%ZIS</b> to set <b>IOST</b> to " <b>C-WHATEVER</b> " and <b>IOXY</b> to the code to position the cursor for that terminal. Modify <b>%ZISS</b> routine to set all the <b>IO</b> variables for your terminal type. The routine contains instructions for a simple modification strategy.

## 8.2.2 NEW PERSON File for Standalone VA FileMan

In general, the files you need to run VA FileMan without Kernel are installed by the **DINIT** routines. However, one file that is referenced by VA FileMan is *not* included (i.e., NEW PERSON [#200] file). Kernel supplies the NEW PERSON (#200) file.

This section describes the fields in the NEW PERSON (#200) file directly accessed by VA FileMan; this is a small subset of the fields in Kernel's NEW PERSON (#200) file. The fields described contain characteristics of the VA FileMan users. If you set up a file in the proper global location with these fields, you enhance standalone VA FileMan's functionality. However, the NEW PERSON (#200) file is *not* required to run standalone VA FileMan.

The NEW PERSON (#200) file needs to be established with a global root of **^VA(200,**. Use the information in [Table 87](#) when creating the file using the **Modify File Attributes** [DIMODIFY] option.



**REF:** For a description of USER() function, see the "[USER](#)" section.

The fields directly accessed are shown in [Table 87](#):

**Table 87: System Management—NEW PERSON (#200) File Fields that Enhance Standalone VA FileMan**

Field Name	Field #	Node;Piece	Description
NAME	.01	0;1	Identifies the user. It is pointed to and displayed in several places. A <b>FREE TEXT</b> field holding from <b>3 to 30 characters</b> .
INITIAL	1	0;2	Used by the <b>USER()</b> function. A <b>FREE TEXT</b> field holding from <b>2 to 5 characters</b> .
TITLE	8	0;9	Used by the <b>USER()</b> function. A <b>POINTER TO A FILE</b> field that points to the TITLE (#3.1) file located at <b>^DIC(3.1,</b> . You need to define this file to use <b>USER("T")</b> .
NICK NAME	13	.1;4	Used by the <b>USER()</b> function. A <b>FREE TEXT</b> field holding from <b>1 to 10 characters</b> .

Field Name	Field #	Node;Piece	Description
FILE RANGE	31.1	1;1	Used to assign numbers to newly created files. A <b>FREE TEXT</b> field with the format <i>nnnnn-nnnnn</i> .
TEXT TERMINATOR	31.2	1;4	Holds the default text terminator used by the Line Editor. A <b>FREE TEXT</b> field holding from <b>1 to 5 characters</b> .
PREFERRED EDITOR	31.3	1;5	Holds the user's Preferred Editor. A <b>POINTER TO A FILE</b> field that points to the ALTERNATE EDITOR (#1.2) file.
TYPE AHEAD	200.09	200;9	Used to determine if "type ahead" is allowed. A <b>SET OF CODES</b> field: <ul style="list-style-type: none"> <li>• "Y" = Allowed</li> <li>• "N" = Not allowed. Default is "N"</li> </ul>

In addition to these fields accessed directly by VA FileMan, Kernel uses the NEW PERSON (#200) file to set up VA FileMan key variables. You can define additional NEW PERSON (#200) file fields to use to define these local variables as shown in [Table 88](#):

**Table 88: System Management—NEW PERSON (#200) File Fields to Define Key Variables in VA FileMan**

Field Name	Field #	Node;Piece	Description
Internal Entry Number	—	—	Used to set <b>DUZ</b> for the user. There is no defined <b>.001</b> field on the NEW PERSON (#200) file.
FILE MANAGER ACCESS CODE	3	0;4	Used to set <b>DUZ(0)</b> for user. A <b>FREE TEXT</b> field from <b>1 to 15 characters</b> .
LANGUAGE	200.07	200;7	Used to set <b>DUZ("LANG")</b> . A <b>POINTER TO A FILE</b> field that points to the LANGUAGE (#.85) file identifying the user's language.
TIMED READ (# OF SECONDS)	200.1	200;10	Used to set <b>DTIME</b> for the user. A <b>NUMERIC</b> field with a value of <b>1 to 99999</b> .

When these additional fields are defined, you can use them in a signon routine to set these key variables.

Of course, you can choose to place additional information about the users in the NEW PERSON (#200) file. If you add other fields to the NEW PERSON (#200) file, use field numbers greater than 10,000 and use subfile numbers with at least 5 digits following the decimal place. Also, place the fields in global nodes subscripted with numbers greater than 10,000. If you numberspace your data elements in this way, you can avoid conflicts, if you later install Kernel's NEW PERSON (#200) file.

## 8.3 ^%ZOSF Nodes

### 8.3.1 Manually Setting ^%ZOSF Nodes

VA FileMan's screen-oriented utilities execute M operating system-specific code contained in ^%ZOSF nodes. Running the **DINZMGR** routine in the Manager Account, as explained in the *VA FileMan Installation Guide*, sets the necessary nodes. However, **DINZMGR** can only set the nodes for some operating systems. If you want to use screen-oriented features but your operating system is *not* supported by **DINZMGR**, you *must* set the nodes yourself.

The ^%ZOSF nodes and their meanings are listed in [Table 89](#). Remember that the values stored in these nodes *must* be M code that, when executed, perform the indicated function. For example, **X ^%ZOSF("EOFF")** would need to turn off the echo to the current device.

**Table 89: System Management—Description of the ^%ZOSF Nodes**

^%ZOSF Node	Description
<b>EOFF</b>	Turns off echo to the <b>\$I</b> device.
<b>EON</b>	Turns on echo to the <b>\$I</b> device.
<b>NO-TYPE-AHEAD</b>	Turns off type-ahead for the <b>\$I</b> device.
<b>RM</b>	Sets the <b>\$I</b> width to <b>X</b> characters.
<b>TRMOFF</b>	Resets terminators to normal.
<b>TRMON</b>	Turns on all control characters as terminators.
<b>TRMRD</b>	Returns in <b>Y</b> what terminated the last read.
<b>TYPE-AHEAD</b>	Allows type-ahead for the <b>\$I</b> device.

^%ZOSF Node	Description
XY	Sets \$X=DX and \$Y=DY.

## 8.4 Alternate Editors

### 8.4.1 Setting Up Alternate Editors

VA FileMan supplies two editors:

- Line Editor
- Screen Editors

These editors are used for entering and changing text in **WORD-PROCESSING**-type fields. Other editors can be made available by using the ALTERNATE EDITOR (#1.2) file. Entries in this file specify the M code used to call the Alternate Editor and to restore the VA FileMan environment after work in the editor is complete.

The user can choose an editor in two ways:

1. The PREFERRED EDITOR field in the NEW PERSON (#200) file is a pointer to the ALTERNATE EDITOR (#1.2) file. Any entry in the ALTERNATE EDITOR (#1.2) file can be selected as a **Preferred Editor**. Then, whenever a **WORD-PROCESSING**-type field is presented for editing, the user is automatically switched into this **Preferred Editor**.
2. From the **Line Editor**, the user can switch to any of the editors in the ALTERNATE EDITOR (#1.2) file temporarily by using the **Editor Change** option on the UTILITIES menu. After exiting the chosen editor, the user is returned to the `.



**REF:** For additional information about selecting editors, see the “Choice of Word-processing Editors” section in the *VA FileMan User Manual*.

### 8.4.1.1 ALTERNATE EDITOR File Entries

The requirements of the ALTERNATE EDITOR (#1.2) file's fields are explained in [Figure 95](#) of the creation of an entry in that file:

**Figure 95: System Management—Example of Creating an ALTERNATE EDITOR File Entry**

```
SELECT OPTION: ENTER <ENTER> OR EDIT FILE ENTRIES
INPUT TO WHAT FILE: ALTERNATE EDITOR <ENTER> (2 ENTRIES)
EDIT WHICH FIELD: ALL// <ENTER>
SELECT ALTERNATE EDITOR: VMSEDT
ARE YOU ADDING VMSEDT AS A NEW ALTERNATE EDITOR (THE 3RD)? NO// YES <ENTER>
(YES)
```

An entry needs to be added to the ALTERNATE EDITOR (#1.2) file. The NAME of every entry *must* begin with a unique character. This name is used to select the editor.

The required ACTIVATION CODE FROM DIWE field contains M code that invokes the Alternate Editor. In this case, a routine supplied by Kernel that invokes the VMS editor is called:

```
ACTIVATION CODE FROM DIWE: G ^XTEDTVXD
```

Usually, this code needs to:

1. Extract the word-processing data from the **WORD-PROCESSING**-type field.
2. Present it to the Alternate Editor.
3. Return the edited text to the **WORD-PROCESSING**-type field.

DBS calls should be used to retrieve and file the text.

For example, if the text to be edited were in Field #56 in File #1234 use the following Data Retriever call to extract the text for entry 789:

```
S MYFDA=^TMP(MYDATA, $J) D GETS^DIQ(1234, 789, , 56, , MYFDA)
```

The text returns in nodes descendent from **^TMP("MYDATA", \$J, 1234, "789", 56)**.

After editing is complete, you can file the data from the same global array by using the following commands:

```
S MYFDA=^TMP(MYDATA) D FILE^DIE(, MYFDA)
```



**REF:** For more details of the use of these calls, see the descriptions of the Data Retriever, Filer, and Word-processing Filer in the "Database Server (DBS) API" section in the *VA FileMan Developer's Guide*.

This task of returning the data to the **WORD-PROCESSING**-type field can also be accomplished by code in the [RETURN TO CALLING EDITOR field](#).

If the editor uses any local M variables beginning with **D**, they should be **NEW**ed to avoid problems upon return to VA FileMan. Also, avoid calls to VA FileMan utilities in the fields that accept M code so that variables upon which VA FileMan depends (e.g., **DA** and **DO**) are *not* changed.

The OK TO RUN TEST field contains code that checks to determine whether it is OK to run the editor:

```
OK TO RUN TEST: I ^%ZOSF ( [OS ] ) [ [VAX ] & ( DUZ ( 0 ) [ [ @ ] ] )
```

This field is optional; if nothing is entered, there are no restrictions on the use of the editor. The code should set **\$T** to **true** if it is OK to proceed or to **false** if it is *not*. If the code returns **\$T=false**, the user automatically returns to the Line Editor. In this example, the environment is being checked to ensure that the called editor is available, and it is verified that the user has the required VA FileMan Access Code.

M code entered into the optional RETURN TO CALLING EDITOR field can be used to restore the environment needed by the Line Editor:

```
RETURN TO CALLING EDITOR: <ENTER>
```

Since the user can switch editors at will, it is safest to restore the text to its previous location in the global and to maintain the local symbol table, as mentioned above. If these tasks are accomplished by code in the ACTIVATION CODE FROM DIWE field or by the called routine, this field is unnecessary.



**NOTE:** The edited text *must* be restored; VA FileMan does *not* move text back to the proper global location.

**Figure 96: System Management—Example Where the User is Prompted to Choose an Alternate Editor**

```
DESCRIPTION:  
 1>THE VMS EDITOR.  
 2> <ENTER>  
EDIT OPTION: <ENTER>  
SELECT ALTERNATE EDITOR:
```

The definition of the Alternate Editor is complete. An editor’s developer should provide information needed to set up an ALTERNATE EDITOR (#1.2) file entry. The local site manager has flexibility in setting up the editor. In particular, the OK TO RUN TEST field can be used to enforce local policies regarding access to editors.

## 8.5 COMPILED ROUTINE File

### 8.5.1 COMPILED ROUTINE File Cleanup: ENRLS^DIOZ()

The COMPILED ROUTINE (#.83) file stores a list of numbers used to assign names to compiled sort routines. When a compiled **SORT** template is used in a print, the next available routine number is selected from the COMPILED ROUTINE (#.83) file. The routine name is then set to **^DISZ**, concatenated with up to three **Zeroes** and the number from the COMPILED ROUTINE (#.83) file (e.g., if the number is four, the routine name becomes **^DISZ0004**). After the print finishes, the routine number is released, and the routine is deleted. However, if the system goes down, the number may *not* be released. This procedure allows the site manager to release one or all numbers on the COMPILED ROUTINE (#.83) file for reuse and to delete the associated routines from the system.



**NOTE:** This procedure should *never* be included in application code. It should only be run from the M **Programmer** prompt (aka command prompt). The routine should be run only when the system is inactive to avoid inadvertently deleting a routine that is in use.

## 8.6 Compare Data and Data Dictionaries across Environments

### 8.6.1 Compare Data Dictionaries

The **Transfer Entries** [DITRANSFER] option, **Namespace Compare** [???] option includes the ability to compare the data dictionaries across namespaces/UCIs located on the same servers to help with version control. The routine allows you to get a list of namespaces that can be accessed and the option to pick the file's data dictionary that you wish to compare.

**Figure 97: System Management—Example Where the User Selects to Compare Data Dictionaries**

```
VA FILEMAN VERSION 22.2

ENTER OR EDIT FILE ENTRIES
PRINT FILE ENTRIES
SEARCH FILE ENTRIES
MODIFY FILE ATTRIBUTES
INQUIRE TO FILE ENTRIES
UTILITY FUNCTIONS ...
DATA DICTIONARY UTILITIES ...
TRANSFER ENTRIES
OTHER OPTIONS ...

SELECT VA FILEMAN <TEST ACCOUNT> OPTION: TRANSFER ENTRIES

SELECT TRANSFER OPTION: 3 <ENTER> NAMESPACE COMPARE
UCI: FMVAMC,FMVAMC

START WITH WHAT FILE: OPTION// 200 <ENTER> NEW PERSON (52 ENTRIES)
GO TO WHAT FILE: NEW PERSON// <ENTER> (52 ENTRIES)
COMPARE TO WHAT UCI: EHR// ?
CHOOSE FROM:
%CACHELIB
%SYS
DOCBOOK
EHR
FM220
FM222E
FMOLD
SAMPLES
USER

COMPARE TO WHAT UCI: EHR// <ENTER>

SELECT ONE OF THE FOLLOWING:

1          DATA DICTIONARY ONLY
2          FILE ENTRIES ONLY
3          DATA DICTIONARY AND FILE ENTRIES

ENTER RESPONSE: 3// 1 <ENTER> DATA DICTIONARY ONLY
DISPLAY COMPARISON ON
DEVICE: HOME// 0;P-OTHER;80;99999
```

MAR 24, 2016 PLA.ISC-WASH.DOMAIN.EXT

UCI: FMVAMC,FMVAMC

UCI: EHR

-----  
DATA DICTIONARY #200 (NEW PERSON)

FIELD: EMAIL ADDRESS (#.151)

DESCRIPTION...

FIELD: PREFERRED LANGUAGE

FIELD: LANGUAGE SKILLS

FIELD: COP SUBSCRIPTION

INDEX: ACOPSID^REGULAR WHOLE FILE INDEX FOR THE ERX SUBSCRIBER ID

INDEX: COP^REGULAR WHOLE FILE CROSS REFERENCE OF THE SUBSCRIPTION SUBFILE

INDEX: COPNPI^REGULAR INDEX ON NPI FOR ERX

INDEX: ADEG^UPDATE THE DEGREE FIELD IN THE NAME COMPONENTS FILE.

INDEX: ANAME^UPDATE THE CORRESPONDING ENTRY IN THE NAME COMPONENTS FILE.

## 8.6.2 Compare File Entries

The **Transfer Entries** [DITRANSFER] option, **Namespace Compare** [???] option includes the ability to compare the entries across namespaces located on the same servers to help with version control. The option can display the namespace/UCIs that are available for comparing file entries. Also, you can select option #3, **Data Dictionary and File Entries** option, that compares both the data dictionary and the entries of a file across namespace/UCIs.

**Figure 98: System Management—Example Namespace Compare File Entries**

```
1          DATA DICTIONARY ONLY
2          FILE ENTRIES ONLY
3          DATA DICTIONARY AND FILE ENTRIES

ENTER RESPONSE: 3// 2 <ENTER> FILE ENTRIES ONLY
DISPLAY COMPARISON ON
DEVICE: HOME// 0;P-OTHER;80;99999

MAR 26 2016  PLA.ISC-WASH.DOMAIN.EXT
UCI: FMVAMC,FMVAMC                UCI: EHR
-----

DATE ENTERED: JAN 14,2015          DATE ENTERED: SEP 24,2003
CREATOR: FMUSER,ONE                CREATOR: FMUSER,ONE
TIMESTAMP: 62347,45796             TIMESTAMP: 63263,24079
                                NEW PERSON: CENTRAL,PAID
DATE ENTERED: JAN 14,2015          DATE ENTERED: OCT 23,2003
CREATOR: FMUSER,ONE                CREATOR: FMUSER,ONE
TIMESTAMP: 62347,45796             TIMESTAMP: 63263,24079
                                NEW PERSON: EDILOCKBOX,AUTOMATIC
DATE ENTERED: JAN 14,2015          DATE ENTERED: OCT 23,2003
CREATOR: FMUSER,ONE                CREATOR: FMUSER,ONE
TIMESTAMP: 62347,45796             TIMESTAMP: 63263,24079

SELECT ONE OF THE FOLLOWING:
```

### Format

```
ENRLS^DIOZ([ROUTINE_IEN])
```

### Input Parameters

**routine\_ien:** (Optional) Internal Entry Number of the compiled routine number to be released. If passed as **NULL** or **Zero** or if the parameter string is empty, all of the numbers in the file are released.

## Output

There is no output from this routine. The routine simply sets a flag on one or more entries in the COMPILED ROUTINE (#.83) file, indicating that the entry number is available for reuse.

### 8.6.2.1 Examples

#### 8.6.2.1.1 Example 1

To release routine number three for reuse and to delete routine **^DISZ0003**, do the following:

```
>D ENRLS^DIOZ(3)
```

#### 8.6.2.1.2 Example 2

To release *all* routine numbers for reuse and to delete *all* associated **^DISZnnnn** routines for each routine number "*nnnn*," do the following:

```
>D ENRLS^DIOZ()
```

## 9 List File Attributes

The structures of VA FileMan files are stored in the data dictionary (DD). There, you can find the specifications of every field in every file. Frequently, you need to know the information in the DD (usually field names and descriptions) to successfully access and use the data in VA FileMan's files.

The Data Dictionary Utilities submenu contains the following utilities that show information about files:

- [List File Attributes Option](#)
- [Map Pointer Relations Option](#)
- [Check/Fix DD Structure Option](#)
- [Find Pointers Into a File](#)

### 9.1 List File Attributes Option

To get a listing of the fields in a file (and other file attributes), use the **List File Attributes** [DILIST] option. This listing displays the structure of the file and the characteristics of the fields in the file; it does *not* show entries, records, or any data contained in the file. This information can be very useful when deciding what fields to include in a report, or what fields to edit.

You have your choice of the following formats for the listing:

- [Brief](#)
- [Condensed](#)
- [Standard](#) (or [Modified Standard](#))
- [Custom-Tailored](#)
- [Templates Only](#)
- [Global Map](#)
- [Indexes and Cross-References Only](#)
- [Keys Only](#)

First, choose the file to display information about; you can use either its file number or name.

When you select the file, you have the option of requesting a range of files. If you select a range of files, the file number of the “go to” file *must* be higher than the file number of the “start with” file.

If you are prompted “Select SUB-FILE:”, this indicates that the file you are working with has Subfiles. If you want information only about a Subfile, specify the Subfile at this prompt. If you do *not* choose a Subfile, your listing usually includes information about all fields in the file, including those in all Subfiles.

Next, choose a format for the listing:

**Figure 99: List File Attributes—File Attribute Listing Format Choices**

```
SELECT LISTING FORMAT: STANDARD// ?
ANSWER WITH LISTING FORMAT NUMBER, OR NAME
CHOOSE FROM:
1     STANDARD
2     BRIEF
3     CUSTOM-TAILORED
4     MODIFIED STANDARD
5     TEMPLATES ONLY
6     GLOBAL MAP
7     CONDENSED
8     INDEXES AND CROSS-REFERENCES ONLY
9     KEYS ONLY
```

### 9.1.1 Brief Data Dictionary

When you choose Brief as the data dictionary format, a brief listing is produced; the Brief format is more readable but less complete than the default of a Standard listing. Next, you are asked for a destination for the listing’s output at the “DEVICE:” prompt. You can specify any valid printer or press the **Enter** key to send output to your screen as illustrated in [Figure 100](#):

**Figure 100: List File Attributes—Choosing to Display the Brief Listing**

```
SELECT LISTING FORMAT: STANDARD// BRIEF
ALPHABETICALLY BY LABEL? NO// <ENTER>

DEVICE: <ENTER>
```

[Figure 101](#) is a sample of a Brief data dictionary listing of an elementary file of patients:

**Figure 101: List File Attributes—Example of a Brief Data Dictionary Listing**

```

BRIEF DATA DICTIONARY #16026 -- PATIENT FILE          05/31/91  PAGE 1
SITE: KDEMO V7   UCI: VAH,KXX
-----
NAME                16026,.01   FREE TEXT
                    ANSWER MUST BE 3-30 CHARACTERS IN LENGTH.
SEX                 16026,1     SET
                    [M] FOR MALE;
                    [F] FOR FEMALE;
DATE OF BIRTH      16026,2     DATE
                    TYPE A DATE BETWEEN 1/1/1860 AND 1963
RELIGION            16026,3     POINTER TO RELIGION FILE (#13)
DIAGNOSIS           16026,4     16026.04
  MULTIPLE
DIAGNOSIS           16026.04,.01  FREE TEXT
                    ANSWER MUST BE 3-30 CHARACTERS IN LENGTH.
AGE AT ONSET       16026.04,1     NUMBER
                    TYPE A NUMBER BETWEEN 0 AND 100, 0 DECIMAL DIGITS
HISTORY            16026.04,2     16026.42  WORD-PROCESSING
PROVIDER           16026,5     VARIABLE POINTER
                    FILE  ORDER  PREFIX  LAYGO  MESSAGE
                    6     1     S       N     STAFF PROVIDER
                    16    2     0       Y     OTHER PROVIDER
SSN                16026,6     FREE TEXT
  SOCIAL SECURITY NUMBER  ENTER 9 NUMBERS WITHOUT DASHES.

```

The information in the data dictionary reports originated in the definition of the file and its fields.



**REF:** For a detailed explanation of the source of the information displayed by the **List File Attributes** [DILIST] option, see the "[Creating Files and Fields](#)" section.

This data dictionary listing tells you that for each patient, the following information may be available:

- A NAME that is from **3** to **30 characters** long.
- A recorded SEX of either **m** (MALE) or **f** (FEMALE).
- A DATE OF BIRTH.
- A RELIGION (for a list of all valid religions, you would have to consult a RELIGION file).
- One or more diagnoses and for each DIAGNOSIS; DIAGNOSIS is a Multiple-valued field that has the following information:
  - An AGE AT ONSET
  - A HISTORY
- A PROVIDER (e.g., a primary care physician). For a list of valid PROVIDERs, you would consult the NEW PERSON (#200) file. If the PROVIDER's name does *not* appear, it can then be entered in the NEW PERSON (#200) file, since LAYGO (Learn-As-You-Go) has been allowed.
- A SOCIAL SECURITY NUMBER (SSN).

## 9.1.2 Condensed Data Dictionary

Another format for listing a file's attributes is the Condensed format, and [Figure 102](#) illustrates a Condensed data dictionary listing:

**Figure 102: List File Attributes—Example of a Condensed Data Dictionary Listing**

```

CONDENSED DATA DICTIONARY---PATIENT FILE      (#16026)UCI: XXX,XXX
STORED IN: ^DIZ(16026,                          05/31/91    PAGE 1
-----
                                FILE SECURITY
                                DD SECURITY      : #    DELETE SECURITY: #
                                READ SECURITY    : #    LAYGO SECURITY  : #
                                WRITE SECURITY   : #
CROSS REFERENCED BY:
  NAME(B)
                                FILE STRUCTURE
FIELD      FIELD
NUMBER     NAME
.01        NAME (RF), [0;1]
1          SEX (RS), [0;2]
2          DATE OF BIRTH (RD), [0;3]
3          RELIGION (P13[]), [0;4]
4          DIAGNOSIS (MULTIPLE-16026.04), [1;0]
           .01 DIAGNOSIS (MF), [0;1]
           1   AGE AT ONSET (NJ3,0), [0;2]
           2   HISTORY (MULTIPLE-16026.42), [1;0]
           .01 HISTORY (W), [0;1]
5          PROVIDER (V), [2;1]
6          SSN (RFA), [2;2]

```

The codes in parentheses following the field names in the Condensed data dictionary contain information regarding the specifications of the field.

[Table 90](#) is a complete list of those codes and their meanings:

**Table 90: List File Attributes—Condensed Data Dictionary Codes**

<b>Code</b>	<b>Description</b>
<b>a</b>	The field has been marked for <b>a</b> uditing all the time.
<b>e</b>	The auditing is only on <b>e</b> dit or delete.
<b>A</b>	For Multiples, new subentries can be <b>a</b> dded without being asked.
<b>BC</b>	The data is <b>B</b> oolean <b>c</b> omputed (true or false) and <b>C</b> the data is <b>c</b> omputed.
<b>Cm</b>	The data is <b>M</b> ultiline <b>c</b> omputed.
<b>D</b>	The data is <b>d</b> ate-valued.
<b>DC</b>	The data is <b>d</b> ate-valued, <b>c</b> omputed.
<b>F</b>	The data is <b>F</b> REE TEXT.
<b>I</b>	The data is uneditable.
<b>Jn</b>	To specify a print length of " <b>n</b> " characters.
<b>Jn,d</b>	To specify printing " <b>n</b> " characters with " <b>d</b> " <b>d</b> ecimals.
<b>K</b>	The data is M code.
<b>M</b>	For Multiples, the user is asked for another subentry.
<b>N</b>	The data is <b>N</b> UMERIC-valued.
<b>O</b>	The field has an <b>O</b> UTPUT transform.
<b>Pn</b>	The data is a <b>P</b> OINTER TO A FILE reference to file " <b>n</b> ".
<b>Pn'</b>	LAYGO to the <b>p</b> ointed-to file is <i>not</i> allowed.
<b>R</b>	Entry of data is <b>r</b> equired.
<b>S</b>	The data is from a discreet <b>S</b> ET OF CODES.
<b>V</b>	The data is a <b>V</b> ARIABLE-POINTER.

Code	Description
<b>W</b>	The data is <b>WORD-PROCESSING</b> .
<b>WL</b>	The <b>WORD-PROCESSING</b> data is normally printed in <b>line mode</b> (i.e., <i>without</i> word wrap).
<b>X</b>	Editing is <i>not</i> allowed under the <b>Modify File Attributes</b> [DIMODIFY] option, because the <b>INPUT</b> transform has been modified under the <b>Utility Functions</b> [DIUTILITY] menu.
*	There is a screen associated with a DATA TYPE field value of any of the following: <ul style="list-style-type: none"> <li>• <b>POINTER TO A FILE</b></li> <li>• <b>VARIABLE-POINTER</b></li> <li>• <b>SET OF CODES</b></li> </ul>

For example, the SSN field is required (**R**), is **FREE TEXT (F)**, and is audited (**a**).

### 9.1.3 Standard and Modified Standard Data Dictionaries

The most complete information about a file is obtained by using the Standard data dictionary format, which is the default for the **List File Attributes** [DILIST] option. In addition to detailed information about every field in the file, the Standard data dictionary format gives the file access, identifiers, cross-references, other files pointing to the file, files pointed to by the file, and any templates (including forms and blocks) associated with the file.

#### 9.1.3.1 Standard Format

[Figure 103](#) is a sample data dictionary in Standard format:

**Figure 103: List File Attributes—Example of a Standard Data Dictionary Listing**

```

STANDARD DATA DICTIONARY #16026 -- PATIENT FILE                05/31/91  PAGE 1
STORED IN ^DIZ(16026, (1 ENTRY)  SITE: KDEM0 V7  UCI: VAH,KXX

DATA          NAME          GLOBAL          DATA
ELEMENT       TITLE         LOCATION       TYPE
-----
A SAMPLE FILE CONTAINING SOME OF THE FIELDS FOUND IN A FILE OF PATIENT
INFORMATION IN A HOSPITAL DATABASE.

          DD ACCESS: #
          RD ACCESS: #
          WR ACCESS: #
          DEL ACCESS: #
          LAYGO ACCESS: #
          AUDIT ACCESS: #

CROSS
REFERENCED BY: NAME(B)

          CREATED ON: MAR 22, 1991

16026,.01    NAME          0;1 FREE TEXT (REQUIRED)

          INPUT TRANSFORM: K:$L(X)>30!($L(X)<3)! (X?1P.E) X
          LAST EDITED:    MAR 29, 1991
          HELP-PROMPT:    ANSWER MUST BE 3-30 CHARACTERS IN LENGTH.
          GROUP:          DEMOG
          CROSS-REFERENCE: 16026^B
                          1)= S ^DIZ(16026, B, $E(X,1,30),DA)=
                          2)= K ^DIZ(16026, B, $E(X,1,30),DA)
                          AUTOMATICALLY CREATED REGULAR X-REF USED TO
                          LOOK-UP AND SORT ENTRIES BASED ON THE VALUE IN
                          THE .01 (NAME) FIELD.

16026,1     SEX          0;2 SET (REQUIRED)
  
```

```

                                [M] FOR MALE;
                                [F] FOR FEMALE;
LAST EDITED: MAR 22, 1991
GROUP: DEMOG

16026,2 DATE OF BIRTH 0;3 DATE (REQUIRED)

INPUT TRANSFORM: S %DT=[E] D ^%DT S X=Y K:2630000<X!(1600101>X)
X
LAST EDITED: MAR 22, 1991
HELP-PROMPT: TYPE A DATE BETWEEN 1/1/1860 AND 1963
GROUP: DEMOG
16026,3 RELIGION 0;4 POINTER TO RELIGION FILE (#13)

LAST EDITED: MAR 22, 1991

16026,4 DIAGNOSIS 1;0 MULTIPLE #16026.04
(ADD NEW ENTRY WITHOUT ASKING)
STANDARD DATA DICTIONARY #16026 -- PATIENT FILE 05/31/91 PAGE 2
STORED IN ^DIZ(16026, (1 ENTRY) SITE: KDEMO V7 UCI: VAH,KXX

DATA ELEMENT NAME GLOBAL DATA
TITLE LOCATION TYPE
-----
16026.04,.01 DIAGNOSIS 0;1 FREE TEXT (MULTIPLY ASKED)

INPUT TRANSFORM: K:$L(X)>30!($L(X)<3) X
LAST EDITED: MAR 22, 1991
HELP-PROMPT: ANSWER MUST BE 3-30 CHARACTERS IN LENGTH.
CROSS-REFERENCE: 16026.04^B
1)= S ^DIZ(16026,DA(1),1,[B],$(X,1,30),DA)=
2)= K ^DIZ(16026,DA(1),1,[B],$(X,1,30),DA)

16026.04,1 AGE AT ONSET 0;2 NUMBER

INPUT TRANSFORM: K:+X=X!(X>100)!(X<0)!(X?.E1[.1N.N) X
LAST EDITED: APR 29, 1991
HELP-PROMPT: TYPE A NUMBER BETWEEN 0 AND 100, 0 DECIMAL
DIGITS

16026.04,2 HISTORY 1;0 WORD-PROCESSING #16026.42

16026,5 PROVIDER 2;1 VARIABLE POINTER

FILE ORDER PREFIX LAYGO MESSAGE
6 1 S N STAFF PROVIDER
16 2 0 Y OTHER PROVIDER

LAST EDITED: MAR 22, 1991

16026,6 SSN 2;2 FREE TEXT (REQUIRED) (AUDITED)

SOCIAL SECURITY NUMBER
INPUT TRANSFORM: K:$L(X)>9!($L(X)<9)!(X?9N) X
LAST EDITED: MAR 22, 1991
HELP-PROMPT: ENTER 9 NUMBERS WITHOUT DASHES.
DESCRIPTION: AN ENTRY IS REQUIRED. IF YOU DO NOT KNOW THIS
PATIENT'S SOCIAL SECURITY NUMBER, ENTER
[000000000] TO INDICATE THE NUMBER IS UNKNOWN.

```

```

          GROUP:          DEMOG
          FILES POINTED TO          FIELDS
PROVIDER (#6)          PROVIDER (#5)
PERSON (#16)          PROVIDER (#5)
RELIGION (#13)          RELIGION (#3)
INPUT TEMPLATE(S):
PRINT TEMPLATE(S):
CAPTIONED          USER #0
ZZDIAGPRINT          MAR 29, 1991@12:18  USER #140
  USED TO PRINT INFORMATION FROM THE DIAGNOSIS MULTIPLE.
SORT TEMPLATE(S):
FORM(S)/BLOCKS(S):

```

### 9.1.3.2 Modified Standard Format

Another data dictionary format is the Modified Standard format, which allows you to suppress printing the M code and to restrict the listing to specified groups of fields.

For example, the dialog in [Figure 104](#) eliminates the M code from the Standard listing and only prints those fields in the DEMOG group:

- NAME
- SEX
- DATE OF BIRTH
- SSN

**Figure 104: List File Attributes—Choosing the Modified Standard Data Dictionary Listing**

```

SELECT LISTING FORMAT: STANDARD// MOD <ENTER> IFIED STANDARD
WANT THE LISTING TO INCLUDE MUMPS CODE? N// <ENTER>
WANT TO RESTRICT LISTING TO CERTAIN GROUPS OF FIELDS? NO// Y
INCLUDE GROUP: DEMOG
AND INCLUDE GROUP: <ENTER>

```



**NOTE:** If you answer the question concerning M code **YES** and do *not* specify any groups, the output from the Modified Standard format is the same as that of the Standard format.

## 9.1.4 Custom-Tailored Data Dictionary

The Custom-Tailored format allows you to select attributes of the fields for your report. You decide what information is displayed, and you determine the printed format of the output.



**REF:** For a detailed description of the techniques used to control the format of output, see the “Print: How to Print Reports from Files” section in the *VA FileMan User Manual*.

For this simple example in [Figure 105](#) you accept the default settings:

**Figure 105: List File Attributes—Choosing the Custom-Tailored Data Dictionary Listing**

```
SELECT DATA DICTIONARY UTILITY OPTION: LIST FILE ATTRIBUTES
START WITH WHAT FILE: PATIENT // <ENTER>
GO TO WHAT FILE: PATIENT // <ENTER>
SELECT SUB-FILE: <ENTER>
SELECT LISTING FORMAT: STANDARD// CUSTOM-TAILORED
SORT BY: LABEL// ?
ANSWER WITH ATTRIBUTE NUMBER, OR LABEL
DO YOU WANT THE ENTIRE ATTRIBUTE LIST? Y <ENTER> (YES)
```

The “SORT BY:” prompt allows you to specify the order in which the data dictionary information is displayed.

In [Figure 106](#), the user is asking for the entire list of possible attributes about a field that can be stored in the data dictionary. Typically, no field would have a value for every one of these attributes:

**Figure 106: List File Attributes—Choosing from a List of Field Attributes**

```

CHOOSE FROM:
.001          NUMBER
.01           LABEL
.1            TITLE
.12          VARIABLE POINTER  (MULTIPLE)
.2           SPECIFIER
.23          LENGTH
.24          DECIMAL DEFAULT
.25          TYPE
.26          COMPUTE ALGORITHM
.27          SUB-FIELDS
.28          MULTIPLE-VALUED
.29          DEPTH OF SUB-FIELD
.3           POINTER
.4           GLOBAL SUBSCRIPT LOCATION
.5           INPUT TRANSFORM
1           CROSS-REFERENCE  (MULTIPLE)
1.1         AUDIT
1.2         AUDIT CONDITION
2           OUTPUT TRANSFORM
3           [H]ELP[ ]-PROMPT
4           XECUTABLE [H]ELP[ ]
8           READ ACCESS (OPTIONAL)
8.5        DELETE ACCESS (OPTIONAL)
9           WRITE ACCESS (OPTIONAL)
9.01       COMPUTED FIELDS USED
10         SOURCE
11         DESTINATION  (MULTIPLE)
12         POINTER SCREEN
12.1       CODE TO SET POINTER SCREEN
12.2       EXPRESSION FOR POINTER SCREEN
20         GROUP  (MULTIPLE)
50         DATE FIELD LAST EDITED
999        TRIGGERED-BY POINTER  (MULTIPLE)
TYPE [ ]-[ ] IN FRONT OF NUMERIC-VALUED FIELD TO SORT FROM HI TO LO
TYPE [ ]+[ ] IN FRONT OF FIELD NAME TO GET SUBTOTALS BY THAT FIELD,
[ ]#[ ] TO PAGE-FEED ON EACH FIELD VALUE, [ ]![ ] TO GET RANKING NUMBER,
[ ]@[ ] TO SUPPRESS SUB-HEADER, [ ]][ ] TO FORCE SAVING SORT TEMPLATE TYPE [TEMPLATE
NAME] IN BRACKETS TO SORT BY PREVIOUS SEARCH RESULTS

SORT BY: LABEL// <ENTER>
START WITH LABEL: FIRST// <ENTER>
FIRST PRINT ATTRIBUTE: ?
ANSWER WITH ATTRIBUTE NUMBER, OR LABEL
DO YOU WANT THE ENTIRE 36-ENTRY ATTRIBUTE LIST? N <ENTER> (NO)

```

At this point, you indicate the specific attributes of the fields that you want displayed.

In [Figure 107](#), help regarding the print formatting options is displayed:

**Figure 107: List File Attributes—Help on Print Formatting in the Custom-Tailored Data Dictionary Listing**

TYPE [ & ] IN FRONT OF FIELD NAME TO GET TOTAL FOR THAT FIELD,  
 [ ! ] TO GET COUNT, [ + ] TO GET TOTAL & COUNT, [ # ] TO GET MAX & MIN,  
 [ ] TO FORCE SAVING PRINT TEMPLATE  
 TYPE [ [TEMPLATE NAME] ] IN BRACKETS TO USE AN EXISTING PRINT TEMPLATE  
 YOU CAN FOLLOW FIELD NAME WITH [ ; ] AND FORMAT SPECIFICATION(S)

Now, by using either the label or the corresponding number of the attribute you want, you select the information you want in your customized data dictionary listing:

**Figure 108: List File Attributes—Selecting the Field Attributes to Print**

FIRST PRINT ATTRIBUTE: LABEL  
 THEN PRINT ATTRIBUTE: TYPE  
 THEN PRINT ATTRIBUTE: DATE FIELD LAST EDITED  
 THEN PRINT ATTRIBUTE: <ENTER>  
 HEADING: FIELD SEARCH// CUSTOM-TAILORED OUTPUT  
 DEVICE: <ENTER>

You can save the selected attributes in a **PRINT** template.



**REF:** For details, see the “Print: How to Print Reports from Files” section in the *VA FileMan User Manual*.

The output looks like [Figure 109](#):

**Figure 109: List File Attributes—Example of a Custom-Tailored Data Dictionary Listing**

PATIENT FILE CUSTOM-TAILORED OUTPUT	MAY 31,1991	11:10	PAGE 1
LABEL	TYPE	DATE FIELD	LAST EDITED
-----			
DATE OF BIRTH	DATE/TIME	MAR 22,1991	
DIAGNOSIS	FREE TEXT		
NAME	FREE TEXT	MAR 29,1991	
PROVIDER	VARIABLE-POINTER	MAR 22,1991	
RELIGION	POINTER	MAR 22,1991	
SEX	SET	MAR 22,1991	
SSN	FREE TEXT	MAR 22,1991	



**NOTE:** With the Custom-Tailored format, to get information about fields in a Multiple, you *must* specifically ask for that Multiple by entering its name at the "Select SUB-FILE:" prompt.

### 9.1.5 Templates Only Format

The Templates Only format displays information about the templates (including forms and blocks) associated with a file. The output resembles the last part of the Standard data dictionary output.

### 9.1.6 Global Map

The Global Map format shows the actual structure of the global (file) that contains the data for the file and its templates. This information is of primary interest to developers who can control how data is stored.

[Figure 110](#) is a sample Global Map:

**Figure 110: List File Attributes—Example of a Global Map Data Dictionary Listing**

```
GLOBAL MAP DATA DICTIONARY #16026 -- PATIENT FILE          05/31/91  PAGE 1
STORED IN ^DIZ(16026, (1 ENTRY)  SITE: KDEMO V7  UCI: VAH,KXX
-----
CROSS
REFERENCED BY: NAME(B)

^DIZ(16026,D0,0)= (#.01) NAME [1F] ^ (#1) SEX [2S] ^ (#2) DATE OF BIRTH [3D]
                ==>^ (#3) RELIGION [4P] ^
^DIZ(16026,D0,1,0)=^16026.04A^^ (#4) DIAGNOSIS
^DIZ(16026,D0,1,D1,0)= (#.01) DIAGNOSIS [1F] ^ (#1) AGE AT ONSET [2N] ^
^DIZ(16026,D0,1,D1,1,0)=^16026.42^^ (#2) HISTORY
^DIZ(16026,D0,1,D1,1,D2,0)= (#.01) HISTORY [1W] ^
^DIZ(16026,D0,2)= (#5) PROVIDER [1V] ^ (#6) SSN [2F] ^

INPUT TEMPLATE(S):
^DIE(30)= ZZUPDATE

PRINT TEMPLATE(S):
^DIPT(.01)= CAPTIONED
^DIPT(60)= ZZDIAGPRINT

SORT TEMPLATE(S):
```

An understanding of these data dictionary listings is the key to displaying, changing, and deleting the data in individual file entries.

## 9.1.7 Indexes and Cross-References Only

The Indexes and Cross-References Only format shows the Traditional cross-references and New-Style indexes that are defined on a file.

[Figure 111](#) is a sample Indexes and Cross-References Only data dictionary listing:

**Figure 111: List File Attributes—Example of an Indexes and Cross-References Only Data Dictionary Listing**

```
INDEX AND CROSS-REFERENCE LIST -- FILE #16026          12/24/98    PAGE 1
-----
FILE #16026

TRADITIONAL CROSS-REFERENCES:

B    REGULAR
      FIELD:  NAME (16026,.01)
      DESCRIPTION:  AUTOMATICALLY CREATED REGULAR X-REF USED TO LOOK-UP AND
                    SORT ENTRIES BASED ON THE VALUE IN THE .01 (NAME) FIELD.
                    1)= S ^DIZ(16026,□B□,$E(X,1,30),DA)=□□
                    2)= K ^DIZ(16026,□B□,$E(X,1,30),DA)

NEW-STYLE INDEXES:

KEYA (#6)    RECORD    REGULAR    IR    LOOKUP & SORTING
UNIQUE FOR:  KEY A (#5), FILE #16026
SHORT DESCR: UNIQUENESS INDEX FOR KEY □A□ OF FILE #16026
SET LOGIC:   S ^DIZ(16026,□KEYA□,X(1),X(2),DA)=□□
KILL LOGIC:  K ^DIZ(16026,□KEYA□,X(1),X(2),DA)
WHOLE KILL:  K ^DIZ(16026,□KEYA□)
              X(1):  NAME (16026,.01) (SUBSCR 1)
              X(2):  SSN (16026,6) (SUBSCR 2)

SUBFILE #16026.04

TRADITIONAL CROSS-REFERENCES:

B    REGULAR
      FIELD:  DIAGNOSIS (16026.04,.01)
              1)= S ^DIZ(16026,DA(1),1,□B□,$E(X,1,30),DA)=□□
              2)= K ^DIZ(16026,DA(1),1,□B□,$E(X,1,30),DA)
```

### 9.1.8 Keys Only

The Keys Only format shows the keys that are defined on a file.

[Figure 112](#) is a sample Keys Only data dictionary listing:

**Figure 112: List File Attributes—Example of a Keys Only Data Dictionary Listing**

```
KEY LIST -- FILE #16026                               12/24/98   PAGE 1
-----
FILE #16026
-----
PRIMARY KEY:      A (#5)
UNIQUENESS INDEX: KEYA (#6)
FILE, FIELD: 1) NAME (16026,.01)  2) SSN (16026,6)
```

## 9.2 Map Pointer Relations Option

The **Map Pointer Relations** [DI DDMAP] option on the **Data Dictionary Utilities** [DI DDU] menu creates a graphic representation of the pointer relationships between files. Files are linked by **POINTER TO A FILE** and **VARIABLE-POINTER** field types.



**REF:** These field types are described in the "[Creating Fields](#)" section.

You select an application package, a file, or a group of files to be mapped. If you select a package to map, you are given the opportunity to exclude a file or files from the map.

The initial dialog is displayed in [Figure 113](#):

**Figure 113: List File Attributes—Example of the Dialog Encountered When Using the Map Pointer Relations Option**

```
SELECT DATA DICTIONARY UTILITY OPTION: MAP POINTER RELATIONS

PRINTS A GRAPH OF POINTER RELATIONS IN A DATABASE OF FILEMAN FILES
NAMED IN THE KERNEL PACKAGE FILE (9.4) OR GIVEN SEPARATELY.
WORKS BEST WITH 132 COLUMN OUTPUT!

SELECT PACKAGE NAME: <ENTER>


ENTER FILES TO BE INCLUDED
ADD FILE: PATIENT
ADD FILE: <ENTER>

FILES INCLUDED                16026  PATIENT

ENTER NAME OF FILE GROUP FOR OPTIONAL GRAPH HEADER: PATIENT FILE

DEVICE: HOME// <ENTER>
```

In this instance, only a single file, the PATIENT file, has been selected for mapping. Of course, a more useful and complex map would be produced if an entire package or a large, related group of files were mapped.

 **NOTE:** You *must* have **DD** access to the PACKAGE (#9.4) file and to the files chosen at the “Add FILE:” prompt.

The output consists of three columns. The middle column has the target file or files, each surrounded by a box. This is the file or group of files that you asked to be mapped. To the left, in the first column, are the files and fields that point to the target file. An abbreviated description of the field is shown. To the right, in the last column, are any files that are pointed to by the target file. The output’s heading contains brief descriptions of the codes used.

A possible output for the PATIENT (#2) file is shown in [Figure 114](#):

**Figure 114: List File Attributes—Example of the Output Produced with the Map Pointer Relations Option**

FILE/PACKAGE: PATIENT FILE		DATE: MAY 31, 1991	
FILE (#) POINTER FIELD	POINTER TYPE	(#) FILE POINTER FIELD	FILE POINTED TO
L=LAYGO *=TRUNCATED		N=NORMAL REF. C=XREF. V=VARIABLE POINTER	
S=FILE NOT IN SET M=MULTIPLE			
ADMISSIONS (#16999) CLIENT .....	(N S L)->	16026 PATIENT RELIGION V PROVIDER	-> RELIGION -> PERSON -> PROVIDER

This output shows that the CLIENT field in the ADMISSIONS file points to the PATIENT (#2) file. **LAYGO** additions are allowed, and the ADMISSIONS file is *not* in the set of files being mapped. Further, the RELIGION field in the PATIENT file points to the RELIGION file and the PROVIDER field, a “**v**” (VARIABLE-POINTER), points to the PERSON and PROVIDER files. If the target file points to a file that is *not* in the account, the map shows:

\*\*\* NONEXISTENT FILE \*\*\*



**REF:** For a more elaborate example of a pointer map, see the “Pointer Map” section in the *VA FileMan Technical Manual*.

### 9.3 Check/Fix DD Structure Option

To ensure that the internal structure of your files and subfiles is consistent, use the **Check/Fix DD Structure** [DI DDUCHK] option from the **Data Dictionary Utilities** [DI DDU] menu. You *must* have **READ** access to the files being analyzed. In addition, you need **DD** access for this option to correct erroneous nodes.

This utility looks at a file’s identifiers, cross-references, **POINTER TO A FILE**, **VARIABLE-POINTER**, and **COMPUTED** fields. If there are inconsistencies or conflicts between the information in the data dictionary and the structure of the file’s global nodes, the **Check/Fix DD Structure** [DI DDUCHK] option notes them.

If you want, the **Check/Fix DD Structure** [DI DDUCHK] option corrects inconsistencies found in the data dictionary. The process does *not* change any file structures; it only removes or corrects unnecessary or incorrect **DD** nodes. Data is *not* affected.

The dialog for running this option is simple. You specify the file or files you want to check and indicate whether you want to delete incorrect nodes. Then the progress of the checking is displayed followed by a report of any discrepancies found or any changes made. For example:

**Figure 115: List File Attributes—Example of Dialog and Output Encountered When Using Check/Fix DD Structure Option**

```
SELECT DATA DICTIONARY UTILITY OPTION: CHECK/FIX DD STRUCTURE
CHECK THE DATA DICTIONARY.
START WITH WHAT FILE: 16033
GO TO WHAT FILE: <ENTER>
REMOVE ERRONEOUS NODES? NO// YES
DEVICE: HOME// <ENTER> DECSERVER

CHECKING FILE # 16033
  CHECKING ID NODES FOR Q.
  CHECKING IX NODES.
  CHECKING PT NODES.
  FILE: 16037 FIELD: .01 IS NOT A POINTER.
    ^DD(16033,0,PT,16037,.01) WAS KILLED.
  CHECKING FIELDS....
CHECKING SUBFILE # 16033.04
  CHECKING IX NODES.
  CHECKING FIELDS...
CHECKING SUBFILE # 16033.42
  CHECKING FIELDS.
RETURNING TO SUBFILE 16033.04.
RETURNING TO MAIN FILE.....
CHECKING SUBFILE # 16033.01
  CHECKING IX NODES.
  CHECKING FIELDS.
RETURNING TO MAIN FILE.....
```

In the previous example (Figure 115), the check is being run on a single file. Correction of erroneous nodes has been requested. An incorrect “PT” node was found and deleted.

 **NOTE:** Subfiles are inspected, too.

Application developers might use this tool to clean up their files before export. Site managers may find the reporting function useful for checking a package’s files after installation. Erroneous nodes that are found by this option can be remnants of prior versions of the files; the current install may *not* be to blame.

## 9.4 Find Pointers Into a File Option

The **Find Pointers Into a File** [DDU FIND POINTERS INTO A FILE] option locates the entries in other files that point to your target file. You can specify the scope of the search.

The **Find Pointers Into a File** [DDU FIND POINTERS INTO A FILE] option is located on the **Data Dictionary Utilities** [DI DDU] menu. After you indicate which file you want analyzed, you can limit the extent of the analysis in the following ways:

- Only find pointers to a single entry in the target file.
- Find pointers to all entries in the target file.
- Only find pointers to the target file that point to a non-existent entry in that file (dangling pointers).
- Only find pointers to entries in a **SEARCH** template on the target file. You will see this option only if there are **SEARCH** templates on the target file.

The resulting report is sorted by pointing file. It displays the IEN of the pointed-to entry in the target file, the value of the **.01** field in the pointing file of the entry that contains the pointer, and the Field Name of the pointing field in the pointing file.

In [Figure 116](#), pointers to a single entry in the Patient file are found.

**Figure 116: Data Dictionary Utilities—Example of Dialog and Output Encountered When Using the Find Pointers Into a File Option**

```
SELECT OPTION: DATA DICTIONARY UTILITIES
SELECT DATA DICTIONARY UTILITY OPTION: FIND POINTERS INTO A FILE

THIS UTILITY TRIES TO FIND ALL ENTRIES IN ALL FILES POINTING TO A CERTAIN FILE

SELECT FILE: PATIENT

SELECT ONE OF THE FOLLOWING:

    1      ONE PARTICULAR PATIENT ENTRY
    2      ALL PATIENT ENTRIES
    3      NON-EXISTENT PATIENT ENTRIES
    4      ENTRIES FROM A PATIENT SEARCH TEMPLATE

FIND POINTERS TO: ALL PATIENT ENTRIES// 1 <ENTER> ONE PARTICULAR PATIENT ENTRY
FIND POINTERS TO PATIENT ENTRY: FMPATIENT,TWO <ENTER>          3-3-60    000234567
NO      NON-VETERAN (OTHER)
DEVICE: HOME// <ENTER> TELNET    RIGHT MARGIN: 80// <ENTER>

***PATIENT: FMPATIENT,TWO***
FILE 45 (PTF)
  \1      FMPATIENT,TWO          PATIENT
FILE 59.9 (PBM PATIENT DEMOGRAPHICS)
  \2      JUN 9,2015            PATIENT
FILE 120.8 (PATIENT ALLERGIES)
  \9      FMPATIENT,TWO          PATIENT
FILE 120.85 (ADVERSE REACTION REPORTING)
  \4      JUN 9,2015            PATIENT
FILE 120.86 (ADVERSE REACTION ASSESSMENT)
  \4      FMPATIENT,TWO          NAME
FILE 301.7 (IVM ADDRESS CHANGE LOG)
  \1      JUN 9,2015@08:39:11    PATIENT
  \2      JUN 9,2015@08:39:30    PATIENT
FILE 391.71 (ADT/HL7 PIVOT)
  \2      JUN 9,2015            PATIENT
                                  EVENT POINTER
FILE 405 (PATIENT MOVEMENT)
  \3      JUN 9,2015@08:40:16    PATIENT
  \4      JUN 9,2015@08:40:16    PATIENT
FILE 798.3 (ROR PATIENT EVENTS)
  \4      FMPATIENT,TWO          PATIENT NAME
TYPE <ENTER> TO CONTINUE OR '^' TO EXIT: ^
```

## 9.5 Meta Data Dictionary

By running ^**DDD** during VA FileMan installation, it creates the META DATA DICTIONARY (#.9) file. The Meta Data Dictionary lists all fields in all files in a searchable format.

**Figure 117: List File Attributes—Example Setting Up the Meta Data Dictionary**

```
SELECT OPTION NAME: DIUSER <ENTER> VA FILEMAN

VA FILEMAN VERSION 22.2

ENTER OR EDIT FILE ENTRIES
PRINT FILE ENTRIES
SEARCH FILE ENTRIES
MODIFY FILE ATTRIBUTES
INQUIRE TO FILE ENTRIES
UTILITY FUNCTIONS ...
DATA DICTIONARY UTILITIES ...
TRANSFER ENTRIES
OTHER OPTIONS ...

SELECT DATA DICTIONARY UTILITIES <TEST ACCOUNT> OPTION: ???

'CHECK/FIX DD STRUCTURE'      OPTION NAME: DI DDUCHK
  THIS OPTION LOOKS AT THE INTERNAL STRUCTURE OF FILES AND SUBFILES
  AND DETERMINES IF THERE ARE INCONSISTENCIES OR CONFLICTS BETWEEN THE
  INFORMATION IN THE DATA DICTIONARY AND THE STRUCTURE OF THE FILE'S GLOBAL
  NODES.  THIS OPTION WILL NOTE THEM AND FIX OR DELETE THE INCORRECT NODES.

'FIND POINTERS INTO A FILE'    OPTION NAME: DDU FIND POINTERS INTO A FILE
  THIS UTILITY TRIES TO FIND ALL ENTRIES IN ALL FILES POINTING TO A CERTAIN
  FILE.  AFTER SELECTING THE FILE, THE OPTIONS ARE:

  SELECT ONE OF THE FOLLOWING:

  1      ONE PARTICULAR <FILE NAME> ENTRY
  2      ALL <FILE NAME> ENTRIES
  3      NON-EXISTENT <FILE NAME> ENTRIES
  4      ENTRIES FROM A <FILE NAME> SEARCH TEMPLATE

'LIST FILE ATTRIBUTES'        OPTION NAME: DILIST
  THIS OPTION IS USED TO PRINT DATA DICTIONARY LISTINGS FOR A GIVEN FILE.
  THIS LISTING IS USEFUL FOR PROGRAMMERS, ANALYSTS, AND OTHERS INTERESTED
  IN DATA BASE STRUCTURES.

**> PRESS 'RETURN' TO CONTINUE, '^' TO STOP, OR '?[OPTION TEXT]' FOR MORE
    HELP:

'MAP POINTER RELATIONS'       OPTION NAME: DI DDMAP
  THIS OPTION PRINTS A MAP OF THE POINTER RELATIONS BETWEEN A GROUP OF
  FILES.  THE FILE SELECTION IS FROM THE PACKAGE FILE OR ENTERED
  INDIVIDUALLY.

'UPDATE THE META DATA DICTIONARY'  OPTION NAME: DDU UPDATE META DD
  THIS OPTION WILL UPDATE THE META DICTIONARY FILE.

SHALL I SHOW YOU YOUR SECONDARY MENUS TOO? NO// <ENTER> NO
```



```

2.3226,.02
4 NAME ERROR_MESSAGES_NAME 3.076,.01
5 NAME TITLE_NAME 3.1,.01
PRESS <ENTER> TO SEE MORE, '^' TO EXIT THIS LIST, OR
CHOOSE 1-5: <ENTER>
6 NAME TERMINAL_TYPE_NAME 3.2,.01
7 NAME LINE/PORT_ADDRESS_NAME 3.23,.01
8 NAME COMMUNICATIONS_PROTOCOL_NAME 3.4,.01
9 NAME DEVICE_NAME 3.5,.01
10 NAME SPOOL_DOCUMENT_NAME 3.51,.01
PRESS <ENTER> TO SEE MORE, '^' TO EXIT THIS LIST, OR
CHOOSE 1-10: <ENTER>

```

Continue to press "Enter" until you see the entry highlighted in blue below.

CHOOSE 281-375:

```

376 NAME NEW_PERSON_NAME 200,.01
377 NAME USER_CLASS_NAME 201,.01
378 NAME NURS_SERVICE_POSITION_NAME 211.3,1
379 NAME NURS_LOCATION_NAME 211.4,.01
380 NAME *NURS_MI_CLASS_GROUP_NAME 212.42,.01

```

The numbers shown will vary by site.

```

PRESS <ENTER> TO SEE MORE, '^' TO EXIT THIS LIST, OR
CHOOSE 281-380: 376 <ENTER> NEW_PERSON_NAME 200,.01
ANOTHER ONE:
STANDARD CAPTIONED OUTPUT? YES// <ENTER> (YES)
INCLUDE COMPUTED FIELDS: (N/Y/R/B): NO// BOTH <ENTER> COMPUTED FIELDS AND RECORD
NUMBER
(IEN)

```

The number shown will vary by site.

```

NUMBER: 21324 NAME: NEW_PERSON_NAME
LOOKUP TERM: NAME DATA DICTIONARY NUMBER: 200
FIELD NUMBER: .01
DESCRIPTION: ANSWER MUST BE 3-35 UPPER-CASE CHARACTERS IN LENGTH, AND BE IN
THE FORMAT FAMILY(LAST),GIVEN(FIRST) MIDDLE SUFFIX. ENTER '??' FOR MORE HELP.
ENTER ONLY DATA THAT IS ACTUALLY PART OF THE PERSON'S NAME. DO NOT INCLUDE
EXTRA TITLES, IDENTIFICATION, FLAGS, LOCAL INFORMATION, ETC. ENTER THE
PERSON'S NAME IN 'LAST,FIRST MIDDLE SUFFIX' FORMAT. THIS VALUE MUST BE 3-35
CHARACTERS IN LENGTH AND MAY CONTAIN ONLY UPPERCASE ALPHA CHARACTERS, SPACES,
APOSTROPHES, HYPHENS AND ONE COMMA. ALL OTHER CHARACTERS AND PARENTHETICAL
TEXT WILL BE REMOVED.
BUILD(S) (C): XU*8.0*120
: XU*8.0*135
: XU*8.0*134
: XU*8.0*551 TYPE (C): FREE TEXT
SELECT META DATA DICTIONARY:

```

# 10 Creating Files and Fields

## 10.1 Creating a File

To create a new file, use the **Modify File Attributes** [DIMODIFY] option and enter the new file name (from **3 to 45 characters** in length) when asked:

MODIFY WHAT FILE:

Respond with **YES** when asked "Are you adding 'xxxxxxx' as a new FILE?" (where "xxxxxxx" represents the new file name).

### 10.1.1 Naming a New File

File names should be chosen so that they can easily be distinguished from each other:

MODIFY WHAT FILE: **ENT CLINIC PATIENT**



**TIP:** If different people are creating files, it can be helpful to include their initials, nick names, or other identifying phrases within the file name.



**NOTE:** The name for your new file should *not* contain any arithmetic or string operators like + - \* / \ \_ or punctuation like a colon (:).

An internal number *must* be assigned to this new file. You are prompted with the next available internal file number. You can either simply press **Enter** (a **NULL** response) to accept that number, or enter a number *not* already assigned to a file.

Your new file is now initialized. VA FileMan creates the NAME field (field number **.01**), which is:

- **FREE TEXT.**
- **3 to 30 characters** in length.
- *Non-numeric.*
- Has no leading punctuation.

You see the following message:

**A FREETEXT NAME FIELD (#.01) HAS BEEN CREATED.**

The **.01** field's definition can be modified like any other field: it does *not* have to be **FREE TEXT**; its label need *not* be NAME; and so forth. The **.01** field should be the key attribute of an entry used to identify it and to order the entries for lookups. However, entries can be identified and ordered by other fields through cross-references.

After creating a new file, you can define any number of fields for the new file, as described in the "[Creating Fields](#)" section.



**REF:** For an example of a new file using the **Modify File Attributes** [DIMODIFY] option, see the "[Examples of File and Field Creation](#)" section.

## 10.2 Creating Fields

For any file, you can create fields describing logically related data that pertains to entries in that file. When created, every file automatically receives one field: a NAME field (the **#.01** field). You *must* explicitly define any other fields. All such definitions are made (and changed) with the **Modify File Attributes** [DIMODIFY] option.

When you create a file, after you give the new file a name, you are asked:

SELECT FIELD:

Enter a new field name and respond with **YES** when VA FileMan asks:

ARE YOU ADDING [XXXXXXXXXX] AS A NEW FIELD?

The "**XXXXXXXXXX**" represents the name of the field you entered. After answering **YES** to this prompt, you are now ready to specify what sort of data the new field contains. A field can only be *one* type of data; the choices are listed in [Table 91](#).

## 10.2.1 Screen Mode Field Editing

When using the **Modify File Attributes** [DIMODIFY] option, you can make your field entries in the traditional Scrolling Mode or choose to create, modify, or review a file's field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

To make your entries in Screen Mode, simply press the **Enter** key to accept the default response at the "Do you want to use the screen-mode version? YES//"  
prompt.

[Figure 119](#) illustrates using Screen Mode when editing the **.01** field of the ORDER (#100) file:

**Figure 119: Creating Files and Fields—Choosing Screen Mode When Using the Modify File Attributes Option**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER

SELECT FIELD: .01 <ENTER> ORDER #
```

You are taken into Screen Mode where you can edit the properties of the field, as shown in [Figure 120](#):

**Figure 120: Creating Files and Fields—Example Using the Modify File Attributes Option in Screen Mode**

```
FIELD #.01 IN FILE #100
FIELD LABEL: ORDER # DATA TYPE... NUMERIC

TITLE:
AUDIT:
AUDIT CONDITION:
READ ACCESS:
DELETE ACCESS:
WRITE ACCESS:
SOURCE:
DESCRIPTION... TECHNICAL DESCRIPTION...

IS THIS FIELD MULTIPLE... NO

MANDATORY: YES
HELP-PROMPT: ENTER THE ORDER NUMBER.
XECUTABLE HELP:

NOTE THAT THIS FIELD'S DEFINITION IS NOT EDITABLE

PRESS <PF1>H FOR HELP INSERT
```

Using the Screen Mode version of this option, after entering the field name or number (e.g., **.01** field) after the "Select FIELD:" prompt, you are presented with a ScreenMan form (screen) that can be reviewed and edited like any other. In this example, the most important field on this screen is the DATA TYPE field in the upper right corner; it is a required entry. In this example, required entries in Screen Mode are indicated by a caption with a different color and an underline.



**NOTE:** Screen Mode highlights the captions for required fields with an underline. However, depending on your terminal or terminal emulator software and your personal preferences, the form of the highlight can vary (e.g., some emulators highlight required field captions in reverse video, a different color, with an underline, or any combination of highlights).

**REF:** For more information on ScreenMan forms and Screen Mode, see the "ScreenMan" section in the *VA FileMan User Manual*.

In this case, the DATA TYPE field has been defined as **NUMERIC** and is *not editable* as indicated by the message displayed near the bottom of the screen (i.e., "NOTE THAT THIS FIELD'S DEFINITION IS NOT EDITABLE"); this is because a developer has previously edited the definition in a special way. However, unlike the DATA TYPE field, the value of the HELP-PROMPT field (i.e., "Enter the Order number."), which is the message that is displayed to users when they enter a single question mark (?) while editing the ORDER # field, can be edited. The DESCRIPTION and TECHNICAL DESCRIPTION fields are multi-line **WORD-PROCESSING** fields; to edit them, press the **Enter** key and a separate screen opens (i.e., a "popup" window). The DESCRIPTION is displayed to users who enter two question marks (??) while editing the ORDER # field. The TECHNICAL DESCRIPTION, however, is for internal documentation only.

## 10.2.2 Field Data Types

There are seventeen field data types, as shown in [Table 91](#):

**Table 91: Creating Files and Fields—Data Types**

No.	Data Type	Description
1	<a href="#">DATE/TIME</a>	Dates with or without time stamps.
2	<a href="#">NUMERIC</a>	DATA TYPE fields of <b>NUMERIC</b> , including dollar values.
3	<a href="#">SET OF CODES</a>	Codes that represent values (e.g., <b>1=MALE/2=FEMALE</b> ).
4	<a href="#">FREE TEXT</a>	A single alphanumeric string of characters.
5	<a href="#">WORD-PROCESSING</a>	A Multiline document of text.
6	<a href="#">COMPUTED</a>	A virtual field, values <i>not</i> stored.
7	<a href="#">POINTER TO A FILE</a>	Referencing an entry in some other file.
8	<a href="#">VARIABLE-POINTER</a>	Referencing an entry in a defined set of files.
9	<a href="#">MUMPS</a>	Used by developers to enter M code.
10	<a href="#">BOOLEAN</a>	Boolean field.
11	<a href="#">LABEL REFERENCE</a>	Label Reference field.
12	<a href="#">TIME</a>	Time field.
13	<a href="#">YEAR</a>	Year field.
14	<a href="#">UNIVERSAL TIME</a>	Universal Time field.
15	<a href="#">FT POINTER</a>	Free Text Pointer field.
16	<a href="#">FT DATE</a>	Free Text Date field.
17	<a href="#">RATIO</a>	Ratio field.

You are asked for the DATA TYPE field value for any field you are creating. You *must* pick one of these seventeen choices. Data validation checks are then asked depending on the DATA TYPE field value entered. For some data types, a default “HELP” prompt is automatically composed.



**NOTE:** You can review and change a file’s field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For examples of entering a field attributes in Screen Mode, see the [“Examples of File and Field Creation”](#) section.

### 10.2.3 DATE/TIME Data Type

A DATA TYPE field defined as **DATE/TIME** allows you to enter a minimum and maximum date. You can also indicate whether the date can be entered with an imprecise date (e.g., **JUL 1969**) or with the time-of-day (e.g., **JUL 20@4**). VA FileMan does *not* accept dates *before* **1700**.

For example, when defining a DATA TYPE field value as **DATE/TIME**, you are asked the questions shown in [Figure 121](#):

**Figure 121: Creating Files and Fields—Defining a DATA TYPE Field Value as DATE/TIME in Scrolling Mode (1 of 2)**

```
SELECT FIELD: DATE OF BIRTH
  ARE YOU ADDING DATE OF BIRTH AS A NEW FIELD? NO// Y <ENTER> (YES)


DATA TYPE OF DATE OF BIRTH: DATE/TIME
EARLIEST DATE (OPTIONAL): 1/1/1860 <ENTER> (JAN 01, 1860)
LATEST DATE: 1963 <ENTER> (1963)
CAN DATE BE IMPRECISE (Y/N): YES// <ENTER>
CAN TIME OF DAY BE ENTERED (Y/N): NO// <ENTER> NO
```


If you reply **YES** to the “CAN TIME OF DAY BE ENTERED (Y/N): NO//” prompt, you would then be asked “CAN SECONDS BE ENTERED?”.


**Figure 122: Creating Files and Fields—Defining a DATA TYPE Field Value as DATE/TIME in Scrolling Mode (2 of 2)**

```
WILL DATE OF BIRTH FIELD BE MULTIPLE: NO// <ENTER> (NO)
IS DATE OF BIRTH ENTRY MANDATORY (Y/N): NO// Y <ENTER> YES
...
[HELP]-PROMPT: TYPE A DATE BETWEEN 1/1/1860 AND 1963
REPLACE <ENTER>
DESCRIPTION:
1> <ENTER>
```

A default help prompt is automatically written for you with the DATA TYPE field of **DATE/TIME**. You can change this prompt using the “**Replace ... With**” syntax.

 **NOTE:** This help information is displayed when the user inputs a single question mark (?) when editing this field.

 **NOTE:** The MINIMUM LENGTH, MAXIMUM LENGTH, and PATTERN MATCH control the values a user can *input* for the field. You can control the maximum length of *output* in FileMan reports by setting MAXIMUM LENGTH in the **Input Transform** [DIITRAN] option on the **Utility Functions** [DIUTILITY] menu.

 **NOTE:** You can review and change a file’s field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For an example of entering a DATA TYPE field value defined as **DATE/TIME** in Screen Mode, see the “[Examples of File and Field Creation](#)” section.

## 10.2.4 NUMERIC Data Type

A DATA TYPE field defined as **NUMERIC** requires you to enter the lowest and highest values allowed, the maximum number of decimal digits allowed, and to state whether dollar values are allowed (e.g., \$33).

For example, when defining a DATA TYPE field value as **NUMERIC**, you are asked the questions in [Figure 123](#):

**Figure 123: Creating Files and Fields—Defining a DATA TYPE Field Value as NUMERIC in Scrolling Mode (1 of 2)**

```
SELECT FIELD: AGE AT ONSET
ARE YOU ADDING  AGE AT ONSET  AS A NEW FIELD? NO// Y <ENTER> (YES)

DATA TYPE OF AGE AT ONSET: NUMERIC
INCLUSIVE LOWER BOUND: 0
INCLUSIVE UPPER BOUND: 100
IS THIS A DOLLAR AMOUNT (Y/N): NO// <ENTER>
```

If you answer **YES** at the “IS THIS A DOLLAR AMOUNT (Y/N):” prompt, VA FileMan allows users to precede input data with a dollar sign (\$) and shows up to two decimal places.

**Figure 124: Creating Files and Fields—Defining a DATA TYPE Field Value as NUMERIC in Scrolling Mode (2 of 2)**

```
MAXIMUM NUMBER OF FRACTIONAL DIGITS: 0// <ENTER>
WILL AGE AT ONSET FIELD BE MULTIPLE? NO// <ENTER> (NO)
IS AGE AT ONSET ENTRY MANDATORY (Y/N): NO// <ENTER> NO
....
[HELP]-PROMPT: TYPE A NUMBER BETWEEN 0 AND 100, 0 DECIMAL DIGITS
REPLACE <ENTER>
DESCRIPTION:
1> <ENTER>
```

A default help prompt is automatically written for you with the DATA TYPE field value of **NUMERIC**. You can change this prompt using the “**Replace ... With**” syntax.



**NOTE:** This help information is displayed when the user inputs a single question mark (?) when editing this field.



**NOTE:** You can review and change a file’s field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For an example of entering a DATA TYPE field value defined as **NUMERIC** in Screen Mode, see the "[Examples of File and Field Creation](#)" section.

### 10.2.5 SET OF CODES Data Type

A DATA TYPE field defined as a **SET OF CODES** can be used to restrict a user to just a few possible values (e.g., **YES** or **NO**). When defining a DATA TYPE field value of **SET OF CODES**, enter a valid code and a translation of what each code means. The user can enter the code, the full meaning, or a portion of the full meaning. If the field is set up to require only a one-character response (as shown in [Figure 125](#)), this data type can simplify the user's data entry.

VA FileMan has only a limited amount of space to store the codes and their external values. If the limit is exceeded, you are told "TOO MUCH!--SHOULD BE A 'POINTER', NOT 'SET'." The DATA TYPE field value of **SET OF CODES** is sometimes referred to as a **SET**.

For example, when defining a DATA TYPE field value as a **SET OF CODES**, you are asked questions shown in [Figure 125](#):

**Figure 125: Creating Files and Fields—Defining a DATA TYPE Field Value as SET OF CODES in Scrolling Mode**

```
SELECT FIELD: SEX
ARE YOU ADDING SEX AS A NEW FIELD? NO// Y <ENTER> (YES)

DATA TYPE OF SEX: SET <ENTER> OF CODES
INTERNALLY-STORED CODE: M <ENTER> WILL STAND FOR: MALE
INTERNALLY-STORED CODE: F <ENTER> WILL STAND FOR: FEMALE
INTERNALLY-STORED CODE: <ENTER>
WILL SEX FIELD BE MULTIPLE: NO// <ENTER> (NO)
IS SEX ENTRY MANDATORY (Y/N): NO// Y <ENTER> YES
...
HELP-PROMPT: <ENTER>
DESCRIPTION:
1> <ENTER>
```

In this example, "m" stands for Male and "f" stands for Female. Numbers as well as alphabetic characters can be used.



**NOTE:** You can review and change a file's field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For an example of entering a DATA TYPE field value defined as a **SET OF CODES** in Screen Mode, see the "[Examples of File and Field Creation](#)" section.

## 10.2.6 FREE TEXT Data Type

A DATA TYPE field defined as **FREE TEXT** allows you to enter the maximum and minimum allowable string length of the **FREE TEXT** data. You can also enter an M PATTERN MATCH that input data must match.

For example, when defining a DATA TYPE field value as **FREE TEXT**, you are asked the questions shown in [Figure 126](#):

**Figure 126: Creating Files and Fields—Defining a DATA TYPE Field Value as FREE TEXT in Scrolling Mode (1 of 3)**

```
SELECT FIELD: DIAGNOSIS
ARE YOU ADDING DIAGNOSIS AS A NEW FIELD? NO// Y <ENTER> (YES)

DATA TYPE OF DIAGNOSIS: FREE TEXT
MINIMUM LENGTH: 3
MAXIMUM LENGTH: 30
(OPTIONAL) PATTERN MATCH (IN X): <ENTER>
```

The PATTERN MATCH is written in M code. If input data violates the PATTERN MATCH or the Minimum/Maximum lengths, the data is *not* accepted, and the user is shown the help prompt information.

**Figure 127: Creating Files and Fields—Defining a DATA TYPE Field Value as FREE TEXT in Scrolling Mode (2 of 3)**


```
WILL DIAGNOSIS FIELD BE MULTIPLE? NO// Y <ENTER> (YES)
IS DIAGNOSIS ENTRY MANDATORY(Y/N): NO// <ENTER> NO
SHOULD USER SEE AN ADDING A NEW DIAGNOSIS? MESSAGE FOR NEW
ENTRIES (Y/N): N <ENTER> NO
HAVING ENTERED OR EDITED ONE DIAGNOSIS, SHOULD USER BE ASKED
ANOTHER (Y/N): Y <ENTER> YES
```


With these specifications, the user is *not* given a confirming message when new subentries are added to the Multiple. The user is allowed to enter several diagnoses in a row for a given patient.

**Figure 128: Creating Files and Fields—Defining a DATA TYPE Field Value as FREE TEXT in Scrolling Mode (3 of 3)**

```
....  
....  
[HELP]-PROMPT: ANSWER MUST BE 3-30 CHARACTERS IN LENGTH.  
REPLACE <ENTER>  
DESCRIPTION:  
1> <ENTER>
```

A default help prompt is automatically written for you with the DATA TYPE field value of **FREE TEXT**. You can change this prompt using the “**Replace ... With**” syntax.


 **NOTE:** This help information is displayed when the user inputs a single question mark (?) when editing this field.

 **NOTE:** You can review and change a file’s field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For an example of entering a DATA TYPE field value defined as **FREE TEXT** in Screen Mode, see the “[Examples of File and Field Creation](#)” section.

### 10.2.7 WORD-PROCESSING Data Type

A DATA TYPE field defined as **WORD-PROCESSING** allows entry of unlimited free-text data. The data can be edited, formatted, and printed with word-processing text editors.

 **REF:** For a description of VA FileMan’s native text editors, see the “Screen Editor” and “Line Editor” sections in the *VA FileMan User Manual*.

For example, when defining a DATA TYPE field value as **WORD-PROCESSING**, you are asked the questions shown in [Figure 129](#):

**Figure 129: Creating Files and Fields—Defining a DATA TYPE Field Value as WORD-PROCESSING in Scrolling Mode**

```
SELECT FIELD: HISTORY
ARE YOU ADDING [HISTORY] AS A NEW FIELD? NO// Y <ENTER> (YES)

DATA TYPE OF HISTORY: WORD-PROCESSING
SHALL THIS TEXT NORMALLY APPEAR IN WORD-WRAP MODE? YES// <ENTER> (YES)
....
....
[HELP]-PROMPT: SUBJECTIVE NARRATIVE OF PATIENT[S] PROBLEM HISTORY
DESCRIPTION:
1> <ENTER>
```

If you answer **YES** to the “SHALL THIS TEXT NORMALLY APPEAR IN WORD-WRAP MODE” question, text is automatically wrapped at word boundaries to fit in the column in which it is being printed. Usually, this is the preferred way to print text.



**TIP:** When it is important that lines of text be printed exactly as they were entered, answer **NO** to the “SHALL THIS TEXT NORMALLY APPEAR IN WORD-WRAP MODE” question. Thus, text is output in no-wrap mode. You would probably want a spreadsheet or a restaurant menu printed in no-wrap mode.



**NOTE:** If the column in which no-wrap text is being printed is too short to accommodate the line of text, your printer can break the line in the middle of words or otherwise destroy the formatting of the text.



**NOTE:** You can review and change a file’s field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For an example of entering a DATA TYPE field value defined as **WORD-PROCESSING** in Screen Mode, see the [“Examples of File and Field Creation”](#) section.

## 10.2.8 COMPUTED Data Type

When a DATA TYPE field is defined as **COMPUTED**, its value is determined at the time the field is accessed. This computation is based on an expression stored in the data dictionary. The field value (or data) itself is *not* stored in the data dictionary. The computed expression is constructed using field names, literals or constants, functions, and operators.



**REF:** For a complete explanation of these elements, see the “[Computed Expressions](#)” section.



**NOTE:** The functions referred to above are VA FileMan functions stored in the FUNCTION (#.5) file, *not* M functions. A developer with **Programmer** access can also enter M code in a **COMPUTED** field. The M code *must* set variable **X** to the **COMPUTED** field value.

For example, when defining a DATA TYPE field value as **COMPUTED**, you are asked the questions shown in [Figure 130](#):

**Figure 130: Creating Files and Fields—Defining a DATA TYPE Field Value as COMPUTED in Scrolling Mode (1 of 2)**

```
SELECT FIELD: AGE
ARE YOU ADDING  AGE  AS A NEW FIELD? NO// Y <ENTER> (YES)

DATA TYPE OF AGE: COMPUTED

 COMPUTED-FIELD  EXPRESSION: TODAY - (DATE OF BIRTH) \365.25

....
NUMBER OF FRACTIONAL DIGITS TO OUTPUT (ONLY ANSWER IF NUMBER-VALUED): 2
```

For this example ([Figure 130](#)), you assume the DATE OF BIRTH field was previously created. Thus, you can reference it in the “‘COMPUTED-FIELD’ EXPRESSION:” prompt. Also, the “NUMBER OF FRACTIONAL DIGITS TO OUTPUT (ONLY ANSWER IF NUMBER-VALUED):” prompt is asking whether you should enter the number of digits that should normally appear to the right of the decimal point when this field is displayed.

**Figure 131: Creating Files and Fields—Defining a DATA TYPE Field Value as COMPUTED in Scrolling Mode (2 of 2)**

```
SHOULD VALUE ALWAYS BE INTERNALLY ROUNDED TO 2 DECIMAL PLACES? NO// Y <ENTER>
(YES)
WHEN TOTALLING THIS FIELD, SHOULD THE SUM BE COMPUTED FROM
  THE SUMS OF THE COMPONENT FIELDS? NO// <ENTER> (NO)
LENGTH OF FIELD: 8// <ENTER>
```



**NOTE:** You can review and change a file's field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For an example of entering a DATA TYPE field defined as **COMPUTED** in Screen Mode, see the "[Examples of File and Field Creation](#)" section.

## 10.2.9 POINTER TO A FILE Data Type

A DATA TYPE field defined as a **POINTER TO A FILE** requires you to enter the name or number of the pointed-to file and that file *must* already exist (defined previously).

For example, when defining a DATA TYPE field value as **POINTER TO A FILE**, you are asked the questions shown in [Figure 132](#):

**Figure 132: Creating Files and Fields—Defining a DATA TYPE Field Value as POINTER TO A FILE in Scrolling Mode (1 of 3)**

```
SELECT FIELD: RELIGION
ARE YOU ADDING RELIGION AS A NEW FIELD? NO// Y <ENTER> (YES)

DATA TYPE OF RELIGION: POINTER <ENTER> TO A FILE
POINT TO WHICH FILE: RELIGION
```

The pointed to file *must* already exist on your system. If you enter a single question mark (?) at the "POINT TO WHICH FILE:" prompt, you are presented with a list of the available files.

**Figure 133: Creating Files and Fields—Defining a DATA TYPE Field Value as POINTER TO A FILE in Scrolling Mode (2 of 3)**

```
SHOULD [ADDING A NEW RELIGION FILE ENTRY] ([LAYGO])  
BE ALLOWED WHEN ANSWERING THE [RELIGION] QUESTION? NO// <ENTER> (NO)
```

By answering **NO** to this prompt, users who are editing patient data are *not* able to add a new entry on-the-fly to the RELIGION file. This prompt depends on whether you have **LAYGO** access to the file or *not*.

**Figure 134: Creating Files and Fields—Defining a DATA TYPE Field Value as POINTER TO A FILE in Scrolling Mode (3 of 3)**

```
WILL RELIGION FIELD BE MULTIPLE? NO// <ENTER> (NO)  
IS RELIGION ENTRY MANDATORY (Y/N): NO// <ENTER> NO  
....  
[HELP]-PROMPT: <ENTER>  
DESCRIPTION:  
1> <ENTER>
```



**NOTE:** You can review and change a file's field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For an example of entering a DATA TYPE field value defined as **POINTER TO A FILE** in Screen Mode, see the "[Examples of File and Field Creation](#)" section.

## 10.2.10 VARIABLE-POINTER Data Type

A DATA TYPE field defined as a **VARIABLE-POINTER** (as with the **POINTER TO A FILE** DATA TYPE) requires you to enter the names or numbers of the pointed-to files and those files *must* already exist (defined previously). Additionally, an order, message, and prefix *must* be associated with each file.

For example, when defining a DATA TYPE field value as **VARIABLE-POINTER**, you are asked the questions shown in [Figure 135](#):

**Figure 135: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (1 of 5)**

```
SELECT FIELD: PROVIDER
ARE YOU ADDING  PROVIDER  AS A NEW FIELD? NO// Y <ENTER> (YES)

DATA TYPE OF PROVIDER: VARIABLE-POINTER
SELECT VARIABLE POINTER: PROVIDER
```

You answer the “Select VARIABLE POINTER:” prompt with the name or number of an existing file.

**Figure 136: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (2 of 5)**

```
ARE YOU ADDING  PROVIDER  AS A NEW VARIABLE-POINTER? NO// Y <ENTER> (YES)

VARIABLE-POINTER: PROVIDER// <ENTER>
MESSAGE: STAFF PROVIDER
ORDER: 1
PREFIX: S
SHOULD USER BE ALLOWED TO ADD A NEW ENTRY: NO <ENTER> NO
```

The MESSAGE is part of the online help associated with the VARIABLE-POINTER field when a single question mark (?) is entered during editing. In this example, the PROVIDER file MESSAGE is associated with “Staff Provider.”

The several pointed-to files are searched based on their ORDER. In this example, since the ORDER for the PROVIDER **VARIABLE-POINTER** is one, the PROVIDER file is the first file searched.

The PREFIX field is used to reference a particular pointed-to file. To see the entries in a particular file, you would enter that file’s PREFIX followed by a period and a question mark at the **VARIABLE-POINTER**’s field name (e.g., in this example, entering “S.?” would give you the option to list the entries in the PROVIDER file). If you want to refer to only one of the several pointed-to files, put that file’s PREFIX followed by a period at the

**VARIABLE-POINTER**'s field name (e.g., in this example, entering "S." would refer you to the PROVIDER file).

By answering **NO** to the "SHOULD USER BE ALLOWED TO ADD A NEW ENTRY:" prompt, the user is *not* allowed to add new entries on-the-fly to the PROVIDER file (the same as the RELIGION field entered as a **POINTER TO A FILE**). If you had answered **YES**, then new entries could be added to the PROVIDER file.

**Figure 137: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (3 of 5)**

```
SELECT VARIABLE-POINTER: 16 <ENTER> PERSON
  ARE YOU ADDING PERSON AS A NEW VARIABLE-POINTER? (THE 2ND)? NO// Y <ENTER>
(YES)
VARIABLE-POINTER: PERSON// <ENTER>
MESSAGE: OTHER PROVIDER
ORDER: 2
PREFIX: 0
SHOULD USER BE ALLOWED TO ADD A NEW ENTRY: YES <ENTER> YES
```

In this example ([Figure 137](#)), a second **VARIABLE-POINTER** was created to the (fictitious) PERSON (#16) file. By answering **YES** to the "SHOULD USER BE ALLOWED TO ADD A NEW ENTRY:" prompt, users who are editing patient data are allowed to add entries to the (fictitious) PERSON file.

**Figure 138: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (4 of 5)**

```
SELECT VARIABLE-POINTER: <ENTER>
```

You stop identifying files for a **VARIABLE-POINTER** by simply pressing the **Enter** key without any additional entries at the "Select VARIABLE-POINTER:" prompt.

**Figure 139: Creating Files and Fields—Defining a DATA TYPE Field Value as VARIABLE-POINTER in Scrolling Mode (5 of 5)**

```
WILL PROVIDER FIELD BE MULTIPLE? NO// <ENTER> (NO)
IS PROVIDER ENTRY MANDATORY (Y/N): NO// <ENTER> NO
HELP-PROMPT: <ENTER>
DESCRIPTION:
1> <ENTER>
```

After entering both **VARIABLE-POINTERS**, when you enter a single question mark (?) at the “PROVIDER” field prompt, you see the help message shown in [Figure 140](#):

**Figure 140: Creating Files and Fields—Example of Help Associated with a VARIABLE-POINTER Field**

```
PROVIDER: ?  
  ENTER ONE OF THE FOLLOWING:  
    S.ENTRYNAME TO SELECT A STAFF PROVIDER  
    O.ENTRYNAME TO SELECT A OTHER PROVIDER  
  
  TO SEE THE ENTRIES IN ANY PARTICULAR FILE TYPE <PREFIX.??>
```

In this example, if you simply enter a name at the “PROVIDER:” prompt, then the system searches each of the **VARIABLE-POINTER** field files for the name you have entered.

If a match is found, the system asks you if it is the correct entry. However, if you know the file the entry should be in, then you can speed processing by using the following syntax to select an entry:

- **PREFIX.entry name**
- **MESSAGE.entry name**
- **File Name.entry name**



**NOTE:** You do *not* need to enter the entire file name or message to direct the lookup. Using the first few characters suffices.



**NOTE:** You can review and change a file’s field attributes easily by running the **Modify File Attributes** [DIMODIFY] option in Screen Mode.

**REF:** For an example of entering a DATA TYPE field defined as a **VARIABLE-POINTER** in Screen Mode, see the [“Examples of File and Field Creation”](#) section.

### 10.2.11 MUMPS Data Type

Those with **Programmer** access can define a field with a DATA TYPE field value of **MUMPS**. This **MUMPS**-valued field is designed specifically to contain executable M code. The code entered into this kind of field is verified to be valid M code that conforms to VA programming standards.

DATA TYPE fields defined as **MUMPS** are usually used in files that are part of developer tools systems. For example, the OPTION (#19) file is part of the MENU MANAGEMENT

system used in the VA for assigning menus and associated actions to each computer user. The ENTRY ACTION field on the OPTION (#19) file is defined as a DATA TYPE field of **MUMPS**. This field allows a developer who is creating an option to enter M code to do any setup and initialization that is needed before the end user can do the action allowed by the option.

### 10.2.12 BOOLEAN Data Type

A field defined as a **BOOLEAN** data type can have only two entry choices:

- **YES**
- **NO**

The internal values of the **BOOLEAN** data type are:

- **1—YES**
- **0—NO**

Example:

External: YES

Internal: 1

External: NO

Internal: 0

### 10.2.13 LABEL REFERENCE Data Type

A field defined as a **LABEL REFERENCE** data type is designed to store a tag and routine entry of the format, **TAG^ROUTINE**. It is stored as a **FREE TEXT** field.

Example:

External: TAG^ROUTINE

Internal: TAG^ROUTINE

### 10.2.14 TIME Data Type

A field defined as a **TIME** data type can accept many of the **DATE/TIME** entries but only stores the **TIME** portion.

Example:

External: 15:09:43

Internal: 150943

### 10.2.15 YEAR Data Type

A field defined as a **YEAR** data type can accept many of the date entries but only stores the **YEAR** portion.

Example:

External: 2016

Internal: 3160000

### 10.2.16 UNIVERSAL TIME Data Type

A field defined as a **UNIVERSAL TIME** field can accept many of the **DATE/TIME** entries and stores the date/time in a format with the local time and includes an indicator showing the offset from Universal Time.

The first **14 characters** of the internal storage of the **UNIVERSAL TIME** data type are exactly like the current **DATE/TIME** data type that includes seconds. The **3 characters** in position **15**, **16**, and **17** indicate the UTC time offset in **five (5) minute** increments. In the example below: **(440-500)/12=-5**, this is a negative **five hour** offset from UTC.

Example:

External: JAN 6,2016@08:03:36 (UTC-5:00)

Internal: 3160106.080336440

### 10.2.17 FT POINTER Data Type

A field defined as a **FT POINTER** field works like the **POINTER** data type but internally stores the free text that was returned from the pointed-to value.

Example:

External: PATCH,USER

Internal: PATCH,USER

### 10.2.18 FT DATE Data Type

A field defined as a **FT DATE** field works like the **DATE/TIME** data type but internally stores the free text that was input by the user to determine the date.

Example:

External: T-1

Internal: T-1

## 10.2.19 RATIO Data Type

A field defined as a **RATIO** field is designed to accept two numbers with a colon (:) between the two numbers. It is formatted and stored like a mathematical ratio.

Example:

External: 1:14

Internal: 1:14

## 10.3 Multiple-Valued Field (Multiples)

When you create a DATA TYPE field value of any of the following:

- [DATE/TIME](#)
- [NUMERIC](#)
- [SET OF CODES](#)
- [FREE TEXT](#)
- [POINTER TO A FILE](#)
- [VARIABLE-POINTER](#)
- [BOOLEAN](#)
- [LABEL REFERENCE](#)
- [TIME](#)
- [YEAR](#)
- [UNIVERSAL TIME](#)
- [FT POINTER](#)
- [FT DATE](#)
- [RATIO](#)

After entering type-specific information, you are asked the following:

WILL FIELD BE MULTIPLE: NO//

Answering **YES** to this prompt means that:

- There can be more than one occurrence of a data value for this field in the entry (e.g., more than one DIAGNOSIS in a PATIENT [#2] file entry)
- Subfields can later be associated with this field (e.g., each DIAGNOSIS could have a DATE OF ONSET)

The "MULTIPLE:" prompt is *not* asked for DATA TYPE field values of **WORD-PROCESSING**, since, by definition, such types take Multiline values.

Two special questions are asked about Multiple-valued fields:

SHOULD USER SEE AN ADDING NEW ENTRY? MESSAGE FOR NEW ENTRIES (Y/N)

Answering **NO** here means that the new Diagnosis (or whatever the Multiple is named) gets added *without* a verification prompt being asked.

HAVING ENTERED OR EDITED ONE, SHOULD USER BE ASKED ANOTHER (Y/N):

Answering either of the following:

- **YES**—User is prompted to put in several diagnoses, one right after the other.
- **NO**—User is *not* prompted to enter a second value.

## 10.4 Making a Field Mandatory

For all DATA TYPE field values except **COMPUTED**, as you create the field, you are asked the following:

IS XXXXXXXX ENTRY MANDATORY (Y/N): NO//

The "xxxxxxxx" represents the name of the field. If you answer **YES**, the user of your file is *not* allowed to skip the field *without* entering data for a particular entry.

## 10.5 Field Number Sequences

It is often useful to sequence the fields so that when you are using the **Enter or Edit File Entries** [DIEDIT] option and you ask to edit **ALL** fields, you will see the field questions presented in a natural order. If you want to add a CURRENT AGE (#2.5) field to the PATIENT (#2) file and place it between DATE OF BIRTH (#2) and RELIGION (#3), the dialog would be as shown in [Figure 141](#):

**Figure 141: Creating Files and Fields—Example of “Sequencing” a Field**

```
SELECT FIELD: 2.5
ARE YOU ADDING A NEW FIELD: NO// Y <ENTER> (YES)
LABEL: CURRENT AGE
FIELD NUMBER: 2.5// <ENTER>
```

You could have specified any number between two and three.

## 10.6 NUMBER (.001) Field

All files have unique numbers associated with each of their entries. Defining the NUMBER field allows you to use the Internal Entry Number (IEN, also called the record number) as you would any other field. Usually, this means that someone (e.g., **Herr Doktor, ONE FM PROVIDER**, in the case of the [fictitious] MOZART WORK file) has gone to the trouble of creating a numbering scheme for the entries. To set up a file in which a unique Internal Entry Number is always matched with each entry name, you can create a field numbered **.001** for the file:

**Figure 142: Creating Files and Fields—Creating a NUMBER (#.001) Field**

```
SELECT FILE: MOZART WORK
SELECT FIELD: .001
ARE YOU ADDING A NEW FIELD? NO// YES <ENTER> (YES)
LABEL: KOECHEL NUMBER
FIELD NUMBER: .001// <ENTER>

DATA TYPE OF KOECHEL NUMBER: NUMERIC
INCLUSIVE LOWER BOUND: 1
INCLUSIVE UPPER BOUND: 626
IS THIS A DOLLAR AMOUNT (Y/N): NO// <ENTER>
MAXIMUM NUMBER OF FRACTIONAL DIGITS: 0// <ENTER>
HELP PROMPT: TYPE A NUMBER BETWEEN 1 AND 626, 0 DECIMAL DIGITS.
REPLACE <ENTER>
DESCRIPTION:
1> <ENTER>
```

The dialog in [Figure 143](#) is what would normally create a **NUMERIC**-valued field. In this case, you are describing the file's PRIMARY KEY, or Internal Entry Number.

Once such a **.001** field is defined, you can create a new file entry that might look like [Figure 143](#):

**Figure 143: Creating Files and Fields—Example of Creating a New File Entry with a .001 Field Defined**

```
SELECT MOZART WORK: EINE KLEINE NACHTMUSIK
ARE YOU ADDING A NEW MOZART WORK? NO// YES <ENTER> (YES)
KOECHEL NUMBER: 525
```



**NOTE:** The PRIMARY KEY was added with VA FileMan 22.2.

More importantly, with a **.001** field defined, an entry in the file can always be looked up by the Internal Entry Number (IEN), irrespective of any other cross-referencing that exists for the file, as shown in [Figure 144](#):

**Figure 144: Creating Files and Fields—Looking Up an Entry in a File Using the IEN**

```
SELECT MOZART WORK: 525 <ENTER> EINE KLEINE NACHTMUSIK
```

Record Numbers *must* always be positive and canonic (i.e., they *cannot* contain alpha suffixes, leading **Zeroes**, or trailing fractional **Zeroes**).

Incidentally, the **.001** field example ([Figure 142](#)) illustrates how using the **Modify File Attributes** [DIMODIFY] option can force a field to have a particular number (**.001** in this case). It is done by entering the new Number first, and then the new Label.

### 10.6.1 Forced Lookups Using Numbers

Number-meaningful lookups can be forced by prefixing the numeric input with the ` (grave accent). If FMPATIENT,FIVE's Internal Entry Number (IEN) in the PATIENT (#2) file is 355, they could be identified as shown in [Figure 145](#):

**Figure 145: Creating Files and Fields—Looking Up an Entry in a File Using the FMPATIENT,FIVE's IEN**

```
SELECT PATIENT NAME: `355 <ENTER> FMPATIENT,FIVE
```

If the **.01** field of a file is a pointer to another file, an entry can be looked up by prefacing the IEN of the entry in the pointed-to file with `` (double grave accent). For example, the **.01** field of the PATIENT ALLERGIES (#120.8) file is a pointer to the PATIENT (#2) file. If the IEN of PATIENT (#2) file entry FMPATIENT,ONE is 4, then you could choose the PATIENT ALLERGIES (#120.8) file entry that points to FMPATIENT,ONE, as shown in [Figure 146](#):

**Figure 146: Creating Files and Fields—Looking Up an Entry in a File Using the FMPATIENT,ONE’s IEN**

```
OUTPUT FROM WHAT FILE: PATIENT ALLERGIES// <ENTER> (4 ENTRIES)
SELECT PATIENT ALLERGIES: ``4 <ENTER> FMPATIENT,ONE 3-3-60
```

## 10.7 Changing and Deleting Fields

- [Changing Field Attributes](#)
- [Changing a Field’s DATA TYPE Value](#)
- [Deleting an Existing Field](#)

### 10.7.1 Changing Field Attributes

After creating a field in a file, you can return to **change** or **delete** the field within the **Modify File Attributes** [DIMODIFY] option, simply by entering the field name (or number) when asked:

```
SELECT FIELD:
```

When you return to the field in this option, you can change the field’s:

- Label (Name)
- Title (long form of its name)
- Audit and Audit Conditions (to indicate which fields should be audited)
- Read/Delete/Write
- Source
- Destination
- Group
- Description of the field, a **WORD-PROCESSING** field (what the user sees after entering two question marks)

- Technical Description

After you are presented with these attributes of the field, you can change the attributes defined during the initial definition of the field as described in the "[Creating Fields](#)" section.

For example, say you have created an SSN field (Social Security Number) in the PATIENT (#2) file and would now like to edit the field:

**Figure 147: Editing a Field—LABEL, TITLE, and AUDIT Attributes**

```
SELECT FIELD: SSN
LABEL: SSN// <ENTER>
TITLE: SOCIAL SECURITY NUMBER
AUDIT: YES, ALWAYS
AUDIT CONDITION: <ENTER>
```



**REF:** Auditing is described in the "[Auditing](#)" section.

**Figure 148: Editing a Field—ACCESS Privileges Attributes**

```
READ ACCESS (OPTIONAL): <ENTER>
DELETE ACCESS (OPTIONAL): <ENTER>
WRITE ACCESS (OPTIONAL): <ENTER>
```



**REF:** Control of various kinds of access to files is described in the "[Data Security](#)" section.

**Figure 149: Editing a Field—SOURCE, DESTINATION, GROUP Attributes**

```
SOURCE: <ENTER>
SELECT DESTINATION: <ENTER>
SELECT GROUP: DEMOG
```

A GROUP is a shorthand way for the user to refer to several fields at once when using the **Print File Entries** [DIPRINT] or the **Enter or Edit File Entries** [DIEDIT] options. Here, SSN is being assigned to the DEMOG group.

**Figure 150: Editing a Field—DESCRIPTION Attributes**

```
DESCRIPTION:
1>AN ENTRY IS REQUIRED. IF YOU DO NOT KNOW THIS PATIENT'S SOCIAL
2>SECURITY NUMBER, ENTER 000000000 TO INDICATE THE NUMBER IS
3>UNKNOWN.
EDIT OPTION: <ENTER>
TECHNICAL DESCRIPTION:
1> <ENTER>
```

The DESCRIPTION and TECHNICAL DESCRIPTION attributes document the use and meaning of the field. The information in DESCRIPTION is shown to the user when two question marks are entered at the "EDIT Option:" prompt. When initially creating a field, you are prompted for the DESCRIPTION field after the 'HELP'-PROMPT. The TECHNICAL DESCRIPTION is displayed only when the data dictionary is printed.



**NOTE:** Prior to VA FileMan 21.0 you were allowed to enter a Help Frame for field documentation; that attribute is no longer supported.

**Figure 151: Editing a Field—DATA TYPE, LENGTH, PATTERN MATCH, MANDATORY 'HELP' PROMPT Attributes**

```
DATA TYPE OF SSN: FREE TEXT// <ENTER>
MINIMUM LENGTH: 9// <ENTER>
MAXIMUM LENGTH: 9// <ENTER>
(OPTIONAL) PATTERN MATCH (IN [X]): X?9N// <ENTER>
IS SSN ENTRY MANDATORY (Y/N): Y// <ENTER>
[HELP]-PROMPT: ANSWER MUST BE 9 CHARACTERS IN LENGTH
REPLACE ... WITH ENTER 9 NUMBERS WITHOUT DASHES, E.G., 666456789.
REPLACE <ENTER>
ENTER 9 NUMBERS WITHOUT DASHES, E.G., 666456789.
```

To illustrate the use of GROUPs, add the NAME, DATE OF BIRTH, and SEX fields into the DEMOG group:

**Figure 152: Editing a Field—Adding Fields to a GROUP (1 of 2)**

```
SELECT FIELD: NAME
LABEL: NAME// ^GROUP
SELECT GROUP: DEMOG
DESCRIPTION:
1> <ENTER>
TECHNICAL DESCRIPTION:
1> <ENTER>

DATA TYPE OF NAME: FREE TEXT// ^
```

DATE OF BIRTH and SEX can be added to the GROUP in the same way. Now, when using the **Enter or Edit File Entries** [DIEDIT] option, you could do the following:

**Figure 153: Editing a Field—Adding Fields to a GROUP (2 of 2)**

```
EDIT WHICH FIELD: DEMOG
1  DEMOG  NAME
2  DEMOG  SEX
3  DEMOG  DATE OF BIRTH
EDIT WHICH FIELD: <ENTER>

SELECT PATIENT NAME: FMPATIENT,55
NAME:  FMPATIENT,ONE// <ENTER>
SEX:   MALE// <ENTER>
DATE OF BIRTH:  JAN 3, 1955// <ENTER>
```

## 10.7.2 Changing a Field's DATA TYPE Value

Within the **Modify File Attributes** [DIMODIFY] option, you can change the DATA TYPE field value itself. There are limitations on the sort of changes you can make. These are listed below.

You *must* be very careful in making such changes if you already have file data entered, because there is no guarantee that the old data matches the newly specified criteria (e.g., field length). However, if you do change a field definition, you are asked if you want existing data checked for inconsistencies. A list of any discrepancies is printed. If more than one discrepancy is found, you can save the list of discrepant entries in a template. To generate this list later, use the **Verify Fields** [DIVERIFY] option on the **Utility Functions** [DIUTILITY] menu.

The following restrictions apply to changing the definitions of existing DATA TYPE field values in a file:

- **Multiple**-valued fields *cannot* be changed to single-valued fields or vice versa. **Multiple**-valued fields can only be defined when creating a field.
- **COMPUTED** fields *cannot* be changed to other types of fields or vice versa.
- **WORD-PROCESSING**-type fields should only be changed into **Multiple**-valued **FREE TEXT** fields.
- Only a Multiple-valued **FREE TEXT** field can be changed into a **WORD-PROCESSING** field, and only if no other subfields are defined for that field.
- **POINTER TO A FILE** type fields *cannot* be changed to **VARIABLE-POINTER** type fields or vice versa.

### 10.7.3 Deleting an Existing Field

Deleting a field and its definition is done by deleting the field Name (LABEL). Delete the field by typing the at-sign (@) after the display of a field's LABEL when using the **Modify File Attributes** [DIMODIFY] option:

**Figure 154: Editing a Field—Deleting a Field and its Definition**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: PATIENT

SELECT FIELD: SEX
LABEL: SEX// @
    SURE YOU WANT TO DELETE THE ENTIRE [SEX] FIELD? YES
OK TO DELETE [SEX] FIELDS IN THE EXISTING ENTRIES? YES
```



**CAUTION:** If you answer **NO** to the “OK TO DELETE” question, data conflicts can occur in the future, if you create new fields. It is advisable to always delete existing entries. Only a developer can delete the entries after you have answered **NO**.

## 10.8 Examples of File and Field Creation

The following examples of creating files/fields or editing fields in a file are illustrated using Screen Mode.

- [File Creation](#)
- [DATE/TIME Fields](#)
- [SET OF CODES Field](#)
- [FREE TEXT Field](#)
  - [Carets \(^\) in a FREE TEXT Field](#)
  
- [WORD-PROCESSING Field](#)
- [COMPUTED Field](#)
- [POINTER TO A FILE Field](#)
- [VARIABLE-POINTER Field](#)
- [BOOLEAN Field](#)

- [LABEL REFERENCE Field](#)
- [TIME Field](#)
- [YEAR Field](#)
- [UNIVERSAL TIME Field](#)
- [FT POINTER Field](#)
- [FT DATE Field](#)
- [RATIO Field](#)
- [Creating a Multiple:](#)
  - [Subfields](#)
  - [Numeric Subfield](#)



**NOTE:** These examples assume that the user does *not* have **Programmer** access.

**REF:** For explanations of additional capabilities available to the developer, see the “Advanced File Definition” section in the *VA FileMan Developer’s Guide*.

## 10.8.1 File Creation

[Figure 155](#) illustrates the file definition dialog you see when creating a new file using the **Modify File Attributes** [DIMODIFY] option. In this case, you create the ORDER (#100) file (a standard VA FileMan file):

**Figure 155: Modify File Attributes Option—Creating a File**

```

SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER
ARE YOU ADDING ORDER AS A NEW FILE? NO// Y <ENTER> (YES)
FILE NUMBER: 99000// 100

...SORRY, HOLD ON...
A FREETEXT NAME FIELD (#.01) HAS BEEN CREATED.

SELECT FIELD: NAME

```

VA FileMan prompts you to enter the file name. If it is a new file, VA FileMan asks you to confirm that you want to add a new file. The default file number to the left of the //

(e.g., **99000**) is related to a site number that is assigned to your computer when VA FileMan is initialized. This file number is within the range of numbers assigned to your site. Be sure to follow local policies when assigning file numbers.

As you can see from this example ([Figure 155](#)), when a file is created a field with the label **NAME** and number **.01** is automatically created. When creating a new file, you can change the definition of this field in the same way you can change the definition of any other field.

When you select a field (e.g., the NAME field), you are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 156](#):

**Figure 156: Modify File Attributes Option—Defining the NAME (#.01) Field in Screen Mode**

```
FIELD #.01 IN FILE #100
FIELD LABEL: NAME DATA TYPE... FREE TEXT

TITLE:
AUDIT:
AUDIT CONDITION:
READ ACCESS:
DELETE ACCESS:
WRITE ACCESS:
SOURCE:
DESCRIPTION... TECHNICAL DESCRIPTION...

IS THIS FIELD MULTIPLE... NO

MANDATORY: YES
HELP-PROMPT: NAME MUST BE 3-35 CHARACTERS, NOT NUMERIC OR STARTING WITH PU
EXECUTABLE HELP:

COMMAND: PRESS <PF1>H FOR HELP INSERT
```

## 10.8.2 DATE/TIME Fields

In [Figure 157](#), the DATA TYPE field for the RELEASE DATE/TIME (#.68) field in the ORDER (#100) file has a DATA TYPE field value of **DATE/TIME**:

**Figure 157: Modify File Attributes Option—Editing a DATE/TIME Field in Screen Mode**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: RELEASE DATE/TIME
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 158](#):

**Figure 158: Modify File Attributes Option—Defining a DATA TYPE Field as DATE/TIME in Screen Mode**

```
FIELD #.68 IN FILE #100
FIELD LABEL: RELEASE DATE/TIME          DATA TYPE... DATE
-----
AU |                                     |
   | EARLIEST DATE: _                 |
   | LATEST DATE: _                   |
   | CAN DATE BE IMPRECISE: YES       |
   | CAN TIME OF DAY BE ENTERED: NO   |
   | CAN SECONDS BE ENTERED: NO      |
   | IS TIME REQUIRED: NO              |
D  |-----|
                                     |
IS THIS FIELD MULTIPLE... NO

MANDATORY: NO .
HELP-PROMPT: ENTER THE DATE/TIME THIS ORDER WAS RELEASED TO THE SERVICE
XECUTABLE HELP:

-----
COMMAND:                                PRESS <PF1>H FOR HELP  INSERT
```

In Screen Mode, whenever the DATA TYPE field is editable, a “popup” window appears, containing editable attributes of the field that are pertinent to its specific DATA TYPE field value. **DATE/TIME**-type fields do *not* have any required entries. You can accept all the default values within the “popup” window, simply by closing the window by pressing the **<F1>C** keys.



**NOTE:** To delete the entire field, enter an at-sign (@) at the “FIELD LABEL:” prompt.

### 10.8.3 SET OF CODES Field

In [Figure 159](#), the DATA TYPE field for the FLAGGED (#.61) field in the ORDER (#100) file has a value of **SET OF CODES**:

**Figure 159: Modify File Attributes Option—Editing a SET OF CODES Field in Screen Mode**

```

SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: FLAGGED
  
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 160](#):

**Figure 160: Modify File Attributes Option—Defining a DATA TYPE Field as SET OF CODES in Screen Mode**

```

FIELD #.61 IN FILE #100
FIELD LABEL: FLAGGED DATA TYPE... SET
-----
CODE: 0 WILL STAND FOR: NO
CODE: 1 WILL STAND FOR: YES
AUDIT CODE: WILL STAND FOR:
R CODE: WILL STAND FOR:
DEL CODE: WILL STAND FOR:
WR CODE: WILL STAND FOR:
CODE: WILL STAND FOR:
DESC CODE: WILL STAND FOR:
CODE: WILL STAND FOR:
CODE: WILL STAND FOR:
CODE: WILL STAND FOR:
HE CODE: WILL STAND FOR:
XECUT
-----
COMMAND: PRESS <PF1>H FOR HELP INSERT
  
```

In Screen Mode, whenever the DATA TYPE field is editable, a “popup” window appears, containing editable attributes of the field that are pertinent to its specific DATA TYPE field value. **SET OF CODES**-type fields require input of all the allowable “Internal” values (i.e., “CODE:” prompt), and their “External” equivalents (i.e., “WILL STAND FOR:” prompt).

Using the **<Arrow Up>** and **<Arrow Down>** keys makes it easy to edit the codes. It is permissible to leave a blank row, but every Internal Code (i.e., “CODE:” on the left) *must* have a corresponding External Code (“WILL STAND FOR:” on the right), and vice versa.

## 10.8.4 FREE TEXT Field

In [Figure 161](#), the DATA TYPE field for the REASON FOR FLAG (#.67) field in the ORDER (#100) file has a value of **FREE TEXT**:

**Figure 161: Modify File Attributes Option—Editing a FREE TEXT Field in Screen Mode**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: REASON FOR FLAG
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 162](#):

**Figure 162: Modify File Attributes Option—Defining a Data Type as FREE TEXT in Screen Mode**

```
FIELD #.67 IN FILE #100
FIELD LABEL: REASON FOR FLAG DATA TYPE... FREE TEXT
  MINIMUM LENGTH: 1
  MAXIMUM LENGTH: 80
AUDIT C| PATTERN MATCH (IN [X]): X[]?1P.E
  REA
DELETE ACCESS:
WRITE ACCESS:
SOURCE:
DESCRIPTION... TECHNICAL DESCRIPTION...

IS THIS FIELD MULTIPLE... NO

MANDATORY: NO
HELP-PROMPT: ENTER THE REASON FOR THE FLAG.
XECUTABLE HELP:

COMMAND: PRESS <PF1>H FOR HELP INSERT
```

In Screen Mode, whenever the DATA TYPE field is editable, a “popup” window appears, containing editable attributes of the field that are pertinent to its specific DATA TYPE field value. **FREE TEXT**-type fields require input for the “MINIMUM” and “MAXIMUM” lengths of the field. In this example, users *must* enter from **1 to 80 characters** for this field.

If included, the PATTERN MATCH *must* be written in M. For example, you can ensure that the data value does *not* start with a punctuation character by entering a PATTERN MATCH of:

```
X[?1P.E
```

This is a good check to make on a field that is allowed to be just **one character** in length. Sometimes, users can mistype and answer a field prompt with "/" (same key as ?) or some other meaningless punctuation character. A PATTERN MATCH check such as "X?1P.E" keeps that kind of mistake out of your database.

#### 10.8.4.1 Carets (^) in a FREE TEXT Field

If you are going to have carets (^) in a **FREE TEXT**-type field, it is advisable to create the field on a node by itself. You should create the field as usual, but when VA FileMan asks for the ^-PIECE POSITION, reply with **E1,<maximum length>**, as shown in [Figure 163](#).



**NOTE:** Putting multiple fields on the same node where some of the fields use the Em,n format may cause the inclusion of padding when displaying the data.

**Figure 163: Modify File Attributes Option—Carets (^) in a FREE TEXT Field: Piece Position**

```
SELECT FIELD: SPECIAL SITUATION
ARE YOU ADDING [SPECIAL SITUATION] AS A NEW FIELD (THE 3RD)? NO// Y <ENTER>
(YES)
FIELD NUMBER: 11// 50

DATA TYPE OF SPECIAL SITUATION: FREE TEXT
MINIMUM LENGTH: 3
MAXIMUM LENGTH: 200
(OPTIONAL) PATTERN MATCH (IN [X]):
WILL MY TRY FIELD BE MULTIPLE? NO// <ENTER> (NO)

SUBSCRIPT: 0// 50

The E1,200 means that the field occupies positions 1 through 200 of the node.

^-PIECE POSITION: 1// E1,200
IS MY TRY ENTRY MANDATORY (Y/N): NO// <ENTER> NO
....
[HELP]-PROMPT: ANSWER MUST BE 3-200 CHARACTERS IN LENGTH.
REPLACE <ENTER>
XECUTABLE [HELP]:
DESCRIPTION:
NO EXISTING TEXT
EDIT? NO//
```

## 10.8.5 WORD-PROCESSING Field

In [Figure 164](#), the DATA TYPE field for the ORDER TEXT (#.11) field in the ORDER (#100) file has a value of **WORD-PROCESSING**:

**Figure 164: Modify File Attributes Option—Editing a WORD-PROCESSING Field in Screen Mode**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: ORDER TEXT
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 165](#):

**Figure 165: Modify File Attributes Option—Defining a DATA TYPE Field as WORD-PROCESSING in Screen Mode**

```
FIELD #.01 IN SUB-FILE #772.02 OF FILE #772
FIELD LABEL: MESSAGE TEXT DATA TYPE... WORD-PROCESSING

SHALL THIS TEXT NORMALLY APPEAR IN WORD-WRAP MODE: YES
A | SHALL | CHARACTERS IN THIS TEXT BE TREATED LIKE ANY OTHER CHARACTERS: NO

WRITE ACCESS:
SOURCE:
DESCRIPTION... TECHNICAL DESCRIPTION...

IS THIS FIELD MULTIPLE... NO

MANDATORY: NO
HELP-PROMPT: THE TEXT OF THE INCOMING MESSAGES FOR THIS TRANSMISSION.
XECUTABLE HELP:

COMMAND: PRESS <PF1>H FOR HELP INSERT
```

In Screen Mode, whenever the DATA TYPE field is editable, a “popup” window appears, containing editable attributes of the field that are pertinent to its specific DATA TYPE field value. With **WORD-PROCESSING**-type fields. VA FileMan asks two questions in the “popup” window:

- “SHALL THIS TEXT NORMALLY APPEAR IN WORD-WRAP MODE:”  
If you answer **YES** to this question, text is automatically wrapped at word boundaries to fit in the column in which it is being printed. **Yes** is the default.
- “SHALL “|” CHARACTERS IN THIS TEXT BE TREATED LIKE ANY OTHER CHARACTERS:”  
If you answer **NO** to this question, the vertical bar (|) character is ignored. **No** is the default.



**TIP:** When it is important that lines of text be printed exactly as they were entered, answer:

- **NO** to the prompt, [Figure 165](#), “SHALL THIS TEXT NORMALLY APPEAR IN WORD-WRAP MODE:” question, and
- **YES** to the prompt, [Figure 165](#), “SHALL “|” CHARACTERS IN THIS TEXT BE TREATED LIKE ANY OTHER CHARACTERS:”

## 10.8.6 COMPUTED Field

In [Figure 166](#), the DATA TYPE field for the JUST RELEASED (#1000) field in the ORDER (#100) file has a value of **COMPUTED**:

**Figure 166: Modify File Attributes Option—Editing a COMPUTED Field in Screen Mode**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: JUST RELEASED
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 167](#):

**Figure 167: Modify File Attributes Option—Defining a DATA TYPE Field as COMPUTED in Screen Mode**

```
FIELD #1000 IN FILE #100
FIELD LABEL: JUST RELEASED DATA TYPE... COMPUTED
-----
COMPUTED-FIELD EXPRESSION:
A | #.68+2>TODAY
  |
  | TYPE OF RESULT: BOOLEAN
  | NUMBER OF FRACTIONAL DIGITS TO OUTPUT:
  | SHOULD VALUE ALWAYS BE ROUNDED:
  | WHEN TALLING, SHOULD SUMS BE SUMS OF COMPONENT FIELDS:
  | LENGTH OF FIELD: 3 POINT TO FILE:
-----
IS THIS FIELD MULTIPLE... NO

MANDATORY: NO
HELP-PROMPT:
XECUTABLE HELP:

COMMAND: PRESS <PF1>H FOR HELP INSERT
```

In Screen Mode, whenever the DATA TYPE field is editable, a “popup” window appears, containing editable attributes of the field that are pertinent to its specific DATA TYPE field value. With **COMPUTED**-type fields VA FileMan displays the characteristics of the computed expression in the “popup” window.

**i** **REF:** The syntax of these expressions is explained fully in the “[Computed Expressions](#)” section.

Field #.68 is the RELEASE DATE/TIME field. The JUST RELEASED field is **TRUE** if the RELEASE DATE/TIME was less than two days ago.

**i** **NOTE:** You can specify this virtual value to be **BOOLEAN**, **STRING-VALUED**, **DATE-VALUED**, or **NUMERIC**. Only in the last case are the three fields following the “TYPE OF RESULT” prompt editable.

## 10.8.7 POINTER TO A FILE Field

In [Figure 168](#), the DATA TYPE field for the WHO ENTERED (#3) field in the ORDER (#100) file has a value of **POINTER TO A FILE**:

**Figure 168: Modify File Attributes Option—Editing a POINTER TO A FILE Field in Screen Mode**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: WHO ENTERED
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 169](#):

**Figure 169: Modify File Attributes Option—Defining a DATA TYPE Field as POINTER TO A FILE in Screen Mode**

```
FIELD #13 IN FILE #100
FIELD LABEL: WHO ENTERED DATA TYPE... POINTER
-----
A | POINT TO WHICH FILE: NEW PERSON
  | SHALL [ ]ADDING A NEW FILE ENTRY ( [ ]LAYGO [ ] ) BE ALLOWED: NO
  |-----
WRITE ACCESS:
SOURCE:
DESCRIPTION... TECHNICAL DESCRIPTION...
IS THIS FIELD MULTIPLE... NO
MANDATORY: NO
HELP-PROMPT: ENTER THE NAME OF THE PERSON WHO ENTERED THIS ORDER.
EXECUTABLE HELP:
-----
COMMAND: PRESS <PF1>H FOR HELP INSERT
```

In Screen Mode, whenever the DATA TYPE field is editable, a “popup” window appears, containing editable attributes of the field that are pertinent to its specific DATA TYPE field value. With **POINTER TO A FILE**-type fields, VA FileMan asks you to enter the pointed-to file name in the “popup” window. The file that is pointed to (in this case, the NEW PERSON [#200] file) *must* already exist on your system. If you enter a ? (single question mark) at this prompt, you are presented with a list of the available files from which you can choose.

By answering **NO** to the “LAYGO” question, you ensure that users who are editing the “WHO ENTERED” data are *not* able to add a new entry on-the-fly to the NEW PERSON (#200) file.

### 10.8.8 VARIABLE-POINTER Field

In [Figure 170](#), the DATA TYPE field for the ITEM ORDERED (#7) field in the ORDER (#100) file has a value of **VARIABLE-POINTER**:

**Figure 170: Modify File Attributes Option—Editing a VARIABLE-POINTER Field in Screen Mode**

```

SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: ITEM ORDERED
  
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 171](#):

**Figure 171: Modify File Attributes Option—Defining a DATA TYPE Field as VARIABLE-POINTER in Screen Mode**

```

FIELD #7 IN FILE #100
FIELD LABEL: WHO ENTERED DATA TYPE... VARIABLE-POINTER
-----
VARIABLE-POINTER FILE #1: OPTION ORDER... 1
VARIABLE-POINTER FILE #2: LAB TEST ORDER... 2
AU VARIABLE-POINTER FILE #3: ORDER...
VARIABLE-POINTER FILE #4: ORDER...
VARIABLE-POINTER FILE #5: ORDER...
VARIABLE-POINTER FILE #6: ORDER...
-----
VARIABLE-POINTER #1
D MESSAGE: PROTOCOL
PREFIX: MISC
SHOULD USER BE ALLOWED TO ADD A NEW ENTRY: NO
SCREEN:
EXPLANATION OF SCREEN:
XE
-----
COMMAND: PRESS <PF1>H FOR HELP INSERT
  
```

In Screen Mode, whenever the DATA TYPE field is editable, a “popup” window appears, containing editable attributes of the field that are pertinent to its specific DATA TYPE field value. With **VARIABLE-POINTER**-type fields, VA FileMan asks you to enter the

pointed-to file name and its order in the first "popup" window. In this case, the first VARIABLE-POINTER file entered was the OPTION (#19) file, which would already have to exist. The ORDER was also set to one. When you press the **Enter** key at the "ORDER" prompt for each VARIABLE-POINTER, there is an additional "popup" window (i.e., a "popup window within the popup window"). The additional questions pertaining to the VARIABLE-POINTER file you are currently entering appear in this secondary "popup" window.

As you can see in this example, the OPTION (#19) file's MESSAGE, is associated with "PROTOCOL." Since its ORDER is one, it is the first file searched. The PREFIX "MISC" can also be used to refer to the OPTION (#19) file. Just as with the WHO ENTERED POINTER TO A FILE field (previously described), users *cannot* add new options to the OPTION (#19) file on-the-fly when they are entering an ITEM ORDERED in the ORDER file because the "SHOULD USER BE ALLOWED TO ADD A NEW ENTRY:" prompt is **NO**.

### 10.8.9 BOOLEAN Field

[Figure 172](#) shows the addition/editing of a field of data type **BOOLEAN**. There are no "popup" edits to perform for the **BOOLEAN** data type.



**NOTE:** To delete the entire field, enter an at-sign (@) at the "FIELD LABEL:" prompt.

**Figure 172: Addition/Editing of a Field of Data Type BOOLEAN**

```

FIELD #12 IN FILE #123000001
FIELD LABEL: BOOLEAN          DATA TYPE... BOOLEAN

      TITLE:
      AUDIT:
AUDIT CONDITION:
      READ ACCESS:
      DELETE ACCESS:
      WRITE ACCESS:
      SOURCE:
DESCRIPTION...      TECHNICAL DESCRIPTION...

                        IS THIS FIELD MULTIPLE... NO

      MANDATORY: NO
      HELP-PROMPT:
XECUTABLE HELP:

COMMAND:                                PRESS <F1>H FOR HELP INSERT

```

## 10.8.10 LABEL REFERENCE Field

Since the LABEL REFERENCE field can include a caret (^), VA FileMan using extract storage instead of piece storage and uses "E1,99" as the default at the PIECE POSITION prompt. It is advisable to create the field on a node by itself.

**i** **NOTE:** Putting multiple fields on the same node where some of the fields use the Em,n format can cause the inclusion of padding when displaying the data.

[Figure 173](#) shows the addition/editing of a field of data type **LABEL REFERENCE**. The **LABEL REFERENCE** data type has one "popup" question, which asks the developer if "PARAMETERS ALLOWED:" with the **LABEL REFERENCE** entry.

**i** **NOTE:** To delete the entire field, enter an at-sign (@) at the "FIELD LABEL:" prompt.

**Figure 173: Addition/Editing of a Field of Data Type LABEL REFERENCE**

```
FIELD #13 IN FILE #123000001
FIELD LABEL: LABEL REFERENCE          DATA TYPE... LABEL REFERENCE
TITLE:
AUDIT:
AUDIT CONDITION:
READ ACCESS:
DELETE ACCESS:
WRITE ACCESS:
SOURCE:
DESCRIPTION...      TECHNICAL DESCRIPTION...
IS THIS FIELD MULTIPLE... NO
MANDATORY: NO
HELP-PROMPT:
EXECUTABLE HELP:
PARAMETERS ALLOWED: YES
COMMAND:                PRESS <F1>H FOR HELP  INSERT
```

## 10.8.11 TIME Field

Figure 174 shows the addition/editing of a field of data type **TIME**. The **TIME** data type has one "popup" question, which asks the developer if "SECONDS ALLOWED: YES//"  
with the **TIME** entry.



**NOTE:** To delete the entire field, enter an at-sign (@) at the "FIELD LABEL:" prompt.

**Figure 174: Addition/Editing of a Field of Data Type TIME**

```
FIELD #14 IN FILE #123000001
FIELD LABEL: TIME          DATA TYPE... TIME
TITLE:
AUDIT:
AUDIT CONDITION:
READ ACCESS:
DELETE ACCESS:
WRITE ACCESS:
SOURCE:
DESCRIPTION...           TECHNICAL DESCRIPTION...
IS THIS FIELD MULTIPLE... NO
MANDATORY: NO
HELP-PROMPT:
XECUTABLE HELP:
SECONDS ALLOWED: YES//
COMMAND:                  PRESS <F1>H FOR HELP INSERT
```

## 10.8.12 YEAR Field

Figure 175 shows the addition/editing of a field of data type **YEAR**. The **YEAR** data type has one "popup" question, which asks the developer the "EARLIEST DATE:" that is allowed with the **YEAR** entry.



**NOTE:** To delete the entire field, enter an at-sign (@) at the "FIELD LABEL:" prompt.

**Figure 175: Addition/Editing of a Field of Data Type YEAR**

FIELD LABEL: YEAR	FIELD #15 IN FILE #123000001	DATA TYPE... <b>YEAR</b>
TITLE:		
AUDIT:		
AUDIT CONDITION:		
READ ACCESS:		
DELETE ACCESS:		
WRITE ACCESS:		
SOURCE:		
DESCRIPTION...	TECHNICAL DESCRIPTION...	
	IS THIS FIELD MULTIPLE... NO	
MANDATORY: NO		
HELP-PROMPT:		
XECUTABLE HELP:		
<hr/>		
EARLIEST DATE:	<b>1/1/2017</b>	
COMMAND:	PRESS <F1>H FOR HELP	<b>INSERT</b>

## 10.8.13 UNIVERSAL TIME Field

Figure 176 shows the addition/editing of a field of data type **UNIVERSAL TIME**. There are no “popup” edits to perform for the **UNIVERSAL TIME** data type.



**NOTE:** To delete the entire field, enter an at-sign (@) at the “FIELD LABEL:” prompt.

**Figure 176: Addition/Editing of a Field of Data Type UNIVERSAL TIME**

FIELD LABEL: UTC	FIELD #8 IN FILE #123000001	DATA TYPE... <b>UNIVERSAL TIME</b>
TITLE:		
AUDIT:		
AUDIT CONDITION:		
READ ACCESS:		
DELETE ACCESS:		
WRITE ACCESS:		
SOURCE:		
DESCRIPTION...	TECHNICAL DESCRIPTION...	
	IS THIS FIELD MULTIPLE... NO	
MANDATORY: NO		
HELP-PROMPT:		
XECUTABLE HELP:		
COMMAND:	PRESS <F1>H FOR HELP	<b>INSERT</b>

## 10.8.14 FT POINTER Field

[Figure 177](#) shows the addition/editing of a field of data type **FT POINTER**. The **FT POINTER** data type asks the developer to enter an open global root to the pointed-to file ("POINTER:" prompt). It also asks if adding new entries are allowed ("LAYGO: YES//"  
prompt).



**NOTE:** To delete the entire field, enter an at-sign (@) at the "FIELD LABEL:" prompt.

**Figure 177: Addition/Editing of a Field of Data Type FT POINTER**

```
FIELD #9 IN FILE #123000001
FIELD LABEL: FT POINTER          DATA TYPE... FT POINTER

    TITLE:
    AUDIT:
AUDIT CONDITION:
    READ ACCESS:
    DELETE ACCESS:
    WRITE ACCESS:
    SOURCE:
DESCRIPTION...      TECHNICAL DESCRIPTION...

                        IS THIS FIELD MULTIPLE... NO

    MANDATORY: NO
    HELP-PROMPT: ENTER THE FREE TEXT POINTER
XECUTABLE HELP:

-----
POINTER: VA(200,
LAYGO: YES//

COMMAND:                PRESS <F1>H FOR HELP  INSERT
```

## 10.8.15 FT DATE Field

[Figure 178](#) shows the addition/editing of a field of data type **FT DATE**. There are several "popup" edits to perform for the **FT DATE** data type.

- EARLIEST DATE: The earliest possible date allowed.
- IMPRECISE DATE: Entering "YES" indicates that the date does *not* require a day of the month.
- TIME OF DAY: Entering "YES" indicates that a Time value can be entered.
- TIME REQUIRED: Entering "YES" indicates that a Time value *must* be entered.
- SECONDS ALLOWED: Entering "YES" indicates that a seconds can be entered with the time value.



**NOTE:** To delete the entire field, enter an at-sign (@) at the "FIELD LABEL:" prompt.

**Figure 178: Addition/Editing of a Field of Data Type FT DATE**

```
FIELD #10 IN FILE #123000001
FIELD LABEL: FT DATE          DATA TYPE... FT DATE
TITLE:
AUDIT:
AUDIT CONDITION:
READ ACCESS:
DELETE ACCESS:
WRITE ACCESS:
SOURCE:
DESCRIPTION...              TECHNICAL DESCRIPTION...
IS THIS FIELD MULTIPLE... NO
MANDATORY: NO
HELP-PROMPT:
XECUTABLE HELP:
-----
EARLIEST DATE: 1/1/2017
IMPRECISE DATE: NO//
TIME OF DAY: YES//
TIME REQUIRED: YES
SECONDS ALLOWED: YES//
COMMAND:                    PRESS <F1>H FOR HELP INSERT
```

## 10.8.16 RATIO Field

Figure 179 shows the addition/editing of a field of data type **RATIO**. The **RATIO** data type asks the developer to enter MINIMUM and MAXIMUM values for each numeric value in the **RATIO**.



**NOTE:** To delete the entire field, enter an at-sign (@) at the "FIELD LABEL:" prompt.

**Figure 179: Addition/Editing of a Field of Data Type RATIO**

```
FIELD #11 IN FILE #123000001
FIELD LABEL: RATIO          DATA TYPE... RATIO
TITLE:
AUDIT:
AUDIT CONDITION:
  READ ACCESS:
  DELETE ACCESS:
  WRITE ACCESS:
SOURCE:
DESCRIPTION...             TECHNICAL DESCRIPTION...
                             IS THIS FIELD MULTIPLE... NO
MANDATORY: NO
HELP-PROMPT: ENTER THE RATIO
XECUTABLE HELP:
-----
LEFT SIDE MINIMUM: (0-9999): 1
LEFT SIDE MAXIMUM: (1-9999): 10
RIGHT SIDE MINIMUM: (1-9999): 1
RIGHT SIDE MAXIMUM: (1-9999): 20
COMMAND:                PRESS <F1>H FOR HELP  INSERT
```

## 10.8.17 Creating a Multiple

Figure 180 illustrates creating a Multiple field. This example simulates creating the RESPONSES (#4.5) field in the ORDER (#100) file. It has a data type of **NUMERIC**:

**Figure 180: Modify File Attributes Option—Creating a Multiple in Screen Mode**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: RESPONSES
ARE YOU ADDING RESPONSES AS A NEW FIELD? NO// Y <ENTER> (YES)
FIELD NUMBER: 4.5// <ENTER>
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in Figure 181:

**Figure 181: Modify File Attributes Option—Defining a DATA TYPE Field as a NUMERIC Multiple in Screen Mode**

```
FIELD LABEL: RESPONSES          FIELD #4.5 IN FILE #100
                                DATA TYPE... NUMERIC

    TITLE:
    AUDIT:
AUDIT CONDITION:
    READ ACCESS:
    DELETE ACCESS:
    WRITE ACCESS:
    SOURCE:
DESCRIPTION...      TECHNICAL DESCRIPTION...

                                IS THIS FIELD MULTIPLE... YES

| SHOULD USER SEE AN ADDING A NEW ENTRY MESSAGE: NO
| HAVING ENTERED OR EDITED ONE MULTIPLE, SHOULD USER BE ASKED ANOTHER: NO

COMMAND:                                PRESS <PF1>H FOR HELP  INSERT
```

A Multiple field is created just as any other, except that the “IS THIS FIELD MULTIPLE...” question is answered **YES**. In Screen Mode, after answering **YES** to this question, a “popup” window appears containing editable attributes pertinent to a Multiple field. With Multiple-type fields, VA FileMan asks if you want users to be notified when they are adding new entries and if users should be asked if they want to make another entry. In this case, the user answered **NO** to both questions.

### 10.8.17.1 Subfields

To create or edit Subfields of a Multiple field, select a Multiple-valued field (e.g., the RESPONSES Multiple field, created in Section [10.8.17](#)):

**Figure 182: Modify File Attributes Option—Editing a Multiple’s Subfield in Screen Mode**

```
SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: ORDER// <ENTER>

SELECT FIELD: RESPONSES <ENTER> (MULTIPLE)
```

You are taken into a ScreenMan form where you can edit the properties of the field, as shown in [Figure 183](#):

**Figure 183: Modify File Attributes Option—Reviewing/Editing the Properties of a Multiple Data Type Field in Screen Mode**

```
MULTIPLE FIELD #4.5 IN FILE #100

MULTIPLE-FIELD LABEL: RESPONSES
  READ ACCESS:
  WRITE ACCESS:
  SOURCE:

COMMAND: PRESS <PF1>H FOR HELP INSERT
```

In Screen Mode, after entering a Multiple field, a special screen appears that displays information about the Multiple as a whole.



**NOTE:** If you wanted to delete the entire Multiple field, you would enter an at-sign (@) at the “MULTIPLE-FIELD LABEL” prompt.

After viewing this screen, you can proceed to add fields to the Multiple or to edit existing Subfields. This is done at the “Select <FIELD NAME> SUB-FIELD:” prompt that is displayed when you exit Screen Mode (see [Figure 184](#)).

### 10.8.17.2 Numeric Subfield

After selecting a Multiple-valued field (e.g., the RESPONSES Multiple field in the ORDER [#100] file), you can enter or modify the Multiple's subfields by entering the field's number or name (label) at the "Select <FIELD NAME> SUB-FIELD:" prompt (where <FIELD NAME> represents the name of the Multiple field).

A **.01** field with the same name as the Multiple field was added automatically when the field was identified as a Multiple. The **.01** field is the identifying key for an entry in the Subfile (Multiple); it is like a file's **.01** field, which is the file's identifying key. In [Figure 184](#), the **.01** subfield is edited to have a DATA TYPE of **NUMERIC**:

**Figure 184: Modify File Attributes Option—Example of a .01 Subfield of a Multiple**

```
SELECT RESPONSES SUB-FIELD: .01 <ENTER> ITEM ENTRY
```

You are taken into a ScreenMan form where you can edit the properties of the subfield, as shown in [Figure 185](#):

**Figure 185: Modify File Attributes Option—Defining a Data Type Field as a NUMERIC Subfield in Screen Mode**

```
FIELD #.01 IN SUB-FILE #100.045 OF FILE #100
FIELD LABEL: ITEM ENTRY          DATA TYPE... NUMERIC

-----
AU | INCLUSIVE LOWER BOUND: 1
   | INCLUSIVE UPPER BOUND: 9999999
   | IS THIS A DOLLAR AMOUNT: NO
   | MAXIMUM NUMBER OF FRACTIONAL DIGITS: 0
-----

SOURCE:
DESCRIPTION...      TECHNICAL DESCRIPTION...

IS THIS FIELD MULTIPLE... YES

MANDATORY: NO
HELP-PROMPT: TYPE A NUMBER BETWEEN 1 AND 9999999, 0 DECIMAL DIGITS
EXECUTABLE HELP:

COMMAND:                                PRESS <PF1>H FOR HELP  INSERT
```

In Screen Mode, while editing a sub-field of a Multiple, you notice that the heading at the top of the screen reminds you that you are now editing a field within a Multiple (e.g., "Field #.01 in Sub-File #100.045 of File #100").

Also, whenever the DATA TYPE field is editable, a “popup” window appears, containing editable attributes of the field that are pertinent to its specific DATA TYPE field value. With **NUMERIC**-type fields, VA FileMan asks you to enter the following:

- INCLUSIVE LOWER BOUND (e.g., set to **1**)
- INCLUSIVE UPPER BOUND (e.g., set to **9999999**).
- IS THIS A DOLLAR AMOUNT: You are asked if the numeric value is a dollar amount (e.g., set to **NO**).
- MAXIMUM NUMBER OF FRACTIONAL DIGITS: If decimal digits are allowed (e.g., set to **0**).

A default help prompt is automatically written for you with the DATA TYPE field of **NUMERIC**. In this case, the following English message was built automatically based on the specifications entered by the user:

**TYPE A NUMBER BETWEEN 1 AND 9999999, 0 DECIMAL DIGITS**

As always, you can accept the default help prompt or change it using the **Replace ... With** syntax.



**NOTE:** This help information is displayed when the user inputs a single question mark (?) when editing this field.

# 11 File Utilities

Various file utilities are provided as options on the **VA FileMan's Utility Functions** [DIUTILITY] menu.



**NOTE:** Some additional functionality for modifying files is contained in the separate **Modify File Attributes** [DIMODIFY] option, which is located on the main **VA FileMan** [DIUSER] menu.

## 11.1 Verify Fields

The **Verify Fields** [DIVERIFY] option uses a field's definition to verify the data stored in a file. After invoking this option, you can ask to verify all existing values of a particular field by entering its label at the "VERIFY WHICH FIELD:" prompt; or you can ask that *all* fields at a given file level be verified by entering **ALL** at the prompt.

In addition to checking the validity of the data stored in a file, Verify Fields checks the cross-references on the file's data. Among the items checked for are:

- Dangling pointers in cross-references.
- Cross-reference values exceeding **30 characters** if a field's Label has been translated into the user's language, the report of errors for that field will show the translated Label.

If more than one discrepancy is found between the current definition and the data on file, you are asked if you want to save the list of those entries containing the inconsistent data in a template. Later, you would be able to "SORT BY:" the entries in this template to display or edit them.



**REF:** For information on how to execute or avoid executing any part of the **INPUT** transform when the **Verify Fields** [DIVERIFY] option is being run, see the "Input Transform" section in the "Advanced File Definition" section in the *VA FileMan Developer's Guide*.

**NOTE:** Some parts of a field's **INPUT** transform (whose main purpose is to validate data as a user enters it) may be inappropriate when being executed in the context of the **Verify Fields** [DIVERIFY] option.

**NOTE:** The “DISPLAY OPTION?” field for the **Verify Fields** [DIVERIFY] option *must* be set to “**YES**” to display a final “Press RETURN to continue” at the end of the report.

## 11.2 Cross-Reference a Field or File

### Traditional Cross-References:

- Types of Traditional Cross-references
- Edit a Traditional Cross-reference
- Create a Traditional Cross-reference
- Delete a Traditional Cross-reference

### New-Style Cross-References:

- Edit a New-Style Cross-reference
- Create a New-Style Cross-reference
- Delete a New-Style Cross-reference

There are seven types of Traditional cross-references and two types of New-Style cross-references available. Generally, a cross-reference in VA FileMan specifies that some action is performed when the field’s value is entered, changed, or deleted. For several types of cross-references, the action consists of putting the value into a list (i.e., an index used when looking up an entry or when sorting). The regular cross-reference is used for sorting and for lookup; you can limit it to sorting only. The Key Word in Context (KWIC), mnemonic, and SOUNDEX cross-references are also used for lookup.

You can sort a file on any field (except a **WORD-PROCESSING**-type field) whether a cross-reference exists for the field. However, sorting is done more quickly and efficiently if a regular cross-reference exists on the field.

When a file is created, a Traditional cross-reference on the NAME (#.01) field is automatically established. You can add or delete cross-references at any time using the **Cross-Reference a Field or File** [DIXREF] option located on the **Utility Functions** [DIUTILITY] menu. This option can also be used to enter a description of a cross-reference and to prevent the cross-reference from being deleted.


You can create a cross-reference on a Multiple field, a Multiple’s subfields, or on any other field type except a **WORD-PROCESSING**-type field. For example, the PATIENT

(#2) file contains the AGE AT ONSET subfield in the DIAGNOSIS Multiple. If you create a regular cross-reference for a field in a Multiple, you can choose in what context the cross-reference is used. You might want to cross-reference the whole file by AGE AT ONSET (so that a report sorted by AGE AT ONSET could be produced efficiently). Alternately, you might want to cross-reference only an individual patient's diagnoses by onset age (so that a lookup of diagnosis could be done using AGE AT ONSET).

## 11.2.1 Types of Traditional Cross-references

**Table 92: File Utilities—Traditional Cross-references**

Cross-reference	Description
REGULAR	The field value is sorted and stored in the cross-reference. The regular cross-reference is used for sorting. It can also be used when looking up entries. The cross-reference that is automatically created on the NAME (#.01) field when a file is created is a regular cross-reference; this is the "B" cross-reference.
KWIC	Key Word in Context (i.e., each word of three or more letters in the field value becomes a separate cross-reference). A space is considered the primary word separator. For example, <b>ONE FMUSER</b> can be looked up under either <b>ONE</b> or <b>FMUSER</b> . Uppercase or lowercase two-letter words (e.g., <b>IN, AN, OR, and IS</b> are <i>not</i> considered key text. The words <b>THE, AND, THEN, FOR, FROM, OTHER, THAN, WITH, THEIR, SOME, and THIS</b> (upper- or lowercase) are <i>not</i> considered key text. Quotation marks are also <i>not</i> considered key text.  You can also specify that KWIC separates words at most punctuation marks except quotation marks (e.g., <b>ONE-FMUSER, ONE/FMUSER</b> , etc., are found with <b>FMUSER</b> ). A list of punctuation marks is presented for your selection.
MNEMONIC	The field's values are cross-referenced along with the NAME (#.01) field cross-reference (e.g., the MAIDEN NAME field's values are found along with NAME values in any lookup). Typically, the cross-reference on the NAME field is searched first when doing a lookup.
MUMPS	Those with <b>Programmer</b> access can create special cross-references by putting M code into the <b>SET</b> and <b>KILL</b> logic of a cross-reference. You can use the M code entered to accomplish any task that <i>must</i> be done when the value in a field is entered, changed, or deleted.

Cross-reference	Description
SOUNDEX	The field's value is transformed into a <b>four-character</b> string representing its phonetic properties. That string becomes the cross-reference. For example, soundex transformation would access <b>GONZALEZ, GONZELES, Gonzales, and Gonsalless</b> as equivalents; entry of any one of these forms looks up all the others automatically.
TRIGGER	Whenever the field is updated, a different field can be automatically updated at the same time.   <b>REF:</b> For more details, see the "Trigger Cross-References" section in the <i>VA FileMan Developer's Guide</i> .
BULLETIN	Whenever a field is updated, a MailMan message is sent notifying specified users that an update has occurred. The Bulletin cross-reference is only available when VA FileMan is installed with MailMan.

## 11.2.2 Edit a Traditional Cross-reference

To edit a Traditional cross-reference, identify the field or subfield you wish to edit. VA FileMan displays the type of cross-references on the field and offers you the choices of Edit, Delete, or Create. Select **Edit** at this prompt and you can edit or add a No Deletion message and enter a description of the cross-reference.

**Figure 186: File Utilities—Editing a Traditional Cross-Reference (1 of 2)**

```

SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY FUNCTIONS OPTION: CRO <ENTER> SS-REFERENCE A FIELD OR FILE

WHAT TYPE OF CROSS-REFERENCE (TRADITIONAL OR NEW)? TRADITIONAL// <ENTER>

MODIFY WHAT FILE: TEST

SELECT FIELD: .01 <ENTER> NAME

CURRENT CROSS-REFERENCE IS REGULAR  INDEX OF FILE

CHOOSE E (EDIT)/D (DELETE)/C (CREATE): E

NO DELETION MESSAGE: NO, DO NOT DELETE THIS X-REFI

```

This **FREE TEXT** message indicates that the cross-reference *cannot* be deleted. If a message is retained, the cross-reference *cannot* be deleted. The user sees this message whenever an attempt is made to delete the cross-reference.

**Figure 187: File Utilities—Editing a Traditional Cross-Reference (2 of 2)**

```
DESCRIPTION:  
1>USED FOR LOOKUP ON AND SORTING BY NAME.  
2><ENTER>
```

The description appears in a standard DD listing.



**TIP:** It is important to describe cross-references that are unusual or especially critical. Consider describing all MUMPS, trigger, and bulletin cross-references.

### 11.2.3 Create a Traditional Cross-reference

If you would like to create a Traditional cross-reference for a field, proceed in manner shown in [Figure 188](#):

**Figure 188: File Utilities—Creating a Traditional Cross-Reference**

```
SELECT OPTION: UTI <ENTER> LITY FUNCTIONS  
SELECT UTILITY FUNCTIONS OPTION: CRO <ENTER> SS-REFERENCE A FIELD OR FILE  
  
WHAT TYPE OF CROSS-REFERENCE (TRADITIONAL OR NEW)? TRADITIONAL// <ENTER>  
  
MODIFY WHAT FILE: TEST  
  
SELECT FIELD: 1 <ENTER> DATE  
  
NO CURRENT CROSS-REFERENCE  
WANT TO CREATE A NEW CROSS-REFERENCE FOR THIS FIELD? NO// YES  
  
CROSS-REFERENCE NUMBER: 1// <ENTER>  
  
SELECT TYPE OF INDEXING: REGULAR// <ENTER>  
WANT CROSS-REFERENCE TO BE USED FOR LOOKUP AS WELL AS FOR SORTING?  
YES// <ENTER>  
  
NO DELETION MESSAGE: <ENTER>  
DESCRIPTION:  
1>LOOKUP AND SORTING CAN BE DONE BY DATE USING THIS REGULAR  
2>CROSS-REFERENCE.  
3><ENTER>
```

## 11.2.4 Delete a Traditional Cross-reference

Figure 189 shows how to delete a Traditional cross-reference:

Figure 189: File Utilities—Deleting a Traditional Cross-Reference

```
SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY FUNCTIONS OPTION: GRO <ENTER> SS-REFERENCE A FIELD OR FILE

WHAT TYPE OF CROSS-REFERENCE (TRADITIONAL OR NEW)? TRADITIONAL// <ENTER>

MODIFY WHAT FILE: TEST

SELECT FIELD: 1 <ENTER> DATE

CURRENT CROSS-REFERENCE IS REGULAR  INDEX OF FILE

CHOOSE E (EDIT)/D (DELETE)/C (CREATE): D
ARE YOU SURE THAT YOU WANT TO DELETE THE CROSS-REFERENCE? NO// YES <ENTER> ...OK
```

## 11.2.5 New-Style Cross-References

Two types of New-Style cross-references are available: Regular and MUMPS. They are like their Traditional cross-reference counterparts, but New-Style cross-references offer some unique advantages:

- **Compound Cross-references**—You can create not only *simple cross-references* that are based on a single field, but *compound cross-references*, cross-references that are based on more than one field in a file. For example, in a regular New-Style “C” index you can store both the Name and ID Number of a record as subscripts in a single index:

```
^DIZ(1000, , , FMPATIENT, 25, A56789, 14) =
^DIZ(1000, , , FMPATIENT, 10, D1234, 5) =
```

To create this kind of index with a Traditional cross-reference, you would have to create two MUMPS-type cross-references, one on the NAME field and one on the ID Number field. New-Style cross-references allow you to define this compound cross-reference once as a regular index that VA FileMan can use for lookup and sorting.

- **Field- or Record-Level Execution**—Since a Traditional cross-reference is defined on a particular field, the action associated with that cross-reference is performed whenever the field is edited. With New-Style cross-references, you can specify that the action associated with a cross-reference is performed only once after the entire record has been edited, typically at the end of the editing session. Record-level execution would normally be selected for compound cross-references.

If the “**C**” index in the example above were defined using Traditional MUMPS-type cross-references, and both the NAME and ID Number fields were contained in a single **INPUT** template, the index would be updated when the NAME field was edited, and then again when the ID Number was edited. But if the cross-reference were defined as a New-Style compound index with record-level execution, the index would be updated only once after the entire record was edited, after changes to both the NAME and ID Number fields had been completed.

- **Code to Kill the Entire Index**—This is code that VA FileMan can execute to remove an entire index from a file. This can make re-indexing a file much more efficient. To delete an index, VA FileMan can execute the *Kill Entire Index Code*, instead of looping through all the record in a file and removing each record’s index one at a time.
- **Activity**—New-Style cross-references can have an Activity of “**R**” and/or “**I**” to allow you to control whether the cross-reference should be fired during **Reindexing** and/or **Installation (KIDS)**. If you call **IX^DIK**, **IX1^DIK**, or **IXALL^DIK** or if you select the **Re-Index File [DIRDEX]** option located on VA FileMan’s **Utility Functions** menu [**DIUTILITY**] to re-index all cross-references, only those New-Style cross-references that contain an “**R**” in Activity are fired.

If you explicitly select a cross-reference in an **EN^DIK**, **EN1^DIK**, or **ENALL^DIK** call or in the **Re-Index File [DIRDEX]** option on VA FileMan’s **Utility Functions [DIUTILITY]** menu, that cross-reference is fired regardless of its Activity. Also, when a field is edited, VA FileMan ignores Activity and fires all cross-references on that field; though, you can control whether a cross-reference is fired by entering Set and Kill Conditions.



**REF:** For more information on the **Re-Index File [DIRDEX]** option and limiting re-indexing on some files, see the “[Re-Index File Option](#)” and “[Limits on Reindexing Files](#)” sections.

- **Collation**—You can specify forwards or backwards collation, the direction in which VA FileMan’s lookup utilities loop through a subscript in an index when entries are returned or displayed to the user. This is especially useful for dates. Developers can store dates in their natural internal VA FileMan date format and still display entries in the date index in reverse date order.
- **Lookup Prompt**—Each subscript on an index in the **INDEX (#.11)** file can be assigned a **LOOKUP PROMPT**. This prompt is used as the prompt for entry of the

lookup value during classic VA FileMan lookup ^**DIC** calls. If *not* filled in, VA FileMan defaults to use the name of the field for that subscript value, if there is one.

- **Computed Values**—Those with **Programmer** access can have any value in the cross-reference be computed; the value is determined from M code that sets the variable **X**.
- **Subscript Transforms**—Those with **Programmer** access can define the following on subscripts in an index:
  - **Transform for Storage**—Code that transforms the internal value of a field before it is stored as a subscript in the index.
  - **Transform for Display**—Code that transforms the value stored in the index back to a form that can be displayed to the user.
- **SET and Kill Conditions**—Those with **Programmer** access can enter M code that specifies whether the **SET** or **KILL** logic is fired. The M code sets the variable **X** to Boolean true only if the logic should be executed. The “before” and “after” values are available in the **X**, **X1**, and **X2** arrays (see [Table 93](#)).
- **The X, X1, and X2 Arrays**—Those with **Programmer** access can reference the **X**, **X1**, and **X2** arrays in the **SET** and **KILL** logic and the **SET** and **KILL** conditions of New-Style cross-references. When a field is edited and the cross-reference logic is executed, the field’s corresponding **X1** array element contains the old value of the field, the **X2** array element contains the new value of the field, and the **X** array element contains either the old or new value, depending on whether the **SET** logic, **SET** condition, **KILL** logic, or **KILL** condition is being executed:

**Table 93: File Utilities—X, X1, and X2 Arrays**

<b>Array</b>	<b>Value in KILL Logic/KILL Condition</b>	<b>Value in SET Logic/SET Condition</b>
<b>X(order#)</b>	Old value	New value
<b>X1(order#)</b>	Old value	Old value
<b>X2(order#)</b>	New value	New value

The variables **X**, **X1**, and **X2** always equal **X(1)**, **X1(1)**, and **X2(1)**, respectively.

If an order number in the cross-reference refers to the **.01** field, **X1(order#)** is set to **NULL** when the **SET** logic and **SET** condition are executed during record creation. Similarly, **X2(order#)** is set to **NULL** when the **KILL** logic or condition are executed during record deletion.

- **Key Support**—A regular New-Style index can be used as the Uniqueness Index for a key. VA FileMan ensures that all fields in a Uniqueness Index have values (are *not* **NULL**), and that those values, taken collectively, are unique across all records in the file.



**REF:** For more information on keys and how to create them, see the “[Key Definition](#)” section.

## 11.2.6 Edit a New-Style Cross-reference

To edit a New-Style cross-reference, identify the file or subfile you wish to edit. VA FileMan displays the cross-references on the file and offers you the choices of Edit, Delete, or Create:

**Figure 190: File Utilities—Editing a New-Style Cross-Reference**

```
SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY FUNCTIONS OPTION: CRO <ENTER> SS-REFERENCE A FIELD OR FILE

WHAT TYPE OF CROSS-REFERENCE (TRADITIONAL OR NEW)? TRADITIONAL// NEW
MODIFY WHAT FILE: TEST
SELECT SUBFILE: <ENTER>

CURRENT INDEXES ON FILE #16026:
 75   [ ]COMP[ ] INDEX
 85   [ ]XR202[ ] INDEX
106   [ ]H[ ] INDEX
116   [ ]AC[ ] INDEX
141   [ ]C[ ] INDEX

CHOOSE E (EDIT)/D (DELETE)/C (CREATE): EDIT

WHICH INDEX DO YOU WISH TO EDIT? C
```



**NOTE:** The numbers displayed to the left of each cross-reference are the Internal Entry Number of the cross-reference stored in the INDEX (#.11) file.

You are then taken into a ScreenMan form where you can edit the properties of the New-Style cross-reference, as shown in [Figure 191](#):

**Figure 191: File Utilities—Editing a New-Style Cross-Reference in Screen Mode**

```
NUMBER: 106                                EDIT AN INDEX                                PAGE 1 OF 2
-----
      FILE: 16026                            ROOT FILE: 16026
      INDEX NAME: H                          ROOT TYPE: INDEX FILE

SHORT DESCRIPTION: TEST
DESCRIPTION (WP):  (EMPTY)

      TYPE: REGULAR


      ACTIVITY: IR
      EXECUTION: FIELD

      USE: LOOKUP & SORTING

      Alternatively, this field can be NULL/blank, see Figure 196.

      DO NOT REINDEX: NO RE-INDEXING ALLOWED

COMMAND:                                     PRESS <PF1>H FOR HELP  INSERT
```

 **NOTE:** For additional help, enter a single question mark (?) or two question marks (??) at any prompt.

## 11.2.7 Create a New-Style Cross-Reference

To create a New-Style cross-reference, proceed in the manner shown in [Figure 192](#):

**Figure 192: File Utilities—Creating a New-Style Cross-Reference**

```
SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY FUNCTIONS OPTION: GRO <ENTER> SS-REFERENCE A FIELD OR FILE

WHAT TYPE OF CROSS-REFERENCE (TRADITIONAL OR NEW)? TRADITIONAL// NEW
MODIFY WHAT FILE: TEST
SELECT SUBFILE: <ENTER>

CURRENT INDEXES ON FILE #16026:
75   [ ]COMP[ ] INDEX
85   [ ]XR202[ ] INDEX
106  [ ]H[ ] INDEX
116  [ ]AC[ ] INDEX
141  [ ]C[ ] INDEX

CHOOSE E (EDIT)/D (DELETE)/C (CREATE): CREATE
WANT TO CREATE A NEW INDEX FOR THIS FILE? NO// YES

TYPE OF INDEX: REGULAR// <ENTER>

WANT INDEX TO BE USED FOR LOOKUP & SORTING
OR SORTING ONLY: LOOKUP & SORTING// <ENTER>

INDEX NAME: J// <ENTER>
```



**NOTE:** The numbers displayed to the left of each cross-reference are the Internal Entry Number of the cross-reference stored in the INDEX (#.11) file.

You are then taken into a ScreenMan form where you can edit the properties of the New-Style cross-reference, as shown in [Figure 193](#):

**Figure 193: File Utilities—Creating a New-Style Cross-Reference in Screen Mode**

```
NUMBER: 142                                EDIT AN INDEX                                PAGE 1 OF 2
-----
      FILE: 16026                            ROOT FILE: 16026
      INDEX NAME: J                          ROOT TYPE: INDEX FILE

SHORT DESCRIPTION: TEST
DESCRIPTION (WP): (EMPTY)

      TYPE: REGULAR

      ACTIVITY: IR
      EXECUTION: FIELD

      USE: LOOKUP & SORTING

      Alternatively, this field can indicate "NO RE-INDEXING ALLOWED", see Figure 194.

      DO NOT REINDEX:

COMMAND:                                     PRESS <PF1>H FOR HELP  INSERT
```



**NOTE:** For additional help, enter a single question mark (?) or two question marks (??) at any prompt.

## 11.2.8 Delete a New-Style Cross-Reference

Figure 194 shows you how to delete a New-Style cross-reference:

**Figure 194: File Utilities—Deleting a New-Style Cross-Reference**

```
SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY FUNCTIONS OPTION: GRO <ENTER> SS-REFERENCE A FIELD OR FILE

WHAT TYPE OF CROSS-REFERENCE (TRADITIONAL OR NEW)? TRADITIONAL// NEW
MODIFY WHAT FILE: TEST
SELECT SUBFILE: <ENTER>

CURRENT INDEXES ON FILE #16026:
 75  [ ]COMP[ ] INDEX
 85  [ ]XR202[ ] INDEX
106  [ ]H[ ] INDEX
116  [ ]AC[ ] INDEX
141  [ ]C[ ] INDEX

CHOOSE E (EDIT)/D (DELETE)/C (CREATE): DELETE

WHICH INDEX DO YOU WISH TO DELETE? 141 <ENTER> C
ARE YOU SURE YOU WANT TO DELETE THE INDEX? NO// YES

INDEX DEFINITION DELETED.
REMOVING OLD INDEX ... DONE!

PRESS RETURN TO CONTINUE: <ENTER>
```

## 11.3 Identifier



**NOTE:** If you want to uniquely identify an entry in your file by a combination of fields, and to force that uniqueness, then you most likely want to create a KEY on your file, rather than using Identifier fields (which do *not* force uniqueness). If a field is part of the PRIMARY KEY for a file, then it should *not* be marked as an Identifier as well.

**REF:** For more information on creating a KEY, see the “[Key Definition](#)” section.

An identifier is a designation you can give to a field that you want *permanently* associated with the **.01** field (NAME) of a file. The SSN field of our PATIENT file example has been defined as an identifier field. Each time a patient’s entry is referenced, the SSN is displayed to help positively identify the entry. When a new entry is added to the file, the user is asked to provide the SSN.

A field that is *not* multiple-valued can be specified as an identifier for a file simply by using the **Identifier** [DIIDENT] option available on the **Utility Functions** [DIUTILITY] menu.

A multiple-valued field *cannot* be designated as an identifier; however, a subfield of that multiple-valued field *can* be designated as an identifier for the Multiple. The DIAGNOSIS field in the PATIENT file *cannot*, for example, be designated as an identifier, but its subfield AGE AT ONSET can be designated as an identifier of DIAGNOSIS. This feature is discussed in more detail later in this section.

These are the steps for setting up the sample SSN field as an identifier:

**Figure 195: File Utilities—Example of Setting a Field as an Identifier**

```
MODIFY WHAT FILE: PATIENT
SELECT FIELD: SSN
WANT TO MAKE [SSN] AN IDENTIFIER? NO// Y <ENTER> (YES)
WANT TO DISPLAY SSN WHENEVER A LOOKUP IS DONE
ON AN ENTRY IN THE [PATIENT] FILE? YES // <ENTER> (YES)
```

Here, the positive answer to the last question causes the patient's SSN value to show up whenever a lookup on a patient is done. For example:

**Figure 196: File Utilities—Example of an Identifier Field Displayed When Doing a Lookup**

```
SELECT PATIENT NAME: FMPATIENT
  1 FMPATIENT,25 000223333
  2 FMPATIENT,29 000114444
CHOOSE 1-2:
```

An identifier field is *not* asked if its **WRITE** access security does *not* match the VA FileMan Access Code of the user. If the identifier field has been specified (in the **Modify File Attributes** [DIMODIFY] option) as a required field, the user *must* type a valid answer to its prompt when it is asked as an identifier; otherwise, the entry just created is deleted.

Using the caret key (^) for jumping is *not* allowed for identifier fields in the **Enter or Edit File Entries** [DIEDIT] option when adding a new entry. If you attempt to use the caret key in a field designated as an identifier in an edit session, the entry just created is deleted. Since the SSN field in our example is mandatory and an identifier, this ensures that every patient in our PATIENT file has an SSN recorded.

As mentioned above, you could make the AGE AT ONSET subfield an identifier for the DIAGNOSIS field Multiple as follows:

**Figure 197: File Utilities—Example of a Subfield as an Identifier**

```
SELECT UTILITY OPTION: IDEN <ENTER> TIFIER
SELECT FIELD: DIAG <ENTER> NOSIS (MULTIPLE)
SELECT DIAGNOSIS SUB-FIELD: AGE AT ONSET
WANT TO MAKE  AGE AT ONSET  AN IDENTIFIER? NO// Y <ENTER> (YES)
WANT TO DISPLAY AGE AT ONSET WHENEVER A LOOKUP IS DONE
ON AN ENTRY IN THE  DIAGNOSIS  FILE? YES// N <ENTER> (NO)
```

As a result of this dialog, every time a new DIAGNOSIS for a patient is entered, the AGE AT ONSET would be asked. The AGE AT ONSET would *not*, however, be automatically displayed at subsequent DIAGNOSIS lookups.

To drop a field's status as an identifier, simply return to the **Identifier** [DIIDENT] option, select the field, and answer **YES** to the question:

**Figure 198: File Utilities—Deleting an Identifier Field**

```
FIELD IS ALREADY AN IDENTIFIER; WANT TO DELETE IT? NO// Y
```

## 11.4 Re-Index File Option

Use the **Re-Index File** [DIRDEX] option when you create a new cross-reference on a field that already contains data, and you want to reindex the file. The dialog shown in [Figure 199](#) is presented when reindexing a file:

**Figure 199: File Utilities—Sample Dialog When Re-Indexing a File**

```
DO YOU WISH TO RE-CROSS-REFERENCE ONE PARTICULAR INDEX? NO// <ENTER>
OK, ARE YOU SURE YOU WANT TO KILL OFF THE EXISTING INDEX? NO// Y <ENTER> (YES)
DO YOU THEN WANT TO [RE-CROSS-REFERENCE]? YES// <ENTER>
```

All the cross-references for the file are fired except for bulletins. This dialog executes triggers and MUMPS cross-references.

If a file contains more than one cross-reference, you can get a list of them by entering a single question mark (?) in response to the “DO YOU WISH TO RE-CROSS-REFERENCE ONE PARTICULAR INDEX?” prompt. You can then reindex a single cross-reference or all of cross-references.

### 11.4.1 Limits on Reindexing Files

There are some files that should *not* be reindexed! When those files are inadvertently reindexed, it can cause major problems and necessitate restores from backups. Currently, there is no way to prevent such files from being reindexed, except for putting “DO NOT RE-INDEX” in the file description, which is ineffective. The **Re-Index File** [DIRDEX] option is a powerful, useful tool, but it can cause a lot of damage when one is *not* paying attention.

Patch DI\*22\*167 allows individual cross-references to be marked “Do Not Re-Index,” and the **Re-Index File** [DIRDEX] option respects that. APIs that perform reindexing also respect that, with the following exceptions:

- APIs that reindex a single record ignore the no-reindex restriction, which includes the following APIs:
  - **EN^DIK**
  - **EN1^DIK**
  - **EN2^DIK**
  - **IX^DIK**
  - **IX1^DIK**
  - **IX2^DIK**

- A cross-reference is reindexed if it is specifically named in an API call, regardless of whether it is marked “**Do Not Re-Index**,” which includes the following APIs:
  - **ENALL^DIK**
  - **ENALL2^DIK**



**REF:** For more information on these APIs, see the *VA FileMan Developer’s Guide*.

Traditional regular cross-references (e.g., **B** and **C** cross-references) are always reindexed, and *cannot* be marked “Do Not Re-Index.” All other cross-references can be marked “**Do Not Re-Index**,” because reindexing them might cause problems.

The following cross-reference types can be marked “Do Not Re-Index:”

- All New-Style cross-references
- Bulletin cross-references
- MUMPS cross-references
- Trigger cross-references

To mark a cross-reference “Do Not Re-Index,” use the **Cross-Reference a Field or File** [DIXREF] option under the **Utility Functions** [DIUTILITY] menu.



**CAUTION:** “Do Not Re-Index” can only be undone by **KILLing** the “NOREINDEX” node in the DD.



**REF:** For more information on the **Cross-Reference a Field or File** [DIXREF] option, see the “[Cross-Reference a Field or File](#)” section.

## 11.5 INPUT Transform (Syntax)

If you have **Programmer** access, you can edit a field’s **INPUT** transform or syntax checker. You can also set the Maximum Length for a field’s output.



**REF:** For a detailed description of the **INPUT** transform, see the “Input Transforms” section in the *VA FileMan Developer’s Guide*.

## 11.6 Edit File

The **Edit File** [DIEDFILE] option available on the VA FileMan **Utility Functions** [DIUTILITY] menu displays the various attributes of a file you specify in Screen Mode (i.e., invokes a ScreenMan form).

[Figure 200](#) is an example using the Edit File option with the ORDER (#100) file in Screen Mode:

**Figure 200: File Utilities—Choosing the Edit File Option**

```
SELECT VA FILEMAN OPTION: UTILITY <ENTER> FUNCTIONS
    VERIFY FIELDS
    CROSS-REFERENCE A FIELD IDENTIFIER
    RE-INDEX FILE
    INPUT TRANSFORM (SYNTAX)
    EDIT FILE
    OUTPUT TRANSFORM
    TEMPLATE EDIT
    UNEDITABLE DATA
    MANDATORY/REQUIRED FIELD CHECK
    KEY DEFINITION
SELECT UTILITY FUNCTIONS OPTION: EDIT <ENTER> FILE
MODIFY WHAT FILE: ORDER// <ENTER>
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>
```

You are then taken into a ScreenMan form where you can edit the properties of the file, as shown in [Figure 201](#):

**Figure 201: File Utilities—Using the Edit File Option in Screen Mode**

```
FILE NAME: ORDER          DESCRIPTION...          (FILE # 100)
      SELECT APPLICATION GROUP:
            DEVELOPER:
DATA DICTIONARY ACCESS: #
            READ ACCESS: #
            WRITE ACCESS: #
            DELETE ACCESS: #
            LAYGO ACCESS: #
            AUDIT ACCESS: #
            DD AUDIT: NO
ASK OK WHEN LOOKING UP AN ENTRY: NO
      FILE SCREEN:
      POST-SELECTION ACTION:
      LOOK-UP PROGRAM:
      CROSS-REFERENCE ROUTINE: ORD2

COMMAND:                PRESS <PF1>H FOR HELP  INSERT
```

You can use the **Edit File** [DIEDIT] option to:

- **Edit the Name of a File**—Edit the file name at the “FILE NAME:” prompt.
- **Delete a File**—If you enter an at-sign (@) at the “FILE NAME:” prompt, you are given the choice of deleting the *entire file* and its data attribute dictionary (including all templates and file definitions) or just deleting the *current individual entries in the file*. However, you *cannot* delete a file that is pointed to by another file.
- **Enter or Edit the Description of a File**—You can enter or edit the word-processing text description for documenting the file at the “DESCRIPTION...” prompt. This description appears in the Standard, Modified Standard, and Global Map format data dictionary listings.
- **Enter or Edit the Application Group**—You can enter or edit the Application Group at the “Select APPLICATION GROUP:” prompt. Enter a namespace (from **2 to 4 characters**) indicating a package accessing this file.
- **Enter or Edit the Developer’s Name**—You can enter or edit the name of the package developer at the “DEVELOPER:” prompt. Entering two question marks (??) lets you choose from a list of names.

- **Enter or Edit the File Access Parameters**—You can enter or edit the security access to a file by making entries at any of the following prompts:
  - "DATA DICTIONARY ACCESS:"
  - "READ ACCESS:"
  - "WRITE ACCESS:"
  - "DELETE ACCESS:"
  - "LAYGO ACCESS:"
  - "AUDIT ACCESS:"



**REF:** File security is fully explained in the "[Data Security](#)" section.

- **Turn Auditing On/Off for a File's Data Dictionary**—If you want to turn auditing on for data dictionary changes, enter **YES** at the "DD AUDIT:" prompt. Answer **NO**, if you do *not* want to audit data dictionary changes.



**REF:** Auditing is fully explained in the "[Auditing](#)" section.

- **Ask/Do Not Ask Users to Confirm Their Entry Selection**—If you want users who select an entry in a file (for any lookup purpose) to confirm their entry selection by answering positively at the "...OK?" prompt, answer **YES** at the "ASK 'OK' WHEN LOOKING UP AN ENTRY:" prompt. If you do *not* want users to confirm their entry selection, answer **NO** at the "ASK 'OK' WHEN LOOKING UP AN ENTRY:" prompt. The default is **NO**.



**TIP:** Use this feature on files containing many similar or confusingly named entries (e.g., files for drugs).

- **Enter a File Screen**—A line of MUMPS code can be entered here. It should set the **\$T** switch **TRUE** or **FALSE**. At the time of execution **Y** is the number of a file entry, which you want to FILTER for lookup. Thus, this code is a 'permanent **DIC("S")**' for the file.



**CAUTION: Misuse of this can disable the file!**

For example, this is the file screen for the NEW PERSON (#200) file: **I \$\$\$SCR200^XUSER.**

- **Enter or Edit a Post-Selection Action (only available when you have Programmer access)**—If you have **Programmer** access, you can write M code for a Post-Selection Action, for entries in this file.



**REF:** Post-Selection Action is explained in the *VA FileMan Developer's Guide*.

- **Enter a Lookup Routine (only available when you have Programmer access)**—If you have **Programmer** access, you also can enter an existing lookup routine. To do this, enter a routine namespace (from **3 to 6 characters**, no **^**) at the "LOOK-UP PROGRAM:" prompt. The name you choose for the lookup routine *must* be a routine currently on the system. This special lookup routine is executed instead of the standard VA FileMan lookup logic, whenever a call is made to **^DIC**.
- **Specify that Cross-references on a File Should be Compiled (only available when you have Programmer access)**—If you have **Programmer** access, you also can specify that cross-references on a file should be compiled. To do this, enter a routine namespace (from **3 to 6 characters**, no **^**) at the "CROSS-REFERENCE ROUTINE:" prompt. This becomes the namespace of the compiled routines. If a *new* routine name is entered, but the cross-references are *not* compiled currently, the routine name is automatically deleted.

To **stop** the use of the compiled cross-references, enter an at-sign (**@**) at the "CROSS-REFERENCE ROUTINE:" prompt. At this point, the cross-references are considered uncompiled, and VA FileMan does *not* use the routine for re-indexing. If you decide later to recompile the cross-references, you are shown the routine name previously used so that you can easily reuse the same routine name.

Stopping the use of the compiled cross-reference does *not* delete the compiled routines. If you want, you can delete those routines manually.

## 11.7 OUTPUT Transform

Sometimes, you might want to display a field differently from the way in which it is stored. For example, a Social Security Number can be entered and stored as **nine digits**, but you may want it to always be displayed with punctuating hyphens. The **Output Transform** [DIOTRAN] option allows you to make this kind of specification by associating with any field a computed expression that operates on the value of that field



**REF:** For details about using M code in an **OUTPUT** transform, see the “OUTPUT Transform” topic in the “Advanced File Definition” section in the *VA FileMan Developer’s Guide*.

If you want the SSN field to always appear with inserted dashes, answer the prompts as shown in In [Figure 202](#):

**Figure 202: File Utilities—Example of Creating an OUTPUT Transform**

```
SELECT OPTION: UTI <ENTER> LITIES
SELECT UTILITY OPTION: OUT <ENTER> PUT TRANSFORM
MODIFY WHAT FILE: PATIENT
SELECT FIELD: SSN
SSN OUTPUT TRANSFORM: $E(SSN,1,3)_-_- $E(SSN,4,5)_-_- $E(SSN,6,9)
```



### REMEMBER:

- The transform does *not* apply when you are inputting data; thus, do *not* enter the dashes when using the Enter or Edit File Entries [DIEDIT] option.
- To retrieve the internal, stored value of a field that has an OUTPUT transform, you can refer to the INTERNAL(SSN) function.
- The internal form of the date is automatically invoked when you are sorting by a DATE/TIME valued field.

## 11.8 Template Edit

The **Template Edit** option available on the VA FileMan **Utility Functions** menu [DIUTILITY] is used to edit each of the three types of VA FileMan templates:

- **INPUT**
- **PRINT**
- **SORT**

For each template type, a *two*-screen ScreenMan form is used. This allows you to edit templates in Screen Mode.

The **first** screen of the pair allows you to change the access privileges of the template you are editing:

- **READ ACCESS**—This access controls which class of users [i.e., **DUZ(0)**] get to *use* the template.
- **WRITE ACCESS**—This access controls which class of users gets to *change* the template.

The **first** screen also allows you to enter a DESCRIPTION for the purpose of documenting what the template does. This DESCRIPTION is printed on a "TEMPLATES ONLY" data dictionary list, and in the "TEMPLATES" section of other data dictionary listings.

The **second** screen allows you to edit the *contents* of a template. To "jump" to the second screen from the first screen in a Screen Mode, you need only press the **<PF1><Arrow Down>** from wherever you are on the current screen.



**NOTE:** The **first** screen provides the usual kind of field-by-field help in response to entering a single question mark (?); all help messages are displayed in the lower portion of the screen. Also, entering **<PF1>H** provides general ScreenMan help.

The **second** screen, however, does *not* provide help on individual entries. Thus, if you are building a complicated new template from scratch, it is still a good idea to use the traditional, interactive Scrolling Mode with the **Enter or Edit File Entries** [DIEDIT] and **Print File Entries** [DIPRINT] options.

Figure 203 is an example of the **first** screen of a **PRINT** template using the **Template Edit** option:

**Figure 203: File Utilities—Example of the *First* Screen of a PRINT Template**

```

SELECT UTILITY FUNCTIONS OPTION: TEMPLATE <ENTER> EDIT
MODIFY WHAT FILE: NEW PERSON// <ENTER>
SELECT TEMPLATE FILE: PRINT <ENTER> TEMPLATE

SELECT PRINT TEMPLATE: XUFILEINQ
  1 XUFILEINQ (NOV 04, 2004@11:29) FILE #200
  2 XUFILEINQHDR (APR 09, 1992@12:01) FILE #200
CHOOSE 1-2: 1 <ENTER> XUFILEINQ (NOV 04, 2004@11:29) FILE #200
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

```

You are then taken into a ScreenMan form where you can edit the template properties, as shown in Figure 204:

**Figure 204: File Utilities—Editing a PRINT Template’s Properties in Screen Mode (*First* Screen)**

```

TEMPLATE NAME: XUFILEINQ          TEMPLATE TYPE:
      (COMPILED AS [^XUFILEO] ROUTINE)

DATE LAST MODIFIED: NOV 4,2004@11:29
DATE LAST USED: MAY 17,2012
  READ ACCESS: @
  WRITE ACCESS: @
  USER #:

DESCRIPTION...

HEADER:
[XUFILEINQHDR]
SUB-HEADER SUPPRESSED:

      (PRINT FIELDS ON NEXT PAGE...)

-----
EXIT      SAVE      NEXT PAGE      REFRESH

ENTER A COMMAND OR [^] FOLLOWED BY A CAPTION TO JUMP TO A SPECIFIC FIELD.

COMMAND: NEXT          PRESS <PF1>H FOR HELP INSERT

```

The dates shown following the “DATE LAST MODIFIED” and “DATE LAST USED” prompts are for informational purposes only and are *not* editable. Also, if a template has been “compiled” into a set of routines, an informational message is displayed near the top of the screen (e.g., “Compiled as ‘**^XUFILEO** routine”).

On the **second** screen of the form, you see the **SORT**, **PRINT**, or **INPUT** fields. Thus, you can use this second screen to edit the specific template fields.

[Figure 205](#) is an example of the **second** screen of a **PRINT** template using the **Template Edit** option:

**Figure 205: File Utilities—Editing a PRINT Template’s Properties in Screen Mode (Second Screen)**

```

EDITING PRINT TEMPLATE  XFILEINQ
===== [ INSERT ] ===== ( FILE 200 ) >===== [ <PF1>H=HELP ]=====
$$ (#3=@:PROGRAMMER ACCESS TO ALL FILES,1:);C38;L35;
ACCESSIBLE FILE
  NUMBER;C1;L10;FILE#
  ACCESSIBLE FILE;C12;L25
  DATA DICTIONARY ACCESS;R3;DD
  DELETE ACCESS;R5;DELETE
  LAYGO ACCESS;R5;LAYGO
  READ ACCESS;R4;READ
  WRITE ACCESS;R5;WRITE
  AUDIT ACCESS;R5;AUDIT

<=====T=====T=====T=====T=====T=====T=====T=====T=====

```

As you can see from this example, fields under a Multiple field (e.g., ACCESSIBLE FILE) are *indented*. As you edit, add, and delete subfields here, you *must preserve the indentation*. The same holds true for [Relational Navigation](#) within the template; fields jumped to are in a different file and are indented an extra three spaces each. You do *not* have to indent each new level exactly three spaces; however, there *must* be some extra spaces. Then, if necessary, “un-indent” the same number of spaces to get back to a previous level.

If a **SORT** template has a user number (i.e., USER #), only that user can use that **SORT** template in the VA FileMan **Print File Entries** [DIPRINT] option. To remove this restriction, simply delete the user number by entering an at-sign (@) at the “USER #” prompt.

For **SORT** templates, you can also use the *first* screen of the **Template Edit** option to associate a particular **PRINT** template with a **SORT** template. Thus, whenever that **SORT** template is invoked in the **Print File Entries** [DIPRINT] option, the associated **PRINT** template is used by default, with *no* "FIRST PRINT FIELD:" prompt being displayed to the user.

**Figure 206: File Utilities—Example of a SORT Template (First Screen)**

```

SELECT UTILITY FUNCTIONS OPTION: TEMPLATE <ENTER> EDIT
MODIFY WHAT FILE: NEW PERSON// <ENTER>
SELECT TEMPLATE FILE: SORT <ENTER> TEMPLATE

SELECT SORT TEMPLATE: XUUF AA <ENTER> XUUF AA
                        (JUL 01, 1987@10:51) USER #9999 FILE #200
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

```

You are then taken into a ScreenMan form where you can edit the properties of the template, as shown in [Figure 207](#):

**Figure 207: File Utilities—Editing a SORT Template’s Properties in Screen Mode (First Screen)**

```

TEMPLATE NAME: XUUF AA

DATE LAST MODIFIED: JUL 1,1987
DATE LAST USED: JAN 29,1999
READ ACCESS: #
WRITE ACCESS: #
USER #: ██████████

DESCRIPTION...

PRINT TEMPLATE: XUUF AA

(SORT FIELDS ON NEXT PAGE...)

-----
EXIT    SAVE    NEXT PAGE    REFRESH
ENTER A COMMAND OR [^] FOLLOWED BY A CAPTION TO JUMP TO A SPECIFIC FIELD.

COMMAND: NEXT ██████████ PRESS <PF1>H FOR HELP INSERT

```

Editing **SORT** template fields is particularly tricky; however, most **SORT** templates have only three or so sort levels.

Figure 208 is an example of the *second* screen of a **SORT** template using the **Template Edit** option:

**Figure 208: File Utilities—Editing a SORT Template’s Properties in Screen Mode (Second Screen)**

```
EDITING SORT TEMPLATE [XUUFAA]
===== [ INSERT ] ===== < (FILE 200) > ===== [ <PF1>H=HELP ] =====
SORT BY: DATE/TIME OF ATTEMPT
FROM: JAN 1 1999
TO: T_
    WITHIN DATE/TIME OF ATTEMPT, SORT BY: USER
    FROM:
    TO:
        WITHIN USER, SORT BY: TYPE OF FAILED ATTEMPT
        FROM:
        TO:

<=====T=====T=====T=====T=====T=====T=====T=====T=====
```

The specifications for each successive level of sorting are indented further to the right. You can add or insert sort levels; however, each sort group of lines *must* be indented further to the right than the sort group above it. For each level of sorting, except when the sorting is on a Boolean value, there should be a “**From:**” line and a “**To:**” line. You can also have a fourth line that says “**ASK**” or “**DON’T ASK**,” for sort ranges other than first-to-last. Remember to indent each line in a sort group by the same number of spaces.

## 11.9 Uneditable Data

The Uneditable Data option allows you to specify that a field *cannot* be edited or deleted by a user.

If editing is attempted, the field’s value, along with a No Editing message, is displayed. If, however, the value is part of a subfield, the deletion of the entire entry in the Multiple-valued field is allowed, unless the **.01** field of the Multiple itself is made uneditable.

You can also use this option to remove the uneditable restriction on a field.

## 11.10 Mandatory/Required Field Check

The Mandatory/Required Field Check option checks that fields that are key fields or designated as required contain data. It can check one, a series, or all required entries in a file. If an entry lacks data in a required or key field, a report like [Figure 209](#) is furnished:

**Figure 209: File Utilities—Mandatory/Required Field Check Report**

REQUIRED-FIELD-CHECK ENTRY	FILE: 16026	ZZPATIENT DD-NUMBER/PATH	FIELD	PAGE 1
3	FMPATIENT,25	16026 DIZ(16026,3	SSN	

If all required and key fields contain data, then the NO REQUIRED FIELD IS MISSING message is displayed.

You can store the results in a template.

## 11.11 Key Definition

The following topics are covered:

- Create a Key
- Edit a Key
- Delete a Key
- Verify a Key

Using the Key Definition option, VA FileMan allows you to define keys on a file or subfile. A key is a group of fields that, taken collectively, uniquely identifies a record. All fields in a key *must* have values (*must not* be **NULL**) and those values, taken together, *must* be unique across all records in the file or subfile. VA FileMan enforces KEY INTEGRITY whenever records are added or edited.

Exactly one key in a file *must* be designated the PRIMARY KEY. All other keys are SECONDARY KEYS. While VA FileMan enforces the integrity of both primary and SECONDARY KEYS, the PRIMARY KEY is VA FileMan's principal means of looking up entries in the file. VA FileMan prompts for lookup values for each of the PRIMARY KEY fields and considers a record a match only if it matches *all* lookup values. The **.01** field should be a part of the PRIMARY KEY.

Keys are also useful when transporting data to another system using the Kernel Installation and Distribution System (KIDS). Since the key fields uniquely identify a

record, it is easy to decide whether a record being brought in to the target system needs to be merged to a record that already exists, or whether it is a new record.

Associated with each key is a *Uniqueness Index*, a regular, New-Style cross-reference. The Uniqueness Index helps VA FileMan enforce KEY INTEGRITY and is used during lookup. When you create a new key, you can have VA FileMan create a new Uniqueness Index automatically for you, or you can select an existing index to be the Uniqueness Index of the key. The index you select, though, *must meet the following criteria*:

- It *must* be a regular, New-Style cross-reference.
- It *must* be used for lookup and sorting; it *cannot* have a name that starts with the letter "A".
- It *cannot* have any **SET** or **KILL** conditions.
- It *must* consist of only field-type cross-reference values, all of which are used as subscripts (i.e., it can contain no computed values).
- No subscripts can have transforms.

### 11.11.1 Create a Key

To create a key, do the following:

**Figure 210: File Utilities—Creating a Key**

```
SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY OPTION: KEY <ENTER> DEFINITION

MODIFY WHAT FILE: ZZPATIENT// <ENTER>
SELECT SUBFILE: <ENTER>

THERE ARE NO KEYS DEFINED ON FILE #16026.
WANT TO CREATE A NEW KEY FOR THIS FILE? NO// YES

ENTER A NAME FOR THE NEW KEY: A// <ENTER>
CREATING NEW KEY [A] ...
```

You are then taken into Screen Mode (i.e., ScreenMan form) where you can edit the properties of the key. Enter a single question mark (?) or two question marks (??) at any prompt for additional help.

**Figure 211: File Utilities—Creating a Key in Screen Mode**

```

NUMBER: 5                                EDIT A KEY                                PAGE 1 OF 1
-----
FILE: 16026                               NAME: A                                PRIORITY: PRIMARY

KEY FIELDS:
=====
FIELD          SEQ NO.  FILE          FIELD NAME
-----
UNIQUENESS INDEX:

INDEX DETAILS...

COMMAND:                                PRESS <PF1>H FOR HELP    INSERT

```

On this screen ([Figure 211](#)), in the “KEY FIELDS” section, you can select the fields you wish to include in this key and assign each field a sequence number. The sequence number determines the order in which the fields appear as subscripts in the Uniqueness Index. If you select the key fields in this manner, leave the Uniqueness Index field blank. When you exit the form, VA FileMan prompts you for a name for the Uniqueness Index, and then creates the index automatically for you, as shown in [Figure 212](#):

**Figure 212: File Utilities—Creating the Uniqueness Index Automatically**

```

I[M] GOING TO CREATE A NEW UNIQUENESS INDEX TO SUPPORT KEY [A] OF FILE #16026.

INDEX NAME: C// <ENTER>

ONE MOMENT PLEASE ...
BUILDING NEW INDEX ... DONE!

PRESS RETURN TO CONTINUE:

```

Alternatively, you can leave the information in the KEY FIELDS section blank and select an existing Uniqueness Index. When you exit the form, VA FileMan checks that the information in the KEY FIELDS section is consistent with the selected Uniqueness Index. If there is a conflict, you are asked for a method to resolve the conflict. In this case, select Option #2, "Make Key match Uniqueness Index," as shown in [Figure 213](#):

**Figure 213: File Utilities—Resolving a Conflict with the Key Fields and Uniqueness Index**

```

THE KEY FIELDS AND THE FIELDS IN THE UNIQUENESS INDEX DON'T MATCH.

  SELECT ONE OF THE FOLLOWING:

      1      RE-EDIT THE KEY
      2      MAKE KEY MATCH UNIQUENESS INDEX (ALSO SELECTED ON UP-ARROW)

ENTER RESPONSE: 2 <ENTER> MAKE KEY MATCH UNIQUENESS INDEX (ALSO SELECTED ON UP-
ARROW)

  MODIFYING FIELDS IN KEY ... DONE!

```

### 11.11.2 Edit a Key

To edit a key, identify the file or subfile you wish to edit. VA FileMan displays the cross-references on the file and offers you the choices of Edit, Delete, or Create.

**Figure 214: File Utilities—Editing a Key**

```

SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY OPTION: KEY <ENTER> DEFINITION

MODIFY WHAT FILE: ZZPATIENT// <ENTER>
SELECT SUBFILE: <ENTER>

KEYS DEFINED ON FILE #16026:

  A PRIMARY KEY    UNIQUENESS INDEX: C
    FIELD(S):      1) NAME (#.01)
                   2) SSN (#.02)

CHOOSE V (VERIFY)/E (EDIT)/D (DELETE)/C (CREATE): EDIT

WHICH KEY DO YOU WISH TO EDIT? A// <ENTER>

```

You are then taken into Screen Mode (i.e., ScreenMan form) where you can edit the properties of the key. Enter a single question mark (?) or two question marks (??) at any prompt for additional help.

### 11.11.3 Delete a Key

Figure 215 shows how to delete a key.



**NOTE:** You are also given the option of deleting the Uniqueness Index of the key.

**Figure 215: File Utilities—Deleting a Key**

```
SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY OPTION: KEY <ENTER> DEFINITION

MODIFY WHAT FILE: ZZPATIENT// <ENTER>
SELECT SUBFILE: <ENTER>

KEYS DEFINED ON FILE #16026:

  A PRIMARY KEY      UNIQUENESS INDEX: C
    FIELD(S):      1) NAME (#.01)
                  2) SSN (#.02)

CHOOSE V (VERIFY)/E (EDIT)/D (DELETE)/C (CREATE): D <ENTER> ELETE

WHICH KEY DO YOU WISH TO DELETE? A// <ENTER>
ARE YOU SURE YOU WANT TO DELETE THE KEY? NO// Y <ENTER> ES

  KEY [A] OF FILE #16026 DELETED.

DO YOU WANT TO DELETE THE [C] UNIQUENESS INDEX (#6) ON FILE #16026 PREVIOUSLY
USED BY KEY [A] OF FILE #16026? YES

INDEX DEFINITION DELETED.
REMOVING OLD INDEX ... DONE!
```

## 11.11.4 Verify a Key

When you verify the integrity of a key, VA FileMan checks that all fields in the key have values (are *not* **NULL**), and that those field values, taken together, are unique across all records in the file. Any problems are reported. You can also save the entries that violate KEY INTEGRITY in a template.

Figure 216: File Utilities—Verifying a Key

```
SELECT OPTION: UTI <ENTER> LITY FUNCTIONS
SELECT UTILITY OPTION: KEY <ENTER> DEFINITION

MODIFY WHAT FILE: ZZPATIENT// <ENTER>
SELECT SUBFILE: <ENTER>

KEYS DEFINED ON FILE #16026:

  A PRIMARY KEY      UNIQUENESS INDEX: KEYA
    FIELD(S):      1) NAME (#.01)
                  2) SSN (#.02)

CHOOSE V (VERIFY)/E (EDIT)/D (DELETE)/C (CREATE): V <ENTER> ERIFY

WHICH KEY DO YOU WISH TO VERIFY? A// <ENTER>
STORE THESE ENTRY ID[S] IN TEMPLATE: <ENTER>

DEVICE: HOME// <ENTER> TELNET TERMINAL

KEY INTEGRITY CHECK                      DEC 31, 1998  09:23    PAGE 1
-----
                KEY: A (#5), FILE #16026
UNIQUENESS INDEX: KEYA (#6)

ENTRY #  NAME                ERROR
-----  -
1        FMPATIENT,10        DUPLICATE KEY A (#5)
2        FMPATIENT,10        DUPLICATE KEY A (#5)
3        FMPATIENT,25        MISSING KEY FIELD(S):
                               SSN [16026,.02]
```

In this example, records #1 and #2 have the same key, and record #3 is missing a value for SSN (Field #.02).

# 12 Auditing

VA FileMan auditing tracks changes to:

- Data in a field—Auditing a Data Field.
- File’s structure—Auditing a Data Dictionary.

Fields are identified for auditing by using the **Modify File Attributes** [DIMODIFY] option or by using the **Turn Data Audit On/Off** option (see the “[Turning Data Field Audit On/Off](#)” section). Thus, a user with limited VA FileMan access does *not* need the **Modify File Attributes** [DIMODIFY] option to initiate an audit.

Files are identified for auditing by using the **Edit File** [DIEDFILE] option.



**CAUTION: Auditing can be resource intensive; it can slow your system considerably. It can also use considerable disk space. Use it with discretion.**

The **Audit Menu** [DIAUDIT], which is locked with the **XUAUDITING** security key, is located under the **Other Options** [DIOOTHER] menu:

**Figure 217: Auditing—Audit Options**

```
SELECT VA FILEMAN <TEST ACCOUNT> OPTION: OTHER OPTIONS

FILEGRAMS ...
AUDIT MENU ...
SCREENMAN ...
STATISTICS
VA FILEMAN MANAGEMENT ...
DATA EXPORT TO FOREIGN FORMAT ...
EXTRACT DATA TO FILEMAN FILE ...
IMPORT DATA
BROWSER
DATA ACCESS CONTROL ...
DATA MAPPING ...SELECT OTHER OPTIONS <TEST ACCOUNT> OPTION: AUDIT MENU

SELECT AUDIT MENU <TEST ACCOUNT> OPTION: ?

FIELDS BEING AUDITED
MONITOR A USER
PURGE DATA AUDITS
PURGE DD AUDITS
TURN DATA AUDIT ON/OFF
SHOW PAST CHANGES TO DATA DICTIONARIES

SELECT AUDIT MENU <TEST ACCOUNT> OPTION:
```

The use of these options is described in the topics that follow in this section.

## 12.1 Auditing a Data Field

- [Overview](#)
- [Setting a Data Field Audit](#)
- [Turning Data Field Audit On/Off](#)
- [Reviewing the Data Field Audit Trail](#)
- [Tracking Data Field Audits](#)
- [Purging a Data Field Audit Trail](#)

### 12.1.1 Overview

Use the **Enter or Edit File Entries** [DIEDIT] option to add, change, or delete data values. For example, an admission clerk can use the **Enter or Edit File Entries** [DIEDIT] option to enter a Date of Birth of "**JAN 20, 1949**" for patient ONE FMPATIENT. Later, a nurse could use the same option to change the value to "**JAN 20, 1950**". You might like to know who made the successive changes and when; the data field audit capability is designed to record this information.

When changes like the ones mentioned in the example above are made to an audited field, the following information is stored in the **AUDIT** (#1.1) file:

- Date and time the change was made.
- User's name who made the change.
- Old and new data values.

In other words, starting an audit on a data field provides an ongoing chronological list of who made what changes to data values of fields you are auditing.

## 12.1.2 Setting a Data Field Audit

To begin auditing data changes in a field, use the **Modify File Attributes** [DIMODIFY] option, choose the file and fields that should be audited and answer **YES** or **EDITED** to the "AUDIT:" prompt. The possible responses to the "AUDIT:" prompt are:

**Table 94: Auditing—"AUDIT" Prompt Response**

Response	Description
<b>YES, ALWAYS</b>	Indicates the audit will occur when data is initially entered, changed or deleted.
<b>EDITED OR DELETED</b>	Means that the audit only occurs when changes are made or values deleted ( <i>not</i> when the data is initially entered).
<b>NO</b>	Turns the audit off.
@	Deletes the current audit status (effectively turning audit off), <i>not</i> the field.

The example in [Figure 218](#) initiates a Data Field audit on the DATE OF BIRTH field of the PATIENT (#2) file. Only changes to pre-existing data are recorded:

**Figure 218: Auditing—Example of a Data Field Audit**

```

SELECT OPTION: MOD <ENTER> IFY FILE ATTRIBUTES
DO YOU WANT TO USE THE SCREEN-MODE VERSION? YES// <ENTER>

MODIFY WHAT FILE: PATIENT <ENTER> (1630 ENTRIES)

SELECT FIELD: DATE OF BIRTH
LABEL: <ENTER>
TITLE: <ENTER>
AUDIT: EDITED <ENTER> EDITED OR DELETED
.
.
.

```

If you have **Programmer** access, you are also presented with the "AUDIT CONDITION:" prompt. Here, you can restrict the data entries that are audited. Enter a line of M code that *must* evaluate true or false. If it evaluates true, an audit record of the value change is made. Otherwise, no auditing is done for that data entry.



**REF:** For more information on the audit condition, see the "Advanced File Definition" section in the *VA FileMan Developer's Guide*.



**NOTE:** If auditing is *not* turned on for the field with a **YES** or **EDITED** answer to the "AUDIT:" prompt, the field is *not* audited, even if code is entered at the "AUDIT CONDITION:" prompt.

Fields with a DATA TYPE field of **COMPUTED** *cannot* be audited.

To initiate an audit of data fields or of a data dictionary, you *must* have **AUDIT** access to the files you want to audit.



**REF:** For more information about **AUDIT** access, see the "[Data Security](#)" section.

### 12.1.3 Turning Data Field Audit On/Off

Users who do *not* have the **Modify File Attributes** [DIMODIFY] option (**DD** access) to files can be given the **Turn Data Audit On/Off** [DIAUDIT TURN ON/OFF] option. This option is found on VA FileMan's **Audit Menu** [DIAUDIT]. It is used to begin or end an audit trail for fields. The user *must* have **AUDIT** access to the file to set data audits. No other access is necessary. The Turn Data Audit On/Off option is called DIAUDIT TURN ON/OFF; it can be granted to users with Kernel's menu management options. No other attributes in the field definition can be affected by use of this option.

Use of the **Turn Data Audit On/Off** [DIAUDIT TURN ON/OFF] option is very similar to using the **Modify File Attributes** [DIMODIFY] option:

Figure 219: Auditing—Turning a Data Audit On

```
SELECT VA FILEMAN <TEST ACCOUNT> OPTION: OTHER OPTIONS
    FILEGRAMS ...
    AUDIT MENU ...
    SCREENMAN ...
    STATISTICS
    VA FILEMAN MANAGEMENT ...
    DATA EXPORT TO FOREIGN FORMAT ...
    EXTRACT DATA TO FILEMAN FILE ...
    IMPORT DATA
    BROWSER
    DATA ACCESS CONTROL ...
    DATA MAPPING ...SELECT OTHER OPTIONS <TEST ACCOUNT> OPTION: AUDIT
<ENTER> MENU
SELECT AUDIT MENU <TEST ACCOUNT> OPTION: ?
    FIELDS BEING AUDITED
    MONITOR A USER
    PURGE DATA AUDITS
    PURGE DD AUDITS
    TURN DATA AUDIT ON/OFF
    SHOW PAST CHANGES TO DATA DICTIONARIES
SELECT AUDIT MENU <TEST ACCOUNT> OPTION: TURN <ENTER> DATA AUDIT ON/OFF
AUDIT FROM WHAT FILE: PATIENT <ENTER> (1630 ENTRIES)
SELECT FIELD: DATE OF BIRTH <ENTER> Y
AUDIT Y//: EDITED <ENTER> OR DELETED
AUDIT CONDITION: ??
ENTER MUMPS CODE THAT WILL SET $T TO 1 FOR AUDIT TO TAKE PLACE.
AUDIT CONDITION: ^
```

To end the audit trail, simply re-enter the **Turn Data Audit On/Off** [DIAUDIT TURN ON/OFF] option and turn the audit off by entering NO at the "AUDIT:" prompt:

**Figure 220: Auditing—Turning a Data Audit Off**

```
SELECT AUDIT MENU <TEST ACCOUNT> OPTION: TURN <ENTER> DATA AUDIT ON/OFF
AUDIT FROM WHAT FILE: PATIENT// <ENTER> (1630 ENTRIES)
SELECT FIELD: DATE OF BIRTH <ENTER> E
AUDIT: E// NO

SELECT FIELD: DATE OF BIRTH <ENTER> N
AUDIT: N//
```

## 12.1.4 Reviewing the Data Field Audit Trail

### 12.1.4.1 CAPTIONED Output with Audit Trail

There are several different ways to retrieve audit information. The simplest is to examine all the audited past values for a particular entry, or a set of entries. Say you are reviewing patient entries in the PATIENT (#2) file. Using either the **Inquire to File Entries** [DIINQUIRE] or the **Print File Entries** [DIPRINT] options, specify the [CAPTIONED] output. If the file has audited data on record, you are asked whether you want to see the audit trail on the patients displayed.

Suppose the PATIENT (#2) file has been edited so that patient "FMPATIENT" was first assigned a DOB of **1/20/49**, but that subsequently this DOB was changed to **1/20/50**.

**Figure 221: Auditing—CAPTIONED Output with Audit Trail**

```
SELECT VA FILEMAN <TEST ACCOUNT> OPTION: INQUIRE <ENTER> TO FILE ENTRIES
OUTPUT FROM WHAT FILE: PATIENT// <ENTER> (1630 ENTRIES)
SELECT PATIENT NAME: FMPATIENT, ONE
ANOTHER ONE: <ENTER>
STANDARD CAPTIONED OUTPUT? YES// <ENTER> (YES)
INCLUDE COMPUTED FIELDS: (N/Y/R/B): NO// <ENTER> - NO RECORD NUMBER (IEN), NO
COMPUTED FIELDS
DISPLAY AUDIT TRAIL? NO// YES

NAME: FMPATIENT, ONE DOB: JAN 20, 1950
CHANGED FROM [JAN 20, 1949] ON MAR 15, 2001@16:48:53 BY USER #16
CREATED ON MAR 13, 2001@19:41:36 BY USER #330
```

### 12.1.4.2 Computed-Field Functions for Retrieving Audit Changes

Three powerful new functions allow retrieval of audit data. As with all Computed Expressions, use of these functions is in the context of the entry in the (audited) file.

- **PRIORVALUE(Field\_name)** returns values of the "Field\_name" field that existed before a change was made.
- **PRIORDATE(Field\_name)** returns the corresponding date/time at which a change to the field value was made.
- **PRIORUSER(Field\_name)** returns the corresponding user number ("**DUZ**") who made the change to the field value.

Each of these Functions is multi-valued, since the Field value for the entry may have been changed more than once in the past. The Functions "**1ST**", "**2ND**", "**LAST**", "**COUNT**" can be used with these Multiples. Thus, the Computed Expression **COUNT(PRIORVALUE(NAME))>4** means "the NAME on this entry has been changed more than four times in the past". To find any entry whose NAME used to contain "**FMUSER**", search for **PRIORVALUE(NAME)["FMUSER"]**.

### 12.1.4.3 AUDIT File

A more specialized way to examine audits is to query the AUDIT (#1.1) file to obtain audit information. The set of all past audited events for a given VA FileMan file is itself a VA FileMan file, which can be queried or printed. The entries are identified as follows:

1. By their internal entry number in the AUDIT (#1.1) file.
2. By the internal entry number of the edited entry from the file being audited.
3. By the date/time of the audited event.

Suppose "**FMPATIENT,ONE**" is the **355th** entry in the PATIENT (#2) file.

**Figure 222: Auditing—AUDIT File: Query**

```
OUTPUT FROM WHAT FILE: PATIENT// 1.1 <ENTER> AUDIT
AUDIT FROM WHAT FILE: PATIENT// <ENTER> (1630 ENTRIES)
SELECT PATIENT AUDIT: 1 <ENTER> 355      3-15-2001@16:48:53
ANOTHER ONE: <ENTER>
STANDARD CAPTIONED OUTPUT? YES// <ENTER> (YES)
INCLUDE COMPUTED FIELDS: (N/Y/R/B): NO// Y <ENTER> (YES)
```

Responding with **YES** at the "Include COMPUTED fields: (N/Y/R/B): NO//"  
prompt shows the AUDIT (#1.1) file **COMPUTED** fields:

- ENTRY NAME
- FIELD NAME
- OLD VALUE
- NEW VALUE.

A **NO** response does *not* show these fields.

[Figure 223](#) shows the output that is produced:

**Figure 223: Auditing—AUDIT File: Output**

NUMBER: 1	INTERNAL ENTRY NUMBER: 355
DATE/TIME RECORDED: JAN 06, 2003@12:42:08	
FIELD NUMBER: 2	USER: FMUSER,FIVE
OLD INTERNAL VALUE: 2490120	DATATYPE OF OLD VALUE: DA
NEW INTERNAL VALUE: 2500120	DATATYPE OF NEW VALUE: DA
ENTRY NAME (C): FMPATIENT,ONE	FIELD NAME (C): DATE
OLD VALUE (C): JAN 06, 2003	NEW VALUE (C): FEB 03, 2003

If you deleted FMPATIENT's DATE OF BIRTH, the corresponding entry in the AUDIT (#1.1) file would show the NEW VALUE as **<deleted>**. If the entire entry for FMPATIENT were deleted, there would be no value for the ENTRY NAME field in any of the audit listings for that entry. The last audit entry would show **<deleted>** for the NEW VALUE of the DATE OF BIRTH field.

Since the entries in the AUDIT (#1.1) file can be sorted just as flexibly as those in any other VA FileMan file, you could easily, for example, search for all audits that occurred between a range of DATE/TIME RECORDED values.

## 12.1.5 Tracking Data Field Audits

Use the **Fields Being Audited** [DIAUDITED FIELDS] option to list the audited fields from a file or a specified range of files. After selecting the **Fields Being Audited** [DIAUDITED FIELDS] option, give a file (or a range) at the "START WITH What File:" and "GO TO What File:" prompts.

You receive a report with the following data ([Figure 224](#)):

- **FILE**—File number
- **NUMBER**—Field number
- **LABEL**—Field name
- **TYPE**—Type of field being audited
- **AUDIT**—Type of audit being performed.

The listing in [Figure 224](#) shows all fields flagged for auditing in a file range:

**Figure 224: Auditing—Sample Listing Showing Fields Flagged for Auditing**

AUDITED FIELDS					JAN 20, 1989	10:10	PAGE 1
FILE	NUMBER	LABEL	TYPE	AUDIT	-----		
16	2	DOB	DATE/TIME	YES, ALWAYS			
40.5	10	LOCATION	POINTER	EDITED OR DELETED			
1036	1	SEX	SET	EDITED OR DELETED			
1036	7	SSN	FREE TEXT	YES, ALWAYS			

## 12.1.6 Purging a Data Field Audit Trail

Use the **Purge Data Audits** [DIAUDIT PURGE DATA] option to purge the audit trail. It purges the records used for auditing data fields for a specified file. Purging audit trails is *not* an automatic feature; it *must* be done manually.

You should do either of the following:

- Turn auditing off on the files you are purging while you are doing the purge.

Or

- Leave auditing on but purge the file when *not* many users are on the system.

If you purge when auditing is on and people are using the file in question, it is possible that you might end up with incomplete audit records on the audited file.



**CAUTION: Purged audit records *cannot* be recovered!**

[Figure 225](#) illustrates purging *selected* records from the audit trail:

**Figure 225: Auditing—Choosing to Purge Only *Selected* Data Audit Records**

```
SELECT VA FILEMAN <TEST ACCOUNT> OPTION: OTHER OPTIONS
FILEGRAMS ...
AUDIT MENU ...
SCREENMAN ...
STATISTICS
VA FILEMAN MANAGEMENT ...
DATA EXPORT TO FOREIGN FORMAT ...
EXTRACT DATA TO FILEMAN FILE ...
IMPORT DATA
BROWSER
DATA ACCESS CONTROL ...
DATA MAPPING ...SELECT OTHER OPTIONS <TEST ACCOUNT> OPTION: AUDIT
<ENTER> MENU
SELECT AUDIT MENU <TEST ACCOUNT> OPTION: ?
FIELDS BEING AUDITED
MONITOR A USER
PURGE DATA AUDITS
PURGE DD AUDITS
TURN DATA AUDIT ON/OFF
SHOW PAST CHANGES TO DATA DICTIONARIES
SELECT AUDIT MENU <TEST ACCOUNT> OPTION: PURGE DATA <ENTER> AUDITS
AUDIT FROM WHAT FILE: PATIENT <ENTER> (1630 ENTRIES)
DO YOU WANT TO PURGE ALL DATA AUDIT RECORDS? NO// <ENTER>
```

Answering **NO** to the “DO YOU WANT TO PURGE ALL DATA AUDIT RECORDS? NO//” prompt allows you to specify which entries to purge.

**Figure 226: Auditing—Listing Internal Entry Numbers for Data Audit Fields for Possible Purging**

```
PURGE AUDIT RECORDS BY: INTERNAL ENTRY NUMBER// ?? <ENTER>

CHOOSE FROM:
.001          NUMBER
.01           INTERNAL ENTRY NUMBER
.02           DATE/TIME RECORDED
.03           FIELD NUMBER
.04           USER
.05           RECORD ADDED
.06           ACCESSED
1            ENTRY NAME
1.1          FIELD NAME
2            OLD VALUE
2.1          OLD INTERNAL VALUE
2.2          DATATYPE OF OLD VALUE
3            NEW VALUE
3.1          NEW INTERNAL VALUE
3.2          DATATYPE OF NEW VALUE
4.1          MENU OPTION USED
4.2          PROTOCOL OR OPTION USED

TYPE '-' IN FRONT OF NUMERIC-VALUED FIELD NAME TO SORT FROM HIGH TO LOW.
TYPE '+' IN FRONT OF FIELD NAME TO GET SUBTOTALS BY THAT FIELD'S VALUES.
'#' TO PAGE-FEED ON EACH FIELD VALUE, '!' TO GET RANKING NUMBER
'@' TO SUPPRESS SUB-HEADER, ']' TO FORCE SAVING TEMPLATE
TYPE ';'TXT' AFTER FREE-TEXT FIELDS TO SORT NUMBERS AS TEXT
TYPE [TEMPLATE NAME] IN BRACKETS TO SORT BY PREVIOUS SEARCH RESULTS
TYPE 'BY(0)' TO DEFINE RECORD SELECTION AND SORT ORDER

PURGE AUDIT RECORDS BY: INTERNAL ENTRY NUMBER//
```

You can specify which records to purge by referencing any field in the AUDIT (#1.1) file. You can select the records by using the same criteria available at the "SORT BY:" prompt.



**REF:** For more details on the "SORT BY:" prompt, see the "Print: How to Print Reports from Files" section in the *VA FileMan User Manual*.

**Figure 227: Auditing—Purging Selected Audit Records from a File**

```
PURGE AUDIT RECORDS BY: INTERNAL ENTRY NUMBER// USER
START WITH USER: FIRST// FMUSER
GO TO USER: LAST// FMUSER
  WITHIN USER, PURGE AUDIT RECORDS BY: <ENTER>

DEVICE: <ENTER> TELNET PORT   RIGHT MARGIN: 80// <ENTER>

...HMMM, LET ME PUT YOU ON HOLD FOR A SECOND...

PURGE OF AUDIT DATA:  PATIENT          FEB 21, 1990   13:26   PAGE 1
-----
4 RECORDS PURGED.
```

The dialog in [Figure 228](#) purges *all* records from the PATIENT (#2) file's audit trail:

**Figure 228: Auditing—Purging All Audit Records from a File**

```
AUDIT FROM WHAT FILE: PATIENT <ENTER> (1630 ENTRIES)
DO YOU WANT TO PURGE ALL DATA AUDIT RECORDS? NO// YES
ARE YOU SURE? NO// YES

DELETED
```

## 12.2 Auditing a Data Dictionary

In addition to auditing changes to data values, changes to data dictionaries are audited.



The Description and Technical Description field changes are *not* audited.

- [Setting Automatic Data Dictionary Auditing](#)
- [Reviewing the Data Dictionary Audit Trail](#)
- [Purging a Data Dictionary Audit Trail](#)
- [Auditable Word Processing Fields](#)
- [Word-Processing Fields Can be Made Uneditable](#)
- [Reviewing a User's Data Access](#)

### 12.2.1 Setting Automatic Data Dictionary Auditing

Substantive Data Dictionary changes are automatically recorded. No action is needed to start this auditing. Auditing results are saved in the DD AUDIT (#.6) file. Changes to

some documentation-related attributes (e.g., Description and Technical Description) are *not* audited.

## 12.2.2 Reviewing the Data Dictionary Audit Trail

To see what changes were made to a specific file's data dictionary, use the **Inquire to File Entries** [DIINQUIRE] or **Print File Entries** [DIPRINT] options and identify the DD AUDIT (#.6) file as the file of choice. To see all changes made to data dictionaries during a period of time use the **Show Past Changes to Data Dictionaries** [DIAUDIT SHOW PAST CHG TO DDs] option.

[Figure 229](#) is an example of how to identify the changes made to a data dictionary:

**Figure 229: Auditing—Choosing to Review a Data Dictionary Audit**

```
SELECT VA FILEMAN <TEST ACCOUNT> OPTION: INQ <ENTER> UIRE TO FILE ENTRIES
OUTPUT FROM WHAT FILE: PATIENT//: .6 <ENTER> DD AUDIT
AUDIT FROM WHAT FILE: PATIENT
SELECT PATIENT SUB-FILE: <ENTER>
```



**NOTE:** You only see the "<FILE NAME> SUB-FILE" prompt if the file contains a Subfile. To display audit information for Subfile, specify it here.

**Figure 230: Auditing—Specifying a Data Dictionary Audit**

```
SELECT PATIENT DD AUDIT: ?
ANSWER WITH PATIENT DD AUDIT NUMBER, OR FIELD NUMBER, OR DATE UPDATED, OR USER
CHOOSE FROM:
1      2      02-20-90 FMUSER,FIVE
2      3      02-20-90 FMUSER,FIVE
```

In [Figure 230](#) the entries in the DD AUDIT (#.6) file are identified by the field number (2 and 3 in this example), the date of the change (02/20/90 for both entries), and the person making the change (FMUSER,FIVE for both).

**Figure 231: Auditing—Reviewing a Data Dictionary Audit**

```
SELECT PATIENT DD AUDIT: 1
ANOTHER ONE: 2
ANOTHER ONE: <ENTER>
STANDARD CAPTIONED OUTPUT? YES// <ENTER>
DISPLAY COMPUTED FIELDS? NO// <ENTER>

NUMBER: 1                FIELD NUMBER: 2
TYPE: EDIT DATE          UPDATED: FEB 20, 1990@17:54:36
USER: FMUSER,FIVE        ATTRIBUTE NAME: LABEL
ATTRIBUTE NUMBER: .01     FILE NUMBER: 999000
OLD VALUE(S): DATE OF BIRTH NEW VALUE(S): DOB

NUMBER: 2                FIELD NUMBER: 3
TYPE: EDIT DATE          UPDATED: FEB 21, 1990@11:54:03
USER: FMUSER,FIVE        ATTRIBUTE NAME: LABEL
ATTRIBUTE NUMBER: .01     FILE NUMBER: 999000
OLD VALUE(S): CURRENT AGE NEW VALUE(S): AGE
```

The example in [Figure 231](#) indicates that a user named FMUSER,FIVE modified the (fictitious) PATIENT (#999000) file on 2/20 and 2/21/90. This person edited the LABEL (.01 attribute) of Fields #2 and #3:

- The LABEL of Field #2 was changed from DATE OF BIRTH to DOB.
- The LABEL of Field #3 was changed from CURRENT AGE to AGE.

The number of the first change as it was recorded in the DD AUDIT (#.6) file is **1** and the second is **2**.

If, instead, you want a complete listing of all data dictionary changes regardless of file, use the **Show Past Changes to Data Dictionaries** [DIAUDIT SHOW PAST CHG TO DDs] option. This option prompts you for the starting date of the audits you want to see. By default, all changes since the last purge of data dictionary audits will be shown. The audits are sorted by File Number and then in reverse date/time. The field, field attribute, and user making the change are shown as well as the before and after values.

In [Figure 232](#), two files were changed; multiple changes were made to the first and one to the second.

**Figure 232: Auditing—Reviewing DD Changes for Time Period**

```

SELECT VA FILEMAN <TEST ACCOUNT> OPTION: OTHER OPTIONS

    FILEGRAMS ...
    AUDIT MENU ...
    SCREENMAN ...
    STATISTICS
    VA FILEMAN MANAGEMENT ...
    DATA EXPORT TO FOREIGN FORMAT ...
    EXTRACT DATA TO FILEMAN FILE ...
    IMPORT DATA
    BROWSER
    DATA ACCESS CONTROL ...
    DATA MAPPING ...SELECT OTHER OPTIONS <TEST ACCOUNT> OPTION: AUDIT
<ENTER> MENU

SELECT AUDIT MENU <TEST ACCOUNT> OPTION: ?

    FIELDS BEING AUDITED
    MONITOR A USER
    PURGE DATA AUDITS
    PURGE DD AUDITS
    TURN DATA AUDIT ON/OFF
    SHOW PAST CHANGES TO DATA DICTIONARIES

SELECT AUDIT MENU <TEST ACCOUNT> OPTION: SHOW <ENTER> PAST CHANGES TO DATA
DICTIONARIES
SHOW DATA DICTIONARY CHANGES SINCE: FIRST// T-3 <ENTER> (MAR 12, 2013)
DEVICE: HOME// <ENTER> TELNET PORT    RIGHT MARGIN: 80// <ENTER>

DATA DICTIONARY CHANGES, LOCAL SERIALS FILE(#680) SINCE MAR 12,2013
FIELD                ATTRIBUTE                                USER NUMBER
-----
ADDED DATE           TYPE                                MAR 12,2013@16:04:24    1
  FROM: D
  TO: DA
TITLE                TYPE                                MAR 12,2013@16:02:01    1
  FROM: RP680.5A
  TO: RP680.5

DATA DICTIONARY CHANGES, VA FILEMAN CHANGE FILE(#1009) SINCE MAR 12,2013
FIELD                ATTRIBUTE                                USER NUMBER
-----
REVIEW DISPOSITION   TYPE                                MAR 13,2013@13:28:21    1
  FROM: F:FAILED. DOESN'T WORK.;D:NOT DOCUMENTED.;1:ALL OKAY.;T:NOT
  TESTED.;R:ROLLED BACK. N/A.;
  TO: F:FAILED. DOESN'T WORK.;D:NOT DOCUMENTED.;1:ALL OKAY.;T:NOT
  TESTED.;R:ROLLED BACK. N/A.;DT:NEITHER TESTED NOR DOCUMENTED.;

```

### 12.2.3 Purging a Data Dictionary Audit Trail

Use the **Purge DD Audits** [DIAUDIT PURGE DD] option to erase *all* audit trails used in auditing data dictionaries (including Subfiles that have their own data dictionaries) for a specified file. Purging is *not* an automatic feature; it *must* be done manually.

You should do either of the following:

- Turn auditing off on the files you are purging while you are doing the purge.

OR

- Leave auditing on but purge the file when *not* many users are on the system.

If you purge when auditing is on and people are using the file in question, it is possible that you might end up with incomplete audit records on the audited file.

The dialog in [Figure 233](#) results in purging **selected** data dictionary audit records for the user **FMUSER**:

**Figure 233: Auditing—Purging Selected Data Dictionary Audit Records**

```
SELECT AUDIT MENU <TEST ACCOUNT> OPTION: PURGE DD <ENTER> AUDITS
AUDIT FROM WHAT FILE: PATIENT <ENTER> (1630 ENTRIES)
SELECT PATIENT SUB-FILE: <ENTER>
DO YOU WANT TO PURGE ALL DD AUDIT RECORDS? NO// NO

PURGE DD AUDIT RECORDS BY: FIELD NUMBER// USER
START WITH USER: FIRST// FMUSER
GO TO USER: LAST// FMUSER
  WITHIN USER, PURGE DD AUDIT RECORDS BY: <ENTER>
DEVICE: <ENTER> TELNET PORT   RIGHT MARGIN: 80// <ENTER>

HOLD ON, PLEASE...

PURGE OF DD AUDIT:  PATIENT FILE      FEB 21, 1990   14:45   PAGE 1
-----
9 RECORDS PURGED.
```

The dialog in [Figure 234](#) results in purging **all** data dictionary audit records for the PATIENT (#2) file:

**Figure 234: Auditing—Purging All Data Dictionary Audit Records**

```
AUDIT FROM WHAT FILE: PATIENT <ENTER> (1630 ENTRIES)
SELECT PATIENT SUB-FILE: <ENTER>
DO YOU WANT TO PURGE ALL DD AUDIT RECORDS? NO// YES
ARE YOU SURE? NO// YES

DELETED
```

## 12.2.4 Auditable Word Processing Fields

VA FileMan security has been improved by allowing **WORD-PROCESSING** fields to be audited. The functionality allows storing auditable information when information is added, edited, updated, or deleted.

Users *must* have the **XUAUDITING** Security Key to set this functionality.

**Figure 235: Auditing—Auditable Word Processing Fields**

```
SELECT OTHER OPTIONS <TEST ACCOUNT> OPTION: AUDIT MENU
SELECT AUDIT MENU <TEST ACCOUNT> OPTION: TURN <ENTER> DATA AUDIT ON/OFF
AUDIT FROM WHAT FILE:

Enter a File name that has WORD-PROCESSING fields.

SELECT FIELD:

Enter a WORD-PROCESSING field Name.

AUDIT: Y// ?
CHOOSE FROM:
Y      YES, ALWAYS
N      NO
E      EDITED OR DELETED
```

## 12.2.5 Word-Processing Fields Can be Made Uneditable

With VA FileMan 22.2, reference files and clinically significant text can now be protected from subsequent change.

**Figure 236: Auditing—Uneditable Data**

```
SELECT UTILITY FUNCTIONS <TEST ACCOUNT> OPTION: UNEDIT <ENTER> ABLE DATA
MODIFY WHAT FILE:
Type a File name that has WORD-PROCESSING field and press Enter.
SELECT FIELD:
Type a name of the WORD-PROCESSING field and press Enter.
WANT TO PREVENT ALL USERS FROM CHANGING OR DELETING DATA VALUES
THAT ARE ENTERED FOR THE '<FIELD NAME>' FIELD? NO// YES <ENTER> (YES)
...FIELD IS NOW UNEDITABLE
```

## 12.2.6 Reviewing a User's Data Access

You can see when a specified user has accessed a file by using the **Monitor A User** [DIAUDIT MONITOR USER] option . To begin the report, you need to identify the following:

- User you want to monitor.
- File of interest.
- Date.

The resulting report indicates which entries the user edited and the first and last time the user edited them. You can obtain details of the edits made by using the **Inquire to File Entries** [DIINQUIRE] or **Print File Entries** [DIPRINT] options to query the AUDIT (#1.1) file as described above.



**NOTE:** This option only reports edits for files during periods for which data auditing is turned on.

**Figure 237: Auditing—Sample User Access Report**

```
SELECT AUDIT MENU <TEST ACCOUNT> OPTION: MONITOR <ENTER> A USER
SELECT A USER WHO HAS SIGNED ON TO THIS SYSTEM: FMUSER,FIVE
SELECT AUDITED FILE: VA FILEMAN ROUTINE
START WITH DATE: FIRST// <ENTER>
DEVICE: HOME// <ENTER> TELNET PORT    RIGHT MARGIN: 80// <ENTER>

VA FILEMAN ROUTINE RECORDS ACCESSED BY FMUSER,FIVE (DUZ=1)    PAGE 1
                                EARLIEST ACCESS              LATEST ACCESS
-----
DIFROM                          NOV 14,2012@15:02:14
DIFROM1                          NOV 14,2012@15:59:29
DINVTM                          NOV 30,2012@14:12:24
DINVTX                          NOV 30,2012@15:29:19
```

## 13 Data Security

VA FileMan has facilities for screening access to entire files and to data within files on a field-by-field basis. Access to previously defined templates is also controlled.

When used with Kernel, there are two possible mechanisms for control of user access:

- Associating an Access Code with every user and with every file, field, and template on the system. The user's Access Code is stored in the local variable.
- Doing a lookup into a user's entry in the NEW PERSON (#200) file to see if the file in question is available to that user. The method is in effect only if Kernel's File Access Security has been installed on the system. It takes precedence over the Access Code method for file-level security. Field and template security are unaffected by Kernel's File Access Security; they remain enforced by Access Code.

These methods for data security are described in this section.

### 13.1 Security at the File Level

There are two methods of controlling access to files:

- Access Code Security on Files
- File Access Security (Formerly Part 3 of Kernel)

On a particular system only one method is in effect at a particular time. If File Access Security has been installed, it controls file-level security; otherwise, access is controlled by Access Codes. Obviously, if you are using VA FileMan without Kernel, only Access Code security is possible.

#### 13.1.1 Access Code Security on Files

The Access Code is a string of characters that correspond to functional data access categories. For example:

- Uppercase **A** might correspond to entry/editing of Administrative data.
- Lowercase **a** might correspond to viewing of Administrative data.
- Uppercase **F** might correspond to entering or editing Fiscal data.
- Lowercase **f** might correspond to viewing of Fiscal data.

Typically, you go through a password-checking signon process before using VA FileMan. During signon, you are identified by the system and your Access Code is set. For

example, it might be equal to **Aaf**, if you are identified as someone entitled to view and change Administrative data, but only to view (search and print) Fiscal data. If you lack any such security clearance, the default value of the Access Code entry is simply **NULL**.

[Table 95](#) lists the Access Codes used to control access to files in six different ways:

**Table 95: Data Security—File Access Codes**

<b>Access Code</b>	<b>Description</b>
<b>READ</b>	Controls use of the file by the following options: <ul style="list-style-type: none"> <li>• <b>Print File Entries</b> [DIPRINT]</li> <li>• <b>Search File Entries</b> [DISEARCH]</li> <li>• <b>Inquire to File Entries</b> [DIINQUIRE]</li> <li>• <b>Statistics</b> [DISTATISTICS]</li> <li>• <b>List File Attributes</b> [DILIST]</li> <li>• <b>Transfer File Entries</b> (transfer-from file)</li> </ul>
<b>WRITE</b>	Controls use of <b>Enter or Edit File Entries</b> [DIEDIT] and <b>Transfer File Entries</b> (transfer-to file) options.
<b>DELETE</b>	Controls deletion of an entire entry in the <b>Enter or Edit File Entries</b> [DIEDIT] or the <b>Transfer Entries</b> [DITRANSFER] options.
<b>LAYGO</b>	Controls creating a new entry within the <b>Enter or Edit File Entries</b> [DIEDIT] option. You <i>must</i> have <b>LAYGO</b> as well as <b>WRITE</b> access to a file to add new entries. Additionally, you <i>must</i> have <b>WRITE</b> access on the field level to all required identifiers.
<b>DD</b>	Controls use of the <b>Modify File Attributes</b> [DIMODIFY] option and <b>Utility Functions</b> menu (Data Dictionary).
<b>AUDIT</b>	Controls the setting of auditing characteristics and the deletion of audit trails.

All these controls are based on the value of the Access Code. When you access a file, under any of these options, you are *not* allowed to access any file that is protected unless your current Access Code either equals an at-sign (@) or contains at least **one character** in common with the protection code string of the file. The at-sign (@) is generally reserved for use by developers; it gives **Programmer** access.

Any new file that you create with a code string in your Access Code is automatically given the following access equal to that code string:

- **READ**
- **WRITE**
- **DELETE**
- **LAYGO**
- **AUDIT**

To change these codes later, use the **Edit File** [DIEDFILE] option on the **Utility Functions** [DIUTILITY] menu. Be sure when doing so that your own Access Code contains the codes you want to add or equals the at-sign (@).

### 13.1.2 File Access Security (Formerly Part 3 of Kernel)

VA FileMan also can perform a lookup into a user's record and see if a specific file has been assigned. If your systems manager has run the special conversion to have access controlled by the ACCESSIBLE FILE Multiple in the NEW PERSON (#200) file, then access to a file is *not* based on an Access Code. Rather, a lookup is done in your record to see if you are allowed access to the file in question. If your VA FileMan Access Code is the at-sign (@), you are allowed access to all files, even if this special conversion has been performed.

Access to files is granted to you by the systems manager who uses Kernel's File Access Security system.



**REF:** For detailed information about File Access Security, see the [Kernel application documentation on the VA software Document Library \(VDL\)](#).

## 13.2 Protection for Fields in a File

Your Access Code is checked to control access to data fields within each file.



**NOTE:** File Access Security does *not* affect field-level protection.

**Table 96: Data Security—Field Access Codes**

Access Code	Description
<b>WRITE</b>	The <b>Enter or Edit File Entries</b> [DIEDIT] option looks at WRITE access for every field used to see if <b>WRITE</b> access has been defined. If the field's <b>WRITE</b> Access Code has no characters in common with your Access Code, you are <i>not</i> able to see that field.
<b>DELETE</b>	The <b>Enter or Edit File Entries</b> [DIEDIT] option looks at the <b>DELETE</b> access of a field when you try to delete a value for that field (with the at-sign [@]). If there is a <b>DELETE</b> access string and if it has no characters in common with your Access Code, you are prohibited from deleting.
<b>READ</b>	The <b>Inquire to File Entries</b> [DIINQUIRE], <b>Print File Entries</b> [DIPRINT], and <b>Search File Entries</b> [DISEARCH] options look at the <b>READ</b> access for every field accessed. If this <b>READ</b> Access Code has no characters in common with your Access Code, you <i>cannot</i> see data stored in the field.

There is an exception to this field protection: If the invoking program has set the Access Code equal to the at-sign (@), all fields are accessible. The at-sign (@) is also considered the developer's Access Code to the *entire* data dictionary.

To enter or edit a field's **READ**, **DELETE**, or **WRITE** access:

1. Use the **Modify File Attributes** [DIMODIFY] option.
2. Indicate the field to be protected.
3. Enter the desired codes when prompted for the following information:
  - "READ ACCESS:"
  - "DELETE ACCESS:"
  - "WRITE ACCESS:"

Any code character entered *must* be in your current Access Code, unless you have the at-sign (@).

### 13.3 Protection for Templates

When you create an **INPUT**, **SORT**, or **PRINT** template, your Access Code is assigned to that template for **READ** access and **WRITE** access. Anyone else subsequently using the template *must* have a code that has a character in common with the template's **READ** access. Anyone who is allowed to change the template *must* have a code with a character in common with the template's **WRITE** access. When the template is changed, these Access Codes can also be changed.

Print the PRINT TEMPLATE (#.4) file, the SORT TEMPLATE (#.401) file, and the INPUT TEMPLATE (#.402) file to display the **READ** access and **WRITE** access fields for templates.

Every user on the system has a user number. This number is sometimes called **DUZ**, because it is stored in the local variable **DUZ**. The user number is the internal entry number in the NEW PERSON (#200) file, if Kernel is installed. VA FileMan uses the user number for security for **SEARCH** templates. When a **SEARCH** template is created, it is assigned the **DUZ** of its creator. Only someone with that **DUZ** can use the template. You can change the assigned USER # with the **Template Edit** [DITEMP] option. If the USER # is deleted, everyone can use the **SEARCH** template.

## 14 Transferring File Entries

The **Transfer Entries** [DITRANSFER] option contains two options:

- **Transfer File Entries**
- **Compare/Merge File Entries**

These options allow you to compare and/or merge two entries in a single file or to transfer entries from one file to another. For example, you can combine data from two different entries into one of the two. This could happen in a patient database when the same patient has been inadvertently entered twice with the name spelled slightly differently.



**CAUTION:** Once you have merged file entries, the merging *cannot* be undone. Care *must* be taken that data is *not* mistakenly lost.

### 14.1 Transfer File Entries Option

The **Transfer File Entries** option can be used for several purposes. You can use it to:

- Merge two entries in the same file.
- Transfer one or more records from one file to another file.
- Copy a data dictionary into a new file.

However, the **Compare/Merge File Entries** option (described in Section [14.2](#), "[Compare/Merge File Entries Option](#)") should usually be used to merge entries in the same file; it is specifically designed for that task.

You *must* have **READ** access for the file you are transferring from and **WRITE** access for the file you are transferring to. If you are deleting entries after the transfer, you need **DELETE** access as well.



**REF:** For the details of file security, see the "[Data Security](#)" section.

## 14.1.1 Transferring Data within the Same File

You can use the **Transfer File Entries** option to merge two entries that are in the same file. To do this:

1. Identify the input and output file as the same.
2. Identify the two entries.

Data values are then transferred from the **FROM** entry to the **TO** entry. [Figure 238](#) shows the simple dialog:

**Figure 238: Transferring File Entries—Transferring Data within a File**

```
SELECT OPTION: TRANSFER ENTRIES

SELECT TRANSFER OPTION: ?
ANSWER WITH TRANSFER OPTION NUMBER, OR NAME
CHOOSE FROM:
1      TRANSFER FILE ENTRIES
2      COMPARE/MERGE FILE ENTRIES

SELECT TRANSFER OPTION: 1 <ENTER> TRANSFER FILE ENTRIES

INPUT TO WHAT FILE: PATIENT
TRANSFER FROM FILE: PATIENT
TRANSFER DATA INTO WHICH PATIENT: FMPATIENT,77
TRANSFER FROM PATIENT: FMPATIENT,90
WANT TO DELETE THIS ENTRY AFTER IT'S BEEN TRANSFERRED? NO//
```

If the **TO** entry (**77 FMPATIENT** in [Figure 238](#)) already has a value on file for a given field, that value is preserved (i.e., it is *not* overwritten by the corresponding field value in the **FROM** entry). This rule applies to **WORD-PROCESSING** data fields also. If the recipient has any text on file, corresponding text from the sender's entry is *not* merged with it. Thus, if you decide to delete the **FROM** entry (**90 FMPATIENT** in [Figure 238](#)), you can lose some data.

In the case of distinct Multiple-valued subfields, merging takes place. The subentries in the **FROM** entry would be added to the Multiple in the **TO** entry. Further, if two subentries have the same **.01** value and if any of the subfields are blank in the **TO** entry, the **FROM** subentry's data is placed in the blank subfield. In this way, data is added to Multiples in the same way that it is added to files.

For example, suppose **DIAGNOSIS** is the label of the **.01** field of a Multiple and **AGE AT ONSET** is a subfield in that Multiple. If **77 FMPATIENT** has a **DIAGNOSIS** of "**Angina**" and **90 FMPATIENT** has one of "**Diabetes**", **77 FMPATIENT** ends up with both "**Angina**" and "**Diabetes**". Further, if both **77 FMPATIENT** and **90 FMPATIENT** had "**Angina**", but **77 FMPATIENT** had no **AGE AT ONSET** for that subentry and **90**

**FMPATIENT** did have one, the **90 FMPATIENT**'s AGE AT ONSET data for "**Angina**" would be transferred to **77 FMPATIENT**.

[Figure 239](#) illustrates the transfer of data values from one entry to another in more detail. (These two entries will also be used in the discussion of the **Compare/Merge File Entries** option.) Before the transfer, the **Inquire to File Entries** [DIINQUIRE] option displays these two entries from the (fictitious) SCHOLAR file:

**Figure 239: Transferring File Entries—Example Displaying Two Records in a File *Prior* to a Transfer**

```
NAME: FMPATIENT,23 A.  SSN: 000-99-9999
SUBJECT AREA: PHILOSOPHY
TOPICS: 23[S PARADOX
TOPICS: PRINCIPLE OF EXCLUDED MIDDLE
TOPICS: SET THEORY
TOPICS: THEORY OF TYPES
TOPICS: LIAR PARADOX
```

```
NAME: FMPATIENT,23 H.  DATE OF BIRTH: 1872  SSN: 000-88-8888
SUBJECT AREA: MATHEMATICS
TOPICS: 24[S PARADOX
TOPICS: SET THEORY
TOPICS: THEORY OF TYPES
TOPICS: AXIOM OF INFINITY
```

The transfer is then initiated with the dialog shown in [Figure 240](#):

**Figure 240: Transferring File Entries—Initiating a Transfer of File Entries**

```
SELECT TRANSFER OPTION: TRANS <ENTER> FER FILE ENTRIES

INPUT TO WHAT FILE: SCHOLAR
TRANSFER FROM FILE: SCHOLAR
TRANSFER DATA INTO WHICH SCHOLAR: FM
    1  FMPATIENT,23 A.
    2  FMPATIENT,23 H.
CHOOSE 1-2: 1
TRANSFER FROM SCHOLAR: FMPATIENT,23 H.
WANT TO DELETE THIS ENTRY AFTER IT[S TRANSFERRED? NO// YES
...EXCUSE ME, LET ME THINK ABOUT THAT A MOMENT....
```

Data values are then merged such that no pre-existing values are overwritten but new values are added. The DATE OF BIRTH is added since the pre-existing value was **NULL**. Different subentries in the TOPICS Multiple are added to the pre-existing list of topics. The **Inquire to File Entries** [DIINQUIRE] option shows the result:

**Figure 241: Transferring File Entries—Results *after* a Transfer of File Entries**

```
NAME: FMPATIENT,23 A. DATE OF BIRTH: 1872 SSN: 000-99-9999
SUBJECT AREA: PHILOSOPHY
TOPICS: 23[S PARADOX
TOPICS: PRINCIPLE OF EXCLUDED MIDDLE
TOPICS: SET THEORY
TOPICS: THEORY OF TYPES
TOPICS: LIAR PARADOX
TOPICS: AXIOM OF INFINITY

THE ENTRY FOR FMPATIENT,23 H. HAS BEEN DELETED.
```

### 14.1.2 Transferring Entries between Files

You can use the **Transfer File Entries** option to move *all* or a *group* of entries from one file to an entirely separate file. To do this:

1. Answer the "Input to what File:" prompt and the "TRANSFER FROM FILE:" prompt with different file names.
2. Specify whether transferred entries should be added all as new, or whether they should be merged with existing entries.
3. Specify whether transferred entries should be deleted in the original file.
4. Specify which entries to transfer, by entering sort criteria.

For transfer to occur, the NAME (#.01) fields of both files *must* have matching LABEL fields and DATA TYPE fields. In this way, VA FileMan can identify corresponding entries. Values of the fields can then be transferred. Only those fields where the LABEL and DATA TYPE fields match are transferred. Before the transfer is done, you are told which fields have their data transferred.

The dialog presented when transferring entries to another file is presented in [Figure 242](#). In this instance, you are transferring the contents of the (fictitious) SCHOLAR file to the (fictitious) NEW SCHOLAR file. The (fictitious) NEW SCHOLAR file already exists, and it does have some entries whose NAME field matches those in the (fictitious) SCHOLAR file.

**Figure 242: Transferring File Entries—Transferring Entries from one File to Another**

```
SELECT OPTION: TRANSFER ENTRIES
SELECT TRANSFER OPTION: TRANSFER FILE ENTRIES
INPUT TO WHAT FILE: NEW SCHOLAR
TRANSFER FROM FILE: NEW SCHOLAR// SCHOLAR
[NAME] FIELDS, [SSN] FIELDS, [DATE OF BIRTH] FIELDS, [SUBJECT AREA]
FIELDS,[TOPICS] FIELDS, WILL BE TRANSFERRED
WANT TO MERGE TRANSFERRED ENTRIES WITH ONES ALREADY THERE? NO// YES
WANT EACH ENTRY TO BE DELETED AS IT'S TRANSFERRED? NO
TRANSFER ENTRIES BY: NAME// <ENTER>
START WITH NAME: FIRST// <ENTER>
DEVICE: HOME// <ENTER>
```

You can specify whether you want entries with the same NAME (#.01) field to be merged when the transfers are made, or whether each transferred entry and subentry should become a distinct new entry in the target file. (In this case, the answer was **YES** to merge.) You can also specify whether the entries should be deleted from the “from” file as they are transferred. (In this case ([Figure 242](#)), the answer was **NO** to delete.)

A simple report is created that lists the entries that were transferred. You can route that list to a printer using the “DEVICE:” prompt.

In addition, you have considerable control over which entries are transferred. Your answers to the “TRANSFER ENTRIES BY:” and “START WITH:” prompts select entries in the same way that you specify sort criteria when describing a print output.



**REF:** For more information on sort criteria and print output, see the “Print: How to Print Reports from Files” section in the *VA FileMan User Manual*.

For example, if you only wanted to transfer scholars born after **1900** to the (fictitious) NEW SCHOLAR file, you could answer the “TRANSFER ENTRIES BY:” prompt as shown in [Figure 243](#):

**Figure 243: Transferring File Entries—Selecting Specific Entries for Transfer**

```
TRANSFER ENTRIES BY: NAME// DATE OF BIRTH>1900
WITHIN DATE OF BIRTH>1900, TRANSFER ENTRIES BY: <ENTER>
```



**NOTE:** The **Transfer File Entries** option can be used to purge files. You can define a (fictitious) SCHOLAR ARCHIVE file containing a subset of the fields that are in the original file (e.g., the fields: NAME, SSN, and DATE OF BIRTH); you can then simply transfer into this separate file all or a selected group of entries from the original file, deleting the entries as they are transferred.

### 14.1.3 Transferring Entries into a New File

You can use the **Transfer File Entries** option to create a new file. To do this:

1. At the “Input to what File:” prompt, enter the name of a nonexistent file.
2. If you have **Programmer** access, you are prompted for the global location for the new file.
3. When you specify the file to transfer *from*, you can request that the data dictionary of that file be copied as the data dictionary for your new file.

[Figure 244](#) is an example of using the **Transfer File Entries** option to create a new file:

**Figure 244: Transferring File Entries—Using the Transfer File Entries Option to Create a New File**

```
INPUT TO WHAT FILE: SCHOLAR COPY
  ARE YOU ADDING SCHOLAR COPY AS A NEW FILE? NO// Y <ENTER> (YES)
  FILE NUMBER: 16031// <ENTER>

INTERNAL GLOBAL REFERENCE: ^DIZ(16031, // <ENTER>
...SORRY, I'M WORKING AS FAST AS I CAN...
  A FREETEXT NAME FIELD (#.01) HAS BEEN CREATED.
TRANSFER FROM FILE: SCHOLAR
DO YOU WANT TO TRANSFER THE SCHOLAR DATA DICTIONARY INTO YOUR NEW
FILE? YES
```



**NOTE:** You are asked the global reference question only if you have **Programmer** access.

Answering **YES** copies (or “clones”) the data definitions of the old file. Then, if the old file has had any templates created, you are asked:

DO YOU WANT TO COPY [SCHOLAR] TEMPLATES INTO YOUR NEW FILE?

Once you have created a new file with identical field and template descriptions, you can transfer entries into it. This is a method of copying a file.

## 14.2 Compare/Merge File Entries Option

The **Compare/Merge File Entries** [???] option allows you to compare the data value of two entries before merging them into one entry. Furthermore, this option provides you with an opportunity to identify the data values from either entry that are used to create the final merged entry. Both entries involved *must* be in the same file to use this option.

### 14.2.1 Comparing Entries

You can use the **Compare/Merge File Entries** option as a simple tool to compare entries. To do this:

1. Identify a file.
2. Identify the two entries to be compared.
3. At the “MERGE ENTRIES AFTER COMPARING THEM?” prompt, enter **NO**.

In [Figure 245](#), two similar entries in the (fictitious) SCHOLAR file are used.

**Figure 245: Transferring File Entries—Selecting Entries to Compare in a File (1 of 2)**

```
SELECT TRANSFER OPTION: COMPARE/MERGE FILE ENTRIES
COMPARE ENTRIES IN WHAT FILE: SCHOLAR
COMPARE SCHOLAR: FMPATIENT
```

VA FileMan responds to this abbreviated response with the matching entries in the list shown in [Figure 246](#):

**Figure 246: Transferring File Entries—Selecting Entries to Compare in a File (2 of 2)**

```
1 FMPATIENT,23 A.
2 FMPATIENT,23 H.
CHOOSE 1-2: 1
WITH SCHOLAR: FMPATIENT,23 H.

NOTE: USE THIS OPTION ONLY DURING NON-PEAK HOURS IF MERGING ENTRIES IN A
FILE THAT IS POINTED-TO EITHER BY MANY FILES, OR BY LARGE FILES.

MERGE ENTRIES AFTER COMPARING THEM? NO// <ENTER>
```

Here you choose whether to simply compare entries or to merge them. If you are merging, the process of repointing entries in other files from the merged-from entry to the merged-to entry can be time-consuming or can create many tasked jobs. This is more likely if the file is pointed to by many files or if the files that point to it have many entries. In these situations, consider merging at times when your system is *not* busy.

**Figure 247: Transferring File Entries—Comparison Output**

```
DO YOU WANT TO DISPLAY ONLY THE DISCREPANT FIELDS? NO// <ENTER>
DEVICE: <ENTER> RIGHT MARGIN: 80// <ENTER>

COMPARISON OF SCHOLAR FILE ENTRIES          FEB 14, 1991  11:59    PAGE 1
SCHOLAR          FMPATIENT,23 A.          FMPATIENT,23 H.
-----
*** NAME          FMPATIENT,23 A.          FMPATIENT,23 H.
*** SSN           000-99-9999          000-88-8888
*** SUBJECT AREA  PHILOSOPHY          MATHEMATICS
*** DATE OF BIRTH 1872

PRESS RETURN TO CONTINUE OR ^_ TO EXIT:
```

The asterisks (“\*\*\*”) appearing in front of the field label indicate the entries contain different data in those fields. This simple report compares the data in each field in the two entries.

## 14.2.2 Merging Entries

You can use the **Compare/Merge File Entries** option to merge entries. To do this:

1. Identify a file.
2. Identify the two entries to be compared.
3. At the “MERGE THESE ENTRIES AFTER COMPARING THEM?” prompt, enter **YES**.
4. Choose which entry supplies the default values.
5. Choose to retain or delete the “merge from” record.
6. Optionally adjust, field-by-field, which entry supplies the default value.
7. Choose to proceed with the merge, summarize before merging, or re-edit the merge criteria.

The Merge process is described in detail via the example in [Figure 248](#):

**Figure 248: Transferring File Entries—Merging Entries in a File**

```
MERGE THESE ENTRIES AFTER COMPARING THEM? NO// YES
1 FMPATIENT,23 A.
2 FMPATIENT,23 H.
```

After choosing a file and two entries to compare, enter **YES** at the “MERGE THESE ENTRIES AFTER COMPARING THEM? NO//” prompt to merge the entries as well as compare them.

You now specify which of the two entries are used to supply the default values. The entry you choose is the “merge to” entry that contains the merged data, as shown in [Figure 249](#):

**Figure 249: Transferring File Entries—Choosing which File Entry will Serve as the Default Entry**

```
NOTE: RECORDS WILL BE MERGED INTO THE ENTRY SELECTED FOR THE DEFAULT.
WHICH ENTRY SHOULD BE USED FOR DEFAULT VALUES (1 OR 2)? 1
*** RECORDS WILL BE MERGED INTO FMPATIENT,23 A.
```

You next indicate the disposition of the “merge from” entry. You can choose to delete it or retain it unmodified. You can also redirect pointers that point to the “merged from” entry to point to the “merge to” entry. Free text pointers are *not* redirected.

**Figure 250: Transferring File Entries—Deleting the “Merged From” File Entry**

```
DO YOU WANT TO DELETE THE MERGED FROM ENTRY AFTER MERGING ? YES
DO YOU WANT TO REPOINT ENTRIES POINTING TO THIS ENTRY? YES
ENTER FILE TO EXCLUDE FROM REPOINT/MERGE: <ENTER>
```



**NOTE:** You can also choose to exclude pointers from specified files in the repointing process. In this example ([Figure 250](#)), all pointers are repointed.

**Figure 251: Transferring File Entries—Setting Up the Merge Output**

```
DO YOU WANT TO DISPLAY ONLY THE DISCREPANT FIELDS? NO// <ENTER>
DEVICE: <ENTER>
```

Press the **Enter** key at the “DEVICE:” prompt so that you can continue to control how the merge is to proceed. Since the merge is interactive, it is *not* appropriate to release control to a printer or other output device. While working at a keyboard printer is possible, it is *not recommended*, because you are *not* able to monitor the selection of data values that is described below.

VA FileMan displays the data from the entries being merged. Brackets ([ ]) indicate the values that are used to create the final merged entry. If the data values in both entries are the same, no brackets are shown. To start, the values for “FMPATIENT,23 A.” are bracketed because that entry was chosen as the default entry. To switch the value that goes into the merged entry, enter that field’s number.

Figure 252 shows the initial defaults in brackets and the response to switching the value for the SSN field:

**Figure 252: Transferring File Entries—Merge Output (1 of 2)**

COMPARISON OF SCHOLAR FILE ENTRIES			FEB 14, 1991	11:59	PAGE 1
SCHOLAR			[FMPATIENT,23 A.]	FMPATIENT,23 H.	
-----					
1.	NAME	[FMPATIENT,23 A.]	FMPATIENT,23 H.		
2.	SSN	[000-99-9999]	000-88-8888		
3.	SUBJECT AREA	[PHILOSOPHY]	MATHEMATICS AND PHILOSOPHY		
4.	DATE OF BIRTH	[1872]			
DEFAULT IS ENCLOSED IN BRACKETS, E.G., [FMPATIENT,23 A.]					
ENTER 1-4 TO CHANGE A DEFAULT VALUE, ^ TO EXIT REPORT, RETURN TO CONTINUE: <b>2</b>					
COMPARISON OF SCHOLAR FILE ENTRIES			FEB 14, 1991	11:59	PAGE 1
SCHOLAR			[FMPATIENT,23 A.]	FMPATIENT,23 H.	
-----					
1.	NAME	[FMPATIENT,23 A.]	FMPATIENT,23 H.		
2.	SSN	000-99-9999	[000-88-8888]		
3.	SUBJECT AREA	[PHILOSOPHY]	MATHEMATICS AND PHILOSOPHY		
4.	DATE OF BIRTH	[1872]			
DEFAULT IS ENCLOSED IN BRACKETS, E.G., [FMPATIENT,23 A.]					
ENTER 1-4 TO CHANGE A DEFAULT VALUE, ^ TO EXIT REPORT, RETURN TO CONTINUE: <b>&lt;ENTER&gt;</b>					



**NOTE:** The DATE OF BIRTH field (Figure 252) from **FMPATIENT,23 H.** is bracketed because the default entry has no value for that field; it is **NULL**. When the default (“merged to”) entry has no data in a field, data is always brought over from the “merged from” entry. You have no control over this aspect of the merging. Even if you attempt to switch the data value to the **NULL** value and remove the brackets, the data is still brought into the “merged to” field.

If there is more than one screen of field information, VA FileMan displays additional screens. In this case, there is one field that is a Multiple. The **Compare/Merge File Entries** option neither displays data from **WORD-PROCESSING** or **Multiple** fields nor allows the selection of data for the final “merge to” entry. The number of subentries in the Multiples of the two fields is displayed:

**Figure 253: Transferring File Entries—Merge Output (2 of 2)**

```

COMPARISON OF PATIENT FILE ENTRIES      FEB 14, 1991  11:59  PAGE 2
SCHOLAR          [FMPATIENT,23 A.]      FMPATIENT,23 H.
-----
NOTE: MULTIPLES WILL BE MERGED INTO THE TARGET RECORD.

1.  [ ] TOPICS [ ]          [ ] 5 ENTRIES [ ]          [ ] 4 ENTRIES [ ]

ENTER RETURN TO CONTINUE: <ENTER>

```

The merging of data for a **WORD-PROCESSING** field or for a Multiple and its subfields is done in the same way that the **Transfer File Entries** option does it.

Although VA FileMan is ready to perform the merge now, you are prompted with other options—just in case you are *not* ready. To cancel the merge, enter the caret (^) at the “ACTION:” prompt to exit.

**Figure 254: Transferring File Entries—Merge Options**

```

OK.  [ ] M READY TO DO THE MERGE.

      SELECT ONE OF THE FOLLOWING:
          P PROCEED TO MERGE THE DATA
          S SUMMARIZE THE MODIFICATIONS BEFORE PROCEEDING
          E EDIT THE DATA AGAIN BEFORE PROCEEDING

ACTION:

```

These three options are (Figure 254):

- **P**—[PROCEED](#) to merge the data
- **S**—[SUMMARIZE](#) the modifications before proceeding
- **E**—[EDIT](#) the data again before proceeding

### 14.2.2.1 PROCEED

If you enter **P** or **PROCEED** at the "ACTION:" prompt, then VA FileMan proceeds to merge the data specified:

**Figure 255: Transferring File Entries—Merge PROCEED Option**

```
ACTION: P <ENTER> ROCEED TO MERGE THE DATA.  
  
I WILL NOW MERGE ALL SUBFILES IN THIS FILE ...  
THIS MAY TAKE SOME TIME, PLEASE BE PATIENT.  
  
I WILL NOW REPOINT ALL FILES THAT POINT TO THIS ENTRY ...  
THIS MAY TAKE SOME TIME, PLEASE BE PATIENT.  
  
GATHERING FILES AND CHECKING [PT] NODES  
  
MERGING ENTRIES  
  
MERGE COMPLETE  
  
DELETING FROM ENTRY
```

The merging of the data is now complete.



**CAUTION:** Be careful when using this option, because the "merge from" entry can be deleted, and data could be lost.

### 14.2.2.2 SUMMARIZE

To review the changes that are made to the “merge to” entry, enter an **S** or **SUMMARIZE** at the “ACTION:” prompt *before* proceeding to merge, as shown in [Figure 256](#):

**Figure 256: Transferring File Entries—Merge SUMMARIZE Option**

```
ACTION: S <ENTER> UMMARIZE THE MODIFICATIONS BEFORE PROCEEDING
SUMMARY OF MODIFICATIONS TO FMPATIENT,23 A.
FIELD   OLD VALUE  NEW VALUE
-----
2.  SSN    000-99-9999  000-88-8888
4.  DATE OF BIRTH  1872
NOTE: MULTIPLES WILL BE MERGED INTO THE TARGET RECORD
ENTER RETURN TO CONTINUE: <ENTER>
```

### 14.2.2.3 EDIT

Enter an **E** or **EDIT** at the “ACTION:” prompt to change the decisions made earlier. If you choose this action, you are again shown the values in both entries, with your current selections in brackets, and can switch the data going into the “merge to” entry.

## 15 Extract Tool

Using the Extract Tool, you can move or copy data from logical records in VA FileMan files to a new VA FileMan file. This new file may either *permit* users to modify its contents or *prevent* users from modifying its contents and can be available for online inquiries and print processes. If this new file is used to store archived data, any options and utilities that create new entries or update existing entries are restricted. Options and utilities that update the data dictionary are also restricted.

### 15.1 Extract Overview

The following is an overview of the process of using the Extract Tool to extract entries from a file:

1. Identify the files and fields from which to extract data by using information in the data dictionary listings.
2. Build a destination file by creating a new field for each field in the source file.
3. Select the source file entries from which data is extracted by creating a **SEARCH/SORT** template.
4. Select the fields from which data is extracted by creating an **EXTRACT** template.
5. Move the extracted data to the destination file by using the **Update Destination File** [DIAX UPDATE] option.
6. Purge the selected entries from the active database.

### 15.2 Important Items to Note

Before beginning the extract process, consider each of the following important facts about the Extract Tool:

- An extract activity is file-specific, *not* user-specific. Anyone with access to the file and the Extract Tool options can complete or change an existing extract activity.
- When the extracted data is moved to the destination file, the source entries in the primary file are blocked from selection.
- A Subfile *cannot* be extracted by itself. At least one field from every Multiple level *above* the Subfile *must* also be extracted. If no field is extracted at the next higher level, the **.01** field at that level will automatically be extracted.
- You can extract identifiers and/or key fields from the source file to the destination file, so that records in the destination file can be uniquely identified.

- An **EXTRACT** template is the only type of **PRINT** template that can be used by the **Update Destination File** [DIAX UPDATE] option.
- The ARCHIVAL ACTIVITY (#1.11) file contains a brief history that describes who performed the various extract activity steps and when the steps were completed.
- The extract activity can be canceled at any time before the **Purge Extracted Entries** [DIAX PURGE] option is used.
- The **Purge Extracted Entries** [DIAX PURGE] option deletes all source data in the primary file from which you extracted data.
- Selected entries *cannot* be purged until they have been moved to the destination file.
- A second extract from a file *cannot* be performed until the active extract activity has been completed, either by purging or canceling.

### 15.2.1 Source File

The term source file represents the primary file and any other files that can be referenced by extended pointers. The primary file is the starting file from which you extract your data. The term extract field refers to any field in the source file.

### 15.2.2 Destination File

The term “destination file” represents the VA FileMan file that stores the extracted data. The destination file can be located anywhere on the network that is recognized by the system. To create this file, you can select either of the following two VA FileMan options:

- **Modify File Attributes** [DIMODIFY], which is located on the **VA FileMan** [DIUSER] menu.
- **Modify Destination File** [DIAX MODIFY], which is located on the **Extract Data To Fileman File** [DIAX EXTRACT MENU] menu.

For each extract field in the source file, a corresponding field in the destination file *must* exist. Certain DATA TYPE field values can optionally be resolved to external form before moving the data to the destination file. For example, data extracted from a DATA TYPE field of **POINTER TO A FILE** can be moved to a **FREE TEXT**-type field in the destination file, if external form of data is moved; or such data can be moved to a **NUMERIC**-type field, if the internal value is moved.



**REF:** For more information, see the [“Mapping Information”](#) section.

The destination file uses a file level attribute called ARCHIVE FILE. The following is a description of this flag:

- **YES**—This is an archive file, and users *cannot* modify or delete the data or the data dictionary. Any data dictionary changes may invalidate the archived data.
- **NO (or NULL)**—There are no restrictions on the file.

If you need to update an archive file's data dictionary, you *must* convert the old data to the new data dictionary format.

Updates to an archive file are allowed only through either of the following:

- **Update Destination File** [DIAX UPDATE] extract option
- **EXTRACT^DIAXU** API.



**REF:** For more information on the Extract Tool API (**EXTRACT^DIAXU**), see the *VA FileMan Developer's Guide*.

Only Regular, KWIC, and Soundex-type cross-references are *recommended* for archive files. No other types of cross-references should be created.

If you are building a destination file that will store archived data, set the ARCHIVE FILE flag to **YES** (do this with the **Modify Destination File** [DIAX MODIFY] option). Setting the ARCHIVE FILE flag to **YES** prevents users from modifying or deleting the data in the file or the file's data dictionary while using VA FileMan options or API calls. Users are also prevented from deleting file entries while using VA FileMan options or API calls.

### 15.3 Mapping Information

Mapping information identifies the relationship between the data in the source file and the data in the destination file. When you create your **EXTRACT** template, you will enter the name of the field in the source file and identify its intended location in the destination file. You will need to ensure that the DATA TYPE field value of the field in the destination file is compatible with the DATA TYPE field value of the extract field. The compatibility of the DATA TYPE field values is validated when the fields are specified during template creation.

[Table 97](#) recommends the DATA TYPE field values to use, depending on the DATA TYPE field value of the extract field:

**Table 97: Extract Tool—DATA TYPE Field Value Recommendations**

<b>DATA TYPE Field Value of Extract Field</b>	<b>DATA TYPE Field Value of Destination Field</b>
<b>DATE/TIME</b>	1) <b>DATE/TIME</b> , internal form of data is moved. 2) <b>FREE TEXT</b> , external form of data is moved.
<b>NUMERIC</b>	<b>NUMERIC</b> or <b>FREE TEXT</b> .
<b>SET OF CODES</b>	1) <b>FREE TEXT</b> , if external form of the <b>SET OF CODES</b> is moved. 2) <b>SET OF CODES</b> , if internal form of the <b>SET OF CODES</b> is moved. Users <i>must</i> make sure the <b>SET OF CODES</b> fields are identical in both the source file and the destination file data dictionaries.
<b>FREE TEXT</b>	<b>FREE TEXT</b> .
<b>WORD-PROCESSING</b>	<b>WORD-PROCESSING</b> .
<b>COMPUTED</b>	<b>FREE TEXT</b> , <b>DATE/TIME</b> , or <b>NUMERIC</b> .
<b>POINTER TO A FILE</b>	1) <b>NUMERIC</b> , if internal form of data is moved. 2) Non-pointer field type ( <b>FREE TEXT</b> , <b>NUMERIC</b> , or <b>DATE/TIME</b> ), if external form of data is moved.
<b>VARIABLE-POINTER</b>	1) Non-pointer field type, if external form of data is moved; (if the <b>.01</b> fields of the pointed-to files have different DATA TYPE field values, DATA TYPE field values of destination field should be <b>FREE TEXT</b> ).

<b>DATA TYPE Field Value of Extract Field</b>	<b>DATA TYPE Field Value of Destination Field</b>
	2) <b>FREE TEXT</b> , if internal form of data is moved.
<b>MUMPS</b>	<b>MUMPS.</b>
<b>Multiples</b>	<b>Multiples.</b>
<b>Backward Pointers</b>	<b>Multiples.</b>

The following are additional guidelines that you *must* follow while creating your destination file:

- If you are extracting a **SET OF CODES**-type field and you are mapping it to a **FREE TEXT**-type field, use a maximum length of the same—or greater than—length as the longest external value in the SET OF CODES field. If you are mapping the **SET OF CODES**-type field to a **SET OF CODES**-type field, create the corresponding field in the destination file, using the same specifications as the extract field.
- If you are extracting DATA TYPE field values of any of the following:
  - **FREE TEXT**
  - **DATE/TIME**
  - **NUMERIC**
  - **WORD-PROCESSING**
  - **MUMPS**

Create the corresponding field in the destination file, using the same specifications as the extract field.

- If you are extracting a Multiple-type field, create the corresponding field in the destination file as a Multiple-type field. Multiples in the source file are moved to Multiples in the destination file, following the DATA TYPE field value recommendations listed in [Table 97](#). The structure of the Multiple in the destination file should be the same as that in the source file down to the lowest level Multiple that you extract. When extracting data in a Subfile, at least one field from every Multiple level above the Subfile *must* also be extracted. If you do *not*

specify a field to extract at a higher level, the **.01** field at that level is automatically extracted.

- If you are extracting a Backward Extended Pointer-type field, create the corresponding field in the destination file as a Multiple. The Extract Tool resolves Backward Pointers. Thus, their values are moved to Multiples in the destination file.
- If the field you are extracting has an **OUTPUT** transform, make sure the **INPUT** transform of the destination field can receive the data in the format generated by the **OUTPUT** transform.

## 15.4 ARCHIVAL ACTIVITY File

To learn about the status of an extract activity you can enter a question mark at most of the prompts in the **Extract Data To Fileman File** [DIAX EXTRACT MENU] menu options (aka **Extract Tool** menu option). Using the **Inquire to File Entries** [DIINQUIRE] option on the ARCHIVAL ACTIVITY (#1.11) file yields information about past or pending activities. Those activities created by the Extract Tool are referred to as extract activities. The amount of information displayed depends on the status of the extract activity. The ARCHIVAL ACTIVITY (#1.11) file contains the following information:

- **DUZ** of the individual performing the extract activity.
- Status of the extract activity (e.g., EDITED or UPDATED).
- Dates on which the activities were performed.
- Number of entries extracted.
- Source file number.
- **SEARCH/SORT** and **PRINT** templates used in the extract activity.

Beginning with VA FileMan 20.0 and later, the ARCHIVAL ACTIVITY (#1.11) file contains data about both archiving and extract activities. A file can have only *one* active activity at a time, either of the following:

- Archiving activity
- Extract activity

You can only select an extract activity from the **Extract Data To Fileman File** [DIAX EXTRACT MENU] menu (aka Extract Tool menu). When you use the **Inquire to File Entries** [DIINQUIRE] option, the word **EXTRACT** appears for all extract activities.

## 15.5 Extract Steps

The order of the options on the **Extract Data To Fileman File** [DIAX EXTRACT MENU] menu (aka Extract Tool menu), reflects the sequence of steps in which you ordinarily perform your extract activity.

To access the **Extract Data To Fileman File** [DIAX EXTRACT MENU] menu (aka Extract Tool menu options, start at the **Other Options** [DIOTHER] menu.

[Figure 257](#) is a sample of the dialog that you encounter.

**Figure 257: Extract Tool—Extract Data To Fileman File [DIAX EXTRACT MENU] Menu Options**

```
SELECT VA FILEMAN <TEST ACCOUNT> OPTION: OTHER OPTIONS

FILEGRAMS ...
AUDIT MENU ...
SCREENMAN ...
STATISTICS
VA FILEMAN MANAGEMENT ...
DATA EXPORT TO FOREIGN FORMAT ...
EXTRACT DATA TO FILEMAN FILE ...
IMPORT DATA
BROWSER
DATA ACCESS CONTROL ...
DATA MAPPING ...

SELECT OTHER OPTIONS <TEST ACCOUNT> OPTION: EXTRACT <ENTER> DATA TO
FILEMAN FILE

1 SELECT ENTRIES TO EXTRACT
2 ADD/DELETE SELECTED ENTRIES
3 PRINT SELECTED ENTRIES
4 MODIFY DESTINATION FILE
5 CREATE EXTRACT TEMPLATE
6 UPDATE DESTINATION FILE
7 CANCEL EXTRACT SELECTION
8 PURGE EXTRACTED ENTRIES
9 VALIDATE EXTRACT TEMPLATE

SELECT EXTRACT DATA TO FILEMAN FILE <TEST ACCOUNT> OPTION:
```

## 15.5.1 Select Entries to Extract Option (1 of 9)

The **Select Entries to Extract** [DIAX SELECT] option initiates the extract activity. In this option, the entries are selected and stored in a template and an entry in the ARCHIVAL ACTIVITY (#1.11) file is created. Entries are selected in the same manner as the **Search File Entries** [DISEARCH] option.



**REF:** For guidance on selecting entries, see the "Search" section in the *VA FileMan User Manual*.

The **Select Entries to Extract** [DIAX SELECT] option performs the following functions:

1. During the search phase, the search criteria for selecting entries are specified and *must* be stored in a template.



**NOTE:** If you want to extract a subentry contained in a Multiple field, you *must* extract the entire entry.

2. During the sort phase, which is indicated by the "SORT BY" prompt, you can enter additional restrictions on the entries to be selected. If no further restrictions are required, simply accept the defaults provided at the "SORT BY" and "START WITH" prompts.
3. During the print phase, which is indicated by the "PRINT FIELD" prompt, VA FileMan gathers the entries specified in the search and sort phases and adds the internal entry numbers of the selected entries to the **SEARCH** template. Although specifying print fields is *not* required, the print process *must* be run to completion. Simply press the **Enter** key at the "PRINT FIELD" prompt or specify actual fields to print a report of identifying information for the extracted records.

In the sample dialog in [Figure 258](#), notice the sequence in which the search, sort, and print prompts appear:

**Figure 258: Extract Tool—Search, Sort, and Print Options When Selecting Entries to Extract**

```

EXTRACT FROM WHAT FILE: CHANGE
-A- SEARCH FOR CHANGE FIELD: .01 <ENTER> NO.
-A- CONDITION: LESS THAN
-A- LESS THAN: 900

-B- SEARCH FOR CHANGE FIELD: <ENTER>

IF: A// <ENTER> NO. LESS THAN 900

STORE RESULTS OF SEARCH IN TEMPLATE: ZZTEST TEMPLATE
ARE YOU ADDING ZZTEST TEMPLATE AS A NEW SORT TEMPLATE? NO// Y <ENTER> (YES)

SORT BY: VERSION
START WITH VERSION: FIRST// <ENTER>
WITHIN VERSION, SORT BY: <ENTER>
FIRST PRINT FIELD: .01 <ENTER> NO.
THEN PRINT FIELD: VERSION
THEN PRINT FIELD: PROGRAMMER
THEN PRINT FIELD: <ENTER>
HEADING: CHANGE EXTRACT SEARCH REPLACE <ENTER>
DEVICE: <ENTER>

```

The resulting output looks like [Figure 259](#):

**Figure 259: Extract Tool—Select Entries to Extract Output**

```

CHANGE EXTRACT SEARCH          AUG 30, 1992  10:59  PAGE 1
NO.          VERSION          PROGRAMMER
-----
101          17.10          FMPROGRAMMER,25
102          17.32          FMPROGRAMMER,26
103          17.35          FMPROGRAMMER,26
      3 MATCHES FOUND.

```

After you use this option, VA FileMan marks the ARCHIVAL ACTIVITY (#1.11) file entry with a status of SELECTED. If an unfinished extract activity exists for a file and you select this same file for a subsequent extract activity, you see the message shown in [Figure 260](#):

**Figure 260: Extract Tool—Example of a Notice Regarding an Outstanding Extract Activity**

```

THERE IS ALREADY AN OUTSTANDING EXTRACT ACTIVITY.
PLEASE FINISH IT OR CANCEL IT.

```

Since the ARCHIVAL ACTIVITY (#1.11) file maintains a record of both your extract and archiving activities, you see the italicized word *archiving* whenever the outstanding file activity is an archiving one. To add or delete entries from the **SEARCH/SORT** template you just created, use the **Add/Delete Selected Entries** [DIAX ADD/DELETE] option.

## 15.5.2 Add/Delete Selected Entries Option (2 of 9)

When you wish to add entries to the extract activity or you wish to delete an entry or entries, use the **Add/Delete Selected Entries** [DIAX ADD/DELETE] option. This option provides an easy way to eliminate undesired entries or to add the ones needed to your list of entries to extract. Like the **Inquire to File Entries** [DIINQUIRE] option, this option displays a selected entry and then asks if you want to delete or add the entry. If you modify the list, then the activity's status in the ARCHIVAL ACTIVITY (#1.11) file changes from **SELECTED** to **EDITED**.

You can only use this option to modify your list before the entries are moved to the destination file. If you need to change the extract activity list after the destination file is updated, you need to cancel the extract activity and start a new extract activity.

To use the **Add/Delete Selected Entries** [DIAX ADD/DELETE] option, select the extract activity you want to modify by entering the archival activity number, source file number, or source file name. Then select the entry to be added or deleted.

[Figure 261](#) depicts the sequence you follow when adding an entry to the extract activity:

**Figure 261: Extract Tool—Using the ADD/DELETE SELECTED ENTRIES Option**

```
SELECT EXTRACT OPTION: ADD/DELETE <ENTER> SELECTED ENTRIES
SELECT EXTRACT ACTIVITY: ?
ANSWER WITH ARCHIVAL ACTIVITY ARCHIVE NUMBER, OR FILE
CHOOSE FROM:
3 CHANGE 08-30-92 SELECTED SELECTOR:FMEMPLOYEE,2
EXTRACT

SELECT EXTRACT ACTIVITY: 3 <ENTER> 08-30-92 SELECTED
SELECTOR:FMEMPLOYEE,2 EXTRACT

SELECT CHANGE NO.: 330
NO.: 330 VERSION: 17.09
PROGRAMMER: FMPROGRAMMER,27 ROUTINE: DIL2
DATE CHANGED: OCT 24, 1995

ADD THIS ENTRY TO THE EXTRACT SELECTION? YES// <ENTER>
```



**NOTE:**

- Entering two question marks (??) at the “Select EXTRACT ACTIVITY:” prompt displays a list of file entries.
- The phrase “\*on EXTRACT list\*” appears next to those entries that are currently part of the extract activity.
- The “DELETE this entry...?” prompt appears whenever you select an entry that is currently on the extract list.
- The “ADD this entry...?” prompt appears whenever you select an entry that is *not* among the items on the list.

### 15.5.3 Print Selected Entries Option (3 of 9)

To display the list of entries you have selected, use the **Print Selected Entries** [DIAX PRINT] option. This option uses the standard VA FileMan interface for printing.



**REF:** For guidance on printing entries, see the “Print: How to Print Reports from Files” section in the *VA FileMan User Manual*.

[Figure 262](#) depicts the type of dialog you encounter when printing a list of entries to be extracted:

**Figure 262: Extract Tool—Using the PRINT SELECTED ENTRIES Option**

```

SELECT EXTRACT OPTION: PRINT <ENTER> SELECTED ENTRIES
SELECT EXTRACT ACTIVITY: 3 <ENTER> CHANGE 08-30-92  EDITED
SELECTOR:FMEMPLOYEE,J  EXTRACT

ENTER A REGULAR PRINT TEMPLATE NAME OR FIELDS YOU WISH TO SEE
PRINTED ON THIS REPORT OF RECORDS TO BE EXTRACTED.

FIRST PRINT FIELD: [ZZTEST TEMPLATE]

```

The output looks like [Figure 263](#):

**Figure 263: Extract Tool—PRINT SELECTED ENTRIES Option Output**

CHANGE EXTRACT ACTIVITY NO.	VERSION	AUG 30, 1992 11:09 PROGRAMMER	PAGE 1
-----	-----	-----	-----
101	17.10	FMPROGRAMMER,25	
102	17.32	FMPROGRAMMER,25	
103	17.35	FMPROGRAMMER,25	
330	17.09	FMPROGRAMMER,30	

### 15.5.4 Modify Destination File Option (4 of 9)

You can use either the **Modify File Attributes** [DIMODIFY] option or the **Modify Destination File** [DIAX MODIFY] options when you are ready to create the destination file that receives your extracted data. You can also use these options to correct discrepancies that you noticed while you were building your **EXTRACT** template. The two options are nearly identical. However, one major difference exists: the **Modify Destination File** [DIAX MODIFY] option prompts for a new file attribute: ARCHIVE FILE (see the "[Destination File](#)" section). If you use the **Modify File Attributes** [DIMODIFY] option to create the destination file, you need to access the **Modify Destination File** [DIAX MODIFY] option to set the ARCHIVE FILE flag.

[Figure 264](#) is a sample of the type of dialog that you encounter when modifying your destination file:

**Figure 264: Extract Tool—Using the MODIFY DESTINATION FILE Option (1 of 2)**

```
SELECT EXTRACT OPTION: MODIFY <ENTER> DESTINATION FILE

THIS OPTION ALLOWS YOU TO BUILD A FILE WHICH WILL STORE DATA
EXTRACTED FROM OTHER FILES. WHEN CREATING FIELDS IN THE
DESTINATION FILE, ALL DATA TYPES ARE SELECTABLE. HOWEVER, ONLY A
FEW DATA TYPES ARE ACCEPTABLE FOR RECEIVING EXTRACTED DATA.

PLEASE SEE YOUR USER MANUAL FOR MORE GUIDANCE ON BUILDING THE
DESTINATION FILE.

MODIFY WHAT FILE: CHANGE EXTRACT
```

From this point on, you see the usual dialog while creating a new file and creating fields.

Once you have finished creating your destination file, you see the dialog in [Figure 265](#):

**Figure 265: Extract Tool—Using the MODIFY DESTINATION FILE Option (2 of 2)**

```
SELECT FIELD: <ENTER>
ARCHIVE FILE? NO// ?

ENTER EITHER [Y] OR [N]

ARCHIVE FILE? NO// ??
  [YES] WILL NOT ALLOW MODIFICATIONS OR DELETIONS OF DATA OR DATA
  DICTIONARY
  [NO] WILL PLACE NO RESTRICTIONS ON THE FILE.

ARCHIVE FILE? NO// <ENTER>

SELECT EXTRACT OPTION:
```

### 15.5.5 Create Extract Template Option (5 of 9)

When selecting destination fields for data to be extracted into, keep in mind that the **INPUT** transforms of the destination fields are executed for each field value. For an extracted record, the value of each field in the record is tested against the **INPUT** transform of its destination field. If any field fails the **INPUT** transform, the extract for the entire record fails. Make sure the **INPUT** transforms on the destination fields are appropriate for the data being extracted.



**NOTE:** If you are extracting a subrecord using the **EXTRACT^DIAXU** entry point and its **FILING\_LEVEL** parameter, and a value fails the **INPUT** transform, only the extract of the Subrecord fails.

When you are ready to build an **EXTRACT** template, you *must* select the **Create Extract Template** [DIAX CREATE] option. Using this option, you identify not only the field you wish to extract from the source file but also its corresponding field in the destination file. The **EXTRACT** template is the only type of **PRINT** template used in the **Update Destination File** [DIAX UPDATE] option.

Building an **EXTRACT** template requires entering valid field numbers or field names at the “EXTRACT FIELD” prompt. Since VA FileMan stores **EXTRACT** templates in the PRINT TEMPLATE (#.4) file, this option uses the term “PRINT TEMPLATE” instead of “EXTRACT TEMPLATE” in the dialog. For each extract field that you identify in the source file, at the “MAP TO” field prompt, enter the destination file field name or field number that receives the data. Only those fields defined in the **EXTRACT** template appear in the destination file.

Keep in mind that the value of each field in an extracted record is tested against the **INPUT** transform of its destination field. If any value fails its destination field's **INPUT** transform, the extract for the entire record fails. Make sure the **INPUT** transforms on the destination fields are appropriate for the data you are extracting.



**NOTE:** If you are extracting a subrecord using the **EXTRACT^DIAXU** entry point and its **FILING\_LEVEL** parameter, and a value fails the **INPUT** transform, only the extract of the Subrecord fails.

When you arrive at the "STORE EXTRACT LOGIC IN TEMPLATE:" prompt, enter the name that you wish to assign to your new **EXTRACT** template. To edit an existing **EXTRACT** template, on the other hand, simply enter its name at the "FIRST EXTRACT FIELD:" prompt—using the following format:

[[**EXTRACT** **TEMPLATENAME**]

[Figure 266](#) is a sample of the dialog that you encounter when you are ready to build an **EXTRACT** template:

**Figure 266: Extract Tool—Using the CREATE EXTRACT TEMPLATE Option**

```
SELECT EXTRACT OPTION: CREATE <ENTER> EXTRACT TEMPLATE

THIS OPTION LETS YOU BUILD A TEMPLATE WHERE YOU SPECIFY FIELDS TO EXTRACT AND
THEIR CORRESPONDING MAPPING IN THE DESTINATION FILE.

FOR MORE DETAILED DESCRIPTION OF REQUIREMENTS ON THE DESTINATION FILE, PLEASE
SEE YOUR VA FILEMAN USER MANUAL.

OUTPUT FROM WHAT FILE: CHANGE <ENTER> (956 ENTRIES)
DESTINATION FILE: CHANGE EXTRACT <ENTER> (0 ENTRIES)

FIRST EXTRACT CHANGE FIELD: .01 <ENTER> NO.
MAP NO. TO CHANGE EXTRACT FIELD: .01 <ENTER> NO.

THEN EXTRACT CHANGE FIELD: VERSION
MAP VERSION TO CHANGE EXTRACT FIELD: VERSION

THEN EXTRACT CHANGE FIELD: PROGRAMMER
MAP PROGRAMMER TO CHANGE EXTRACT FIELD: PROGRAMMER

STORE EXTRACT LOGIC IN TEMPLATE: CHANGE EXTRACT
ARE YOU ADDING CHANGE EXTRACT AS A NEW PRINT TEMPLATE? NO// YES <ENTER> (YES)
```

While you are creating your **EXTRACT** template, VA FileMan performs a few validation checks. Inspecting the extract field and its corresponding field in the destination file, VA FileMan checks to see if both fields are compatible in several important areas, including:

- Data type
- Minimum length
- Maximum length
- Minimum values
- Maximum values

If a discrepancy exists, VA FileMan displays an error message such as the statement shown in [Figure 267](#):

**Figure 267: Extract Tool—Example of a Notice Regarding a Discrepancy**

```
PROGRAMMER FIELD IN CHANGE EXTRACT FILE SHOULD HAVE A MAXIMUM  
LENGTH OF AT LEAST 30 CHARACTERS.
```

After VA FileMan displays an error message about your destination field, you can continue building your template. You are *not*, however, able to update the destination file until you have corrected the problem.

[Figure 268](#) shows the warning that you see when any source field and its corresponding destination field fail one of the validation checks:

**Figure 268: Extract Tool—Example of the Warning Message When the Validation Check Fails**

```
THE DESTINATION FILE DATA DICTIONARY SHOULD BE MODIFIED PRIOR TO  
ANY MOVEMENT OF EXTRACT DATA!
```

At any “MAP ‘FIELD NAME’ TO ‘FILE NAME’ FIELD:” prompt, entering two question marks (“??”) yields a list of the selectable fields in the destination file. The list gets shorter as fields are selected to ensure that no two extract fields map information to a single field in the destination file.

## 15.5.6 Update Destination File Option (6 of 9)

Once you have used the **Update Destination File** [DIAX UPDATE] option, the extracted data from the source file is moved to the destination file. After you enter the name of the **EXTRACT** template that you wish to use, VA FileMan makes sure the template's mapping information is correct and acceptable and then populates the destination file, adding entries as new records. VA FileMan does *not*, however, check to see if any of those records to be moved already exist in the destination file. Since this two-step process can be quite time-consuming, it can be queued at the "DEVICE:" prompt.

[Figure 269](#) is a sample of the dialog:

**Figure 269: Extract Tool—Using the UPDATE DESTINATION FILE Option**

```
SELECT EXTRACT OPTION: UPDATE <ENTER> DESTINATION FILE
SELECT EXTRACT ACTIVITY: 3 <ENTER> CHANGE 08-31-92      EDITED
SELECTOR:FMEMPLOYEE,J   EXTRACT

YOU MUST ENTER AN EXTRACT TEMPLATE NAME.  THIS EXTRACT TEMPLATE
WILL BE USED TO POPULATE YOUR DESTINATION FILE.
PRINT TEMPLATE: CHANGE EXTRACT <ENTER> **EXTRACT**      (AUG
30,1992)          USER #2  FILE #16000

EXCUSE ME, THIS WILL TAKE A FEW MOMENTS...
CHECKING THE DESTINATION FILE...

IF ENTRIES CANNOT BE MOVED TO THE DESTINATION FILE, AN EXCEPTION
REPORT WILL BE PRINTED.

SELECT A DEVICE WHERE TO PRINT THE EXCEPTION REPORT.

QUEUEING TO THIS DEVICE WILL QUEUE THE UPDATE PROCESS.
EXCEPTION REPORT DEVICE: Q <ENTER> UEUE TO PRINT ON
DEVICE: PRINTER
```

After the destination file has been updated, VA FileMan changes the extract activity status from **SELECTED** or **EDITED** to **UPDATED DESTINATION FILE**. At this point, the entries from the source file are no longer available on lookups. This protective measure prevents you from attempting to edit the selected source file entries, so that they contain the same data as the corresponding destination file entries.

The Exception Report in [Figure 270](#) is printed when the Extract Tool fails to move all data in a source entry into the destination file. A failed **INPUT** transform is one possible cause of such a failure. In this case, the incomplete entry in the destination file is deleted. The source entry is *not* locked, and its internal entry number is deleted from the extract list. The total number of entries extracted is reduced by the total numbers of entries appearing on the exception report.

**Figure 270: Extract Tool—Exception Report**

```
EXTRACT ACTIVITY EXCEPTION REPORT          JUN 27, 1996  PAGE: 1
-----
EXTRACT ACTIVITY: 9          ARCHIVER: FEMPLOYEE,J

THE FOLLOWING ENTRIES IN THE [TEST] FILE WERE NOT MOVED BY THE
EXTRACT TOOL

ENTRY # 9 WAS NOT PROCESSED BECAUSE:
  THE VALUE [NEW] FOR FIELD FTEXT MULT LABEL IN FTEXT MULT SUB-FIELD IN FILE TEST
  IS NOT
  VALID.

ENTER # 30 WAS NOT PROCESSED BECAUSE:
  THE VALUE [NEW] FOR FIELD FTEXT MULT LABEL IN FTEXT MULT SUB-FIELD IN FILE TEST
  IS NOT
  VALID.

*** PLEASE KEEP THIS FOR FUTURE REFERENCE ***
```

The following is a list of recommended steps to take when an exception report is printed:

1. Finish the active extract activity by purging or canceling.
2. Determine the problem with the source entry and fix it.
3. If there are several entries on the exception report, start another extract activity. Your **SEARCH/SORT** template can be reused to use the same search specifications.
4. Adjust the extract list to match the list of entries on the exception report by using the **Add/Delete Selected Entries** [DIAX ADD/DELETE] option.
5. Proceed as before.

For exceptions caused by **INPUT** transforms, keep in mind that the value of each field in an extracted record is tested against the **INPUT** transform of its destination field. If any value fails its destination field's **INPUT** transform, the extract for the entire record fails.

Make sure the **INPUT** transforms on the destination fields are appropriate for the data you are extracting.



**NOTE:** If you are extracting a subrecord using the **EXTRACT^DIAXU** entry point and its **FILING\_LEVEL** parameter, and a value fails the **INPUT** transform, only the extract of the Subrecord fails.

### 15.5.7 Purge Extracted Entries Option (7 of 9)

If you have **DELETE** access to the primary file, you can use the **Purge Extracted Entries** [DIAX PURGE] option to delete extracted data from the primary file (e.g., CHANGE file). After you have purged your entries, VA FileMan updates the ARCHIVAL ACTIVITY (#1.11) file.

If you attempt to purge an extract activity that lacks the status UPDATED DESTINATION FILE, you encounter the message shown in [Figure 271](#):

**Figure 271: Extract Tool—Example of a Notice from VA FileMan When Extract Activity Lacks the Status UPDATED DESTINATION FILE**

```
DATA HAS NOT YET BEEN MOVED TO THE DESTINATION FILE!
```

When purging extracted data, you encounter a dialog much like the one in [Figure 272](#):

**Figure 272: Extract Tool—Using the PURGE EXTRACTED ENTRIES Option (1 of 2)**

```
SELECT EXTRACT OPTION: PURGE <ENTER> EXTRACTED ENTRIES
SELECT EXTRACT ACTIVITY: 3 <ENTER> CHANGE 08-30-92 UPDATED
DESTINATION FILE SELECTOR:FMEMPLOYEE,J EXTRACT
```

If the source file has fields from other files pointing to it, the Extract Tool displays the text shown in [Figure 273](#):

**Figure 273: Extract Tool—Using the PURGE EXTRACTED ENTRIES Option (2 of 2)**

```
THE RECORDS ABOUT TO BE PURGED SHOULD NOT BE [ ]POINTED TO[ ] BY OTHER
RECORDS TO MAINTAIN DATABASE INTEGRITY.

THIS OPTION WILL DELETE DATA FROM BOTH CHANGE
AND FROM THE ARCHIVAL ACTIVITY FILE.
ARE YOU SURE YOU WANT TO CONTINUE? NO// YES

THE ENTRIES WILL BE DELETED IN INTERNAL NUMBER ORDER.

<< 4 ENTRIES PURGED >>
```



**CAUTION:** As you can see (Figure 273), entering a YES response to the “Are you sure you want to continue? NO//” prompt deletes the entries immediately!

### 15.5.8 Cancel Extract Selection Option (8 of 9)

You can cancel an extract activity any time before the entries are purged by using the **Cancel Extract** [DIAX CANCEL] option. If the extract activity status is **UPDATED DESTINATION FILE**, which means the entries have already been moved to the destination file, you see a warning notice. At this point, you can roll back or delete the new entries that were created while using the **Update Destination File** [DIAX UPDATE] option.

After you have canceled an extract activity, VA FileMan deletes the ARCHIVAL ACTIVITY (#1.11) file reference to the extract activity. In addition, you once again can gain access to all of those source entries that VA FileMan locked during the update of your destination file. To extract data *without* purging the source entries, cancel the extract activity to unlock the selected entries in the source file.

You encounter the dialog in Figure 274 while canceling an extract activity:

**Figure 274: Extract Tool—Using the CANCEL EXTRACT SELECTION Option**

```
SELECT EXTRACT OPTION: CANCEL EXTRACT SELECTION
SELECT EXTRACT ACTIVITY: CHANGE <ENTER> 3 CHANGE 08-31-92
  UPDATED DESTINATION FILE      SELECTOR:FMEMPLOYEE,0 EXTRACT

ARE YOU SURE YOU WANT TO CANCEL THIS EXTRACT ACTIVITY? NO// ??
  ENTER YES TO STOP THIS ACTIVITY AND START AGAIN FROM THE BEGINNING.

ARE YOU SURE YOU WANT TO CANCEL THIS EXTRACT ACTIVITY? NO// YES

THIS EXTRACT ACTIVITY HAS ALREADY UPDATED THE DESTINATION FILE.

DELETE THE DESTINATION FILE ENTRIES CREATED BY THIS EXTRACT ACTIVITY? NO// ??
  ENTER YES TO ROLLBACK THE DESTINATION FILE TO ITS STATE BEFORE
  THE UPDATE.

DELETE THE DESTINATION FILE ENTRIES CREATED BY THIS EXTRACT
ACTIVITY? NO// <ENTER>

>>> DONE <<<
```

To cancel an extract selection so that you can start over, enter **YES** at the “Delete the destination file entries...” prompt. Entering **YES** prevents you from sending a duplicate set of entries to the destination file. If, on the other hand, you simply want to cancel the

extract selection, pressing the **Enter** key at the prompt unlocks the source entries and retains the destination file entry.

### 15.5.9 Validate Extract Template Option (9 of 9)

After you have corrected any discrepancies that VA FileMan might have pointed out while you were creating an **EXTRACT** template, you can use the **Validate Extract Template** [DIAX VALIDATE] option to quickly check your **EXTRACT** template's mapping information. The **Validate Extract Template** [DIAX VALIDATE] option does *not* alter anything in your template. [Figure 275](#) is a sample dialog:

**Figure 275: Extract Tool—Using the VALIDATE EXTRACT TEMPLATE Option**

```
SELECT EXTRACT OPTION: VAL <ENTER> IDATE EXTRACT TEMPLATE
SELECT EXTRACT TEMPLATE: CHANGE <ENTER> EXTRACT **EXTRACT**
(AUG 30, 1992) USER #2 FILE #16000

EXCUSE ME, THIS WILL TAKE A FEW MOMENTS...
CHECKING THE DESTINATION FILE...

TEMPLATE LOOKS OK!
```

## 16 Filegrams



**NOTE:** To use the full capabilities of VA FileMan's Filegram procedures, Kernel 6.5 or later *must* be installed on your system.

Filegrams are a feature in VA FileMan intended for use by system managers and software developers.

A Filegram is a process that moves a record (also called an entry) from a file on one computer system to a duplicate file on another, independent computer system. An independent computer system is defined as a system having its own database.

You can move a Filegram by either of the following methods:

- **Locally**—Sending data from the “live” account at a medical center to a “test” account at the same medical center is an example of moving a Filegram locally.
- **Remotely**—Sending data from a computer in the San Francisco Medical Center to a computer in the Salt Lake City Medical Center is an example of moving a Filegram remotely.

The records you can move by a Filegram can be either of the following types:

- **Physical**—Records stored in one file.
- **Logical**—Related fields stored in different files. Logical records are loaded into Filegrams by using VA FileMan's relational navigational syntax (:).

For successful Filegram installation, the recipient system *must* have the following:

- Kernel 6.5 or later.
- The FILEGRAM HISTORY (#1.12) file.
- The Filegram key to the Filegram submenu.
- A file structure that matches the one reflected by the entry in the Filegram. This is the structure existing on the sender's system.

## 16.1 FILEGRAM-Type Templates

The Filegram process requires you to create an **OUTPUT** template, which is stored in the PRINT TEMPLATE (#.4) file along with regular **PRINT** templates. VA FileMan recognizes the **FILEGRAM-type** and regular **PRINT** templates as **OUTPUT** templates, but their similarity ends there.

**i** **NOTE:** Since **FILEGRAM-type** templates are stored in the PRINT TEMPLATE (#.4) file, the dialogs you encounter in the Filegram process refer to a **FILEGRAM-type** template as a **PRINT** template.

Regular **PRINT** templates already created are screened so that you *cannot* accidentally replace an existing **PRINT** template with a FILEGRAM-type template.

## 16.2 Filegram and Archiving Relationship

**FILEGRAM-type** templates are the only kind of template allowed in the archiving process.

**i** **REF:** For a full description of archiving in VA FileMan, see the "[Archiving](#)" section.

## 16.3 Filegram Menu Options

The **Filegrams** [DIFG] menu, which is located under the **Other Options** [DIOTHER] menu, is shown in [Figure 276](#):

**Figure 276: Filegrams [DIFG] Menu Options**

<u>SELECT OTHER OPTIONS</u> <TEST ACCOUNT> <u>OPTION: FILEGRAMS</u>	
CREATE/EDIT FILEGRAM TEMPLATE	[DIFG CREATE]
**> LOCKED WITH XUFILEGRAM	
DISPLAY FILEGRAM TEMPLATE	[DIFG DISPLAY]
**> LOCKED WITH XUFILEGRAM	
GENERATE FILEGRAM	[DIFG GENERATE]
**> LOCKED WITH XUFILEGRAM	
VIEW FILEGRAM	[DIFG VIEW]
SPECIFIERS	[DIFG SPECIFIERS]
**> LOCKED WITH XUFILEGRAM	
INSTALL/VERIFY FILEGRAM	[DIFG INSTALL]
**> LOCKED WITH XUFILEGRAM	

## 16.4 Using Filegrams

The following is a summary of the basic steps needed to send and install a Filegram:

1. The Filegram **sender** creates a [FILEGRAM-type template](#) for a specified file using the **Create/Edit Filegram Template** [DIFG CREATE] option.

Use the **Display Filegram Template** [DIFG DISPLAY] option to review the template.

2. The **sender** can optionally designate specifiers with the **Specifiers** [DIFG SPECIFIERS] option. These are used for matching existing entries with Filegrams.
3. The **sender** then generates a Filegram for a specific entry using the **Generate Filegram** [DIFG GENERATE] option. The Filegram is placed into a MailMan message. The Filegram sender sends this message to an individual or individuals at a remote or local destination.
4. The **recipient** receives the Filegram with MailMan, reading the mail message containing the Filegram, and forwarding it to **S.DIFG-SRV-HISTORY**. This is a special server that loads the message into the recipient's FILEGRAM HISTORY (#1.12) file and sets up the interface between VA FileMan and MailMan on the target system.
5. If you want to install the Filegram on a system other than the one where you received it, instead of immediately forwarding it to the **S.DIFG-SRV-HISTORY** server, forward the message to an individual on the ultimate target system (who in turn forwards it to *their* **S.DIFG-SRV-HISTORY** server).
6. Both the **sender** and the **recipient** can use the **View Filegram** [DIFG VIEW] option to inspect the Filegram.
7. Then, the **recipient** of the Filegram on the target system uses the **Install/Verify Filegram** [DIFG INSTALL] option to install the Filegram into the destination file.
8. **Senders** and **recipients** can delete a Filegram at any time.

The recipient can choose to modify the **S.DIFG-SRV-HISTORY** server or create another server to aid in the installation of Filegrams.



**REF:** For additional information about setting up servers, see the following documentation on the VA Software Document Library (VDL):

- [Kernel application documentation on the VA software Document Library \(VDL\)](#)
- [MailMan application documentation on the VA software Document Library \(VDL\)](#)

## 16.5 Filegram Steps

### 16.5.1 Create/Edit Filegram Template Option

Use the **Create/Edit Filegram Template** [DIFG CREATE] option to create a [FILEGRAM-type template](#) or edit an existing **FILEGRAM-type** template. A [FILEGRAM-type template](#) is like a regular **PRINT** template without any formatting instructions. You always receive the “STORE FILEGRAM LOGIC IN TEMPLATE:” prompt, no matter how many fields you identify.



**REF:** For information about regular **PRINT** templates, see the “Print: How to Print Reports from Files” section in the *VA FileMan User Manual*.

The **Create/Edit Filegram Template** [DIFG CREATE] option is the first step in developing a Filegram; there is no Filegram without the template. This template is also used in the archiving process. Before using this option, you may wish to familiarize yourself with the files and fields involved.

[Figure 277](#) illustrates how to create a [FILEGRAM-type template](#):

**Figure 277: Filegrams—Creating a FILEGRAM Template (1 of 3)**

```
SELECT FILEGRAMS <TEST ACCOUNT> OPTION: CREATE <ENTER> /EDIT FILEGRAM TEMPLATE
OUTPUT FROM WHAT FILE: CHANGE
FIRST SEND CHANGE FIELD: ??
```

You can enter **ALL** at the “FIRST SEND <FILE NAME> FIELD:” prompt, if you want to include all fields in the file in your Filegram. **ALL** can also be used in existing file navigation paths. Enter **[?]** at this prompt to get a listing of existing [FILEGRAM-type template](#) for the selected file.

Figure 278 shows how two question marks (??) at the " prompt requests a list of the fields in the file.

Figure 278: Filegrams—Creating a FILEGRAM Template (2 of 3)

```
FIRST SEND CHANGE FIELD: ??

CHOOSE FROM:
.01  NAME
1    VERSION
2    TAG
3    ROUTINE
4    CHANGE
5    REPORTER (MULTIPLE)
6    DATE CHANGED
7    PROGRAMMER
9    BUG OR FEATURE
10   PURPOSE
11   DESCRIPTION (WORD-PROCESSING)
```

In Figure 279, the PROGRAMMER field is a pointer to the NEW PERSON (#200) file.

Figure 279: Filegrams—Creating a FILEGRAM Template (3 of 3)

```
FIRST SEND CHANGE FIELD: 1 <ENTER> VERSION
THEN SEND CHANGE FIELD: 3 <ENTER> ROUTINE
THEN SEND CHANGE FIELD: 4 <ENTER> CHANGE
THEN SEND CHANGE FIELD: 5 <ENTER> REPORTER (MULTIPLE)
  FIRST SEND REPORTER SUB-FIELD: .01
  THEN SEND REPORTER SUB-FIELD: <ENTER>
THEN SEND CHANGE FIELD: 6 <ENTER> DATE CHANGED
THEN SEND CHANGE FIELD: 7 <ENTER> PROGRAMMER
THEN SEND CHANGE FIELD: 11 <ENTER> DESCRIPTION (WORD-PROCESSING)
THEN SEND CHANGE FIELD: <ENTER>
STORE FILEGRAM LOGIC IN TEMPLATE: ZZTEST FILEGRAM
ARE YOU ADDING ZZTEST FILEGRAM AS A NEW PRINT TEMPLATE? NQ// Y <ENTER> (YES)
```

The template for your Filegram is now set up. Edit this template just like you would any other **PRINT** template.



**NOTE:** You do *not* have to include the **.01** field, because it is automatically included for use as a lookup value.

To send logical records in a Filegram, simply use file navigation to and from existing files at the "SEND FIELD:" prompts.



**REF:** For a discussion of relational navigation using forward and backward pointers, see the "[Relational Navigation](#)" section.

## 16.5.2 Display Filegram Template Option

The **Display Filegram Template** [DIFG DISPLAY] option displays the [FILEGRAM-type template](#) in a two-column format (like the **Inquire to File Entries** [DIINQUIRE] option). Multiple-type fields are shown last in the display, no matter what their field number.

[Figure 280](#) is an example of the output produced by the **Display Filegram Template** option:

**Figure 280: Filegrams—FILEGRAM Template Output**

```
SELECT FILEGRAMS <TEST ACCOUNT> OPTION: DIS <ENTER> PLAY FILEGRAM TEMPLATE
SELECT FILEGRAM TEMPLATE: ZZTEST FILEGRAM

NAME: ZZTEST FILEGRAM          DATE CREATED: AUG 24, 1989
  READ ACCESS: @                FILE: 1001
  USER #: 29                   WRITE ACCESS: @
  DATE LAST USED: AUG 24, 1989
ORDER: 1                       FILEGRAM FILE: 1001
  LEVEL: 1                     DATE LAST STORED: AUG 24, 1989
FIELD ORDER: 1                 FIELD NUMBER: .01
CAPTION (C): NAME
FIELD ORDER: 2                 FIELD NUMBER: 1
CAPTION (C): VERSION
FIELD ORDER: 3                 FIELD NUMBER: 3
CAPTION (C): ROUTINE
FIELD ORDER: 4                 FIELD NUMBER: 4
CAPTION (C): CHANGE
FIELD ORDER: 5                 FIELD NUMBER: 6
CAPTION (C): DATE CHANGED
FIELD ORDER: 6                 FIELD NUMBER: 7
CAPTION (C): PROGRAMMER
FIELD ORDER: 7                 FIELD NUMBER: 11
CAPTION (C): DESCRIPTION
ORDER: 2                       FILEGRAM FILE: 1001.05
  LEVEL: 2                     PARENT: 1001
  CROSS-REFERENCE: MULTIPLE    USER RESPONSE TO GET HERE: REPORTER
  DATE LAST STORED: AUG 24, 1989
FIELD ORDER: 1                 FIELD NUMBER: .01
CAPTION (C): REPORTER

FIRST PRINT FIELD: S DIFGT=315 D FG^DIFGB;X//
COMPILED (C): N
```

The code in the FIRST PRINT FIELD has special meaning to VA FileMan.

### 16.5.3 Specifiers Option

The Filegram sender uses the **Specifiers** [DIFG SPECIFIERS] option to identify a particular field in the file as a reference point to use when installing the Filegram. The value of this field in the Filegram is compared to values in the entries at the target site. The values *must* match for a Filegram to be installed. If the specifier has a unique value for every entry in the file and is cross-referenced, that cross-reference is used to locate an entry. This reduces the search time and increases accuracy. Specifiers can be compared to identifiers: unlike identifiers, which are used for user interaction purposes, specifiers are used for transaction purposes. Specifiers are optional.

The dialog in [Figure 281](#) creates a specifier in a sample PATIENT (#2) file:

**Figure 281: Filegrams—Example of Creating a Specifier (1 of 2)**

SELECT FILEGRAM OPTION: <b>5 &lt;ENTER&gt;</b> SPECIFIERS
OUTPUT FROM WHAT FILE: <b>PATIENT</b>
SELECT FIELD: <b>3 &lt;ENTER&gt;</b> SSN
WANT TO MAKE SSN A SPECIFIER? NO// <b>YES</b>
IS THE VALUE OF THIS FIELD UNIQUE FOR EACH ENTRY? NO// <b>YES</b>

Answer **YES** only if you have a regular cross-reference on the field that you are making a specifier. A field can be a specifier *without* being unique.

If you answer **YES**, a dialog like [Figure 282](#) occurs:

**Figure 282: Filegrams—Example of Creating a Specifier (2 of 2)**

SELECT ONE OF THE FOLLOWING:
1 C REGULAR
IF ONE OF THE ABOVE PROVIDES A DIRECT LOOK-UP BY SSN, PLEASE ENTER ITS NUMBER OR NAME: <b>1</b>

To delete a specifier:

**Figure 283: Filegrams—Deleting a Specifier**

SELECT FIELD: <b>3 &lt;ENTER&gt;</b> SSN
SSN IS ALREADY A SPECIFIER. DO YOU WANT TO DELETE IT? NO// <b>YES</b>

## 16.5.4 Generate Filegram Option

A Filegram sender uses the **Generate Filegram** [DIFG GENERATE] option. Be sure that the **DUZ** correctly identifies the Filegram sender; the **DUZ** is used to identify the Filegram's sender to the recipient. The option creates a Filegram in MailMan message format after you designate a file, **FILEGRAM-type** template, and a file entry.

Concurrently, it creates a record in the FILEGRAM HISTORY (#1.12) file, which points to the MESSAGE (#3.9) file. The record created in the FILEGRAM HISTORY (#1.12) file is called a Filegram history. The Filegram history allows VA FileMan to differentiate between a Filegram message and a mail message. After the Filegram is placed into the mail message, you can send it to an individual at any established address.

You can only send one entry at a time. [Figure 284](#) illustrates the generation of a Filegram:

**Figure 284: Filegrams—Example of Generating a Filegram**

```
SELECT FILEGRAM OPTION: GENERATE FILEGRAM
OUTPUT FROM WHAT FILE: CHANGE
SELECT FILEGRAM TEMPLATE: ZZTEST FILEGRAM
SELECT CHANGE NO.: 334
SEND MAIL TO: <REDACTED>.VA.GOV
AND SEND TO: <ENTER>
```

## 16.5.5 Receiving Filegrams with MailMan

Filegram messages do *not* appear as NEW mail at the receiving site. After the mail message is received, the Filegram recipient *must* read the mail message. Then, it should be forwarded to **S.DIFG-SRV-HISTORY** on the target system (i.e., the system on which the Filegram is installed). This is a special server used to load the message into the recipient's FILEGRAM HISTORY (#1.12) file and to set up the interface between VA FileMan and MailMan at the target system.

[Figure 285](#) is an example of what the recipient would see when reading and forwarding a Filegram:

**Figure 285: Filegrams—Example of a Filegram Received and Forwarded**

```
SUBJ: FILEGRAM FOR ENTRY #334 IN CHANGE FILE (1001). [#186309]
25 AUG 89 10:00 20 LINES
FROM: SITE,MANAGER IN  IN  BASKET PAGE 1
-----
$DAT^CHANGE^1001^N^
CHANGE^1001^L=334
  BEGIN:CHANGE^1001@1
    SPECIFIER:VERSION^1=17.4
  END:CHANGE^1001
  VERSION^1=17.4
  ROUTINE^3=DICATT5
  CHANGE^4=CHANGED DIED TO DIE0
  DATE CHANGED^6=APR 13, 1987
  PROGRAMMER^7=FMPROGRAMMER
  DESCRIPTION^11=WP
 THIS CHANGE ENABLES INCREDIBLY WONDERFUL THINGS
 TO OCCUR.
.
  REPORTER^5^L=FMEMPLOYEE,10
  BEGIN:REPORTER^1001.05@2
  END:REPORTER^1001.05
  REPORTER^.01=FMEMPLOYEE,10
^
$END DAT

ENTER MESSAGE ACTION (IN IN BASKET): IGNORE// F
FORWARD MAIL TO: S.DIFG-SRV-HISTORY

SENDING A MESSAGE
```

## 16.5.6 View Filegram Option

Entries are made into the FILEGRAM HISTORY (#1.12) file at the sending site when the Filegram is generated and at the receiving site by the S.DIFG-SRV-HISTORY server. The **View Filegram** [DIFG VIEW] option allows the Filegram sender or recipient to inspect the Filegram. Select this option and answer the "Select FILEGRAM HISTORY:" prompt with a question mark to get a listing of available Filegram histories. You can see the Filegram history just created by using the **<Spacebar> <Enter>** or another Filegram history by entering its internal entry number or date/time.

[Figure 286](#) shows you what a simple (*without* pointers) Filegram looks like. The **View Filegram** [DIFG VIEW] option shows this information to a Filegram's sender. The receiver sees additional information about the transmission of the mail message including the network mail path taken to the target system.

**Figure 286: Filegrams—Example of a Simple Filegram (*without* Pointers)**

```
FILEGRAM FOR ENTRY #334 IN CHANGE FILE (#1001).  
  
SENT ON 25 AUG 1989 @ 09:00 BY FMUSER,FOUR  
  
$DAT^CHANGE^1001^N^  
CHANGE^1001^L=334  
  BEGIN:CHANGE^1001@1  
    SPECIFIER:VERSION^1=17.4  
  END:CHANGE^1001  
  VERSION^1=17.4  
  ROUTINE^3=DICATT5  
  CHANGE^4=CHANGED DIED TO DIE0  
  DATE CHANGED^6=APR 13, 1987  
  DESCRIPTION^11=WP  
THIS CHANGE ENABLES INCREDIBLY WONDERFUL THINGS  
TO OCCUR.  
.  
  REPORTER^5^L=FMPATIENT,10  
    BEGIN:REPORTER^1001.05@2  
    END:REPORTER^1001.05  
  REPORTER^.01=FMPATIENT,10  
  ^  
$END DAT
```

## 16.5.7 Install/Verify Filegram Option

The Filegram recipient uses the **Install/Verify Filegram** [DIFG INSTALL] option to install a Filegram from a MailMan message into a file on the target system. Choose a Filegram history by entering its date/time or by using a **<Spacebar><Enter>** (for the last used history) at the "Select FILEGRAM HISTORY:" prompt. The destination file for the Filegram entry *must* be in place on the target system for a successful installation!



**CAUTION:** The installation has a better chance of succeeding if the destination file is a replica of the sending file.

The message "**DONE**" is displayed if the install was successful. If *not* successful, an **UNSUCCESSFUL INSTALLATION** message is returned with an error code.



**REF:** For a list of error codes identifying their meaning, see "**^DIFG**" in the "Filegrams API" section in the *VA FileMan Developer's Guide*.

## 16.5.8 Deleting a Filegram

Delete a Filegram by removing the Filegram's entry from the FILEGRAM HISTORY (#1.12) file. As shown in [Figure 287](#), use the **Enter or Edit File Entries** [DIEDIT] option. Enter the at-sign (@) at the DATE/TIME prompt of the Filegram history you want to delete.

**Figure 287: Filegrams—Deleting a Filegram**

```
SELECT OPTION: ENTER OR EDIT FILE ENTRIES
INPUT TO WHAT FILE: FILEGRAM HISTORY
EDIT WHICH FIELD: ALL// <ENTER>
SELECT FILEGRAM HISTORY: 8-24-1994@11:10:00
DATE/TIME: AUG 24, 1994@11:10:00// @
SURE YOU WANT TO DELETE THE ENTIRE FILEGRAM HISTORY? YES
```

## 17 Archiving



**NOTE:** To use VA FileMan's archiving procedure, Kernel 6.5 or later *must* be installed on your system. To use the **Find Archived Records** option to retrieve records archived with VA FileMan versions earlier than 20.0, Kernel 7.1 or later *must* be installed on your system.

In general computer terms, archiving is a procedure that permits you to remove data from an online database and place that data into a low-cost storage medium (e.g., magnetic tape) for long-term retention.



**CAUTION:** Archiving in VA FileMan is a complete purge that simply clears data from your system. In other words, no data restoration is provided! However, there is a data retrieval option.

This facility is a prototype and supported only as a developer's tool. You *must* access your archiving options directly through VA FileMan from **Programmer** mode. For example:

```
>D Q^DI
```

VA FileMan performs the archiving function by:

1. Searching through file entries using specified criteria.
2. Extracting and transporting the selected entries by Filegrams to temporary storage in the ARCHIVAL ACTIVITY (#1.11) file.
3. Simply writing the data to permanent storage.

The **FILEGRAM-type** template used to transport an archive activity can be created during the archiving session (in the **Archiving** menu options), or an existing **FILEGRAM-type** template created using the **Filegrams** [DIFG] menu options can be used.

## 17.1 Considerations before Archiving

The following summarizes some of the important items to note regarding the archiving facility. Consider them before you begin archiving:

- It is *strongly encouraged* that you to have a current backup of your files before archiving.
- Archiving is *not* user-specific. In other words, archiving is attached to a file *not* a user. For your own protection, be aware that someone other than yourself can complete or change an existing archiving activity.
- Data from logical and physical files can be archived, but only the data from the physical (primary) file can be purged (removed).
- VA FileMan *must* be able to collect and print the data using the search criteria before you can create an archiving activity. The **Select Entries to Archive** option *must* be run to completion such that the selected entries are printed and can thus be stored in the **^DIBT SORT TEMPLATE** global.
- If you plan to keep a hard copy of the printed entries for future reference, design your **PRINT** template to facilitate review. You do *not* need to print all fields' values that are archived, but you can include those that, in combination, uniquely identify the entry. Save the **PRINT** template for future use when archiving.
- When the archived entries are written to temporary storage, the corresponding original entry disappears from the user's view. (A new node, subscripted with **-9**, is added to the original entry so that it is bypassed by the usual VA FileMan calls.)
- If data to be archived is contained in a Multiple field, then the entire entry *must* be archived. You *cannot* archive a subentry by itself.
- A **FILEGRAM-type template** is the only template type allowed in the **Write Entries to Temporary Storage** option.
- You *cannot* have more than one archiving activity on a file at a time.
- Data selected for archiving can be permanently saved to any **sequential** storage media, for example: SDP, a VMS file, magnetic tape, or a removable disk pack.
- Depending on the number of entries involved, be prepared for the search (the **Select Entries to Archive** option) and write (the **Write Entries to Temporary Storage** option) processes to be time-consuming.
- A brief history of who performed the various archive steps and when they were accomplished is saved in the ARCHIVAL ACTIVITY (#1.11) file.
- You can cancel your archiving process, by using the **Cancel Archival Selection** option, at any time before the **Purge Selected Entries** option is used.

- The **Find Archived Entries** option can be used to verify that the archive medium contains all the information intended to be archived.
- The **Purge Selected Entries** option completely deletes data from the file being archived and from the ARCHIVAL ACTIVITY (#1.11) file.
- You *cannot* purge archived entries until you have moved selected entries to permanent storage. Thus, you need *not* worry about losing entries before they are archived.
- You *cannot* start a second archive from a file until you purge or cancel the existing archiving activity on that file.

## 17.2 Archiving Process, including Archiving Options (1-9)

The order of the options on the **Archiving** menu reflects the sequence of steps in which you ordinarily do archiving. Access the **Archiving** menu from the **Other Options** [DIOTHER] menu:

Figure 288: Archiving—Options

```

SELECT OPTION: OTHER OPTIONS
SELECT OTHER OPTION: ARCHIVING
SELECT ARCHIVE OPTION: ? <ENTER>
ANSWER WITH ARCHIVE OPTION NUMBER, OR NAME
CHOOSE FROM:
  1          SELECT ENTRIES TO ARCHIVE
  2          ADD/DELETE SELECTED ENTRIES
  3          PRINT SELECTED ENTRIES
  4          CREATE FILEGRAM ARCHIVING TEMPLATE
  5          WRITE ENTRIES TO TEMPORARY STORAGE
  6          MOVE ARCHIVED DATA TO PERMANENT STORAGE
  7          PURGE SELECTED ENTRIES
  8          CANCEL ARCHIVAL SELECTION
  9          FIND ARCHIVED ENTRIES

```

## 17.2.1 Select Entries to Archive

The **Select Entries to Archive** [???] option creates the archiving activity. It is used similarly to the **Search File Entries** [DISEARCH] option. The **Select Entries to Archive** [???] option is the first step in developing an archiving activity and is very important, since there *cannot* be any archiving without the **SEARCH** template created in this option.



**TIP:** It is important to know which entries you want archived and where you want them stored before you start the archiving process!

This mandatory archiving option really performs three important functions:

1. A search for file entries that meet your specified search criteria or condition (truth test) occurs first. The results of this search are then stored in a template you specify. You *must* store these results in a template!



**REF:** For guidance on how to use the search option procedures, see the "Search" section in the *VA FileMan User Manual*.

2. After storing the results of the search in a template, a sort of the file by any field *must* occur. This sort is as important as the search portion of this option! Do *not* use any sort criteria that contains M code. *Be careful*, if you limit the sort range, you limit the entries selected for archiving. You can delete unwanted entries later by using the **Add/Delete Selected Entries** [DIAX ADD/DELETE] option.
3. Finally, a print of the fields to be archived to a printer or the screen (CRT) *must* occur. Printing fields that uniquely identify the entries being archived gives you a *permanent* record of the archived entries.

You *cannot* create an archiving activity until *at least one* entry is located according to your search criteria, sorted, and printed to a printer or a terminal!



**NOTE:** If a subentry to be archived is contained in a Multiple field, the entire entry *must* be archived.

Figure 289 is an example of the dialog that you can encounter. Notice the sequence of search, sort, and print:

**Figure 289: Archiving—Example of Selecting Entries to Archive**

```

ARCHIVE FROM WHAT FILE: CHANGE
-A- SEARCH FOR CHANGE FIELD: .01 <ENTER> NO.
-A- CONDITION: LESS THAN
-A- LESS THAN: 900

-B- SEARCH FOR CHANGE FIELD: <ENTER>

IF: A// <ENTER> NO. LESS THAN 900

STORE RESULTS OF SEARCH IN TEMPLATE: ZZTEST TEMPLATE
ARE YOU ADDING ZZTEST TEMPLATE AS A NEW SORT TEMPLATE? NO// Y <ENTER> (YES)

SORT BY: VERSION
START WITH VERSION: FIRST// <ENTER>
WITHIN VERSION, SORT BY: <ENTER>

FIRST PRINT FIELD: .01 <ENTER> NO.
THEN PRINT FIELD: VERSION
THEN PRINT FIELD: PROGRAMMER
THEN PRINT FIELD: <ENTER>
HEADING: CHANGE ARCHIVE SEARCH REPLACE <ENTER>
DEVICE: <ENTER>

CHANGE ARCHIVE SEARCH          AUG 30, 1992  10:59  PAGE 1
NO.          VERSION          PROGRAMMER
-----
101          17.10          FMPROGRAMMER,25
102          17.32          FMPROGRAMMER,26
103          17.35          FMPROGRAMMER,26
3 MATCHES FOUND.

```

After using this option, the status of your archiving activity is **SELECTED**. The status of an archiving activity can be any of the following:

- SELECTED
- EDITED
- ARCHIVED (TEMPORARY)
- ARCHIVED (PERMANENT)
- PURGED

You can identify a file at the “ARCHIVE FROM WHAT FILE:” prompt and get the response shown in [Figure 290](#):

**Figure 290: Archiving—Example of a Notice Regarding an Outstanding Archiving Activity**

THERE IS ALREADY AN OUTSTANDING *ARCHIVING* ACTIVITY.  
PLEASE FINISH IT OR CANCEL IT.

This message means that the file you identified has already been selected for archiving and that archiving activity has *not* been completed; a second one *cannot* be started yet. Since the ARCHIVAL ACTIVITY (#1.11) file is being shared by both archiving and extract activities, the italicized word in the message says *extract* if the outstanding activity on the file identified is an extract.

## 17.2.2 Add/Delete Selected Entries

Use the **Add/Delete Selected Entries** [???] option to add entries to or delete them from the archiving activity. This is an easy way to clear out unwanted entries or add needed ones before archiving.

This option uses the **Inquire to File Entries** [DIINQUIRE] option to display the selected entries and then allows you to add or delete an entry. If you add or delete an entry to an established archiving activity, then the status of the activity is changed to **EDITED**.

The **Add/Delete Selected Entries** [???] option does *not* allow you to edit an archiving activity list after the **Write Entries to Temporary Storage** option has been done. If you need to change the archiving activity list after writing to temporary storage, you *must* cancel that archiving activity and start a new one.

When using this option, you first select the archiving activity number that you want to modify. You can also identify the archiving activity by its file number or file name. Then, choose the entry that you want to add or delete.

Figure 291 is an example in which an entry is being added to the archiving activity:

**Figure 291: Archiving—Example of Adding an Entry to the Archival Activity**

```
SELECT ARCHIVE OPTION: ADD/DELETE <ENTER> SELECTED ENTRIES
SELECT ARCHIVAL ACTIVITY: ?
ANSWER WITH ARCHIVAL ACTIVITY ARCHIVE NUMBER, OR FILE
CHOOSE FROM:
1          VA FILEMAN CHANGE          08-05-89    ARCHIVED(PERMANENT)
SELECTOR:FMUSER,FIVE    ARCHIVING
3          CHANGE                    08-30-92    SELECTED
SELECTOR:FMUSER,SIX    ARCHIVING

SELECT ARCHIVAL ACTIVITY: 3

SELECT CHANGE NO.: 330
NO.: 330                      VERSION: 17.09
PROGRAMMER: FMPROGRAMMER,27    ROUTINE: DIL2
DATE CHANGED: OCT 24, 1986

ADD THIS ENTRY TO THE ARCHIVAL SELECTION? YES// <ENTER>
```

If you enter two question marks (??) at the prompt where you select the entry for addition or deletion, a list of the file's entries is displayed. Those that are already part of the archiving activity are identified by "**\*ON ARCHIVE LIST\***". The "ADD this entry TO the ARCHIVAL SELECTION? YES//" question appears if you have selected an entry *not* already on the list for archiving. A "DELETE ..." question appears if you have selected an entry already on the list.

### 17.2.3 Print Selected Entries

The **Print Selected Entries** [???] option displays each entry from a selected archiving activity in a regular print or a Filegram format (depending on the template you identify). The archiving activity entries are printed on whatever device you indicate. You can use this option for an archiving activity with any status except **PURGED**.

[Figure 292](#) illustrates the regular print format:

**Figure 292: Archiving—Printing an Archival Activity in a Regular Format**

```
SELECT ARCHIVE OPTION: PRINT <ENTER> SELECTED ENTRIES
SELECT ARCHIVAL ACTIVITY: 3 <ENTER> CHANGE 08-30-92 EDITED
SELECTOR:FMPATIENT,2 ARCHIVING

ENTER REGULAR PRINT TEMPLATE NAME OR FIELDS YOU WISH TO SEE PRINTED
ON THIS REPORT OF ENTRIES TO BE ARCHIVED.

FIRST PRINT FIELD: [ZZTEST TEMPLATE]

CHANGE ARCHIVAL ACTIVITY          AUG 30, 1992 11:09 PAGE 1
NO.          VERSION              PROGRAMMER
-----
101          17.10                FMPROGRAMMER,25
102          17.32                FMPROGRAMMER,26
103          17.35                FMPROGRAMMER,26
330          17.09                FMPROGRAMMER,30
```

[Figure 293](#) is an example of a Filegram format produced by naming a **FILEGRAM-type** template to print (only a single entry is shown):

**Figure 293: Archiving—Printing an Archival Activity in a Filegram Format**

```
FIRST PRINT FIELD: [ZZTESTFILEGRAM]

CHANGE ARCHIVAL ACTIVITY          AUG 30, 1992 11:09 PAGE 1
-----

      NO.: 330
$DAT^CHANGE^16000^N^
CHANGE^16000^L=330
  BEGIN:CHANGE^16000@1
    SPECIFIER:VERSION^1=17.09
    IDENTIFIER:PROGRAMMER^7=FMPROGRAMMER,30
  END: CHANGE^16000
  NO.^ .01=330
  VERSION^1=17.09
$END DAT
```

## 17.2.4 Create Filegram Archiving Template

The **Create Filegram Archiving Template** [??] option creates a **FILEGRAM-type template**, which is the template used in the **Write Entries to Temporary Storage** [??] option. A Filegram is the tool used for extracting and transporting archived data to temporary storage.



**REF:** For further explanation, see the "[Write Entries to Temporary Storage](#)" section.

You can create a new **FILEGRAM-type template** with this option or edit an existing one created using the Filegram or **Archiving** [??] menu options.



**NOTE:** Only those fields defined in the **FILEGRAM-type template**, the **.01** field, and any PRIMARY KEY or Identifier fields are archived to permanent storage.

The PRIMARY KEY is available as of VA FileMan 22.0.

You create a **FILEGRAM-type template** just as you create a **PRINT** template, except only valid field numbers or names can be entered. You *cannot* include print qualifiers. You always get the "STORE FILEGRAM LOGIC IN TEMPLATE:" prompt, no matter how many fields you identify.

**FILEGRAM-type template** are stored in the PRINT TEMPLATE (#.4) file, so the **FILEGRAM-type template** is referred to as a **PRINT** template in the dialog. However, VA FileMan can distinguish between **PRINT** and **FILEGRAM-type** templates.

If you want to edit an existing **FILEGRAM-type template**, identify it at the "FIRST SEND FIELD:" prompt by entering "[**Filegram templatenam**"]". Otherwise, specify the fields you want included in the archive. (The **.01** field and the file's PRIMARY KEY and Identifier fields will always be sent.) In the example that follows, all fields are being sent.



**NOTE:** The PRIMARY KEY is available as of VA FileMan 22.0.

Figure 294 is an example of creating a **FILEGRAM-type** template:

**Figure 294: Archiving—Example of Creating a FILEGRAM ARCHIVING Template**

```
SELECT ARCHIVE OPTION: CREATE <ENTER> FILEGRAM ARCHIVING TEMPLATE
OUTPUT FROM WHAT FILE: CHANGE
FIRST SEND CHANGE FIELD: ALL
DO YOU MEAN ALL THE FIELDS IN THE FILE? NQ// Y <ENTER> (YES)
THEN SEND CHANGE FIELD: <ENTER>
STORE ARCHIVE LOGIC IN TEMPLATE: CHANGE FILEGRAM
ARE YOU ADDING CHANGE FILEGRAM AS A NEW PRINT TEMPLATE? NQ// YES <ENTER>
(YES)
```

### 17.2.5 Write Entries to Temporary Storage

The **Write Entries to Temporary Storage** [???] option writes your selected archiving activities to the ARCHIVAL ACTIVITY (#1.11) file into a **WORD-PROCESSING** field. This step is in preparation for moving the data to permanent storage. You *cannot* archive to permanent storage unless you use the **Write Entries to Temporary Storage** [???] option first. This file could grow quite large and shrinks only after the **Purge Selected Entries** [???] option has been run.

After using this option, the archived entries appear to be missing from the primary file. This protective measure ensures selected entries *cannot* be edited so that entries in the file match the archived version. The entries are *not* really gone, merely locked.

Also, after using this option, you *cannot* add or delete entries to an archiving activity. If changes in the selection of entries for archiving are necessary, you must cancel this activity and restart.



**CAUTION: This process can be quite time-consuming! It can be queued at the "DEVICE:" prompt. After it is completed, the status of your archiving activity is ARCHIVED (TEMPORARY).**

In [Figure 295](#), the entries in Archiving Activity #3 are being sent to temporary storage using the **ZZTESTFILEGRAM FILEGRAM-type** template. The task is being queued with output sent to PRINTER. The resulting report contains a header, dots, and archiving totals, as shown in [Figure 295](#):

**Figure 295: Archiving—Example of Writing Entries to Temporary Storage**

```

SELECT ARCHIVE OPTION: WRITE <ENTER> ENTRIES TO TEMPORARY STORAGE
SELECT ARCHIVAL ACTIVITY: 3 <ENTER> CHANGE 08-30-92 EDITED
SELECTOR:FMEMPLOYEE,J ARCHIVING

YOU MUST ENTER A FILEGRAM TEMPLATE NAME. THIS FILEGRAM TEMPLATE
WILL BE USED TO ACTUALLY BUILD THE ARCHIVE MESSAGE.

PRINT TEMPLATE: ZZTESTFILEGRAM <ENTER> **FILEGRAM** (AUG 30, 1992) USER
#60 FILE #16000

DEVICE: Q <ENTER> UEUE TO PRINT ON
DEVICE: PRINTER

CHANGE ARCHIVING ACTIVITY AUG 30, 1992 15:01 PAGE 1
-----

. . . .
4 ITEMS HAVE BEEN ARCHIVED

```

## 17.2.6 Move Archived Data to Permanent Storage

Once you have written an archiving activity to temporary storage, you can use the **Move Archived Data to Permanent Storage** [???] option. If you choose an archiving activity that has *not* yet been written to temporary storage, a warning is issued.

This option does several things not necessarily in the order listed here.

1. Prints an Archive Activity report, which prints information from the ARCHIVAL ACTIVITY (#1.11) file, such as the archiver (i.e., the person who selected this option), the archival activity number, an index of all archived entries, and the search criteria used during the Select Entries to Archive process, etc.
2. Builds an index of all archived entries and writes this index at the beginning of the archived file. An item in the index contains the **.01** value of the entry, along with all PRIMARY KEY and Identifier values for that entry.



**NOTE:** The PRIMARY KEY is available as of VA FileMan 22.0.

3. Prompts for Archive Device Label information. This information usually represents some naming convention that systems managers use as physical label for devices

such as tape. If using a disk file, the full disk file name is presented as a default. This device label information, along with the index printed on the Archive Activity report, should be useful in locating an archived entry and the device to which it was archived.

4. Moves archive data to permanent storage. Permanent storage is considered any sequential storage media (e.g., SDP, VMS file, magnetic tape, or disk data set).

You can send to more than one permanent storage location without having to recreate the archiving activity. When the archived data is moved to permanent storage, every line contained in the temporary storage **WORD-PROCESSING** field is simply read and then written to a sequential medium.

The status of your archiving activity after using this option is **ARCHIVED (PERMANENT)**.

In the example in [Figure 296](#), the archiving activity is identified by file name, "**CHANGE.**" It is going to be archived to the specified tape:

**Figure 296: Archiving—Example of Moving Archived Data to Permanent Storage**

```
SELECT ARCHIVE OPTION: MOVE <ENTER> ARCHIVED DATA TO PERMANENT STORAGE
SELECT ARCHIVAL ACTIVITY: CHANGE <ENTER> 3 CHANGE 08-30-92
ARCHIVED(TEMPORARY) SELECTOR:FMEMPLOYEE,J ARCHIVING

NOTE: THIS OPTION WILL 1) PRINT AN ARCHIVE ACTIVITY REPORT TO
SPECIFIED PRINTER DEVICE AND 2) WILL MOVE ARCHIVE DATA TO PERMANENT
STORAGE TO SPECIFIED ARCHIVE STORAGE DEVICE.

SELECT SOME TYPE OF SEQUENTIAL STORAGE MEDIA, SUCH AS SDP, TAPE, OR
DISK FILE (HFS) FOR ARCHIVAL STORAGE.

PRINTER DEVICE: PRINTER

ARCHIVE STORAGE DEVICE: MAGTAPE PARAMETER ([CAVL]:0:2048)
DO YOU WANT YOUR OUTPUT QUEUED? NO// YES
REQUESTED START TIME: NOW// <ENTER>

ARCHIVE DEVICE LABEL: XXXXV3$MUA0:ARCHIVE;083092;3// <ENTER>
```

If the archive storage device is tape, an archive device label is generated for you, as depicted in the previous example. You can override this label by entering your own device label information.

If you select an archive storage device that is *non-sequential*, a warning is issued. You are then given the option to continue the archiving process.



**CAUTION: If you specify a device that can be overwritten (e.g., SDP) or that does *not* electronically store data (e.g., a printer), the data is *forever* irretrievable.**

Depending on what devices are selected, and whether queueing was requested for either device, different warnings are issued informing the user of either steps they need to take or steps that the program takes because of the queueing request.

[Figure 297](#) is an example of an archive activity report:

**Figure 297: Archiving—Example of an Archive Activity Report**

```
ARCHIVE ACTIVITY REPORT                                AUG 30,1992  PAGE: 1
-----
ARCHIVAL ACTIVITY: 3
ARCHIVE DEVICE LABEL INFORMATION:  XXXXV3$MUA0:ARCHIVE;083092;3
PRIMARY ARCHIVED FILE: CHANGE (#16000)
ARCHIVER: FMEMPLOYEE,J
SEARCH CRITERIA:
      .01 LESS THAN 900

INDEX INFORMATION:

NO.          VERSION          PROGRAMMER
101          17.10          FMPROGRAMMER,25
102          17.32          FMPROGRAMMER,26
103          17.35          FMPROGRAMMER,26
330          17.09          FMPROGRAMMER,30

*** PLEASE KEEP THIS FOR FUTURE REFERENCE ***
```

## 17.2.7 **Purge Stored Entries**

Before running the **Purge Stored Entries** [???] option, use the **Find Archived Entries** [???] option to verify that the archive medium contains the complete archived data for an archiving activity by searching for the last record listed on the index.

This option is used to remove archived data from both the archived file (document example is the CHANGE file) and the ARCHIVAL ACTIVITY (#1.11) file. A brief history of who performed the various archiving steps and when is saved in the ARCHIVAL ACTIVITY (#1.11) file.

If you select an archiving activity for purging that has *not* been sent to the archives, then you receive the message shown in [Figure 298](#):

**Figure 298: Archiving—Example of a Notice from VA FileMan When Purging without Archiving Data**

```
DATA HAS NOT YET BEEN ARCHIVED TO PERMANENT STORAGE!
```

You see the dialog in [Figure 299](#) when purging permanently archived data:

**Figure 299: Archiving—Example of Purging Permanently Archived Data**

```
SELECT ARCHIVE OPTION: PURGE <ENTER> STORED ENTRIES  
  
BEFORE YOU PURGE, MAKE SURE THAT YOUR ARCHIVE MEDIUM IS READABLE!  
YOU MAY USE THE FIND ARCHIVED ENTRIES OPTION TO FIND THE LAST  
ARCHIVED RECORD APPEARING ON THE INDEX.  
  
DO YOU WANT TO PROCEED?? NO// YES  
SELECT ARCHIVAL ACTIVITY: 3 <ENTER> CHANGE 08-30-92 ARCHIVED(PERMANENT)  
SELECTOR:FMEMPLOYEE,J ARCHIVING  
  
THIS OPTION WILL DELETE DATA FROM BOTH CHANGE  
AND FROM THE ARCHIVAL ACTIVITY FILE.  
  
ARE YOU SURE YOU WANT TO CONTINUE? NO// YES
```



**CAUTION:** By answering YES at the “Are you sure you want to continue? NO//” prompt, the entries are *immediately* deleted! Be sure this is your decision.

**Figure 300: Archiving—VA FileMan Notifies You of the Number of Entries Purged**

```
THE ENTRIES WILL BE DELETED IN INTERNAL NUMBER ORDER.  
  
<< 4 ENTRIES PURGED >>
```

## 17.2.8 Cancel Archival Selection

The **Cancel Archival Selection** [???] option is used to cancel an archiving activity before the **Purge Selected Entries** [???] option occurs. You are warned if the archiving activity has already been moved to permanent storage.

When you cancel an archiving activity, the entry in the ARCHIVAL ACTIVITY (#1.11) file is deleted. Also, entries that were locked after being moved to temporary storage can again be read and edited.

Figure 301 shows the dialog for canceling an archival activity:

**Figure 301: Archiving—Canceling an Archival Activity**

```
SELECT ARCHIVE OPTION: CAN <ENTER> CEL ARCHIVAL SELECTION
SELECT ARCHIVAL ACTIVITY: CHANGE <ENTER> 3 CHANGE 08-30-92 EDITED
SELECTOR: FMEMPLOYEE,J ARCHIVING

ARE YOU SURE YOU WANT TO CANCEL THIS ARCHIVING ACTIVITY? NO// YES
```



**CAUTION:** By answering YES to this prompt, the entries are *immediately* deleted! Be sure this is your decision.

## 17.2.9 Find Archived Entries

The **Find Archived Entries** [???] option scans an archive file and retrieves an archive entry (or entries) that match the NAME field (#.01) and/or the PRIMARY KEY and/or Identifier fields of the requested entry. On the prompts for lookup values and on the resulting report, PRIMARY KEY and Identifier fields are both called “Identifiers.” The information contained in the archived entry is printed in CAPTIONED format [field name (field number) = value]. This option should only be used with tapes or files archived under VA FileMan 19.0 or later.



**NOTE:** The PRIMARY KEY is available as of VA FileMan 22.0.

Several requests for archived entries can be made at a time. An “(id)” next to a prompt indicates an Identifier, PRIMARY KEY, or Specifier field. One set of all the prompts makes one request.

The matching process is dependent on the presence or absence of an index on the archive file or tape. The matching process for archive files or tapes with no index can be more time-consuming, since it must read the entire archived file to determine all matches. The matching process finds a match when the values of all the answered prompts match that of an archived entry. If a partial value is typed at any prompt, the matching process finds all matches that begin with the partial value for that prompt.

Figure 302 shows the dialog for finding archived entries:

Figure 302: Archiving—Example of Finding Archived Entries

```
SELECT ARCHIVE OPTION: FIND <ENTER> ARCHIVED ENTRIES

THIS OPTION WILL SCAN YOUR ARCHIVED FILE AND WILL ATTEMPT TO
RETRIEVE ENTRIES THAT MATCH THE NAME (.01) FIELD AND/OR THE
IDENTIFIER FIELD(S) OF THE ARCHIVED FILE.

MAGNETIC TAPES SHOULD BE OPENED WITH VARIABLE LENGTH RECORDS.

SEQUENTIAL ARCHIVE DEVICE: HFS <ENTER> DISK FILE
HOST FILE NAME: TMP.TMP// ARC.DAT

YOU ARE READING ARCHIVED INFORMATION FROM THE CHANGE FILE.
DO YOU WANT TO CONTINUE? YES// <ENTER>

MULTIPLE REQUESTS MAY BE MADE.
ONE SET OF ALL PROMPTS MAKES ONE REQUEST.

THIS ARCHIVED FILE CONTAINS AN INDEX OF ALL ARCHIVED ENTRIES.
DO YOU WANT TO SEE THE INDEX NOW? YES// <ENTER>

NO.          VERSION          PROGRAMMER

101          17.10          FMPROGRAMMER,25
102          17.32          FMPROGRAMMER,26
103          17.35          FMPROGRAMMER,26
330          17.09          FMPROGRAMMER,30

ENTER NO.: <ENTER>
ENTER VERSION (ID) : <ENTER>
ENTER PROGRAMMER (ID) : FMPATIENT,30

ENTER NO.: <ENTER>
ENTER VERSION (ID) : <ENTER>
ENTER PROGRAMMER (ID) : <ENTER>

ENTER NO.: <ENTER>
PRINT FOUND ENTRIES TO DEVICE: <ENTER> DECSERVER

SEARCHING ARCHIVED FILE...
FORMATTING FOUND ENTRIES...
PRESS RETURN TO CONTINUE OR ^_ TO EXIT: <ENTER>
ARCHIVE RETRIEVAL LIST                                AUG 30,1992 PAGE: 1
REQUEST: 1
PROGRAMMER = FMPROGRAMMER,2
-----

ARCHIVE FILE: CHANGE (#16000)

LOOKUP VALUE (#.01): 330
IDENTIFIERS:
VERSION (#1) = 17.09
PROGRAMMER (#7) = FMPROGARMMER,30
FIELDS:
FIELD NAME: NO. (#.01) = 330
FIELD NAME: VERSION (#1) = 17.09
```

## 17.2.10 ARCHIVAL ACTIVITY File

The status of an archiving activity is displayed in the listing of activities when a question mark is entered at most of the Archiving options. You can also use the **Inquire to File Entries** [DIINQUIRE] option on the ARCHIVAL ACTIVITY (#1.11) file to obtain information about past or pending archiving activities. The amount of information you receive depends on the status of the archiving activity. The ARCHIVAL ACTIVITY (#1.11) file contains the following information:

- **DUZ** of the individual performing the extract activity.
- Status of the extract activity (e.g., EDITED or UPDATED).
- Dates on which the activities were performed.
- Number of entries extracted.
- Source file number.
- **SEARCH/SORT** and **PRINT** templates used in the extract activity.

Beginning with VA FileMan 20.0, the ARCHIVAL ACTIVITY (#1.11) file contains data about both archiving and extract activities. A file can have only *one* active activity at a time—either an archiving activity or an extract activity. You can only select an extract activity from the Extract Tool options. When you use the **Inquire to File Entries** [DIINQUIRE] option, the word **EXTRACT** appears for all extract activities.

# 18 Meta Data Dictionary

## 18.1 Overview

The META DATA DICTIONARY (#.9) file is a summary of all the VA FileMan files in a VistA installation. It is being provided to enable future development to quickly identify where information is stored within files and fields throughout the VistA system.

[Table 98](#) lists the fields in the META DATA DICTIONARY (#.9) file:

**Table 98: META DATA DICTIONARY (#.9) File Fields**

<b>Filed Name and Number</b>	<b>Description</b>
NAME (#.01)	The NAME field contains the dictionary name followed by the field name with the underscore ( _ ) connecting the two. If the field is a Multiple, then the Multiple name is included.
LOOKUP TERM (#.02)	The LOOKUP TERM is the final field name.
DATA DICTIONARY NUMBER (#.03)	This is the internal entry number (IEN) for the dictionary represented.
FIELD NUMBER (#.04)	This is the internal entry number (IEN) for the field represented.
DATA (#.05)	This is an indicator that the field contains data.
OBJECT NAME (#.06)	The OBJECT NAME is a CamelCase representation of the NAME field.
LAST UPDATED (#.07)	This indicates the last date/time this field was last edited.
DESCRIPTION (#1)	This is the field description.
BUILD(S) (#9.6)	This field displays all the builds where this field was included.
TYPE (#25)	This is the field Data Type.

For example:

**Figure 303: Meta Data Dictionary—Sample Entry for File #200, Field #.01**

```
NUMBER: 21336                                NAME: NEW PERSON_NAME
LOOKUP TERM: NAME                            DATA DICTIONARY NUMBER: 200
FIELD NUMBER: .01                            OBJECT NAME: NEWPERSON.NAME
LAST UPDATED: DEC 30,2015@08:02:19
DESCRIPTION: ANSWER MUST BE 3-35 UPPER-CASE CHARACTERS IN LENGTH, AND BE IN THE
FORMAT FAMILY(LAST),GIVEN(FIRST) MIDDLE SUFFIX. ENTER '??' FOR MORE HELP.
ENTER ONLY DATA THAT IS ACTUALLY PART OF THE PERSON'S NAME. DO NOT INCLUDE EXTRA
TITLES, IDENTIFICATION, FLAGS, LOCAL INFORMATION, ETC. ENTER THE PERSON'S NAME IN
'LAST,FIRST MIDDLE SUFFIX' FORMAT. THIS VALUE MUST BE 3-35 CHARACTERS IN LENGTH
AND MAY CONTAIN ONLY UPPERCASE ALPHA CHARACTERS, SPACES, APOSTROPHES, HYPHENS AND
ONE COMMA. ALL OTHER CHARACTERS AND PARENTHETICAL TEXT WILL BE REMOVED.
BUILD(S) (C): XU*8.0*120
           : XU*8.0*135
           : XU*8.0*134
           : XU*8.0*551
                                           TYPE (C): FREE TEXT
```

## 18.2 ^DDD: Initial Creation

The ^DDD API can be run directly from the M prompt (**D ^DDD**) or by using the **DDU UPDATE META DD** option. This API removes all current entries and fully reproduces the Meta Data Dictionary.

## 18.3 FILELIST^DDD: File List Partial Update

The **FILELIST^DDD** API can be run directly from the M prompt. It requires an array by reference that includes files that are to be updated. For example:

**Figure 304: FILELIST^DDD API—Example**

```
S ARRAY(2)=[] ;PATIENT FILE
S ARRAY(200)=[] ;NEW PERSON FILE
D FILELIST^DDD(.ARRAY)
```

## 18.4 PARTIAL1^DDD: Partial Update using ^DIC(DDD,"%MSC")

The **PARTIAL1^DDD** API makes use of the **^DIC(DDD,"%MSC")** data dictionary variable that includes the date that this file was last updated. The **PARTIAL1^DDD** API updates all files where the **^DIC(DDD,"%MSC")** variable is greater than the date and time stamp that the Meta Data Dictionary was last updated.



**NOTE:** Currently, the **^DIC(DDD,"%MSC")** variable is *not* updated for field description changes.

## 18.5 PARTIAL2^DDD: Partial Update using ^DD(FILE,FIELD,"DT")

The **PARTIAL2^DDD** API makes use of the **^DD(FILE,FIELD,"DT")** variable that includes the date that the field was last updated. The **PARTIAL2^DDD** API updates all files where there is a field that the **^DD(FILE,FIELD,"DT")** variable is equal to or greater than the date and time stamp that the Meta Data Dictionary was last updated.

# 19 Data Synchronization

## 19.1 Overview

The Data Synchronization functionality is designed to be used in association with a Representational State Transfer (REST) service, passed to the Data Synchronization routine to perform automated VA FileMan data dictionary updates.

The main entry point for the Data Synchronization routine is:

**EN^DIFSBLD(JSONIN)**

There are two portions required by the Data Synchronization routine:

- JSON object entry parameter, **JSONIN**.
- External disk file that contains the JSON object formatted update data.

## 19.2 Input JSON Object

Currently, the JSON object entry parameter should contain two values:

- **FULLNAME**
- **FILENAME**

As the project expands more values will be added to the JSON object entry parameter. [Figure 305](#) is an example of the JSON object entry parameter.

The JSON object entry parameter should contain the following elements:

- **FULLNAME:** Contains the full path and file name to the JSON object formatted update data file.
- **FILENAME:** Contains the file name to the JSON object formatted update data file.

**Figure 305: Example of the JSON Object Entry Parameter**

```
{
  "FULLNAME" : "/HOME/FTPTEST/DI_222_6_TEST/TEST_03_19100_V5.JSON",
  "FILENAME" : "TEST_03_19100_V5.JSON"
}
```

## 19.3 Input JSON Data File

The JSON object formatted update data file should contain the following elements:

- **DATABASE:** The root identifier for the file.
  - **FILE:** The VA FileMan file being updated.
  - **RECORD:** A JSON array of the individual data objects.

Within each JSON data object, the field names should match the VA FileMan file CamelCased names found in the Meta Data Dictionary. These names are used to cross-reference to the VA FileMan field numbers.

**Figure 306: Example of a JSON Object Formatted Update Data File**

```
{
  "DATABASE": {
    "FILE": "19100",
    "RECORD": [
      {
        "ID.NAME": "ENTRY 004",
        "START.DATE": "20160705",
        "CODE.TYPE": "P",
        "INFORMATION": "UT NEC NULLA EGET VELIT PHARETRA EUISMOD. CURABITUR SED
LEO LOBORTIS.",
        "ADDRESS": [
          {
            "STREET": "123 MAIN ST.",
            "CITY": "ANY TOWN",
            "STATE": "ANY STATE",
            "ZIP": "99999"
          },
          {
            "STREET": "123 MAIN ST.",
            "CITY": "OTHER TOWN",
            "STATE": "OTHER STATE",
            "ZIP": "00000"
          }
        ],
        "LONG.TEXT": "LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT. DUIS
VARIUS."
      },
      {
        "ID.NAME": "ENTRY 009",
        "START.DATE": "20161105",
        "CODE.TYPE": "P",
        "INFORMATION": "UT NEC NULLA EGET VELIT PHARETRA EUISMOD. CURABITUR SED
LEO LOBORTIS.",
        "ADDRESS": [
          {
            "STREET": "194 E. MAIN ST.",
            "CITY": "ABINGDON",
            "STATE": "VIRGINIA",
            "ZIP": "24210"
          },
          {
            "STREET": "1000 COMMONWEALTH AVE.",
            "CITY": "BRISTOL",
            "STATE": "VIRGINIA",
            "ZIP": "24209"
          }
        ],
        "LONG.TEXT": "LOREM IPSUM DOLOR SIT AMET, CONSECTETUR ADIPISCING ELIT.
PELLENTESQUE A DICTUM EST."
      }
    ]
  }
}
```

## 20 Appendix A—Revision History Archive

This section should be used to show all document revision history prior to the current year.



**REF:** For the most recent, current year document revision history, see the [“Revision History”](#) section.


Date	Revision	Description	Author
12/31/2022	1.5	<p>Updates:</p> <ul style="list-style-type: none"> <li>• Section <a href="#">12.1.2</a>: Revised fields that <i>cannot</i> be audited to just <b>COMPUTED</b> fields.</li> <li>• Section <a href="#">19</a>: <ul style="list-style-type: none"> <li>○ Changes for patch DI*22.2*6: Added Section <a href="#">19</a>, “Data Synchronization.”</li> <li>○ Removed reference to the ENTITY (#1.5) file in Section <a href="#">19.1</a>.</li> </ul> </li> <li>• Made format and content updates throughout this document related to HTML and Word document synchronization project.</li> <li>• Changed all references from “Enterprise Program Management Office (EPMO)” and “DevSecOps (DSO)” to “Software Product Management (SPM).”</li> <li>• Changed all references from “OI&amp;T” to “OIT” throughout.</li> <li>• Updated all references throughout to Kernel manuals to the current, correct title: <ul style="list-style-type: none"> <li>○ <i>Kernel 8.0 and Kernel Toolkit 7.3 Systems Management Guide.</i></li> <li>○ <i>Kernel 8.0 and Kernel Toolkit 7.3 Developer’s Guide.</i></li> </ul> </li> <li>• Reformatted display of file and field names throughout; moved file/field</li> </ul>	VASS Development Team

Date	Revision	Description	Author
		number immediately following the file/field name.	
01/17/2017	1.4	Changes for patch DI*22.2*2: <ul style="list-style-type: none"> <li>• Updated Section 9.2.2. Added eight new data types to <a href="#">Table 91</a>.</li> <li>• Added Sections <a href="#">10.2.12</a> – <a href="#">10.2.19</a>.</li> <li>• Updated Section <a href="#">10.3</a>. Added eight new data types.</li> <li>• Added Sections <a href="#">10.8.9</a> – <a href="#">10.8.16</a>.</li> <li>• Added Section <a href="#">18</a> – Meta Data Dictionary.</li> </ul>	VistA Kernel/VistA Infrastructure (VI) Development Team
11/29/2016	1.3	Changes to support release of Patch DI*22.2*4: <ul style="list-style-type: none"> <li>• Updated Section <a href="#">5.2.2.2</a> to reflect change in behavior of computed expressions as print fields.</li> <li>• Added Section <a href="#">4.2.2.5</a>.</li> <li>• Updated Figure numbering from Figure 78 onward and corresponding cross-references.</li> <li>• Updated ToC, LoF, and LoT.</li> </ul>	VistA Kernel/VistA Infrastructure (VI) Development Team
09/26/2016	1.2	Tech Edit: <ul style="list-style-type: none"> <li>• Updated document for Section 508 conformance.</li> <li>• Updated user entries in <a href="#">Figure 97</a> and <a href="#">Figure 98</a>.</li> <li>• Updated Section <a href="#">9.4</a> and removed PII from <a href="#">Figure 116</a>.</li> <li>• Updated Section <a href="#">9.5</a> and user entries in <a href="#">Figure 117</a> and <a href="#">Figure 118</a>.</li> <li>• Added/Updated Section <a href="#">12.2.4</a> and <a href="#">Figure 235</a>.</li> <li>• Added/Updated Section <a href="#">12.2.5</a> and <a href="#">Figure 236</a>.</li> <li>• Added/Updated Section <a href="#">12.2.6</a> and <a href="#">Figure 237</a>.</li> <li>• Updated “<a href="#">Glossary</a>” table entries.</li> <li>• Updated document styles and</li> </ul>	VistA Kernel/VistA Infrastructure (VI) Development Team

Date	Revision	Description	Author
		formatting to follow current documentation standards and style guidelines.	
09/15/2016	1.1	Changes to support release of Patch DI*22.2*3: <ul style="list-style-type: none"> <li>• Note regarding output device added to Section <a href="#">2.3.2</a> (How to Export Data).</li> <li>• Note regarding option setup added to Section <a href="#">11.1</a> (Verify Fields).</li> <li>• Note regarding auditing of the Description and Technical Description fields added to Section <a href="#">12.2</a> (Auditing a Data Dictionary).</li> </ul>	VistA Kernel/VistA Infrastructure (VI) Development Team
08/03/2016	1.0	Initial release of VA FileMan 22.2 Advanced User Manual.	VA FileMan 22.2 Development Team
TBD	TBD	TBD	TBD

## Glossary

Term	Description
<b>.001 Field</b>	A field containing the internal entry number of the record.
<b>.01 Field</b>	The one field that <i>must</i> be present for every file and file entry. It is also called the NAME field. At a file's creation the <b>.01</b> field is given the label NAME. This label can be changed.
<b>Abbreviated Response</b>	This feature allows you to enter data by typing only the first few characters for the desired response. This feature will <i>not</i> work unless the information is already stored on the computer.
<b>Access Codes</b>	In VA FileMan, a string of codes that determines your security access to files, fields, and templates. In Kernel, you enter an Access Code to identify yourself during signon.
<b>Alerts</b>	Brief online notices that are issued to users as they complete a cycle through the menu system. Alerts are designed to provide interactive notification of pending computing activities, such as the need to reorder supplies or review a patient's clinical test results. Along with the alert message is an indication that the <a href="#">View Alerts</a> common option should be chosen to take further action.
<b>Alternate Editor</b>	One of the text editors available for use from VA FileMan. Editors available vary from site to site. They are entries in the ALTERNATE EDITOR (#1.2) file.
<b>ANSI</b>	American National Standards Institute.
<b>ANSI M</b>	The M (formerly known as MUMPS) programming language is a standard recognized by the American National Standard Institute (ANSI). M stands for Massachusetts Utility Multi-programming System.
<b>API</b>	<p>Program calls provided for use by application developers. APIs allow developers to carry out standard computing activities without needing to duplicate utilities in their own software. APIs also further DBA goals of system integration by channeling activities, such as adding new users, through a limited number of callable entry points. VistA APIs fall into the following three categories:</p> <ul style="list-style-type: none"> <li>• <b>“Supported API”</b>—These are callable routines, which are supported for general use by all VistA applications.</li> <li>• <b>“Controlled Subscription API”</b>—These are callable routines for which you <i>must</i> obtain an Integration Control Registration (ICR; formerly referred to as Integration Agreements [IAs]) to use.</li> <li>• <b>“Private API”</b>—Where only a single application is granted</li> </ul>

Term	Description
	<p>permission to use an attribute/function of another VistA package.</p> <p>These ICRs are granted for special cases, transitional problems between versions, and release coordination.</p>
<b>Array</b>	An arrangement of elements in one or more dimensions. An M array is a set of nodes referenced by subscripts that share the same variable name.
<b>At-Sign (@)</b>	<p>A VA FileMan security Access Code that gives the user <b>Programmer</b>-level access to files and to VA FileMan's developer features. Also, the character @ (i.e., at-sign) is used at VA FileMan field prompts to delete data.</p> <p>See also <a href="#">Programmer Access</a>.</p> <p> <b>CAUTION: Programmer access in VistA is defined as DUZ(0)="@". It grants the privilege to become a developer in VistA. Programmer access allows you to work outside many of the security controls enforced by VA FileMan, enables access to all VA FileMan files, access to modify data dictionaries, etc. It is important to proceed with caution when having access to the system in this way.</b></p>
<b>Audit Trail</b>	The record or log of an ongoing audit.
<b>Auditing</b>	The monitoring and recording of computer use.
<b>Backward Pointer</b>	A pointer to your current file from another file; used in the extended pointer syntax.
<b>Boolean Expression</b>	A logical comparison between values yielding a true or false result. In M, <b>Zero</b> means false and <i>non-Zero</i> (often one) means true.
<b>Callable Entry Point</b>	An authorized developer call that can be used in any VistA application package. The DBA maintains the list of DBIC-approved entry points.
<b>Canonic Number</b>	A number with no leading <b>Zeroes</b> and no trailing <b>Zeroes</b> after a decimal point.
<b>Caption</b>	In ScreenMan, a label displayed on the screen. Captions often identify fields that are to be edited.
<b>Checksum</b>	The result of a mathematical computation involving the individual characters of a routine or file.
<b>Command Area</b>	In ScreenMan, the bottom portion of the screen, which is used to

Term	Description
	display help information and to accept user commands.
<b>Common Menu</b>	The <b>Common</b> menu consists of options that are available to all users. Entering two question marks at the menu select prompt displays any secondary menu options available to the signed-on user, along with the common options available to all users.
<b>Controlled Subscription Integration Control Registration</b>	This applies where the Integration Control Registration (ICR) describes attributes/functions that <i>must</i> be controlled in their use. The decision to restrict the IA is based on the maturity of the custodian package. Typically, these IAs are created by the requesting package based on their independent examination of the custodian package's features. For the IA to be approved the custodian grants permission to other VistA packages to use the attributes/functions of the IA; permission is granted on a one-by-one basis where each is based on a solicitation by the requesting package. An example is the extension of permission to allow a package (e.g., Spinal Cord Dysfunction) to define and update a component that is supported within the Health Summary package file structures.
<b>Cross-Reference</b>	An attribute of a field or a file that identifies an action that should take place when the value of a field is changed. Often, the action is the placement of the field's value into an index. A Traditional cross-reference is defined with a specific field. A New-Style cross-reference is a file attribute and can be composed of one or more fields. New-Style cross-references are stored in the INDEX (#.11) file.
<b>Cursor</b>	On your display terminal, the line or rectangle identifying where your next input is placed on the screen.
<b>Data</b>	A representation of facts, concepts, or instructions in a formalized manner for communication, interpretation, or processing by humans or by automatic means. The information you enter for the computer to store and retrieve. Characters that are stored in the computer system as the values of local or global variables. VA FileMan fields hold data values for file entries.
<b>Data Attribute</b>	A characteristic unit of data such as length, value, or method of representation. VA FileMan field definitions specify data attributes.
<b>Data Dictionary</b>	A record of a file's structure, its elements (fields and their attributes), and relationships to other files. Often abbreviated as DD.
<b>Data Dictionary Access</b>	A user's authorization to write/update/edit the data definition for a computer file. Also known as <b>DD</b> Access.
<b>Data Integrity</b>	This term refers to the condition of patient records in terms of

Term	Description
	completeness and correctness. It also refers to the process in which a particular patient's data is synchronized at all the sites in which that patient receives care.
<b>Data Type</b>	The kind of data stored in a field. For example, the following are VA FileMan DATA TYPE field values: <ul style="list-style-type: none"> <li>• <b>NUMERIC</b></li> <li>• <b>COMPUTED</b></li> <li>• <b>WORD-PROCESSING</b></li> </ul>
<b>Database</b>	An organized collection of data spanning many files. Often, all the files on a system constitute that system's database.
<b>Database Management System (DBMS)</b>	A collection of software that handles the storage, retrieval, and updating of records in a database. A Database Management System (DBMS) controls redundancy of records and provides the security, integrity, and data independence of a database.
<b>Database, National</b>	A database that contains data collected or entered for all VHA sites.
<b>DBA</b>	The Database Administrator (DBA) oversees software development with respect to VistA Standards and Conventions (SAC) such as namespacing. Also, this term refers to the Database Administration function and staff.
<b>DBIA</b>	Database Integration Agreement. See also <a href="#">Integration Control Registration (ICR)</a> .
<b>Decentralized Hospital Computer Program (DHCP)</b>	Decentralized Hospital Computer Program (now known as Veterans Health Information Systems and Technology Architecture [VistA]). VistA software, developed by VA, is used to support clinical and administrative functions at VA Medical Centers nationwide. It is written in M and, via the Kernel, runs on all major M implementations regardless of vendor. VistA is composed of packages that undergo a verification process to ensure conformity with namespacing and other VistA standards and conventions. See also <a href="#">VistA</a> .
<b>Default</b>	A computer-provided response to a question or prompt. The default might be a value pre-existing in a file. Often, you can change a default.
<b>Department of Veterans Affairs (VA)</b>	The Department of Veterans Affairs (formerly known as the Veterans Administration.)
<b>Device</b>	Peripheral connected to the host computer, such as a printer, terminal, disk drive, modem, and other types of hardware and equipment


Term	Description
	associated with a computer. The host files of underlying operating systems may be treated like devices in that they may be written to (e.g., for spooling).
<b>Device Prompt</b>	A Kernel prompt at which you identify where to send your output.
<b>Dictionary</b>	Database of specifications of data and information processing resources. VA FileMan's database of data dictionaries is stored in the FILE of files (#1).
<b>Direct Mode Utility</b>	A developer call that is made when working in direct <b>Programmer</b> mode. A direct mode utility is entered at the MUMPS prompt (e.g., > <b>D</b> ^ <b>XUP</b> ). Calls that are documented as direct mode utilities <i>cannot</i> be used in application software code.
<b>Domain</b>	A site for sending and receiving mail.
<b>Double Quotes ("" )</b>	Symbol used in front of a <b>Common</b> menu option's text or synonym to select it from the <b>Common</b> menu. For example, the <b>five-character</b> string " <b>TBOX</b> " selects the <b>User's Toolbox</b> [XUSERTOOLS] option on the <b>Common</b> menu.
<b>DUZ</b>	Local variable holding the user number that identifies the signed-on user.
<b>DUZ(0)</b>	Local variable that holds the File Manager Access Code of the signed-on user.
<b>Edit Window</b>	In ScreenMan, the area in which you enter or edit data. It is highlighted with either reverse video or an underline. In Screen Editor, the area in which you enter and edit text; the area between the status bar and the ruler.
<b>Entry</b>	A record in a file. "Entry" and "record" are used interchangeably.
<b>EPMO</b>	Enterprise Program Management Office, now known as Product Delivery Service (PDS).
<b>Error Trap</b>	A mechanism to capture system errors and record facts about the computing context such as the local symbol table, last global reference, and routine in use. Operating systems provide tools such as the %ER utility. The Kernel provides a generic error trapping mechanism with use of the ^% <b>ZTER</b> global and ^ <b>XTER*</b> routines. Errors can be trapped and, when possible, the user is returned to the menu system.
<b>Extended Pointers</b>	A means to reference fields in files other than your current file.
<b>Facility</b>	Geographic location at which VA business is performed.

Term	Description
<b>Field</b>	In an entry, a specified area used to hold values. The specifications of each VA FileMan field are documented in the file's data dictionary.
<b>Field Number</b>	The unique number used to identify a field in a file. A field can be referenced by # followed by the field number.
<b>File</b>	A set of related records (or entries) treated as a unit.
<b>File Manager (VA FileMan)</b>	VistA's Database Management System (DBMS). The central component of Kernel that defines the way standard VistA files are structured and manipulated.
<b>Form</b>	In ScreenMan, a group of one or more pages that comprise a complete transaction. Comparable to an <b>INPUT</b> template. See also <a href="#">ScreenMan Forms</a> .
<b>FORUM</b>	The central email system within VistA. Developers use FORUM to communicate at a national level about programming and other issues. FORUM is located at the OI Field Office—Washington, DC (162-2).
<b>FREE TEXT</b>	A DATA TYPE field value that can contain any printable characters.
<b>Full-Screen Editing</b>	The ability to enter data in various locations on the two-dimensional computer display. Compare to scrolling mode.
<b>Global Variable</b>	Variable that is stored on disk (M usage).
<b>Help Prompt</b>	The brief help that is available at the field level when entering one or more question marks.
<b>Histogram</b>	A type of bar graph that indicates frequency of occurrence of values.
<b>Identifier</b>	In VA FileMan, a field that is defined to aid in identifying an entry in conjunction with the NAME field.
<b>Index</b>	An ordered list used to speed retrieval of entries from a file based on a value in some field or fields. Definitions: <ul style="list-style-type: none"> <li>• <b>Simple Index</b>—Refers to an index that stores the data for a single field.</li> <li>• <b>Compound Index</b>—Refers to an index that stores the data for more than one field.</li> </ul> Indexes are created and maintained via cross-references.
<b>INPUT Template</b>	A pre-defined list of fields that together comprise an editing session. Within <b>INPUT</b> templates, subfiles can now be edited in more than one place within the template, so that different subfields can be edited each time.

Term	Description
<b>Integration Control Registration (ICR)</b>	Integration Control Registrations (ICRs) define agreements between two or more VistA software applications to allow access to one development domain by another. VistA software developers are allowed to use internal entry points (APIs) or other software-specific features that are <i>not</i> available to the general programming public. Any software developed for use in the VistA environment is required to adhere to this standard; as such, it applies to vendor products developed within the boundaries of DBA assigned development domains (e.g., MUMPS AudioFax). An ICR defines the attributes and functions that specify access. The DBA maintains and records all ICRs in the Integration Agreement database on FORUM. Content can be viewed using the DBA menu or the Product Delivery Service (PDS) VA Intranet website.
<b>Internal Entry Number (IEN)</b>	The number used to identify an entry within a file. Every record has a unique internal entry number. Often abbreviated as IEN.
<b>IRM</b>	Information Resource Management; now referred to as system administrators. A service at VA medical centers responsible for computer management and system security.
<b>ISO</b>	Information Security Officer.
<b>Kernel</b>	A VistA software package that functions as an intermediary between the host operating system and VistA applications. Kernel includes installation, menu, security, and device services.
<b>Key</b>	<p>A group of fields that, taken collectively, uniquely identifies a record in a file or subfile. All fields in a key <i>must</i> have values. Definitions:</p> <ul style="list-style-type: none"> <li>• <b>Simple Key</b>—Refers to keys that are composed of only one field.</li> <li>• <b>Compound Key</b>—Refers to keys that are composed of more than one field.</li> </ul> <p>Keys are stored in the KEY (#.31) file.</p>
<b>LAN</b>	Local Area Network.
<b>LAYGO</b>	A user's authorization to create a new entry when editing a computer file. An acronym for <b>Learn As You Go</b> .
<b>Line Editor</b>	The VA FileMan editor that lets you input and change text on a line-by-line basis. The Line Editor works in scrolling mode. See also <a href="#">Screen Editor</a> .
<b>Lookup</b>	To find an entry in a file using a value for one of its fields.
<b>MailMan</b>	An electronic mail system (email) that allows you to send messages to

Term	Description
	and receive them from other users via the computer. It is part of VistA.
<b>Menu</b>	A list that includes the names of options from which you can select an activity.
<b>Menu System</b>	The overall Menu Manager logic as it functions within the Kernel framework.
<b>Menu Text</b>	The descriptive words that appear when a list of option choices is displayed. Specifically, the Menu Text field of the OPTION (#19) file. For example, <b>User's Toolbox</b> is the menu text of the <b>XUSERTOOLS</b> option. The option's synonym is <b>TBOX</b> .
<b>Multiple</b>	A VA FileMan DATA TYPE field value that allows more than one value for a single entry. See also <a href="#">Subfile</a> .
<b>MUMPS (M)</b>	Abbreviated as M. The American National Standards Institute (ANSI) computer language used by VA FileMan and throughout VistA. It is software that consists of a high-level programming language and a built-in database. The MUMPS acronym stands for <b>M</b> assachusetts <b>G</b> eneral Hospital <b>U</b> tility <b>M</b> ulti <b>P</b> rogramming <b>S</b> ystem.
<b>Name Field</b>	The one field that <i>must</i> be present for every file and file entry. It is also called the <b>.01</b> field. At a file's creation the <b>.01</b> field is given the label NAME. This label can be changed.
<b>Namespace</b>	A convention for naming VistA package elements. The Database Administrator (DBA) assigns unique character strings for package developers to use in naming routines, options, and other package elements so that packages may coexist. The DBA also assigns a separate range of file numbers to each package.
<b>Navigation</b>	Navigation meanings: <ul style="list-style-type: none"> <li>• Switching your reference point from one file to another.</li> <li>• Moving your cursor around a terminal display or a document using cursor keys and other commands.</li> </ul>
<b>Node</b>	In a tree structure, a point at which subordinate items of data originate. An M array element is characterized by a name and a unique subscript. Thus, the terms: node, array element, and subscripted variable are synonymous. In a global array, each node might have specific fields or "pieces" reserved for data attributes such as name.
<b>Non-Canonic Number</b>	A number with either leading <b>Zeroes</b> or trailing <b>Zeroes</b> after a decimal point. M treats <i>non</i> -canonic numbers as text instead of as numbers.

Term	Description
<b>Non-NULLI</b>	A value other than <b>NULL</b> . A <b>space</b> and <b>Zero</b> are <i>non-NULL</i> values.
<b>NULLI</b>	Empty. A field or variable that has no value associated with it is <b>NULL</b> .
<b>NULL Response</b>	When replying to a prompt, pressing only the <b>Enter</b> key, abbreviated as <b>&lt;Enter&gt;</b> , to enter nothing.
<b>Numeric Expression</b>	An expression whose value is a number. Compare to string expression.
<b>Numeric Field</b>	Response that is limited to a restricted number of digits. It can be dollar valued or a decimal figure of specified precision.
<b>OIT</b>	Office of Information and Technology
<b>OIFO</b>	Office of Information Field Office.
<b>Operator</b>	One of the processes that is done to the elements in an expression to create a value.
<b>Option</b>	A computing activity that you can select, usually a choice from a menu.
<b>Option Name</b>	Name field in the OPTION (#19) file (e.g., XUMAIN for the option that has the menu text "Menu Management"). Options are namespaced according to VistA conventions monitored by the DBA.
<b>Package (Software)</b>	The set of programs, files, documentation, help prompts, and installation procedures required for a given application (e.g., Laboratory, Pharmacy, and PIMS). A VistA software environment is composed of elements specified via the PACKAGE (#9.4) file. Elements include files, associated templates, namespaced routines, and namespaced file entries from the OPTION, HELP FRAME, BULLETIN, and FUNCTION files. As public domain software, VistA software can be requested through the Freedom of Information Act (FOIA).
<b>Pattern Match</b>	In M, an operator that compares the contents of a variable or literal to a specified pattern of characters or kinds of characters.
<b>PDS</b>	Product Delivery Service (PDS).
<b>POINTER TO A FILE</b>	A DATA TYPE field value that contains an explicit reference to an entry in a file. <b>POINTER TO A FILE</b> -type fields are used to relate files to each other.
<b>Popup Page</b>	In ScreenMan, a page that overlays the regular ScreenMan screen to present the contents of a selected Multiple.
<b>Preferred Editor</b>	The editor always entered when you access a <b>WORD-PROCESSING</b> -type field, which is your default editor. Kernel <i>must</i> be present to establish a Preferred Editor.

Term	Description
<b>Primary Key</b>	A Data Base Management System construct, where one or more fields uniquely define a record (entry) in a file (table). The fields are required to be populated for every record on the file, and are unique, in combination, for every record on the file.
<b>Primary Menu</b>	The list of options presented at sign-on. Each user <i>must</i> have a primary menu to sign-on and reach Menu Manager. Users are given primary menus by system administrators. This menu should include most of the computing activities the user needs.
<b>PRINT Template</b>	The stored specifications of a printed report, including fields to be printed and formatting instructions.
<b>Private Integration Control Registration Number</b>	Where only a single application is granted permission to use an attribute/function of another VistA package. These ICRs are granted for special cases, transitional problems between versions, and release coordination. A Private ICR is also created by the requesting package based on their examination of the custodian package's features. Example: one package distributes a patch from another package to ensure smooth installation.
<b>Programmer Access</b>	<p>The ability to use VA FileMan features that are reserved for application developers. Referred to as "having the at-sign (@)" because the at-sign is the <b>DUZ(0)</b> value that grants <b>Programmer</b> access.</p> <p> <b>Programmer access in VistA is defined as DUZ(0)="@".</b> It grants the privilege to become a developer in VistA. Programmer access allows you to work outside many of the security controls enforced by VA FileMan, enables access to all VA FileMan files, access to modify data dictionaries, etc. It is important to proceed with caution when having access to the system in this way.</p>
<b>Prompt</b>	A question or message from the computer that requires the user's response.
<b>Record</b>	A set of data pertaining to a single entity in a file; an entry in a file.
<b>Record Number</b>	See also <a href="#">Internal Entry Number</a> .
<b>Relational Navigation</b>	Changing your current (or primary) file reference to another file. Relational navigation is accomplished by using the extended pointer syntax without specifying a field in the referenced file.
<b>Required Field</b>	A field that <i>cannot</i> be left <b>NULL</b> for an entry.

Term	Description
<b>Reverse Video</b>	The reversal of light and dark in the display of selected characters on a video screen. For example, if text is normally displayed as black letters on a white background, reverse video presents the text as white letters on a black background or vice versa.
<b>Routine</b>	Program or a sequence of instructions called by a program that may have some general or frequent use. M routines are groups of program lines, which are saved, loaded, and called as a single unit via a specific name.
<b>SAC</b>	Standards and Conventions. Through a process of quality assurance, all VistA software is reviewed with respect to SAC guidelines as set forth by the Standards and Conventions Committee (SACC).
<b>SACC</b>	VistA's Standards and Conventions Committee. This Committee is responsible for maintaining the SAC.
<b>Scattergram</b>	A graph in which occurrences of two fields are displayed on an <b>X-Y</b> coordinate grid to aid data analysis.
<b>Screen Editor</b>	VA FileMan's Screen-oriented text editor. It can be used to enter data into any <b>WORD-PROCESSING</b> field using full-screen editing instead of line-by-line editing. See also <a href="#">Line Editor</a> .
<b>ScreenMan</b>	The set of routines that supports Screen-oriented data editing and data display.
<b>ScreenMan Forms</b>	Screen-oriented display of fields, for editing or simply for reading. VA FileMan's Screen Manager is used to create forms that are stored in the FORM (#.403) file and exported with a software application. Forms are composed of blocks (stored in the BLOCK [#.404] file) and can be regular, full screen pages or smaller, "popup" pages.
<b>Screen-Oriented</b>	A computer interface in which you see many lines of data at a time and in which you can move your cursor around the display screen using screen navigation commands. Compare to <a href="#">Scrolling Mode</a> .
<b>Scrolling Mode</b>	The presentation of the interactive dialog one line at a time. Compare to Screen-oriented.
<b>SEARCH Template</b>	A <b>SEARCH</b> template contains the saved results of a search operation. Usually, the actual entries found are stored in addition to the criteria used to select those entries.
<b>Security Key</b>	The purpose of Security Keys is to set a layer of protection on the range of computing capabilities available with a particular software package. The availability of options is based on the level of system

Term	Description
	access granted to each user.
<b>Sensitive Patient</b>	Patient whose record contains certain information, which may be deemed sensitive by a facility, such as political figures, employees, patients with a particular eligibility or medical condition. If a shared patient is flagged as sensitive at one of the treating sites, a bulletin is sent to the <b>DG SENSITIVITY</b> mail group at each subscribing site telling where, when, and by whom the flag was set. Each site can then review whether the circumstances meet the local criteria for sensitivity flagging.
<b>Server</b>	The computer where the data and the Business Rules reside. It makes resources available to client workstations on the network. In VistA, it is an entry in the OPTION (#19) file. An automated mail protocol that is activated by sending a message to a server at another location with the " <b>S.server</b> " syntax. A server's activity is specified in the OPTION (#19) file and can be the running of a routine or the placement of data into a file.
<b>SET OF CODES</b>	A DATA TYPE field value where a short character string is defined to represent a longer value.
<b>Simple Extended Pointers</b>	An extended pointer that uses a pre-existing pointer relationship to access entries in another file.
<b>Site Manger/IRM Chief</b>	At each site, the individual who is responsible for managing computer systems, installing and maintaining new modules, and serving as a liaison to the CIO Field Offices.
<b>Software (Package)</b>	The set of programs, files, documentation, help prompts, and installation procedures required for a given application (e.g., Laboratory, Pharmacy, and PIMS). A VistA software environment is composed of elements specified via the PACKAGE (#9.4) file. Elements include files, associated templates, namespaced routines, and namespaced file entries from the OPTION, HELP FRAME, BULLETIN, and FUNCTION files. As public domain software, VistA software can be requested through the Freedom of Information Act (FOIA).
<b>Sort</b>	To place items in order, often in alphabetical or numeric sequence.
<b>SORT Template</b>	A <b>SORT</b> template contains the stored record of sort specifications. It contains sorting order, as well as restrictions on the selection of entries. Used to prepare entries for printing.
<b>Spacebar Return Spacebar Enter</b>	You can answer a VA FileMan prompt by pressing the <b>Spacebar</b> and then the <b>Enter</b> (or <b>Return</b> key). This indicates to VA FileMan that you would like the last response you were working on at that prompt

Term	Description
	recalled.
<b>Special Queuing</b>	Option attribute indicating that Task Manager should automatically run the option whenever the system reboots.
<b>Stuff</b>	To place values directly into a field, usually with no user interaction.
<b>Subentry</b>	An entry in a Multiple; also called a Subrecord.
<b>Subfield</b>	A field in a Multiple.
<b>Subfile</b>	The data structure of a Multiple. In many respects, a Subfile has the same characteristics as a File.
<b>Subscript</b>	A symbol that is associated with the name of a set to identify a particular subset or element. In M, a numeric or string value that: is enclosed in parentheses, is appended to the name of a local or global variable and identifies a specific node within an array.
<b>Supported Reference Integration Control Registration</b>	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 IA that describes a standard API such as <b>DIE</b> or <b>VADPT</b> . The package that creates/maintains the Supported Reference <i>must</i> ensure it is recorded as a Supported Reference in the IA database. There is no need for other VistA packages to request an IA to use these references; they are open to all by default.
<b>Task Manager (TaskMan)</b>	Kernel module that schedules and processes background tasks (aka TaskMan)
<b>Template</b>	Means of storing report formats, data entry formats, and sorted entry sequences. A template is a permanent place to store selected fields for use later. Edit sequences are stored in the INPUT TEMPLATE (#.402) file, print specifications are stored in the PRINT TEMPLATE (#.4) file, and search or sort specifications are stored in the SORT TEMPLATE (#.401) file.
<b>Terminal Emulation</b>	Using one kind of terminal or computer display to mimic another kind. Often used with PC remote communication applications.
<b>Terminal Type</b>	The designation of the kind of computer peripheral being used (e.g., the kind of video display or printer). Full terminal type functionality is supplied by Kernel.
<b>Trigger</b>	A type of VA FileMan cross-reference. Often used to update values in the database given certain conditions (as specified in the trigger logic). For example, whenever an entry is made in a file, a trigger could

Term	Description
	automatically enter the current date into another field holding the creation date.
<b>Truth Test</b>	An evaluation of an expression yielding a <b>true</b> or <b>false</b> result. In M, usually either of the following is returned from a truth test: <ul style="list-style-type: none"> <li>• <b>1</b>—True.</li> <li>• <b>0</b>—False.</li> </ul>
<b>UCI</b>	User Class Identification, a computing area. The MGR UCI is typically the Manager’s account, while VAH or ROU may be Production accounts.
<b>Up-Arrow</b>	The ^ character (caret); used in VA FileMan for exiting an option or canceling a response. Also used in combination with a field name or prompt to jump to the specified field or prompt.
<b>Upload</b>	Send a file from one computer system to another (usually using communications software).
<b>User Access</b>	This term is used to refer to a limited level of access, to a computer system, which is sufficient for using/operating a package, but does <i>not</i> allow programming, modification to data dictionaries, or other operations that require <b>Programmer</b> access. Any option, for example, can be locked with the <b>XUPROGCODE</b> security key, which means that invoking that option requires <b>Programmer</b> access.  The user’s access level determines the degree of computer use and the types of computer programs available. The System Manager assigns the user an access level.
<b>VA</b>	Department of Veterans Affairs (VA).
<b>VA FileMan (FileMan)</b>	VistA’s Database Management System (DBMS). The central component that defines the way standard VistA files are structured and manipulated.
<b>VAMC</b>	Veterans Affairs Medical Center.
<b>Variable</b>	Character (or group of characters) that refers to a value. M (previously referred to as MUMPS) recognizes three types of variables: <ul style="list-style-type: none"> <li>• <b>Local variables</b>—Exist in a partition of main memory and disappear at sign-off.</li> <li>• <b>Global variables</b>—Stored on disk, potentially available to any user. Global variables usually exist as parts of global arrays. The term “global” may refer either to a global variable or a global array.</li> <li>• <b>Special variables</b>—Defined by systems operations (e.g., <b>\$TEST</b>).</li> </ul>

Term	Description
<b>Verify Code</b>	<p>The Kernel's Sign-on/Security system uses the Verify Code to validate the user's identity. This is an additional security precaution used in conjunction with the Access Code. Verify Codes shall be at least <b>8 characters</b> in length and contain three of the following four kinds of characters:</p> <ul style="list-style-type: none"> <li>• Letters (lowercase)</li> <li>• Letters (uppercase)</li> <li>• Numbers</li> <li>• Characters that are neither letters nor numbers (e.g., #, @, or \$).</li> </ul> <p>If entered incorrectly, the system does <i>not</i> allow the user to access the computer. To protect the user, both codes are invisible on the terminal screen.</p>
<b>VHA</b>	Veterans Health Administration.
<b>VISN</b>	Veterans Integrated Service Network
<b>VistA</b>	<p>The Veterans Health Information Systems and Technology Architecture (VistA), within the Department of Veterans Affairs, is the component of the Veterans Health Administration that develops software and installs, maintains, and updates compatible computer systems in VA medical facilities. (Previously known as the Decentralized Hospital Computer Program [DHCP].)</p>
<b>WAN</b>	Wide Area Network.



**REF:** For a list of commonly used acronyms, terms, and definitions, see the VA Glossary app on the VA Intranet website.