



VA FILEMAN RELEASE NOTES

Version 22.0

March 1999

Department of Veterans Affairs
VISTA Software Development
Infrastructure Product Line

VA FileMan V. 22.0 Release Notes

VA FileMan V. 22.0 introduces native support for keys and compound cross references and thus helps pave the way for the opening of *VISTA* to outside software. Version 22 of FileMan provides increased interoperability with commercial off-the-shelf (COTS) databases that support similar features (i.e., relational databases using Structured Query Language [SQL]). This new version of FileMan also supports structures that have been developed in other ways by packages such as PCMM and CIRN.

For the new keys and indexes, VA FileMan V. 22 introduces two new files:

KEY File

The new KEY file and associated code changes allow developers to define keys on a file or subfile. A key is a group of one or more fields that together uniquely identifies a record in a file. Each key field must have a value, and fields that make up a key must in combination be unique for all records in the file. FileMan enforces key integrity.

INDEX File

The new INDEX file and associated code changes allow developers to create more complicated indexes (cross-references) including indexes on more than one field. See below for additional information.

In addition, numbers have been reserved for new languages on the VA FileMan LANGUAGE file and new ScreenMan forms have been added to allow easy editing of both the data dictionary and of Input and Print templates.

A list of the NOISs that have been resolved with this release of FileMan is included as Appendix A.

■ INDEXES

COMPOUND INDEXES

Now developers can define compound indexes -- indexes that have more than one data-valued subscript. These compound indexes are defined once for the file, rather than for every field in the index. FileMan routines have been modified to use these compound indexes. For example, the routines used to look up an entry in a file (Classic FileMan ^DIC calls, as well as the Database Server calls FIND^DIC and \$\$FIND1^DIC) now allow the developer to specify a compound index for the lookup. Lookup values can be supplied for each of the data values in the index.

RECORD-LEVEL FIRING OF INDEXES

Indexes on the new INDEX file can be flagged to fire at record level. This means that instead of firing the set/kill logic as each field in the index is edited, FileMan waits until the user has finished editing an entry in a file or subfile before firing the set/kill logic. Record-level firing would normally be selected for compound indexes.

COLLATION SEQUENCE

Any subscript of an index on the INDEX file can be flagged for "backward" collation sequence. The Lister (LIST^DIC) and online question mark help for lookups will display the entries from the subscript in backward collation sequence. This is especially useful for dates and allows the developer to store the dates in their natural internal FileMan date format.

MAXIMUM SUBSCRIPT LENGTH

A new Maximum Length field designator allows the developer to set the length of a data subscript in a regular index. In old style indexes, FileMan truncated the data value to 30 characters before storing the value in the index. This new field can be set on any subscript of an index in the new INDEX file.

TRANSFORMS ON INDEX SUBSCRIPTS

Two new fields "Transform for Storage" and "Transform for Display" can be set on subscripts of regular indexes in the new INDEX file. These two new fields are MUMPS type. This allows a user to transform the field before storing it as a subscript on the index, but to have it displayed in a user-readable format. FileMan lookup routines will attempt to find an entry on the index using the lookup value as entered by the user, but will also perform the "Transform for Storage" on the lookup value and attempt to find a match in the index with the resulting value. Then for

any match found, the index value will be passed through the "Transform for Display" code before it is displayed to the end user.

COMPUTED VALUES ON INDEX SUBSCRIPTS

Computed values can be defined as subscripts on an index in the new INDEX file. For example, the current BS5 index on the PATIENT File that allows lookup by the last letter of a persons name followed by the last four digits of their SSN could be defined on the new INDEX file as a computed subscript. The set/kill logic would be fired at the record level, and the index would be defined only once for the file, rather than on both the Name and SSN fields.

LOOKUP PROMPT

Each subscript on an index in the new INDEX file can be assigned a LOOKUP PROMPT. This prompt will be used as the prompt for entry of the lookup value during classic FileMan lookup ^DIC calls. If not filled in, FileMan will default to use the name of the field for that subscript value if there is one.

ACTIVITY FIELD, SET/KILL CONDITIONS

The activity field on an entry in the new INDEX file allows the developer to specify whether the set/kill logic should be fired during re-indexing, and whether the set logic should be fired during installation (KIDS). Set/Kill conditions can be set. The "before" and "after" values of fields are available to this logic in local arrays so that the developer can tailor whether set/kill logic is fired during editing.

KILL ENTIRE INDEX CODE

An index in the new INDEX file can have code defined to kill the entire index from the data global. This can greatly increase the efficiency of calls to re-index an entire file. FileMan can remove an entire index by executing this code, instead of looping through every entry in the file and executing the kill logic for each entry.

■ KEYS

PRIMARY KEY

One of the keys on a file or subfile must be designated the PRIMARY KEY. This Primary Key will be used to "identify" a record when FileMan brings new data into a system during a KIDS installation. Primary Key field values for an entry will be displayed to the user whenever the entry is displayed during classic interactive FileMan lookups (^DIC).

UNIQUENESS INDEX

Associated with every KEY is a Uniqueness Index. This index is used to enforce uniqueness of the key, so every field in the key must be a subscript on the Uniqueness Index and must have no transforms.

■ Options and Calls to Support Indexes and Keys

- **CROSS-REFERENCE FIELD OR FILE OPTION**

The FileMan option CROSS-REFERENCE FIELD OR FILE, under the UTILITY menu, now asks developers whether they want to process New or Traditional indexes. If they select Traditional (the default), then this will work exactly the same way the option worked in previous versions of FileMan. Selecting "New", however, will allow the developer to create new style indexes that are stored in the INDEX file.

- **CLASSIC LOOKUP (^DIC, IX^DIC and MIX^DIC)**

The interactive lookup has been modified to intelligently use entries in the new INDEX file. If the lookup index is compound, the user will be prompted for lookup values for each of the subscripts in the index before the lookup occurs. Online help will display the entries using the collation sequence specified. If a subscript in the index has a transform and the original pass through the index doesn't find a match, FileMan will apply the transform to the lookup value and try again.

The new K flag in DIC(0) will cause FileMan to default to use the Uniqueness Index on the Primary KEY field as the starting index for the lookup, rather than the "B" index in a call to ^DIC.

The developer is no longer required to set DIC("P"). The only exception to this is a few files that are not structured like normal FileMan files. An example of this is the FileMan Audit file where the first subscript is the file number of the file being audited. Some other files have been structured like this in order to use the same data dictionary on several different files. Most of these files belong to the FileMan package.

- **FINDER (FIND^DIC and \$\$FIND1^DIC)**

The Finder routines have been modified to intelligently use entries in the new INDEX file. If the lookup index is compound, the Finder accepts multiple lookup values, one for each subscript in the index. If a subscript in the index has a transform, FileMan will apply the transform to the lookup value and will find any matches to it, along with the original lookup value.

Use of new entries in the FIELDS parameter will allow the developer to control exactly what data will be returned by the call. This makes programming easier and more self-documenting.

The new "C" FLAG supports a new algorithm for doing lookups on names with commas. The new "K" flag specifies that the lookup should be done using the Uniqueness index on the Primary KEY for the file.

- **LISTER (LIST^DIC)**

The Lister has been modified to intelligently use entries in the new INDEX file. If the index is compound, the Lister accepts multiple FROM and PART values, one for each subscript in the index. The output list entries will be returned using the Collation Sequence field on the subscripts.

Use of new entries in the FIELDS parameter will allow the developer to control exactly what data will be returned by the call. This makes programming easier and more self-documenting.

- **REINDEXING**

All of the Reindexing entry points have been modified to intelligently use entries in the new INDEX file. This includes firing Record Level indexes on more than one field (compound indexes) once per record rather than once per field. The "Activity" field on each entry in the INDEX file is checked before firing the set/kill logic to make sure that Reindexing is one of the activities supported by the index. The "Set Condition Code" and "Kill Condition Code" fields are also executed to determine whether or not to fire the set/kill logic.

- **DATA DICTIONARY LISTING**

The DD listings have been modified to display the new INDEX and KEY information for a file. Two new types of DD listings have been added: "INDEXES AND CROSS REFERENCES ONLY" lists only the cross-references (traditional and new) on the file; "KEYS ONLY" lists the keys defined on a file.

- **CLASSIC EDIT (^DIE) AND SCREENMAN (^DDS)**

The Classic FileMan editing routine ^DIE and ScreenMan (^DDS) have been modified to intelligently use entries in the new INDEX file. This includes making sure that local arrays containing the old and new values of each field are available for use in the set and kill logic and that Record-Level indexes are fired once after the editing session. Key integrity is also enforced.

- **FILER (FILE^DIE)**

The Filer has been modified to intelligently use entries in the new INDEX file and to enforce key integrity.

A new "T" (Transaction) flag has been added so that the Filer will validate all the data in the FDA before filing. If any value in the FDA is invalid, nothing is filed.

- **UPDATER (UPDATE^DIE)**

The Updater has been modified to intelligently use the entries in the new Index file and to enforce key integrity.

A new K (Key) flag has been added so that the Updater will use primary key field values instead of the value of the .01 field to do lookups for Finding and LAYGO Finding nodes.

- **VALIDATOR (VAL^DIE)**

The Validator has been modified to check that the field value does not violate key integrity.

- **FIELDS VALIDATOR (VALS^DIE) (New)**

A new Fields Validator procedure has been created to validate data for a group of fields and to convert valid data to internal FileMan format. The integrity of keys affected by the new values is also checked.

- **KEY VALIDATOR (\$\$KEYVAL^DIE) (New)**

A new Key Validator extrinsic function has been created to verify that values contained in the FDA do not produce an invalid key. All keys in which any field in the FDA participates are checked. If the value for a field in a key being checked is not present in the FDA, the value used to verify the key is obtained from the previously filed data.

- **SORT AND PRINT (EN1^DIP)**

The sort logic will automatically check for the existence of usable single field indexes on the new INDEX file, similar to the way that it currently uses traditional regular indexes in order to improve sort efficiency. However, to use compound indexes for sorting, the documented BY(0) input variable must still be used.

The BY(0) information and associated variables that allow the developer to customize sorts by specifying which index is to be used can now be embedded into a SORT template.

- **DIFROM SERVER (USED BY KIDS)**

When data for a file with a Primary KEY is sent during an Install, the Primary KEY field values of each record are compared to those of the records on the file at the target site. This is done in order to determine whether the incoming record needs to be merged to an existing record, or whether a new record must be added. Previously, the match was made on the .01 field and all Identifier fields.

- **VERIFY FIELDS**

If the field is a key field, this option checks the integrity of keys that contain this field. All fields in those keys must be non-null and must in combination be unique across all records in the file.

- **MANDATORY/REQUIRED FIELD CHECK**

If a file has any KEYS defined, this option verifies that all key fields are non-null.

■ Additional New Features

EDITING INPUT AND PRINT TEMPLATES

A new ScreenMan form has been designed to let the developer more easily edit Input and Print templates. It will no longer be necessary to "step" through the entire template in order to edit one item on the template. It will also be easier to see all of the fields in the template at a glance.

MODIFY FILE ATTRIBUTES

A new ScreenMan form has been designed to let the developer more easily edit the data dictionary in the Modify File Attributes option.

NEW LANGUAGE NUMBERS RESERVED ON LANGUAGE FILE

The following numbers have been reserved for new languages on the VA FileMan LANGUAGE file.

- 5 Finnish
- 6 Italian
- 10 Arabic
- 11 Russian

DIFROM NOT USED WITH VERSION 22

The programmer utility DIFROM must not be used by VA developers because it has been replaced by the Kernel Installation and Distribution System (KIDS). DIFROM does not support new VA FileMan V. 22.0 data dictionary structures. If new style Indexes or Keys are added to any file, they will not be transported by DIFROM.

Release Notes

Appendix A—NOISs Resolved in VA FileMan V. 22.0

NOIS	Description
LON-0298-61143	Extended Pointer syntax with Multiples not working: Difficulty using the syntax to print subfield 1.
SPO-0499-51313	Incorrect handling of print field search: On an inquire to file 2, a search for a lowercase print field of "m" results in an error.
LOM-0499-61231	Accessible files causing errors on deleted file: Previously accessible file no longer available to user.
NAS-0499-31177	Print template with \$S and extended file reference: PRINT template with \$Select statement with extended file reference no longer works.
MOU-0499-30917	Kernel Part II & Killer Patch problems: Previously accessible file no longer available to user and cannot use option "Take away all access to a file."
DAY-0399-41473	Bad literal error in ScreenMan: When using ScreenMan to edit word processing fields, if using the full-screen editor and change to the line editor, error occurs.
ISH-0299-41053	DICOMP UNDEF: Word processing functions, relational jumps, and compiled PRINT templates called from code cause errors in particular circumstances.
STL-0299-40098	Computed Field Interpretation: When printing letters, fails to pick up veteran's address.
ISA-0299-10019	DS and D0 Overwritten on Recursive FM Call: Separate but related errors on DS and D0 following a call within a call, but errors eliminated by NEWing DS and D0.
PTH-0199-20481	FileMan displays only 9 digits of IEN: When using FileMan to display the NUMBER field, the last digit is truncated so that only 9 digits are displayed.
OKL-0199-70233	Letter templates: Fails to print data from desired field when printing letters.
GNH-1298-42489	PRINT template editing: PRINT template will not allow creator to edit.
LEB-1298-21858	FileMan infinite loop: When performing a search, if it includes inquiry of the STOP CODE, the job runs continuously.

NOIS	Description
TAM-1298-30328	Cross-reference errors with string too long: Attempt to build a bulletin cross-reference resulted in the length of the node being greater than 255 characters.
HOU-1198-71890	Error: AUDIT+2~DIK1:3: Errors occur when users "^" out of multiple entry pointer fields that point to an AUDITED field.
HOU-1198-71658	Error W1+1~DIR: Errors occur when variable IOM is less than 13 and users enter anything other than "^" at the prompt.
DDC-0998-50060	ListBox displaying duplicate lines: When populating a ListBox using GetList, entries are displayed three or four times per entry in the file.
IHS-0898-52038	Re-index problem in routine DITMGM2C: Routine calls IXALL^DIK instead of IX1^DIK.
ISD-0698-72382	List audit fields multiples: Listing audits won't include audits of multiples.
ISL-0598-50057	Error when reusing a sequence number in KIDS: Error occurs creating patch builds with KIDS, editing the master build definition, attempting to add a patch and reusing a sequence number.
ISL-0398-51938	Default disappears with S in DIC(0) string: In first ^DIC, default disappeared and "O" was appended to DIC(0).
ISF-0398-60828	LIST^DIC and FROM value: The FROM value returned by LIST^DIC is incorrect when index is on a pointer field and there are more entries to return.
ISL-1297-51623	FIND1 call in DIC fails exact match: Occurs whenever the string is over 30 characters.
MAD-1197-40962	Transfer message text: The EDIT option - "Transfer incoming text" does not transfer an original message plus the responses in one action.
WPB-1197-30937	Bulletin cross-referencing other than .01 field: User wants more control over a field that is bulletin triggered when something is added/changed/deleted.
SFC-1197-60723	BY(0) and [TEMPLATES]: Problem with using BY(0) in combination with some template names in BY.

NOIS	Description
ISW-1097-20575	Variable "C" not killed: When doing a DIC look-up, the variable "C" is being set and not NEWed or KILLED.
BHH-0997-41156	Bulletin parameter OLD FIELD NAME will not take: Trying to create a bulletin cross-reference with the OLD NAME (Field) as a parameter produces an error.
HUN-0897-22220	Uncompiling template: Must template be uncompiled before starting a conversion of progress notes (compiled input template routines getting too large)?
BAY-0897-32047	DIS error: User cannot enter more than 26 search elements in DIS.
SFC-0897-61782	\$\$ functions in COMPUTED fields: Functions expressed inside computed fields act differently than code used in print fields.
WPB-0897-30060	ScreenMan – Tab characters: When editing text in ScreenMan, if embedded TABs in wp global, edits look okay on screen but not in what is saved to global.
ATG-0597-32047	DIEZ routines > 10,000 bytes: DIEZ compiles routines greater than 10,000 bytes regardless of the value selected for the routine size.
ISD-0497-70794	Patch #29 Problem: After installing Patch 29 to FileMan, some Mental Health codes stopped working when using IX^DIC.
NOL-0397-72099	Suppressing header/formfeed: How to successfully suppress the formfeed that follows a printout using a call to DIP
ALT-0397-21483	Installation of patch 29 causes voluntary login: Voluntary login terminals fail to interrogate user when duplicate ID code (1st char last name and last 4 of SSN) are on system.
AMA-0297-72067	Can't admit patient: Following installation of patch DG*5.3*111, MAA's can't admit any patients.
NHM-0297-11971	DG*5.3*111 hangs: When installing DG*5.3*111, it stops at 50% complete.
MIA-0297-31963	Patch install problem: When installing DG*5.3*111 it stops at 50% complete.
MIN-0297-41850	Patch DG*5.3*111 still running after 12 hours: When installing DG*5.3*111, it stops at 50% complete.

NOIS	Description
HOU-1296-70848	Error on # too long: When user holds down a number key too long, it errors out.
WBP-0896-20825	Can't edit a PRINT template: Unable to insert a multiple field in a template.
NAS-0696-30808	Undefined error for "?" of empty file: Using Class III software with file definitions but no data, user tried to enter data into files, but attempt was rejected. When user entered a "?," an error was generated.
ISW-1095-23513	Lookup not showing numeric partial matches: FileMan lookups do not show all partial matches with numeric values.
HON-0895-60831	Creating an option, created sort template: When editing PRINT template field of SORT template file, directly or through Utility Functions, it does not utilize FileMan lookups.
MCM-0695-52933	Computed fields not printing: When computed fields are the first field, they don't show up in a print.
MWV-0695-20497	Local template: The IEN of the patient (stored in D0) is overwritten by the IEN of the insurance company from file 36 (stored in X).
BTM-0595-10549	Long FM search criteria: Problems with (1) "A" & "]" and IF condition and (2) using ^ in SEARCH FOR FIELD
MGY-0694-30773	Max String error when creating a bulletin type x-ref: Get "string too long" error while trying to create a bulletin type x-ref with 9 parameters