SAC

The Department of Veterans Affairs M Programming Standards and Conventions.

Table of Contents

Revision History

Date	Description	Author
1/22/1996	Adopted	SACC Committee
7/13/1999	Revised	SACC Committee
12/5/2007	Approved	David Alexander Wally Fort Debbi Lawson (CACI) Randy Morton (Co-Chair) Jean Sheppard Gregory Woodhouse (Chair)
5/1/2008	Added 2.2.6.3 Added "and must be in patch release order" to 2.2.2.2 Corrected font in the description for Actual List Added number sequence to Files section	Albert Consentino Wally Fort Debbi Lawson (CACI) Randy Morton (Co-Chair) Jean Sheppard Gregory Woodhouse (Chair)
2/9/2011	Updated 2.2.3 Changed the reference to the directive Updated 2.2.5 and 2.3.1.1 Modified casing requirements for local variables.	Andrew Bakke Albert Consentino Ron DiMiceli Randy Morton (Co-Chair) Jerry Rutherford Tammy Salewsky Jean Sheppard Jesse Staab Gregory Woodhouse (Chair)
7/6/2011	Updated 2.2.2 Removed the spaces before Package Name, the patch listing, and the Version Date to make the requirement and the example consistent.	Andrew Bakke Albert Consentino Ron DiMiceli Randy Morton (Co-Chair) Jerry Rutherford Tammy Salewsky Jean Sheppard Jesse Staab Gregory Woodhouse (Chair)

Date	Description	Author
8/24/2011	Updated 2.2.4 Clarified the relationship between routine names and labels. Routine names, as well as labels within routines, may only be 8 characters.	Andrew Bakke Albert Consentino Ron DiMiceli Randy Morton (Co-Chair) Jerry Rutherford Tammy Salewsky Jean Sheppard Jesse Staab Gregory Woodhouse (Chair)
6/6/2102	Added 2.5.1.4	Andrew Bakke Albert Consentino Ron DiMiceli Randy Morton (Co-Chair) Jerry Rutherford Tammy Salewsky Jean Sheppard Jesse Staab Gregory Woodhouse (Chair)
8/30/2012	Updated 2.3.3.2 to include $ROLES	Andrew Bakke Albert Consentino Ron DiMiceli Randy Morton (Co-Chair) Jerry Rutherford Tammy Salewsky Jean Sheppard Jesse Staab Gregory Woodhouse (Chair)

Date	Description	Author
5/5/2016	Updated 2.2.1 Clarification of first line format Added 2.9.5 Added Multidivisional requirement. It is required that applications function properly in a multidivisional environment. The SAC does not specify what this means or how multidivisional environment beyond the high level requirement that applications must function correctly in an environment supporting multiple divisions. Updated 2.3.1.1 Local variable names may not exceed sixteen characters. Namespaced variables must be all uppercase characters. All other local variables can be mixed case. Updated 2.2.4 Routine names and Labels are limited to sixteen (16) characters (not including the formal list for parameter passing) and may not contain lower case characters.	Randy Morton

1.1 General Programming Standards and Conventions

The definitions, standards, and conventions within this section apply to all programming and user environments in use in the Veterans Health Information Systems and Technology Architecture (VistA). This includes all applications that use commercial software products as an integral part of VistA (but not to the commercial application itself unless the commercial application is installed in the VistA environment) and to all development environments including, but not limited to programming languages such as M or Pascal and graphical user interface (GUI) environments such as Delphi.

Definitions applicable to this document

Actual List	These are the actual values that are passed into an API or Supported Reference by a caller. For example: when the variable DA is equal to 200, the 200 is the actual parameter used when ^DIE is called. A group of these values are the Actual List.
CONVENTIONS	Programming guidelines which are designed to promote consistency and safety across VistA applications. Exemptions from conventions are not required, but developers are strongly encouraged to follow them. It is also strongly recommended that the developer document any deviation from these conventions.
DBA	Database Administrator
DBIA	DataBase Integration Agreement. See ICR.
EXEMPTION	Authority granted by the SACC to a specific version of a VistA application which allows that application to not comply with a particular section of the SAC for a specified timeframe.
EXTENSIONS	An addition, deletion, or modification to the current Standards and Conventions document. Each extension will contain an appropriate effective date, and may optionally contain an expiration date or event. (The Programming SACC will update the SAC quarterly via extensions, which have the full weight of the original SAC. Every effort will be made to keep all VistA development standards and conventions in one location.)
Formal List	These are the formal values that are passed into an API or Supported Reference by a caller. For example: when the variable DA is equal to 200, DA is the formal parameter used when ^DIE is called. A group of these values are the Formal List.
IA	Integration Agreement. IAs can be accessed via the DBA menu on FORUM.
IAC	Integration Agreement Committee

ICR	Integration Control Registrations (Formerly Database Integration Agreements)
Incremental Lock	Allows multiple locks on the same resource.
KERNEL	The Kernel is a set of software utilities that form the foundation of VistA and include elements that start with the namespaces XG*, XLF*, XOB*, XPD*,XQ*, XT*, XU*, XV*, ZI*, ZO*, ZT*, ZU*.
Ls	Refers to one or more spaces, or a single tab character, at the beginning of the line or between the label and body of the line.
MAILMAN	Mailman is a set of software utilities that form the foundation of VistA 's electronic mail and communications and include elements that start with the namespace XM*.
NAMESPACE	A unique prefix of between 2 and 4 alpha characters assigned by the DBA.
NEW	A way to create a new version of a variable either by explicit declaration or implicitly through parameter passing.
Non-Comment source line	An executable line of code, or a line of text or data referenced by an executable line of code.
PACKAGE	A set of routines, files, options, templates, security keys, screens, bulletins, functions, help frames, forms, blocks, objects, protocols, dialogues, list templates, windows, etc. namespaced according to DBA requirements that function as a unit.
SACC	Standards and Conventions Committee
STANDARD	A rule which all VistA software must follow.
SUPPORTED REFERENCE	Documentation regarding routines, labels, extrinsic functions, files, or global nodes that are accepted and documented by the IAC for use by all packages. Supported references are listed on the DBA menu on FORUM.
VA FileMan	The Database Management System for VistA, with namespaces DD*, DI* and DM*.
VistA	Veterans Health Information Systems and Technology Architecture

Files

1.2 Naming requirements for files used by VistA packages.

1.3 All VA FileMan files in the M Language environment must be number spaced in the number space assigned to the package by the DBA.

1.4 All Windows, DOS, VMS, or other host files created or exported as part of a VistA application shall be namespaced in the namespace assigned by the DBA.

1.5 Packages exporting script files should provide script files for all the terminal emulation packages commonly in use in the VA.

1.6 Packages exporting spreadsheet templates should apply protection to embedded formulas to prevent accidental deletion by a user. Spreadsheet templates should contain documentation describing the purpose of the template, complex functions, and user help.

2 M Language Programming Standards and Conventions

All M-based VistA software will meet the following standards, and comply with the spirit of the conventions.

2.1 The 1995 ANSI/MDC X11.1 Sections 1 and 2 will be adhered to unless explicitly modified by this document.

2.2 Routines (Routine structure and format)

2.2.1 The first line of a routine must be in the following format: routine name<ls>; site/programmer<space>-<space>brief description [optional space];date of last edit [update not required, time is optional].

ZZAA12 ;DALOI/XXX – Example Routine;2/13/07

2.2.1.1 The first line of a routine cannot contain the formal list for parameter passing.

2.2.1.2 Routines generated by VA FileMan or Kernel and other compiled routines used in exporting a package, need not comply with this standard (i.e., 2.2.1).

2.2.2 The second line of a routine must be in the following format: [LABEL-optional]<ls>;;version number;package name;**p_m,...p_n**;version date;Build n where:

;;1.0;PACKAGE;**p_m,…p_n**;Feb 1, 2007;Build 1

2.2.2.1 The version number must be the same on all of the package-namespaced routines.

2.2.2.2 p_m,...p_n are the applied patch numbers, in order of patch release, separated by commas and must be in patch release order. This ";" piece is null if there are no patches.

2.2.2.3 The version date must be the same on all of the package namespaced routines.

2.2.2.3.1 Kernel utilities should be used to update the version date.

2.2.2.4 Build n [optional] will be automatically added to the routines by KIDS transport. The build number may not be manually edited.

2.2.2.5 Routines compiled from templates, cross-references, etc., by VA FileMan during or after package installation are exempt from the second line requirement.

2.2.3 If local modifications to a routine are restricted or prohibited by policy or directive, the third line should contain an appropriate notice. (e.g., "Per VHA Directive 2004-038, this routine should not be modified".) VHA Directives can be found at http://vaww1.va.gov/vhapublications/publications.cfm?Pub=1.

2.2.4 Routine names and Labels are limited to sixteen (16) characters (not including the formal list for parameter passing) and may not contain lower case characters.

2.2.4.1 LABEL+OFFSET references will not be used except for $TEXT references.

2.2.4.2 Lines referenced by $TEXT for use other than to check for the existence of a routine or a line label in that routine must be in the following format: [LABEL-optional]<ls>;;text or M code.

2.2.5 The line body must contain at least 1 printable character, must not exceed 245 characters in length, and must contain only the ASCII characters values 32-126. Line labels, global variable names, system variables, SSVNs, etc. must be uppercase.

2.2.6 Package routine names of the following forms will not be used:

2.2.6.1 NAMESPACE_I* (with the exceptions of Kernel, VA FileMan, and routines created to support the INIT process).

2.2.6.2 NAMESPACE_NTE* (with the exception of the package integrity routines).

2.2.6.3 NAMESPACE_Z*

2.2.7 The maximum routine size, as determined by executing ^%ZOSF("SIZE"), is 20,000 characters. 15,000 of the allowed 20,000 characters may be non-comment source lines (including “;;” comment lines). 5,000 characters are reserved for non-“;;” comments.

2.2.8 Vendor specific subroutines may not be called directly except by Kernel, Mailman, and VA FileMan.

2.2.9 All applications will use documented TaskMan utilities to interface with TaskMan.

2.2.10 Naked references must either be appropriately preceded by the full reference defining it or be documented.

2.2.10.1 An appropriate preceding full reference is one that is on the same physical routine line as the naked reference and has no code between it and the naked reference that branches in any manner to other lines of code or executables.

2.2.10.2 Naked references requiring documentation must be documented within the routine in the immediate vicinity of the naked reference. Naked references that are preceded by a full reference which is outside of the routine where the naked reference is used must have documentation in both the routine containing the full reference and the routine containing the naked reference. This documentation must be in the immediate vicinity of the appropriate reference.

2.2.10.3 Uses of naked references in called utilities are exempt, e.g., S DIC=200,DIC(0)="AEQ",DIC("S")="I $L($P($G(^(1)),"^",9))" D ^DIC is a legitimate use of the naked reference.

2.2.11 % Routines

2.2.11.1 No application will distribute % routines. (Exemptions: Kernel, and VA FileMan)

2.2.11.2 No % routines shall execute variables which could be set by a programmer prior to executing the code.

2.2.11.3 No routine may use VIEW commands using variables as arguments which could be set by a programmer prior to executing the code. (Exemption: Kernel)

2.2.12 Z Routines

2.2.12.1 No application will export routines whose names start with the letter "Z". (Exemption: Kernel)

2.2.13 Routines may not be invoked using the extended reference syntax, i.e., D ^|VAH|TAG^ROUTINE is illegal.

2.3 Variables

2.3.1 Local Variables

2.3.1.1 Local variable names may not exceed sixteen characters. Namespaced variables must be all uppercase characters. All other local variables can be mixed case. Any variable containing lowercase characters must be NEWed at the beginning of the routine, subroutine or DoDot.

2.3.1.2 The full evaluated length of a local variable name including subscripts must not exceed 200 characters. The evaluated length is calculated as follows.

- Example subscripted variable: NAME(sub1,sub2,...,subn)
  (e.g. ($L(NAME)+3) + ($L(sub1) + $L(sub2) + ... +$L(subn)) + (2 * number

of subscripts) +15)
VAR("XXX",123,1,2,0) would evaluate to a string length of 42, (6+11+10+15)=42.

2.3.1.3 System-Wide Variables

2.3.1.3.1 The following are system-wide variables. Any application setting system wide variables must conform to the following definitions.

- AGE - Patient age in years from date of birth to DT expressed as an integer, or, if deceased, the date of death.
- DFN - Internal number of an entry in the PATIENT File (#2).
- DOB - Patient date of birth expressed in internal VA FileMan format.
- SEX - Patient sex; either "F" or "M".
- SSN - Social Security Number with 9 contiguous digits, or 9 digits and a "P".
- VA("BID") - Brief patient identifier up to 7 characters.
- VA("PID") - Patient identifier; up to 15 characters.

2.3.1.3.2 The following variables, referenced elsewhere in this document, are set by Kernel during sign-on, or by VA FileMan, and can be assumed to exist by all VistA applications.

- DT - Current date, without time, in internal VA FileMan format.
- DTIME - Time-out parameter for a read command in seconds.
- DUZ array - Contains user-specific information.
- DUZ(0) is this user's VA FileMan access code.
- DUZ(2) is the internal entry of the institution file.
- DUZ("AG") is the agency code (from the Kernel site param file).
- DUZ("AUTO") is used by Menu Manager to control whether all items on a menu are presented automatically.
- U - Caret (i.e., "^").
- IO - The hardware name of the last selected in/output device.
- IO(0) - The assigned principal device (primary device).

- ION - The logical name of the IO device.
- IOST - The last selected input/output device's subtype from the Terminal Type file.
- IOST(0) - The internal entry number in the Terminal Type file of the last selected IO device's terminal type.
- IOM - The width of the IO device.
- IOSL - The length of the IO device.
- IOF - The code to start output at the top of a page (e.g., W @IOF).
- IOBS - The backspace of the IO device.

2.3.1.4 VistA packages are not allowed to KILL, NEW, SET, MERGE, READ (into) or otherwise modify the variable DUZ or any DUZ array element. (Exemptions: Kernel, and VA FileMan)

2.3.1.5 The variables DT, DTIME, and U have no array elements and shall be initially defined by Kernel or VA FileMan.

2.3.1.5.1 The variable U will not be KILLed or NEWed or changed from the value defined by Kernel or VA FileMan. (It is legal to SET U="^".)

2.3.1.5.2 The variable DT will not be KILLed or NEWed. If changed it must be set using the supported reference S DT=$$DT^XLFDT.

2.3.1.5.3 The variable DTIME may be changed, but must be restored to its original value before exiting the option.

2.3.1.5.3.1 The Kernel supported reference $$DTIME^XUP will reset DTIME to its original value, e.g., S DTIME=$$DTIME^XUP(DUZ).

2.3.1.6 VistA packages are not allowed to KILL, NEW, SET, MERGE, READ (into) or otherwise modify variables beginning with “IO” and any of their array elements except those documented as modifiable in the Kernel System Manual. (Exemptions: Kernel, Mailman, and VA FileMan)

2.3.1.7 VistA packages are not allowed to KILL, NEW, SET, MERGE, READ (into) or otherwise modify variables beginning with “%”. Exceptions to this are the single character variable "%" and the variables set for and/or returned by Kernel, and VA FileMan supported references. (Exemptions: Kernel, VA FileMan and MailMan)

2.3.1.8 A VistA package may declare local variables in its namespace as package-wide. A VistA package may not kill or change another VistA package's package-wide variables.

2.3.1.8.1 The variables and all of their array elements must be described in the package's technical manual.

2.3.1.8.2 Documentation on how to create and kill package-wide variables created by an option that is removed from its exported menu path must be included in the technical manual.

2.3.1.9 All supported references must leave the lock tables and local symbol tables unchanged upon exit with the exception of the following:

- Documented input and output variables (including globals).
- Documented side effects, such as lock table changes, and changes to files.
- Variables within the namespace of the supported reference being called may be changed or killed during execution of the supported reference. (For example, the VA FileMan ^DIC call kills the variable DIE (DIE is within the DI namespace of the ^DIC supported reference), which may exist in the symbol table prior to the call.)
- Variables composed of a single alpha character followed optionally by one numeric.
- The variable %.

These supported references must be documented in the package technical manual and on FORUM with a descriptive list of ALL input and resulting output variables.

2.3.1.10 Naming requirements for variables passed between packages.

2.3.1.10.1 Input variables in an Actual List passed by reference between packages must be package namespaced.

- Legal: D BLD^DIALOG(3500010,,.IBDATA,"IBX")
- Illegal: D BLD^DIALOG(3500010,,.Y,"IBX")

2.3.1.10.2 Variables containing values which will be updated must be namespaced.

- Legal: S DA=10,DR=".01;.104",DIC="^DPT(",DIQ="IBX" D EN^DIQ1
- Illegal: S DA=10,DR=".01;.104",DIC="^DPT(",DIQ="Y" D EN^DIQ1

2.3.2 Global Variables

2.3.2.1 Lowercase characters in global names and global subscripts are prohibited. (Exemption: Cross-references created using field values containing lowercase characters and subscripts used in the ^TMP and ^XTMP globals.)

2.3.2.2 The full evaluated length of a global reference must not exceed 200 characters. The evaluated length is calculated as follows.

Example subscripted variable:

- ^NAME(sub1,sub2,...,subn)
  (e.g ($L(NAME)+3) +($L(sub1) + $L(sub2) + ... +$L(subn)) + (2 * number of subscripts) +15)
  ^TMP("XXX",123,1,2,0) would evaluate to a string length of 42 (6+11+10+15)=42.

2.3.2.3 The KILLing of unsubscripted globals is prohibited and should be protected. (Special instruction to the site will be required to enable the killing of a unsubscripted global. Application developers must document when calls to EN^DIU2 are made to delete files stored in unsubscripted globals).

2.3.2.4 READing, KILLing, SETting or MERGing ^% globals is prohibited. (Exemption: Kernel)

2.3.2.5 All globals must be VA FileMan compatible. ^TMP, ^XTMP and ^UTILITY have a standing exemption from this requirement.

2.3.2.5.1 The global ^TMP will be used as a scratch global within a session. The first subscript shall be $J, or the first two subscripts shall be a package namespaced subscript followed by $J. This global subscript should be killed before and after use.

2.3.2.5.2 The global ^XTMP will be translated, with one copy for the entire VistA production system at each site. The structure of each top node shall follow the format ^XTMP(namespaced- subscript,0)=purge date^create date^optional descriptive information, and both dates will be in VA FileMan internal date format.

2.3.2.6 Fields in VA FileMan files which contain executable code must be write protected in the DD with "@" (e.g., ^DD(file,field,9)="@"), or be defined as VA FileMan data type of M language.

2.3.2.7 References to the DD Global require a formal Database Integration Agreement (DBIA) with the VA FileMan Development team and must be registered with the Database Administrator.

2.3.2.8 All global variables executed by % routines must be in write-protected globals.

2.3.2.9 Extended reference syntax may not be used to reference global variables, i.e., S X=^|VAH|GLOBAL(1,1) is illegal.

2.3.3 Intrinsic (system) Variables

2.3.3.1 The use of lowercase intrinsic variables is prohibited.

2.3.3.2 No VistA package may use the following intrinsic (system) variables unless they are accessed using Kernel or VA FileMan supported references: $D[EVICE], $I[O], $K[EY], $P[RINCIPAL], $ROLES, $ST[ACK], $SY[STEM], $Z*. (Exemptions: Kernel, and VA FileMan)

2.3.3.2.1 $D[EVICE] may check for a binary status, ie 0 or 1.
2.3.3.2.2 $K[EY] May not rely on vendor specific values.

2.3.4 Structured System Variables (SSVNs)

2.3.4.1 The following Structured System Variables may be used only by Kernel, or VA FileMan or through their supported references: ^$CHARACTER, ^$DEVICE, ^$DISPLAY, ^$EVENT, ^$GLOBAL, ^$JOB, ^$LOCK, ^$ROUTINE, ^$SYSTEM, ^$Z*, and ^$WINDOW.

2.4 Commands

2.4.1 BREAK Command

2.4.1.1 Direct use of the BREAK command is prohibited. Use ^%ZOSF("BRK") and ^%ZOSF("NBRK"). (Exemptions: Kernel and VA FileMan)

2.4.2 CLOSE Command

2.4.2.1 Direct use of the CLOSE command is prohibited. Use the supported ^%ZIS* namespaced routines. (Exemptions: Kernel, Mailman and VA FileMan)

2.4.3 HALT Command

2.4.3.1 Direct use of the HALT command is prohibited. Use the supported reference H^XUS. (Exemptions: Kernel, and VA FileMan)

2.4.4 JOB Command

2.4.4.1 Direct use of the JOB command is prohibited. Use the Kernel Task Manager's supported calls to create jobs. (Exemptions: Kernel, and MailMan)

2.4.5 KILL Command

2.4.5.1 The argumentless form of the KILL command is prohibited. (Exemption: Kernel)

2.4.5.2 The exclusive form of the KILL command is prohibited. (Exemptions: Kernel, and VA FileMan)

2.4.6 LOCK Command

2.4.6.1 All LOCKs shall be of the incremental form. (Exemption: Kernel)

- All incremental LOCKS must have a timeout, the timeout must not be less than the value of the Kernel variable “DILOCKTM”.

2.4.7 NEW Command

2.4.7.1 The argumentless form of the NEW command is prohibited.

2.4.7.2 The exclusive form of the NEW command is prohibited.

2.4.8 OPEN Command

2.4.8.1 The use of the OPEN command is prohibited. (Exemptions: Kernel, Mailman and VA FileMan)

2.4.9 READ Command

2.4.9.1 All READ commands shall read into local variables, ^TMP or ^XTMP.

2.4.9.2 All user input READs must have a timeout. If the duration of the timeout is not specified by the variable DTIME and the duration exceeds 300 seconds, documentation in the package technical manual is required.

2.4.9.3 All user input READ commands shall be terminated by a carriage return. (Exemptions: Kernel, and VA FileMan) (Developers desiring to implement escape processing [function keys, arrow keys, etc.] must use Kernel supplied supported references [XGF].)

2.4.10 Transaction Processing Commands

2.4.10.1 VistA packages may use transaction-processing commands as long as the operations have no irreversible side effects.

2.4.11 USE Command

2.4.11.1 The use of the USE command with parameters is prohibited. (Exemptions: Kernel, and VA FileMan)

2.4.12 VIEW Command

2.4.12.1 The use of the VIEW command is prohibited. (Exemptions: Kernel, and VA FileMan)

2.4.13 MWAPI Commands

2.4.13.1 No VistA package may use the MWAPI commands: ESTART, ESTOP, ETRIGGER. (Exemption: Kernel)

2.4.14 Commands

2.4.14.1 The use of Z* commands is prohibited. (Exemptions: Kernel, and VA FileMan)

2.5 Functions

2.5.1 Intrinsic Functions

2.5.1.1 Use of the $NEXT function is prohibited.

2.5.1.2 The use of the $VIEW function is prohibited. (Exemptions: Kernel, and VA FileMan)

2.5.1.3 The use of $Z* functions are prohibited. (Exemptions: Kernel, and VA FileMan)

2.5.1.4 The use of the (non-standard) $INCREMENT function is prohibited. (Exemption: Kernel)

2.5.2 Extrinsic Functions

2.5.2.1 Supported references that use parameters will document the elements of the formal list internally within the routine and in the package technical or programmer manual. Documentation will specify which elements of the formal list are required and which are optional, if any, and those elements which must be passed by reference.

2.5.2.2 Supported extrinsic special variables - extrinsic functions with an empty formal list - will be documented within the routine and in the technical or programmer manual.

2.6 Name Requirements

2.6.1 Unless otherwise approved by the IAC, routine, global, security key, option, template, bulletin, function, screen, help frame, protocol, form, block, list templates, objects, dialogues, remote procedures, and Kernel parameters, etc,. names must be consistent with the assigned namespace.

2.6.2 The following do not have to be namespaced: Mail Groups, File Names, Menu Text,

2.7 Options (Option file entries)

2.7.1 Option selection must be made through Menu Manager. Hardcoded menu management systems are not allowed.

2.7.2 All options in a package must be path independent once the steps described in the technical manual for creating and killing package-wide variables have been taken.

2.7.3 The following must not exist after exiting an option:

- Any documented output variables created by a called supported reference.

- Any documented locks created by a called supported reference.

- Any documented temporary scratch global nodes (e.g., ^TMP and ^UTILITY) created by a called supported reference, with the exception of ^XTMP global data.

- Any local variables, locks, and scratch global nodes (except ^XTMP, or other scratch globals designed to be passed between parts of a package) created by the application.

2.8 Device Handling

2.8.1 All device manipulation will be made through the use of the Kernel supported references. See Sections 2.4.2 and 2.4.8 for specific information about the Close and Open commands. (Exemptions: Kernel and VA FileMan)

2.8.2 Any output to a hard copy device (e.g., printer) must allow for queuing.

2.8.3 Output directed to a hard copy device (e.g., printer) will not start with a form feed or line feeds with the purpose of creating a form feed, and will leave the device at top-of-form when the output is finished.

2.9 Miscellaneous

2.9.1 Application software must use documented Kernel supported references to perform all platform specific functions. (Exemptions: Kernel and VA FileMan)

2.9.2 No data element which may be interpreted as a number may contain more than 15 significant digits.

2.9.3 Packages may phase out supported references (as callable from outside the application and documented by IAC) by providing a minimum 18-month notice to the PROGRAMMER, CHIEF PROJECT MANAGER, and SITEMANAGERS NATIONAL mail groups on FORUM.

2.9.4 Globals and routines will use only the M character set profile.

2.9.5 It is required that applications function properly in a multidivisional environment. The SAC does not specify what this means or how multidivisional environment beyond the high level requirement that applications must function correctly in an environment supporting multiple divisions.

2.10 Conventions

2.10.1 Only Kernel and VA FileMan and existing Supported References may use ^UTILITY.

2.10.2 Tasks should be deleted from Task Manager's list by SETting the variable ZTREQ equal to "@" just prior to the application QUITing.

2.10.3 VA FileMan conventions should be used for editing data and for formatting date and time (see the VA FileMan Users Guide).

2.10.4 Routine documentation

2.10.4.1 Routine line tags referenced from outside the routine should include a comment describing the code's function after the line tag.

2.10.4.2 Any supported references or routines invoked initially from an option or protocol should contain comments explaining the functionality and all input and output variables.

2.10.5 The Data Dictionary may only be edited with FileMan.

2.10.6 READ commands should not be used in the data dictionary.

2.10.7 WRITE commands should not be used in data dictionaries (except for VA FileMan generated ID nodes). The call EN^DDIOL should be used instead.

2.10.8 The proper method of determining if a device is a CRT is to check that the variable IOST starts with the string "C-". (e.g., I $E(IOST,1,2)="C-")

2.10.9 Descriptive information should be included on the third piece of the 0 node of the XTMP global, such as task description and creator DUZ.

2.10.10 The line body of a routine must contain at least 1 character. Generally a single semicolon is used to demarcate a blank line.

2.10.11 Error trapping - To use the new error trapping in a routine, NEW $ESTACK,$ETRAP S $ETRAP="D tag^routine"

3 Interface Programming Standards and Conventions

It is the intention of this section of the Standards and Conventions to provide a consistent path for users as applications migrate from scrolling mode to a screen mode (either ScreenMan, List Manager, or screen oriented editors) to a GUI environment.

3.1 User Interface Standards for Scroll Mode and Screen Mode.

3.1.1 Deletion of a data value, if permissible, must be initiated by the user entering the at-sign "@".

3.1.2 All user-input READs which are in any way evaluated by the application must be escapable by entering a caret "^", which takes an action other than a reread.

3.1.3 All prompts requesting user input must provide additional help when the user enters a question mark ("?"). Any unrecognized or inappropriate response must be handled properly; i.e., at a minimum in a manner similar to the way VA FileMan handles responses. Refer to the VA FileMan User's Manual for more

information. Responses to READs that are in no way evaluated by the application are excluded from this requirement.

3.1.4 In scrolling mode, defaults must be so indicated with a double slash ("//") or "replace" indicating that "replace/with" editing is allowed. The null response (i.e., typing only the RETURN key) shall select the default.

3.1.5 The program must return to the Menu Manager with no more than one intervening read when a user input READ command times out if the argument of the read is in any way evaluated by the application. A timeout at the menu level must halt through H^XUS

3.2 User Interface Conventions for Scroll Mode and Screen Mode.

3.2.1 Developers are encouraged to use the following terminology.

3.2.1.1 Exit - Exit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. If information has been changed, the application may automatically save the information, or prompt the user to save or discard the information.

3.2.1.2 Quit - Like Exit, Quit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. If information has been changed, the application may automatically discard the information, or prompt the user to save or discard the information.

3.2.1.3 Close (or Cancel) allows users to back out of a function or application, one pop-up at a time, until they reach the highest level window. At that point, another Close request has the same effect as an Exit action.

When users Close a pop-up, the application can decide whether to discard or retain the information in that pop-up, depending on how the application wants to establish the default values the next time the pop-up is displayed. If the information is discarded and the pop-up is later redisplayed, the pop-up contains the default values set by the application. If the information is retained and the pop-up is later re-displayed, the pop-up contains the same values as it did when the user Closed the pop-up.

3.2.2 Developers are encouraged to use the following key assignments:

3.2.2.1 PF1 key (or the equivalent key that produces the <ESC>OP sequence) - May result in different actions based on the next key selected.

3.2.2.2 PF2 key (or the equivalent key that produces the <ESC>OQ sequence) - Context-Sensitive Help. Provides context sensitive help about a specific item, field, or window.

3.2.2.3 PF3 key (or the equivalent key that produces the <ESC>OR sequence) - Exit. Exit is defined in 3.2.1.1 above.

3.2.2.4 PF4 key (or the equivalent key that produces the <ESC>OS sequence) - Backtab. Moves the cursor to the previous entry field. The cursor moves from right to left, bottom to top.

3.2.2.5 F10 key - Menu Bar. Moves the cursor to the menu bar, if one is available, at the top of the window or pop-up currently in focus.

3.2.2.6 F12 key - Cancel. Cancel is defined in 3.2.1.3 above.

3.2.2.7 Tab key - Tab. Moves the cursor to the next entry field. The cursor moves from left-to-right, top-to-bottom.

3.2.2.8 PF1,H key sequence - Application Help. Provides information about the particular segment of the application being used.

3.2.3 If a user is waiting for a lock which times out, then appropriate notification should be given to the user.

3.3 User Interface Conventions for GUI Mode.

3.3.1 User Interface Conventions for GUI Mode are detailed in a separate document.

SECTION II A – VHIT ARB DECISION OUTCOME		
☒ **APPROVED**	☐ **DISAPPROVED**	☐ **ESCALATED TO** _____

AUTHORIZATION BY: BRANDT WELKER, ACTING **ARB CHAIR**

SIGNATURE: