



**NAME STANDARDIZATION**  
**Supplement to Patch Description**

**PATCH XU\*8.0\*134**

March 2000

Revised December 2004

Department of Veterans Affairs  
VHA OI Health System Design & Development (HSD&D)  
Infrastructure and Security Services (ISS)



# Revision History

## Document History

The following table displays the revision history for this document. Revisions to the documentation are based on a continuous dialogue with the Infrastructure and Security Services (ISS) Technical Writers and evolving industry standards and styles.

Date	Revision	Description	Author
3/2000	1.0	Initial software and product documentation release via PATCH XU*8.0*134	Tami Winn, San Francisco ISC; Michael Ogi, San Francisco ISC; Susan Strack, San Francisco ISC
12/2004	4.1	Implemented new conventions for displaying TEST data. See <a href="#">Orientation</a> section for details.	Susan Strack, Oakland OIFO

## Patch History

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

## Revision History

# Contents

Revision History .....	iii
Contents .....	v
Figures.....	vii
Orientation .....	ix
Introduction.....	1
Product Description.....	3
<b>User Manual Information .....</b>	<b>7</b>
Relationship Between NEW PERSON Name and Name Components .....	7
New Person File Maintains Standard Form for Names .....	7
New Person Names and Name Components are Synchronized .....	7
Name Components File Preserves Punctuation.....	8
Guidelines for Entering Person Names in VISTA .....	9
Kernel Options Allow Editing of Individual Name Components.....	10
Kernel Options Affected by Name Standardization .....	10
How Does an Option's ScreenMan Form Differ from its Corresponding Input Template? ....	11
Adding a New Entry to the New Person File .....	12
Using the Name Components "Pop-up" Window .....	13
New Person File Reflects Edits Made to the Name Components File .....	14
New VA FileMan Name Formatting Function: XLFMTNAME .....	16
<b>Programmer Manual Information .....</b>	<b>20</b>
Name Components and New Person Files.....	20
Field Descriptions in the New Name Components File (#20).....	20
Changes to Data Dictionary of New Person File (#200).....	22
Steps to Standardize a Name Field .....	24
Source File Data Dictionary .....	24
Name Conversion .....	25
End-User Interfaces .....	25
APIs .....	28
New Supported Reference Integration Agreements .....	28
New Controlled Subscription Integration Agreements .....	45
Modified Kernel API.....	48
<b>Technical Manual Information.....</b>	<b>50</b>
Implementation and Maintenance .....	50
Package Requirements.....	51
Routines .....	51
File List .....	52

Table of Contents

VA FileMan Input Templates.....	53
ScreenMan Forms.....	54
Kernel Options Affected by the Patch .....	55
Kernel Options and Description of Changes .....	56
New VA FileMan FUNCTION: XLFMTNAME – Name Formatting Function .....	57
Menu Diagrams.....	58
Archiving and Purging.....	58
Callable Routines .....	59
External Interfaces (HL7 Components) .....	60
Current VISTA HL7 APIs for Person Name Conversion.....	60
New Kernel APIs for Person Name Conversion .....	60
External Relations.....	62
Package Requirements.....	62
Dependencies.....	62
Integration Agreements (IA) .....	62
Internal Relations .....	64
Namespace.....	64
File Numbers .....	64
Package-wide Variables.....	65
Software Product Security .....	66
File Security .....	66
Glossary .....	68
Appendix A (Data Conversion of the New Person File).....	76
POST^XLFFNAME: New Person Name Conversion .....	76
PRINT^XLFFNAME: Print Report in ^XTMP Global .....	80
Correcting Names That Were Standardized or Parsed Incorrectly .....	81
Index .....	84

# Figures

Figure 1: Comparison between name stored in New Person File versus Name Components File .....	9
Figure 2: Kernel options with associated ScreenMan forms and INPUT templates.....	10
Figure 3: Kernel options affected by Name Standardization .....	12
Figure 4: Adding new users to the NEW PERSON file (#200).....	12
Figure 5: Edits to NAME field of the NEW PERSON file invokes name components "pop-up" window	13
Figure 6: NEW PERSON file is synchronized with NAME COMPONENTS file .....	14
Figure 7: Minimum <i>VISTA</i> packages and patches required .....	51
Figure 8: Exported field definitions and file.....	52
Figure 9: Modified INPUT templates exported with this patch.....	53
Figure 10: Modified ScreenMan forms exported with this patch .....	54
Figure 11: Kernel options with associated ScreenMan forms and INPUT templates.....	55
Figure 12: Kernel options affected by Name Standardization .....	56
Figure 13: Application Programmer Interfaces (APIs) exported with the Name Standardization patch....	59
Figure 14: <i>VISTA</i> HL7 APIs for converting person names.....	60
Figure 15: New Kernel APIs for converting person names .....	61
Figure 16: Patch XU*8.0*134 Supported Reference Integration Agreements .....	62
Figure 17: Patch XU*8.0*134 Controlled Subscription Integration Agreements .....	63
Figure 18: File Security for NAME COMPONENTS (file #200) and NEW PERSON (file #200) .....	67

Table of Figures



# Orientation

## How to Use this Manual

This supplemental documentation to Patch XU\*8.0\*134 is organized into three major parts based on the following functional divisions for inclusion into the Kernel Version 8.0 documentation at a later date:

1. User Manual Information
2. Programmer Manual Information
3. Technical Manual Information

It uses several methods to highlight different aspects of the material. "Snapshots" of computer dialogue (or other online displays) are shown in a non-proportional font and enclosed within a box. User responses to on-line prompts are highlighted in boldface. Boldface is also used to highlight a descriptive word or sentence. The Return or Enter key is illustrated by the symbol **<RET>** when displayed in computer dialogue and is included in examples only when it may be unclear to the reader that such a keystroke must be entered. The following example indicates that you should type two question marks followed by pressing the Return or Enter key when prompted to select an option:

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



Symbol	Description
	Used to inform the reader of general information including references to additional reading material
	<b>Used to caution the reader to take special notice of critical information</b>

Figure i: Documentation Symbol Descriptions

- Descriptive text is presented in a proportional font (as represented by this font). "Snapshots" of computer online displays (i.e., character-based screen captures/dialogs) and computer source code are shown in a *non*-proportional font.
- All uppercase is reserved for the representation of M code, variable names, or the formal name of options, field and file names, and security keys (e.g., the XUPROGMODE key).
- The Enter or Return Key is illustrated as **<Enter>** and is included in examples only when it might be unclear that such a keystroke must be entered.
- Conventions for displaying TEST data in this document are as follows:
  - The first three digits (prefix) of any Social Security Numbers (SSN) will begin with either "000" or "666".
  - Patient and user names will be formatted as follows: [Application Name]PATIENT,[N] and [Application Name]USER,[N] respectively, where "Application Name" is defined in the Approved Application Abbreviations document, located on the [web site] and where "N" represents the first name as a number spelled out and incremented with each new

entry. For example, for the Name Standardization test provider, patient, or user names would be documented as follows: NSPROVIDER,ONE; NSPROVIDER,TWO; NSPATIENT,ONE; NSUSER,ONE; etc; or some variation of this when names are used in descriptive text.



The list of Approved Application Abbreviations can be found at the following Web site:

[http://vista.med.va.gov/iss/strategic\\_docs.asp#sop](http://vista.med.va.gov/iss/strategic_docs.asp#sop)



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

## Reference Material

- a) New Person Name Standardization Software Requirements Specification (SRS) document, dated May 1999.
- b) New Person Name Standardization Software Design Document (SDD) document, dated October 1999.
- c) White Paper on Standardizing Person Names in VISTA (Nov.21, 1995) (Architect's Home Page) which can be found at the following web site: <http://vawww.va.gov/iag/name2.doc>
- d) VISTA data dictionaries for the PATIENT file (#2) and the NEW PERSON file (#200)
- e) HL7 Specification v 2.3 <http://vista2.med.va.gov/vdsi/msg>
- f) ASC Standard X12.3
- g) ANSI HISB Message Standards Developers Subcommittee Common Data Types
- h) Naming Conventions, Initial Requirements Analysis (IRA), last updated 10/19/98
- i) PATIENT NAMES, NPI, etc., - TI NEEDS TO BEGIN TO FORMALIZE STEPS FOR TS, E-mail message thread initiated by Catherine Pfeil, Dated December 4, 1998
- j) PROVIDER ISSUES MEETING MINUTES, FEBRUARY 26-27, 1997 CIO FIELD OFFICE SAN FRANCISCO.
- k) Appendix A of the New Person Name Standardization Software Requirements Specification (SRS) manual.

# Introduction

This supplemental documentation is intended for use in conjunction with the Name Standardization patch (Kernel Patch, XU\*8.0\*134). It outlines the details of the work involved in the Name Standardization patch and gives guidelines for how the generic APIs can be used to standardize the collection and storage of person names across Veterans Health Administration (VHA). The intended audience for this documentation is Information Resource Management (IRM) and Veterans Affairs Medical Center (VAMC) personnel who will be doing the changes to the system. However, it can also be helpful to others in Technical Service, the Program Office, Enterprise *VISTA* Support (EVS), and Technical Integration. This documentation will be incorporated into the Kernel Version 8.0 documentation at a later date.

## Background

The Veterans Health Administration (VHA) does not currently enforce a uniform procedure for recording names of persons, in *VISTA*. This has led to duplicate entries for individuals within systems and problems in matching records across systems. Name suffixes (Jr., III etc.) have added to the problems.

## Purpose

The main impetus of this project is to support the National Provider Index (NPI) project that will assign a national number to every provider who gives service to the Department of Veterans Affairs. It is necessary to define a standard way for names to be entered into the NAME field (#.01) of the NEW PERSON file (#200). This will help in uniquely defining all providers in the file. Another benefit to this project will be the ability to uniquely identify computer users across various VA facilities. This will become especially important as the facilities do more data sharing, and as more computer users have access to data on computers across multiple facilities. It will also be a major step towards supporting data exchange with COTS/GOTS products that rely on a more detailed definition of person names via HL7 segments.



# Product Description

The purpose of the Name Standardization project is to standardize the way person names are stored in *VISTA*. Name Standardization (Patch XU\*8.0\*134) provides utilities that enable *VISTA* applications to standardize the way person names are entered and stored in Veterans Health Administration (VHA) databases.

The Name Standardization release (Patch XU\*8.0\*134) features:

- A standard format for person names in *VISTA*.
- The data conversion of the NEW PERSON file (#200).
- A new NAME COMPONENTS file (#20).
- Changes to the data dictionary of the NEW PERSON file.
- Changes to Kernel options that allow editing of individual name components.
- New Application Programming Interfaces (APIs).
- A new VA FileMan FUNCTION to display names in various formats.

## Standard Format for Person Names in *VISTA*

A standard format for person's names in *VISTA* is being introduced with the Name Standardization project (Patch XU\*8.0\*134). The definition of this new format, referred to as "standard form" or as "standard format," is a person's name entirely in uppercase letters, containing no Arabic numerals (i.e., 1, 2, 3, etc.). The Family Name (last name) portion of a standard name appears to the left of the comma and contains no spaces and no punctuation except hyphens (-). The Given Name (first name), Middle Name, and Suffix (the portion to the right of the comma) contain no punctuation except for hyphens and spaces. NMI and NMN are not used for the Middle Name.

The standard form of a name is:

Family\_name,Given\_name<space>Middle\_name<space>Suffix

## Data Conversion of the New Person File

All personnel who work at VA facilities are recorded in the NEW PERSON file (#200). This file contains data that was previously stored in obsolete *VISTA* files such USER (#3), PROVIDER (#6), and PERSON (#16). Kernel options provide the functionality to enter, edit, terminate, and reactivate users in the NEW PERSON file.

As part of the Post-Installation process for Patch XU\*8.0\*134, a data conversion is run in the NEW PERSON file to convert the .01 field (NAME field) to a standard format. This conversion standardizes names in the NEW PERSON file and parses them into their component parts.

As each name in the NAME field of the NEW PERSON file is converted and parsed, a corresponding entry is created in the new NAME COMPONENTS file, exported with this patch. The components of the parsed name are stored in this new file.



Detailed information regarding the installation of Patch XU\*8.0\*134 and data conversion of the NEW PERSON file can be found in the patch description on the Patch Module in FORUM.

## New Name Components File

In order to facilitate naming standards for the NEW PERSON file, as well as other files with name fields, a new NAME COMPONENTS file (#20) has been created and exported with this patch. This new file holds the component parts of a person's name, which are listed as follows:

- FAMILY (LAST) NAME field (#1)
- GIVEN (FIRST) NAME field (#2)
- MIDDLE NAME field (#3)
- PREFIX field (#4)
- SUFFIX field (#5)
- DEGREE field (#6)

The fields PREFIX field (#4) and DEGREE field (#6), previously shown, can also be used to build formatted names for display; however, these two fields are not considered part of a standard name field. MUMPS cross-references automatically synchronize the fields in the new NAME COMPONENTS file with the name field of the source VISTA file. (In the case of Patch XU\*8.0\*134, the source name field is the .01 field (NAME field) of the NEW PERSON file.)



The component parts of a person's name as listed previously is consistent with the ANSI HISPP Message Standards Developers Subcommittee Common Data Types definitions for element labels and formats, and thus compatible with messaging standards such as Health Level Seven (HL7) and X.12.

## Changes to Data Dictionary of New Person File

The input transform on the NEW PERSON file converts all input into the .01 field (NAME field) into standard form and parses the input into its component parts: Family Name, Given Name, Middle Name, and Suffix. It stores the component parts in the NAME COMPONENTS file.

MUMPS cross-references maintain automatic synchronization between the new NAME COMPONENTS and the NEW PERSON files.

## Changes to Kernel Options Allow Editing of Individual Name Components

The Kernel options themselves, have not been changed. However, the ScreenMan forms and the INPUT templates used by the Kernel options, listed as follows, have been modified to allow users to edit the individual name components of a person's name.

- Add a New User to the System (XUSERNEW),

- Edit an Existing User (XUSEREDIT), and
- Reactivate a User (XUSERREACT)

### **New Application Programming Interfaces (APIs)**

A common set of APIs has been developed for standardizing and retrieving person names to enable *VISTA* applications to standardize person names in their databases.

This project required the coordination between the Information Infrastructure and Patient Management Systems divisions of Technical Services. A common set of APIs has been developed which can be used for standardizing and retrieving person names. To support this effort, each group applied the necessary changes to files under their application's domain (i.e., the NEW PERSON file, and the PATIENT file [#2], respectively). Name Standardization (Patch XU\*8.0\*134) is the Information Infrastructure patch to standardize names in the .01 field of the NEW PERSON file, a critical step in preparing for National Provider Identification.

### **New VA FileMan FUNCTION to Display Names in Various Formats**

A VA FileMan FUNCTION has been created and exported with Name Standardization that returns a formatted name from any *VISTA* file that contains a NAME field. It allows the user to display names in various formats from any file containing a name field that has been standardized and linked to the NAME COMPONENTS file. Patch XU\*8.0\*134 standardizes the .01 field of the NEW PERSON file, so the new FUNCTION will work on that name field.

This FUNCTION will make it easy for non-programmers to include formatted names in selected modes of FileMan output.





# User Manual Information

This is the User Manual section of this supplemental documentation for Patch XU\*8.0\*134. It will be incorporated into the Kernel Systems Manual, Version 8.0 at a later date.

The intended audience for this chapter is Information Resource Management (IRM) and Veterans Affairs Medical Center (VAMC) personnel who will be doing the changes to the system. However, it can also be helpful to others in Technical Service, the Program Office, National *VISTA* Support (NVS), and Technical Integration.

## Relationship Between NEW PERSON Name and Name Components

Name Standardization (Patch XU\*8.0\*134) ensures that names in the NAME field (#.01) of the NEW PERSON file (#200) are in standard form and that each name is kept in synchronization with the component parts of the name as stored in the NAME COMPONENTS file (#20). The following three topics describe this relationship in more detail.

### New Person File Maintains Standard Form for Names

Name Standardization (Patch XU\*8.0\*134) introduces functionality that allows packages to maintain a standard format for names in *VISTA* files. Patch XU\*8.0\*134 also introduces this functionality specifically for the NEW PERSON file. This affects both new users entered into the NEW PERSON file for the first time, as well as edits to existing users in the NEW PERSON file.

Person names entered in the NAME field (.01 field) for the first time, or edits to existing users in the NAME field (.01 field) converted into standard format (also referred to as standard form).

The standard form of a name is:

Family\_name,Given\_name<space>Middle\_name<space>Suffix.

The Family Name (last name) portion of a standard name appears to the left of the comma and contains no spaces and no punctuation except hyphens (-). The Given Name (first name), Middle Name, and Suffix (the portion to the right of the comma) contain no punctuation except for hyphens and spaces.

### New Person Names and Name Components are Synchronized

When a name in the NEW PERSON file is added or edited, a corresponding entry in the new NAME COMPONENTS file (#20) is automatically added or updated. (Patch XU\*8.0\*134 provides APIs to allow other files in *VISTA* packages to update the NAME COMPONENTS file in a similar manner.) Each entry in the NAME COMPONENTS file contains the name broken down into the following component parts:

- FAMILY (LAST) NAME field (#1),
- GIVEN (FIRST) NAME field (#2),
- MIDDLE NAME field (#3), and
- SUFFIX field (#5).

Automatic synchronization is maintained between the fields in the new NAME COMPONENTS file: FAMILY (LAST) NAME (#1), GIVEN (FIRST) NAME (#2), MIDDLE NAME (#3), and SUFFIX (#5), and the source field, which in the case patch XU\*8.0\*134 is the .01 field (NAME field) of the NEW PERSON file.

(For more information on APIs exported with the Name Standardization patch, see the chapter "APIs" in the "Programmer Manual Information" section of this documentation..)


## Name Components File Preserves Punctuation

The standard form of a person name may contain only a limited set of punctuation (e.g., as in the standard form for person names stored in the NEW PERSON file). However, the name components stored in the NAME COMPONENTS file may contain any punctuation except the accent grave (`) and the up-arrow (^). This offers the functionality to build names from the NAME COMPONENTS file in various formats, including all punctuation associated with that name.

For example, suppose the standard form of the name stored in the .01 field of the NEW PERSON file is NSPROVIDER,MERRIE K MD. Since the NAME COMPONENTS file preserves the punctuation intact in the individual name components, this name as stored in the NAME COMPONENTS could look like:

```
PREFIX: MRS .  
GIVEN NAME: MERRIE  
MIDDLE NAME: K.  
FAMILY NAME: NS ' PROVIDER  
SUFFIX: MD
```

So, as one example, a name constructed from the name components can present the name in a readable format for correspondence.

	NS'PROVIDER represents a last name containing an apostrophe (e.g., O'REILLY).
---	---

# Guidelines for Entering Person Names in VISTA

Guidelines have been developed with the Name Standardization project for a new method for entering person names in VISTA. The objective of creating these guidelines is to support the process of standardizing the collection and storage of person names across VHA. A new NAME COMPONENTS file (#20) has been introduced with the Name Standardization release (Patch XU\*8.0\*134). This new file holds the component parts of a person's name, which are listed as follows:

- FAMILY (LAST) NAME,
- GIVEN (FIRST) NAME,
- MIDDLE NAME, and
- SUFFIX.

The following guidelines are provided for the general purpose of entering names in VISTA:

1. When entering a person name into the system, (e.g., In the case of Patch XU\*8.0\*134 you use Kernel options to enter names into the system.) only enter the data that is actually part of the person's name. Do not include extra titles, identification, flags, local information, etc.
2. Enter the person name in the following format:

Family\_(last)\_name,Given\_(first)\_name(s) Middle\_name(s) Suffix(es)

Example: NS ' PROVIDER-DE LEON, JOHN K. JR.

Though FileMan standardizes the name you enter and removes most of the punctuation before storing it in the name field, the punctuation is retained in the NAME COMPONENTS file for use in displaying the name for letters, reports, etc. Therefore, you should enter all appropriate punctuation in the name.

Figure 1 shows the previous example of the name "NSPROVIDER-DELEON, JOHN K JR" which was standardized and stored in the NAME field (.01 field) of the NEW PERSON file, compared to the same standardized name stored in the NAME COMPONENTS file.

New Person File	Name Components File
NSPROVIDER-DELEON, JOHN K JR	Family (last) name: NS ' PROVIDER-DE LEON Given (first) name: JOHN Middle name: K. Suffix: JR.

Figure 1: Comparison between name stored in New Person File versus Name Components File

In forming the standard name, colons (:) and semicolons (;) in the Family Name part are replaced with hyphens (-), and then all punctuation except hyphens are removed, and all spaces are removed. In the other name parts, colons, semicolons, and periods are replaced with spaces, and then all punctuation except for hyphens is removed. Birth position indicators entered as Arabic numerals are changed to their Roman numeral equivalents. (For example, 3rd is changed to III.)

## Kernel Options Allow Editing of Individual Name Components

This chapter provides information about the three Kernel options affected by the Name Standardization release (Patch XU\*8.0\*134). This chapter does not attempt to provide detailed information about how to use these Kernel options. These are documented in detail in the "Sign-On/Security" section of the Kernel Systems Manual, Version 8.0. Use the following URL to access this documentation: <http://vista.med.va.gov/kernel/docs/index.html>.

### Kernel Options Affected by Name Standardization

The Kernel options, associated ScreenMan Forms, and INPUT templates, which are affected by Patch XU\*8.0\*134 are shown in Figure 2. The Kernel options themselves, have not been changed. However, the ScreenMan forms and the INPUT templates used by these options have been modified to allow users the capability to edit the individual name components of a person's name.

Kernel Option and Menu Text	ScreenMan Form	Input Template
XUSERNEW Add a New User to the System	XUNEW USER	XUNEW USER
XUSEREDIT Edit an Existing User	XUEXISTING USER	XUEXISTING USER
XUSERREACT Reactivate a User	XUREACT USER	XUREACT USER

Figure 2: Kernel options with associated ScreenMan forms and INPUT templates

The following is a list of the name component fields stored in the new NAME COMPONENTS file (#20). Each one of these fields (except DEGREE) can be edited using the Kernel options listed in Figure 2:

- FAMILY (LAST) NAME field (#1)
- GIVEN (FIRST) NAME field (#2)
- MIDDLE NAME field (#3)
- PREFIX field (#4)
- SUFFIX field (#5)
- DEGREE field (#6)



The DEGREE field (#6) is updated indirectly via the "ADEG" MUMPS cross-reference on the DEGREE field (#10.6) in the NEW PERSON file (#200).

## How Does an Option's ScreenMan Form Differ from its Corresponding Input Template?

Functionally there is no difference between the ScreenMan forms and INPUT templates invoked by these options. All three options attempt to invoke the associated ScreenMan form first. However, if for some reason the ScreenMan form cannot be invoked (e.g., because the terminal type cannot handle screen-oriented applications), the associated INPUT template for scrolling mode is invoked.

### The ScreenMan Forms

An option's corresponding ScreenMan form is invoked if your terminal type is able to handle screen-oriented applications. If the .01 field (NAME field) in the NEW PERSON file is edited or if the Return (Enter) key is pressed while the cursor is in the NAME field, the individual name components will be displayed in a "pop-up" window so that they can be edited separately.

The Kernel option Add a New User to the System functions slightly differently from the other two options, Edit an Existing User and Reactivate a User. If the option Add a New User to the System has been selected, after the NAME field and all Identifier fields for the new entry have been entered and the ScreenMan form has been invoked, the individual name components appear in a "pop-up" window first for editing. The other two Kernel options position the cursor in the NAME field (i.e., the .01 field of the NEW PERSON file) first for editing. Thus, the timing of when you are prompted to edit the name components varies between the options.

### The INPUT Templates

An option's corresponding INPUT template is invoked if your terminal type is unable to handle screen-oriented applications. The INPUT templates have been modified to prompt for the individual components of a person's name.

## Adding a New Entry to the New Person File

This next set of examples will use the screen-oriented display (i.e., the ScreenMan form) to illustrate the changes to the Kernel options. The INPUT template functions similarly, but in scrolling mode. To demonstrate this, we will add the fictitious DR. MERRIE NS'PROVIDER to the NEW PERSON file for the first time.



The DEGREE field (#6) is updated indirectly via the "ADEG" MUMPS cross-reference on the DEGREE field (#10.6) in the NEW PERSON file (#200).

The three Kernel options affected by the Name Standardization patch are located on the Kernel User Management menu shown in Figure 3. Only the Kernel options affected by Name Standardization are shown.

```
Select Systems Manager Menu Option: user Management

  Add a New User to the System [XUSERNEW]
  Edit an Existing User [XUSEREDIT]
  Reactivate a User [XUSERREACT]
```

Figure 3: Kernel options affected by Name Standardization

After selecting the option Add a New User to the System, Kernel prompts you to enter the person's Name, Initial, Social Security Number, and Sex (i.e., the Identifier fields for the new entry), as shown in Figure 4.

```
Select User Management Option: add <RET> a New User to the System
Enter NEW PERSON's name (Family,Given Middle Suffix): NS' PROVIDER, MARY K. MD
Are you adding 'NSPROVIDER, MARY K. MD' as a new NEW PERSON (the 593RD)?
No// y <RET> (Yes)
Checking SOUNDEX for matches.
No matches found.
Now for the Identifiers.
INITIAL: mkn
SSN: 000555555
SEX: f <RET> FEMALE
```

Figure 4: Adding new users to the NEW PERSON file (#200)

Once the ScreenMan form for editing the properties of the new person has been invoked, the individual name components for that person appear in a new "pop-up" window within the ScreenMan form. This is a "window" that overlays the regular ScreenMan screen in order to present the contents of a selected Multiple as shown, as follows, in Figure 5.

## Using the Name Components "Pop-up" Window

This next example is a continuation of the process of adding a new entry to the NEW PERSON file.

Figure 5 illustrates what happens when we used this "pop-up" window to edit the name, originally entered as Mary to Merrie.

Once the "pop-up" window is invoked, your cursor is positioned in the prompt "Given (First) :". Notice that the change we've made is reflected in a non-editable field at the lower left portion of the "pop-up" window (i.e., NSPROVIDER, MERRIE K MD). This is the name in standard form as it is stored in the .01 field of the NEW PERSON file. All punctuation, except hyphens and the first comma, is removed from the resulting standard form of names stored in the NEW PERSON file.

(For more information on "standard form," see the topic "Standard Format for Person Names in VISTA" in the "Product Description" chapter of this documentation.)

Edit an Existing User		Page 1 of 3
NAME: <b>NSPROVIDER, MARY K MD</b>		
NAME...	<b>NSPROVIDER, MARY K MD</b>	INITIAL: <b>mkn</b>
TITLE:		NICK NAME:
SSN:	<b>000555555</b>	MAIL CODE:
DEGREE:		
Select SEC	Prefix:	NAME COMPONENTS
Want to edi	Given (First): <b>MERRIE</b>	
Want to edi	Middle: <b>K.</b>	
	Family (Last): <b>NS' PROVIDER</b>	
	Suffix: <b>MD</b>	
	<b>NSPROVIDER, MERRIE K MD</b>	
COMMAND: Press <PF1>H for help Insert		

Figure 5: Edits to NAME field of the NEW PERSON file invokes name components "pop-up" window

Edits made to the name components in the pop-up window are made directly to the NAME COMPONENTS file. These edits automatically update the .01 field of the NEW PERSON file.



The PREFIX field in the "pop-up" window, shown in Figure 5, can be entered and stored in the NAME COMPONENTS file. However, Prefix is not part of the standard name stored in .01 field of the NEW PERSON file.

## New Person File Reflects Edits Made to the Name Components File

Once you have made all the necessary edits to the individual name components, press <PF1>C, or press the Return (Enter) key until you've closed the "pop-up" window. Once you've done this, you are back on page 1 of the form. The first field on that page, "NAME..." has been modified so that if you press the Return (Enter) key or edit the name displayed there, the "pop-up" window that displays the name components will re-open.

To move on to the next field, either press the Tab or the down-arrow key.

Notice in Figure 6 that the name Merrie is now reflected in the both the "NAME : " field at the top of the screen, and in the "NAME..." field (i.e., the .01 field of the NEW PERSON file) just beneath it. Once you've closed the "pop-up" window and saved the Screen, any changes you made to the individual name components are reflected in the .01 field (NAME field) of the NEW PERSON file, as well as saved in the NAME COMPONENTS file.

Automatic synchronization is consistently maintained between the .01 field of the NEW PERSON file and the fields in the new NAME COMPONENTS file.

```

                                Edit an Existing User
NAME: NSPROVIDER,MERRIE K MD                                Page 1 of 3
-----
NAME... NSPROVIDER,MERRIE K MD                                INITIAL: mkn
TITLE:                                     NICK NAME:
SSN: 000555555                                       MAIL CODE:
DEGREE:

        PRIMARY MENU OPTION:
Select SECONDARY MENU OPTIONS:

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

        PREFERRED EDITOR:
        Select DIVISION:
        SERVICE/SECTION:
-----
Exit      Save      Next Page      Refresh

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

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

Figure 6: NEW PERSON file is synchronized with NAME COMPONENTS file





## New VA FileMan Name Formatting Function: XLFMTNAME

A new FUNCTION has been installed in the FileMan FUNCTION file (#.5) and is exported with Name Standardization (Patch XU\*8.0\*134). This function can be used within the FileMan Print options, or within word-processing windows used to generate forms. The new FUNCTION allows the user to display names in various formats from any **VISTA** file containing a name field.

Here's how it works:

- The first three parameters passed to the FUNCTION identify the file, record, and field that contain the specific name to be formatted.
- If the NAME field in the **VISTA** file has been standardized and linked to the NAME COMPONENTS file (#20), then the FUNCTION obtains the component parts of that Name from the NAME COMPONENTS file.
- If the FUNCTION can't find the name in the NAME COMPONENTS file, it will obtain the name directly from the **VISTA** file that contains the Name.

Format: XLFMTNAME(FILE#,FIELD#,IENS,FORMAT,FLAGS)

Input Parameters:

**FILE#** (Required). The value of this parameter is the file number of the file containing the name to be returned.

**FIELD#**(Required). The value of this parameter is the field number of the field containing the name to be returned.

**IENS** (Required). The value of this parameter is the Internal Entry number of the entry in the file containing the name to be returned. The Internal Entry number can be followed by a comma to form an IEN string (IENS). However, the comma is not required.

**FORMAT** (Must be defined, or set to a null value). The **FORMAT** parameter controls the general formatting of the output. If passed as a null value, the parameter will default to G as shown in example a). The **FORMAT** parameter can contain one of the following:

- F Family (Last) Name is returned first.
- G Given (First) Name is returned first.
- O The letter O returns only the Family, or Last Name.

**FLAGS** (Must be defined, or set to a null value). Additional flags to control formatting. If passed as a null value, none of the actions controlled by the flags are performed. More than one of these flags can be passed in the **FLAGS** parameter as shown in examples b) and c). The acceptable **FLAG** values are:

C	If "F" is passed in the FORMAT parameter, the name is returned with a comma between the Family (Last) and Given (First) Names. The "C" flag is ignored if the "F" FORMAT is not passed.
D	The Degree is returned (if any).
Dc	The Degree is returned (if any), preceded by a comma and space.
L#	The name is truncated to a maximum length of # characters, where # is an integer between 1 and 256. (See the \$\$NAMEFMT^XLFFNAME call in this section for a description of the pruning algorithm.)
M	Name is returned in Mixed case, with first letters capitalized.
P	The Prefix is returned (if any).
S	Standardizes the name components before building the formatted name (e.g., it removes most punctuation and removes any spaces from the Family name)
Xc	A comma and space will precede the returned suffix (if any).

### Examples:

Note that since this is a FUNCTION from the FileMan FUNCTION file and not a callable routine, it can be used only within FileMan options. This FUNCTION can be used in the FileMan Print option at either the SORT BY or PRINT FIELD prompts. It can also be used in word-processing windows to print form documents, as defined in the FileMan Users Manual.

- a) Suppose we want to print a report from the NEW PERSON file (#200). In this report, we want the NAME field to print with the Given name first. We do not want to print Degrees or Prefixes, we do not want to standardize the name, and we do not want to print a comma and space before the suffix, if any. Using this example, a name with Prefix "MR.", Family name "NSPROVIDER", Given name "JOHN", Middle name "K.", Suffix "JR." and degree "PHD" would print: "JOHN K. NSPROVIDER JR."

Note that the internal entry number (IEN) from File #200 can be passed by using the field name NUMBER, which is used to designate the IEN of the current record on any file. Note also that the FORMAT and FLAGS parameter are passed as null values.

At the PRINT FIELD prompt, enter:

```
FIRST PRINT FIELD: XLFMTNAME(200,.01,NUMBER,"", "")
```

- b) Suppose we have a File #662002 in which the .01 field NAME points to the NEW PERSON file.

In both this example, and in c), we wish to display the NAME fields with the Given name first. We want the names to appear with the first letter of each name in upper case and the rest in lower-case. If Prefix, Suffix or Degree values exist, we wish to include the Prefix, the Suffix preceded by a comma, and the Degree preceded by a comma. A pointed-to name field in the NEW PERSON file with a Prefix "MR.", Family name "NSPROVIDER", Given name "JOHN", Middle Name "K.", Suffix "JR" and Degree "PHD" should be displayed as:

Mr. John K. Nsprovider, Jr, PhD

The .01 NAME field in File #662002 points to File #200. The name field that we wish to format resides in File #200. We must use INTERNAL(NAME) in the IENS parameter to indicate the IEN of the file where the name resides. This gives us the internal pointer value (i.e. the IEN from File #200).

If we wish to print a report from File #662002, displaying the names in the format described previously, then at the PRINT FIELD prompt, enter:

FIRST PRINT FIELD: XLFMTNAME(200,.01,INTERNAL(NAME),"G","DcPXcM")

- c) We can also use the FUNCTION in a word-processing window to generate form documents or letters. Suppose we wish to print a letter for each of the entries in File #662002 as described in example b) listed previously, and that we wish to format the names in the same way. At the start of the letter, we would like to print the NAME field within the salutation. The first line of the word processing field that will be used for the form document could be set up like this:

Dear | XLFMTNAME(200,.01,INTERNAL(NAME),"G","DcPXcM")|:



# Programmer Manual Information

This is the Programmer Manual section of this supplemental documentation for Patch XU\*8.0\*134. It will be incorporated into the Kernel Systems Manual, Version 8.0 at a later date.

The intended audience for this chapter is Information Resource Management (IRM) and Veterans Affairs Medical Center (VAMC) personnel who will be doing the changes to the system. However, it can also be helpful to others in Technical Service, the Program Office, National VISTA Support (NVS), and Technical Integration.

## Name Components and New Person Files

### Field Descriptions in the New Name Components File (#20)

The NAME COMPONENTS file is a new file exported with Name Standardization (Patch XU\*8.0\*134). It will hold the component parts of a name, as shown below. Each entry in the NAME COMPONENTS file is associated with an entry and a name type field in another file. We call this file the source file.

These first three fields shown below (FILE, FIELD, and IENS) are the Primary key for the NAME COMPONENTS file. They uniquely identify the record in the NEW PERSON file (#200). These key fields identify the source file, source field and entry on which the original name resides. The actual name components are Fields #1 through #6 [i.e., FAMILY (LAST) NAME, GIVEN (FIRST) NAME, MIDDLE NAME, PREFIX, SUFFIX, and DEGREE]. The field, NOTES ABOUT NAME (#11), stores the original text of the person's name, only if parenthetical text was removed during the data conversion.

1. **FILE field (#.01)** – This field holds the number of the file or subfile that contains the name. This field must contain a number between 0 and 99999999999, with up to 7 decimal digits.
2. **FIELD field (#.02)** – This field holds the number of the field that contains the name. This is a Number field. This field must contain a number between 0 and 99999999999, with up to 7 decimal digits.
3. **IENS field (#.03)** – This field holds the internal entry number string (IENS) of the entry that contains the name. This is a Free Text field defined as 2-50 characters.
4. **FAMILY (LAST) NAME field (#1)** – This field is referred to as the "last name." This is a Free Text field, defined as 1-35 characters. It must begin with an upper case letter, and must contain only upper case letters, numbers, and punctuation, excluding ^ and \. This is a required field (i.e. it cannot be null).

A new-style record-level MUMPS cross-reference on fields 1, 2, 3, and 5 will build a standard name based on the component parts, prune the result to the maximum length of the source name field, and then update the name field in the source file.

(For information on the pruning algorithm, see the description for the routine to build the name from the component parts in the chapter "APIs" in the "Programmer Manual Information" section of this documentation.)

A new-style record-level MUMPS cross-reference on fields 1, 2, 3, and 5 will update the SIGNATURE BLOCK PRINTED NAME field (#20.2) in the NEW PERSON file, if the source file is 200 and the source field is .01.

5. **GIVEN (FIRST) NAME field (#2)** – This is referred to as the "first name." This is a Free Text field, defined as 1-25 characters. It must begin with an upper case letter, and must contain only upper case letters, numbers, and punctuation, excluding ^ and `. This field is optional (i.e., it can be null). Editing this field will affect the source name field and, if the source file is #200 and the source field is .01, it will affect field 20, the SIGNATURE BLOCK PRINTED NAME in the NEW PERSON file. (See the FAMILY (LAST) NAME field, listed previously, for more information).
6. **MIDDLE NAME field (#3)** – This is a Free Text field, defined as 1-25 characters. It begin start with an upper case letter, and must contain only upper case letters, numbers, and punctuation, excluding ^ and `. It cannot be NMI or NMN. This field is optional (i.e., it can be null). Editing this field will affect the source name field and, if the source file is #200 and the source field is .01, it will affect the Signature Block Printed Name field in the NEW PERSON file. (See the FAMILY (LAST) NAME field, listed previously, for more information).
7. **PREFIX field (#4)** – This is a Free Text field, defined as 1-10 characters. It must begin with an upper case letter, and must contain only upper case letters, numbers, and punctuation, excluding ^ and `. This field is optional (i.e., it can be null).



This field is not included as part of the source name field.

8. **SUFFIX field (#5)** – This is a Free Text field, defined as 1-10 characters. It must begin with an upper case letter or number, and must contain only upper case letters, numbers, and punctuation, excluding ^ and `. This field is optional (i.e., it can be null). Editing this field will affect the source name field and, if the source file is 200 and the source field is .01, it will affect the SIGNATURE BLOCK PRINTED NAME field in the NEW PERSON file. (See the FAMILY (LAST) NAME field, listed previously, for more information).
9. **DEGREE field (#6)** – This is a Free Text field, defined as 1-10 characters. It must begin with an upper case letter, and must contain only upper case letters, numbers, and punctuation, excluding ^ and `. This field is optional (i.e., it can be null).



This field is not included as part of the source name field.

A new-style record-level MUMPS cross-reference on the DEGREE field (#6) of the NAME COMPONENTS file will keep this field in synchronization with the DEGREE field (#10.6) in the NEW PERSON file; that is, if the source file is 200 and the source field is .01, and the NAME COMPONENTS file DEGREE field (#6) is edited, the "ADEG" MUMPS cross-reference will update the corresponding NEW PERSON file DEGREE field (#10.6).

## 10. NOTES ABOUT NAME field (#11)

This is a Free Text field, defined as 3-100 characters. It stores the original text of the person's name. This field is populated ONLY if parenthetical text was removed from the name during the data conversion. Otherwise, the data conversion does NOT populate this field. During the New Person Name Conversion, which is run as part of the post-install routine of Patch XU\*8.0\*134, if a name in the NEW PERSON file contains text in parentheses (), brackets [], or braces {}, that text is removed from the name, and the original name with the parenthetical text is recorded in this field. (This routine is documented in the description for Patch XU\*8.0\*134 on the Patch Module.)



The Standard Name in the NEW PERSON file does not contain a Prefix or a Degree. Therefore, it only populates the following fields in the NAME COMPONENTS file:

- FAMILY (LAST) NAME (#1),
- GIVEN (FIRST) NAME (#2),
- MIDDLE NAME (#3), and
- SUFFIX (#5).

## Changes to Data Dictionary of New Person File (#200)

The input transform of the NEW PERSON file will continue to check that the value of the NAME field (#.01) is 3 to 35 characters in length, contains only one comma, and has at least one character before and one character after the comma. In addition, the input transform will call the Name Standardization Routine to convert the input to the following standard form:

Family\_name,Given\_name<space>Middle\_name<space>Suffix.

The Name Standardization Routine called by the input transform also parses the input into its component parts: Family Name, Given Name, Middle Name, and Suffix. A new "ANAME" MUMPS cross-reference on the .01 field of the NEW PERSON file updates or creates the corresponding entry in the NAME COMPONENTS file with the name components returned by the input transform.

A new NAME COMPONENTS pointer field (#10.1) defined as a pointer to the NAME COMPONENTS file establishes a link between each entry in the NEW PERSON file with the corresponding entry in the NAME COMPONENTS file that contains the component parts of the Name. The "ANAME" MUMPS cross-reference on the .01 field of the NEW PERSON file updates this field, as necessary.

Whenever the DEGREE field (#10.6) in the NEW PERSON file is edited, a new "ADEG" MUMPS cross-reference on this field updates the DEGREE field (#6) in the NAME COMPONENTS file.





## Steps to Standardize a Name Field

### Source File Data Dictionary

Make the following changes to the data dictionary of the source file that contains the name field:

- Modify the input transform of the name field to call STDNAME^XLFNAME to standardize and parse the input.

For example, you might add the following code to the input transform:

```
I $D(X) S nmspNC=X D STDNAME^XLFNAME(.nmspNC,"C") S X=nmspNC
```

This code transforms the user input, which is in the variable X, to standard form, and returns in the nmspNC array the unstandardized name components.

In some circumstances, such as when a new name is created, you may want the nmspNC array to remain defined after the input transform is executed. The cross-reference that updates the NAME COMPONENTS file can then use this array to create the NAME COMPONENTS entry with the unstandardized form of the name components.

(For an example how the input transform can be modified, see the input transform code on the .01 field of File #200 (NEW PERSON) that was distributed with the Name Standardization patch XU\*8.0\*134.)

- Modify the help and description of the name field to describe how names should be entered, and how the entered name will be standardized. See the chapter "Guidelines for Entering Person Names in VISTA" in the "User Manual Information" section of this documentation for the type of information you may want to include here.
- Create a name components pointer field from the source file to the NAME COMPONENTS file. This pointer links each name with the corresponding entry in the NAME COMPONENTS file that holds the components parts of the name, and can be used to navigate from the source file to the corresponding entry in the NAME COMPONENTS file.

For example, the Name Standardization patch has created a new pointer field, NAME COMPONENTS (#10.1), in the NEW PERSON file (#200).

- Create a new-style MUMPS index that optionally uses UPDCOMP^XLFNAME2 and DELCOMP^XLFNAME2 entry points to update the NAME COMPONENTS entry and the name components pointer when the source name changes. (You need to subscribe to IA #3066 to use these APIs. Alternatively, controlled- subscription IA #3041 allows both read and write access to the NAME COMPONENTS file via FileMan tools.)

For example, the Name Standardization patch has created the "ANAME" cross-reference designed to keep the .01 field of the NEW PERSON file in synchronization with the name components in the NAME COMPONENTS file. The index description is located in the INDEX file (#.11) and can be printed via the Indexes Only format of a DD listing of the NEW PERSON file.

- If there's a lookup cross-reference on the name field, replace it with a new-style index with a "transform for lookup" that converts the lookup value to standard form. If FileMan is used to lookup a name in non-standard form, and that non-standard name is not in the index, FileMan automatically executes the transform for lookup to convert the name to standard form, and tries the lookup again.

For example, the Name Standardization patch has replaced the traditional "B" index on the .01 field of the NEW PERSON file with a new-style index with a "transform for lookup." The index description is located in the INDEX file (#.11) and can be printed via the Indexes Only format of a DD listing of the NEW PERSON file.

## Name Conversion

Write a post-install routine that converts existing names to standard form. Loop through all the entries in the file, and for each name:

- Call STDNAME^XLFNAME to obtain the standard form of the name and the name components.
- Call UPDCOMP^XLFNAME2 to create the NAME COMPONENTS entry and to update the name components pointer. You can also pass NAME("NOTES") to UPDCOMP^XLFNAME2 to file information in the NOTES ABOUT NAME field if parenthetical text is stripped to form the standard name. (See IA #3041 and #3066.)
- Update the name field with the standard name.
- Record in ^XTMP the information returned from STDNAME^XLFNAME about ambiguities and potential problems encountered.
- If the standard form of the name is different from the original name, record in ^XTMP that information, as well.
- Write a routine that prints the information in ^XTMP.

(For an example of how this conversion might be written, see the conversion routine POST^XLFNAME distributed with the Name Standardization patch.)

## End-User Interfaces

Modify the interfaces that display and allow editing of the name as follows:

- Modify the interfaces that prompt the user for the name to also prompt for the name components. (IA #3041 is a controlled subscription integration agreement that allows both read and write access via VA FileMan to the fields in the NAME COMPONENTS file.) You can use the newly created name components pointer to navigate from the source file the NAME COMPONENTS file.

- In places where the name is displayed to the end-user, use the \$\$NAMEFMT^XLFNNAME API or the XLFMTNAME FileMan FUNCTION to build the name from the components stored in the NAME COMPONENTS file.-This allows display of the name in a user-friendlier format, because most of the original spacing and punctuation is preserved in the NAME COMPONENTS file.



## APIs

This chapter documents three categories of Kernel Supported Calls as they relate to the Name Standardization patch (Patch XU\*8.0\*134).

1. The first category is titled "New Supported Reference Integration Agreements." This chapter lists and describes the Kernel APIs for Name Standardization that are supported for general use by all **VISTA** applications.
2. The second category is titled "New Controlled Subscription Integration Agreements." This chapter lists and describes the Kernel APIs for Name Standardization for which you must obtain an Integration Agreement ([IA] - Formerly referred to as a DBIA) to use.
3. The third category is titled "Modified Kernel API." This chapter describes the existing Kernel API `$$ADD^XUSERNEW` that was modified by the Name Standardization Patch.

(For a list of the Integration Agreements related to Name Standardization see the chapter "External Relations" in the "Technical Manual Information" section of this documentation.)

### New Supported Reference Integration Agreements

These are the Kernel Supported Reference Integration Agreements (IAs) for Name Standardization. They are listed in alphabetical order by entry point. Supported Reference APIs are open for use by any **VISTA** application as defined by the IA. They have been recorded as a Supported Reference in the IA database on FORUM. It is not required that **VISTA** packages request an IA to use them. The following APIs are alphabetized by Entry Point.

#### **\$\$BLDNAME^XLFNAME: Build Name from Component Parts**

This extrinsic function takes the component parts of a name and returns the name, truncated if necessary, in the following format:

Family\_name,Given\_name<space>Middle\_name<space>Suffix(es)

#### **Format:**

`$$BLDNAME^XLFNAME (. NAME , MAX)`

#### **Input Parameters:**

<b>.NAME</b>	(Required) The component parts of the name:  NAME("FAMILY") = Family (Last) Name NAME("GIVEN") = Given (First) Name(s) NAME("MIDDLE") = Middle Name(s)
--------------	--

	<p>NAME("SUFFIX") = Suffix(es)</p> <p>Alternatively, this array can contain the file number, IENS, and field number of the field that contains the name. If the name has a corresponding entry in the NAME COMPONENTS file, then the name components are obtained from that entry. Otherwise, the name is obtained directly from the file, record, and field specified, and the name components are obtained by making a call to STDNAME^XLFFNAME.</p> <p>NAME("FILE") = Source file number (required)  NAME("IENS") = IENS of entry in the source file (required)  NAME("FIELD") = Source field number (required)</p>
<b>MAX</b>	The maximum length of the Name to be returned. (Default = 256) See " <b>Details:</b> " as follows for a description of the pruning algorithm.

**Details:**

If the MAX input parameter is used, and the resulting name is longer than MAX, the following pruning algorithm is performed to shorten the name:

1. Truncate Middle Name from the right-most position until only the initial character is left;
2. Drop suffix;
3. Truncate Given Name from the right-most position until only the initial character is left;
4. Truncate Family Name from the right-most position;
5. Truncate the name from the right.

**Examples:**

1. Suppose the MYNAME array contains the following elements:

```
MYNAME ("FAMILY") = "NS ' PROVIDER"
MYNAME ("GIVEN") = "JOHN"
MYNAME ("MIDDLE") = "K. "
MYNAME ("SUFFIX") = "JR"
```

Calls to \$\$BLDNAME^XLFFNAME will return the name as follows:

```
$$BLDNAME^XLFFNAME (.MYNAME) → NSPROVIDER, JOHN K JR
$$BLDNAME^XLFFNAME (.MYNAME, 12) → NSPROVIDER, JOH K
```

2. If an entry in the NAME COMPONENTS stores the components of a name stored in the NAME field (#.01) of record number 32 in the NEW PERSON file, and the data in the corresponding record in the NAME COMPONENT file is:

```
FILE = 200
FIELD = .01
IENS = "32, "
GIVEN NAME = "JOHN"
MIDDLE NAME = "K. "
FAMILY NAME = "NS ' PROVIDER"
```

## Programmer Manual Information

```
SUFFIX = "JR"
```

you can set:

```
MYNAME("FILE") = 200  
MYNAME("FIELD") = .01  
MYNAME("IENS") = "32,"
```

and call \$\$BLDNAME^XLFFNAME as in Example 1, listed previously:

```
$$BLDNAME^XLFFNAME(.MYNAME) → NSPROVIDER, JOHN K JR  
$$BLDNAME^XLFFNAME(.MYNAME, 12) → NSPROVIDER, JOH K
```



## \$\$CLEANC^XLFNNAME: Name Component Standardization Routine

This extrinsic function takes a single name component and returns that name in standard format.

### Format:

\$\$CLEANC^XLFNNAME (COMP, FLAGS)

### Input Parameters:

<b>COMP</b>	(Required) The name component to be converted to standard format.	
<b>FLAGS</b>	Flags to control processing. Possible values are:	
	<b>F</b>	<p>If the name component to be converted is the FAMILY (LAST) NAME, pass the "F" flag. With the "F" flag, colons (:), semicolons (;), and commas (,) are converted to hyphens (-). Spaces and all punctuation except hyphens are removed. Leading and trailing spaces and hyphens are removed. Two or more consecutive spaces or hyphens are replaced with a single space or hyphen.</p> <p>Without the "F" flag, the component is converted to upper case. Colons, semicolons, commas, and periods (.) are converted to spaces. All punctuation except for hyphens and spaces are removed. Leading and trailing spaces and hyphens are removed. Two or more consecutive spaces or hyphens are replaced with a single space or hyphen. Birth position indicators 1ST through 10TH are changed to their Roman numeral equivalents.</p>

### Examples:

1. Standardize family (last) name:

```
$$CLEANC^XLFNNAME ("NS' PROVIDER-DE LA ROSA", "F") → NSPROVIDER-
DELA ROSA
```

```
$$CLEANC^XLFNNAME ("NS. PROVIDER", "F") → NSPROVIDER
```

2. Standardize other (non-family) name components:

```
$$CLEANC^XLFNNAME ("E.C.") → E C
```

```
$$CLEANC^XLFNNAME ("RENEE ") → RENEE
```

```
$$CLEANC^XLFNNAME ("MARY ANN") → MARY ANN
```

```
$$CLEANC^XLFNNAME ("JO-ANNE") → JO-ANNE
```

**\$\$FMNAME^XLFNAME: Convert HL7 Formatted Name to Name**

This extrinsic function converts an HL7 formatted name to a name in **VISTA** format.

**Format:**

\$\$FMNAME^XLFNAME (NAME , FLAGS , DELIM)

**Input Parameters:**

<b>[.]NAME</b>	(Required) NAME is the HL7 name to be converted. If the "C" flag is used, the name components are returned in nodes descendent from NAME. (See " <b>Output:</b> " just after this table.)	
<b>FLAGS</b>	Flags to controls processing. Possible values are:	
	<b>C</b>	Return name components in the NAME array. (See " <b>Output:</b> " just after this table.)
	<b>L#</b>	Truncate the returned name to a maximum Length of # characters, where # is an integer between 1 and 256.
	<b>M</b>	Return the name in <b>Mixed</b> case, with the first letter of each name component capitalized.
	<b>S</b>	Return the name in <b>Standardized</b> form.
<b>DELIM</b>	The delimiter used in the HL7 formatted name. (Default = "^")	

**Output:**

<b>NAME</b>	<p>If the FLAGS input parameter contains a "C", the component parts of the name are returned in the NAME array:</p> <p>NAME("FAMILY") = Family (Last) Name  NAME("GIVEN") = Given (First) Name(s)  NAME("MIDDLE") = Middle Name(s)  NAME("SUFFIX") = Suffix(es)</p>
-------------	---

**Details:**

If the L# flag is used, and the resulting name is longer than #, the following pruning algorithm is performed to shorten the name:

1. Truncate Middle Name from the right-most position until only the initial character is left;
2. Drop suffix;
3. Truncate Given Name from the right-most position until only the initial character is left;
4. Truncate Family Name from the right-most position;
5. Truncate the name from the right.

**Examples:**

1. Convert an HL7 formatted name to a **VISTA** name:

```

$$FMNAME^XLFNAME ("NS ' PROVIDER^JOHN^K.^JR^MR.^PHD") →
  NS ' PROVIDER, JOHN K. JR
$$FMNAME^XLFNAME ("NS ' PROVIDER^JOHN^K.^JR^MR.^PHD", "S") →
  NSPROVIDER, JOHN K JR
$$FMNAME^XLFNAME ("NS ' PROVIDER^JOHN^K.^JR^MR.^PHD", "M") →
  Ns 'provider, John K. Jr
$$FMNAME^XLFNAME ("NS ' PROVIDER^JOHN^K.^JR^MR.^PHD", "SL12") →
  NSPROVIDER, JOH K

```

2. Convert an HL7 formatted name where "~" is the delimiter to a standard name:

```

$$FMNAME^XLFNAME ("NS ' PROVIDER~JOHN~K.~JR~MR", "S", "~") →
  NSPROVIDER, JOHN K JR

```

3. Convert an HL7 formatted name to a standard name, and return the components of that name in the MYNAME array:

```

>S MYNAME="NS ' PROVIDER^JOHN^K.^JR^MR.^PHD"

>W $$FMNAME^XLFNAME (.MYNAME, "CS")
NSPROVIDER, JOHN K JR

>ZW MYNAME
MYNAME=NS ' PROVIDER^JOHN^K.^JR^MR.^PHD
MYNAME ("FAMILY")=NS ' PROVIDER
MYNAME ("GIVEN")=JOHN
MYNAME ("MIDDLE")=K.
MYNAME ("SUFFIX")=JR

```

## \$\$HLNAME^XLFNAME: Convert Name to HL7 Formatted Name

This extrinsic function converts a name to HL7 format.

### Format:

\$\$HLNAME^XLFNAME ( [ . ] NAME , FLAGS , DELIM )

### Input Parameters:

<b>[.]NAME</b>	<p>(Required) The component parts of the name to be converted:</p> <p>NAME("FAMILY") = Family (Last) Name (required)  NAME("GIVEN") = Given (First) Name(s) (optional)  NAME("MIDDLE") = Middle Name(s) (optional)  NAME("SUFFIX") = Suffix(es) (optional)  NAME("PREFIX") = Prefix (optional)  NAME("DEGREE") = Degree (optional)</p> <p>Alternatively, this array can contain the file number, IENS, and field number of the field that contains the name. If the name has a corresponding entry in the NAME COMPONENTS file (#20), then the name components are obtained from that entry. Otherwise, the name is obtained directly from the file, record, and field specified, and the name components are obtained by making a call to STDNAME^XLFNAME.</p> <p>NAME("FILE") = Source file number (required)  NAME("IENS") = IENS of entry in the source file (required)  NAME("FIELD") = Source field number (required)</p> <p>Another alternative is to pass in the unsubscripted NAME parameter the name to be converted. \$\$HLNAME^XLFNAME obtains the components parts of that name by making a call to STDNAME^XLFNAME. This alternative is recommended only for names that do not have associated entries on the NAME COMPONENTS file.</p>				
<b>FLAGS</b>	<p>Flags to controls processing. Possible values are:</p> <table border="1"> <tr> <td data-bbox="402 1335 459 1398"><b>L#</b></td> <td data-bbox="459 1335 1341 1398">Truncate the returned name to a maximum Length of # characters, where # is an integer between 1 and 256.</td> </tr> <tr> <td data-bbox="402 1398 459 1430"><b>S</b></td> <td data-bbox="459 1398 1341 1430">Return the name components in the HL7 formatted name in Standardized form.</td> </tr> </table>	<b>L#</b>	Truncate the returned name to a maximum Length of # characters, where # is an integer between 1 and 256.	<b>S</b>	Return the name components in the HL7 formatted name in Standardized form.
<b>L#</b>	Truncate the returned name to a maximum Length of # characters, where # is an integer between 1 and 256.				
<b>S</b>	Return the name components in the HL7 formatted name in Standardized form.				
<b>DELIM</b>	The delimiter to use in the HL7 string. (Default = "^")				

### Details:

If the L# flag is used, and the resulting name is longer than #, the following pruning algorithm is performed to shorten the name:

1. Truncate Middle Name from the right-most position until only the initial character is left;
2. Drop suffix;
3. Truncate Given Name from the right-most position until only the initial character is left;
4. Truncate Family Name from the right-most position;
5. Truncate the name from the right.

**Examples:**

1. Suppose the MYNAME array contains the following elements:

```

MYNAME ("PREFIX") = "MR. "
MYNAME ("GIVEN") = "JOHN"
MYNAME ("MIDDLE") = "K. "
MYNAME ("FAMILY") = "NS ' PROVIDER"
MYNAME ("SUFFIX") = "JR"
MYNAME ("DEGREE") = "PHD"

```

Calls to \$\$HLNAME^XLFFNAME will return the name as follows:

```

$$HLNAME^XLFFNAME (.MYNAME) → NS ' PROVIDER^JOHN^K.^JR^MR.^PHD
$$HLNAME^XLFFNAME (.MYNAME, "", "~") →
  NS ' PROVIDER~JOHN~K.^JR~MR.^PHD
$$HLNAME^XLFFNAME (.MYNAME, "S", "~") → NSPROVIDER~JOHN~K~JR~MR~PHD
$$HLNAME^XLFFNAME (.MYNAME, "L12S") → NSPROVIDER^JOH^K

```

2. If an entry in the NAME COMPONENTS stores the components of a name stored in the NAME field (#.01) of record number 32 in the NEW PERSON file, and the data in the corresponding record in the NAME COMPONENT file is:

```

FILE = 200
FIELD = .01
IENS = "32, "
PREFIX = "MR. "
GIVEN NAME = "JOHN"
MIDDLE NAME = "K. "
FAMILY NAME = "NS ' PROVIDER"
SUFFIX = "JR"
DEGREE = "PHD"

```

you can set:

```

MYNAME ("FILE") = 200
MYNAME ("FIELD") = .01
MYNAME ("IENS") = "32, "

```

and call \$\$HLNAME^XLFFNAME as in Example 1, listed previously, to return the name in various formats.

3. Convert a name passed by value to HL7 format:

```

$$HLNAME^XLFFNAME ("NS ' PROVIDER, JOHN HOWARD II") →
  NS ' PROVIDER^JOHN^HOWARD^II
$$HLNAME^XLFFNAME ("NS ' PROVIDER, JOHN HOWARD II", "S") →
  NSPROVIDER^JOHN^HOWARD^II
$$HLNAME^XLFFNAME ("NS ' PROVIDER, JOHN HOWARD II", "SL10", "~") →
  NSPROVIDER~J~H

```

## NAMECOMP^XLFNAME: Component Parts from Standard Name

This procedure takes a name in standard format and returns in an array the component parts of that name.

### Format:

NAMECOMP^XLFNAME ( .NAME )

### Input Parameters:

<b>.NAME</b>	(Required) NAME is the name in standard format to be parsed. NAMECOMP^XLFNAME returns the component parts of the name in nodes descendent from NAME. (See " <b>Output:</b> " just after this table.)
--------------	--

### Output:

<b>NAME</b>	The component parts of the name are returned in the NAME array passed in.  NAME("FAMILY") = Family (last) Name NAME("GIVEN") = Given (first) Name NAME("MIDDLE") = Middle Name NAME("SUFFIX") = Suffix(es)
-------------	---

### Example:

In this example, the variable MYNAME is set to the standard name, and the NAMECOMP^XLFNAME call is made to return in the MYNAME array the component parts of that name:

```
>S MYNAME="MCDONALD-NSPROVIDER,MARY ANN S MD"
>D NAMECOMP^XLFNAME (.MYNAME)

>ZW MYNAME
MYNAME=MCDONALD-NSPROVIDER,MARY ANN S MD
MYNAME ("FAMILY")=MCDONALD-NSPROVIDER
MYNAME ("GIVEN")=MARY ANN
MYNAME ("MIDDLE")=S
MYNAME ("SUFFIX")=MD
```

## \$\$NAMEFMT^XLFNAME: Formatted Name from Name Components

This extrinsic function returns a name converted to a form useful for display.

### Format:

\$\$NAMEFMT^XLFNAME ( .NAME , FORMAT , FLAGS )

### Input Parameters:

<b>.NAME</b>	<p>(Required) An array that contains the component parts of the name:</p> <p>NAME("FAMILY") = Family (Last) Name (required)  NAME("GIVEN") = Given (First) Name(s) (optional)  NAME("MIDDLE") = Middle Name(s) (optional)  NAME("SUFFIX") = Suffix(es) (optional)  NAME("PREFIX") = Prefix (optional)  NAME("DEGREE") = Degree (optional)</p> <p>Alternatively, this array can contain the file number, IENS, and field number of the field that contains the name. If the name has a corresponding entry in the NAME COMPONENTS file (#20), then the name components are obtained from that entry. Otherwise, the name is obtained directly from the file, record, and field specified, and the name components are obtained by making a call to STDNAME^XLFNAME.</p> <p>NAME("FILE") = Source file number (required)  NAME("IENS") = IENS of entry in the source file (required)  NAME("FIELD") = Source field number (required)</p>																
<b>FORMAT</b>	<p>Controls the general formatting of the output. (Default = G) Possible values are:</p> <table border="1"> <tr> <td data-bbox="396 1178 464 1209"><b>F</b></td> <td data-bbox="464 1178 1339 1209">Return <b>F</b>amily (Last) Name first.</td> </tr> <tr> <td data-bbox="396 1209 464 1241"><b>G</b></td> <td data-bbox="464 1209 1339 1241">Return <b>G</b>iven (First) Name first.</td> </tr> <tr> <td data-bbox="396 1241 464 1276"><b>O</b></td> <td data-bbox="464 1241 1339 1276">Return <b>O</b>nly the Family (Last) Name</td> </tr> </table>	<b>F</b>	Return <b>F</b> amily (Last) Name first.	<b>G</b>	Return <b>G</b> iven (First) Name first.	<b>O</b>	Return <b>O</b> nly the Family (Last) Name										
<b>F</b>	Return <b>F</b> amily (Last) Name first.																
<b>G</b>	Return <b>G</b> iven (First) Name first.																
<b>O</b>	Return <b>O</b> nly the Family (Last) Name																
<b>FLAGS</b>	<p>Flags to controls processing. Possible values are:</p> <table border="1"> <tr> <td data-bbox="396 1312 464 1402"><b>C</b></td> <td data-bbox="464 1312 1339 1402">If the "F" format is used, return a <b>C</b>omma between the Family (Last) and Given (First) Names. Otherwise, the Family (Last) Name and the Given (First) Name are separated by a space. (Ignored if the "F" format is not used.)</td> </tr> <tr> <td data-bbox="396 1402 464 1434"><b>D</b></td> <td data-bbox="464 1402 1339 1434">Return the <b>D</b>egree.</td> </tr> <tr> <td data-bbox="396 1434 464 1465"><b>Dc</b></td> <td data-bbox="464 1434 1339 1465">Return the <b>D</b>egree preceded by a <b>c</b>omma and space.</td> </tr> <tr> <td data-bbox="396 1465 464 1556"><b>L#</b></td> <td data-bbox="464 1465 1339 1556">Truncate the returned name to a maximum <b>L</b>ength of # characters, where # is an integer between 1 and 256. See "<b>Details:</b>" as follows for a description of the pruning algorithm.</td> </tr> <tr> <td data-bbox="396 1556 464 1629"><b>M</b></td> <td data-bbox="464 1556 1339 1629">Return the name in <b>M</b>ixed case, with the first letter of each name component capitalized.</td> </tr> <tr> <td data-bbox="396 1629 464 1661"><b>P</b></td> <td data-bbox="464 1629 1339 1661">Return the <b>P</b>refix.</td> </tr> <tr> <td data-bbox="396 1661 464 1692"><b>S</b></td> <td data-bbox="464 1661 1339 1692">Standardize the name components before building formatted name.</td> </tr> <tr> <td data-bbox="396 1692 464 1719"><b>Xc</b></td> <td data-bbox="464 1692 1339 1719">Precede the Suffi<b>X</b> with a comma and space.</td> </tr> </table>	<b>C</b>	If the "F" format is used, return a <b>C</b> omma between the Family (Last) and Given (First) Names. Otherwise, the Family (Last) Name and the Given (First) Name are separated by a space. (Ignored if the "F" format is not used.)	<b>D</b>	Return the <b>D</b> egree.	<b>Dc</b>	Return the <b>D</b> egree preceded by a <b>c</b> omma and space.	<b>L#</b>	Truncate the returned name to a maximum <b>L</b> ength of # characters, where # is an integer between 1 and 256. See " <b>Details:</b> " as follows for a description of the pruning algorithm.	<b>M</b>	Return the name in <b>M</b> ixed case, with the first letter of each name component capitalized.	<b>P</b>	Return the <b>P</b> refix.	<b>S</b>	Standardize the name components before building formatted name.	<b>Xc</b>	Precede the Suffi <b>X</b> with a comma and space.
<b>C</b>	If the "F" format is used, return a <b>C</b> omma between the Family (Last) and Given (First) Names. Otherwise, the Family (Last) Name and the Given (First) Name are separated by a space. (Ignored if the "F" format is not used.)																
<b>D</b>	Return the <b>D</b> egree.																
<b>Dc</b>	Return the <b>D</b> egree preceded by a <b>c</b> omma and space.																
<b>L#</b>	Truncate the returned name to a maximum <b>L</b> ength of # characters, where # is an integer between 1 and 256. See " <b>Details:</b> " as follows for a description of the pruning algorithm.																
<b>M</b>	Return the name in <b>M</b> ixed case, with the first letter of each name component capitalized.																
<b>P</b>	Return the <b>P</b> refix.																
<b>S</b>	Standardize the name components before building formatted name.																
<b>Xc</b>	Precede the Suffi <b>X</b> with a comma and space.																

**Details:**

If the L# flag is used, and the resulting name is longer than #, the following pruning algorithm is performed to shorten the name:

1. Drop Degree;
2. Drop Prefix;
3. Truncate Middle Name from the right-most position until only the initial character is left;
4. Drop suffix;
5. Truncate Given Name from the right-most position until only the initial character is left;
6. Truncate Family Name from the right-most position;
7. Truncate the name from the right.

**Examples:**

1. Suppose the MYNAME array contains the following elements:

```
MYNAME ("PREFIX") = "MR."
MYNAME ("GIVEN") = "JOHN"
MYNAME ("MIDDLE") = "K."
MYNAME ("FAMILY") = "NS ' PROVIDER"
MYNAME ("SUFFIX") = "JR"
MYNAME ("DEGREE") = "PHD"
```

Calls to \$\$NAMEFMT^XLFFNAME will return the name as follows:

```
$$NAMEFMT^XLFFNAME (.MYNAME, "F") → NS ' PROVIDER JOHN K. JR
$$NAMEFMT^XLFFNAME (.MYNAME, "F", "C") → NS ' PROVIDER, JOHN K. JR
$$NAMEFMT^XLFFNAME (.MYNAME, "F", "CS") → NSPROVIDER, JOHN K JR
$$$$NAMEFMT^XLFFNAME (.MYNAME, "F", "CSD") → NSPROVIDER, JOHN K JR PHD
$$$$NAMEFMT^XLFFNAME (.MYNAME, "F", "CDcXc") → NS ' PROVIDER, JOHN K.,
    JR, PHD
$$$$NAMEFMT^XLFFNAME (.MYNAME, "F", "CSL12") → NSPROVIDER, JOH K
$$$$NAMEFMT^XLFFNAME (.MYNAME, "F", "CMD") → Ns ' provider, John K. Jr
    PhD
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G") → JOHN K. NS ' PROVIDER JR
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G", "D") → JOHN K. NS ' PROVIDER JR PHD
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G", "Dc") → JOHN K. NS ' PROVIDER JR,
    PHD
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G", "P") → MR. JOHN K. NS ' PROVIDER JR
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G", "Xc") → JOHN K. NS ' PROVIDER, JR
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G", "PDcXc") → MR. JOHN K.
    NS ' PROVIDER, JR, PHD
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G", "PDcXcM") → Mr. John K.
    NS ' PROVIDER, Jr, PhD
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G", "S") → JOHN K NSPROVIDER JR
$$$$NAMEFMT^XLFFNAME (.MYNAME, "G", "SL12") → JOH K NSPROVIDER
```



```

$$NAMEFMT^XLFFNAME (.MYNAME, "O") → NS' PROVIDER
$$NAMEFMT^XLFFNAME (.MYNAME, "O", "S") → NSPROVIDER
$$NAMEFMT^XLFFNAME (.MYNAME, "O", "M") → Ns' Provider
$$NAMEFMT^XLFFNAME (.MYNAME, "O", "L3") → O'B

```

2. If an entry in the NAME COMPONENTS stores the components of a name stored in the NAME field (#.01) of record number 32 in the NEW PERSON file (#200), and the data in the corresponding record in the NAME COMPONENT file is:

```

FILE = 200
FIELD = .01
IENS = "32, "
PREFIX = "MR. "
GIVEN NAME = "JOHN"
MIDDLE NAME = "K. "
FAMILY NAME = "NS' PROVIDER"
SUFFIX = "JR"
DEGREE = "PHD"

```

you can set:

```

MYNAME ("FILE") = 200
MYNAME ("FIELD") = .01
MYNAME ("IENS") = "32, "

```

and call \$\$NAMEFMT^XLFFNAME as in Example 1, listed previously, to return the name in various formats.

## STDNAME^XLFNAME: Name Standardization Routine

This procedure parses a name and converts it into the following standard format:

Family\_name,Given\_name<space>Middle\_name<space>Suffix(es)

A name in standard format is entirely in uppercase, and contains no Arabic numerals. The Family\_name (last name) portion of a standard name appears to the left of the comma and contains no spaces and no punctuation except hyphens (-). The other parts of a standard name (the portion to the right of the comma) contain no punctuation except for hyphens and spaces. NMI and NMN are not used for the Middle\_name.

STDNAME^XLFNAME optionally returns in an array the component parts of the name. It also optionally returns information in an array about possible problems encountered during the conversion of the name to standard form and the parsing of the name into its component parts.

### Format:

STDNAME^XLFNAME ( .NAME, FLAGS, .AUDIT)

### Input Parameters:

<b>.NAME</b>	<p>(Required) NAME is the name to be converted to standard format. It is assumed that the name is in the general format:</p> <p style="text-align: center;">Family_name,Given_name(s) Middle_name Suffix(es)</p> <p>If the "F" flag is not used, and the name contains no comma, it is assumed the name is in the general format:</p> <p style="text-align: center;">Given_name(s) Middle_name Family_name Suffix(es)</p> <p>The standard form of the name is returned in the NAME variable. If the "C" flag is passed in, the components of the name are returned in nodes descendent from NAME. (See "<b>Output:</b>" just after this table.)</p>								
<b>FLAGS</b>	<p>Flags to control processing. Possible values are:.</p> <table border="1" style="width: 100%;"> <tr> <td data-bbox="462 1356 532 1419"><b>C</b></td> <td data-bbox="532 1356 1341 1419">Return name components in the NAME array. (See "<b>Output:</b>" just after this table.)</td> </tr> <tr> <td data-bbox="462 1419 532 1545"><b>F</b></td> <td data-bbox="532 1419 1341 1545">If the name passed in the NAME input parameter does not contain a comma, assume it is the Family Name only. For example, if the name input is "NS PROVIDER", return the name as "NSPROVIDER" instead of "JAMES,ST"</td> </tr> <tr> <td data-bbox="462 1545 532 1577"><b>G</b></td> <td data-bbox="532 1545 1341 1577">Don't return AUDIT("GIVEN") even if the Given Name is missing.</td> </tr> <tr> <td data-bbox="462 1577 532 1640"><b>P</b></td> <td data-bbox="532 1577 1341 1640">Remove text in parentheses (), brackets [], or braces {} from the name. If such text is actually removed, return AUDIT("STRIP").</td> </tr> </table>	<b>C</b>	Return name components in the NAME array. (See " <b>Output:</b> " just after this table.)	<b>F</b>	If the name passed in the NAME input parameter does not contain a comma, assume it is the Family Name only. For example, if the name input is "NS PROVIDER", return the name as "NSPROVIDER" instead of "JAMES,ST"	<b>G</b>	Don't return AUDIT("GIVEN") even if the Given Name is missing.	<b>P</b>	Remove text in parentheses (), brackets [], or braces {} from the name. If such text is actually removed, return AUDIT("STRIP").
<b>C</b>	Return name components in the NAME array. (See " <b>Output:</b> " just after this table.)								
<b>F</b>	If the name passed in the NAME input parameter does not contain a comma, assume it is the Family Name only. For example, if the name input is "NS PROVIDER", return the name as "NSPROVIDER" instead of "JAMES,ST"								
<b>G</b>	Don't return AUDIT("GIVEN") even if the Given Name is missing.								
<b>P</b>	Remove text in parentheses (), brackets [], or braces {} from the name. If such text is actually removed, return AUDIT("STRIP").								
<b>.AUDIT</b>	<p>If provided, this is an array that STDNAME^XLFNAME returns if there are any ambiguities or possible problems in standardizing the name or parsing the name into component parts. (See "<b>Output:</b>" just after this table.)</p>								

**Output:**

<p><b>NAME</b></p>	<p>NAME is set to the name that was input converted to standard format.</p> <p>If the FLAGS input parameter contains a "C", the component parts of the name are returned in the NAME array:</p> <p style="padding-left: 40px;">NAME("FAMILY") = Family (Last) Name  NAME("GIVEN") = Given (First) Name(s)  NAME("MIDDLE") = Middle Name  NAME("SUFFIX") = Suffix(es)</p>
<p><b>AUDIT</b></p>	<p>AUDIT is set to the original name that was passed in NAME.</p> <p>In addition, if there were any problems in the interpretation of the NAME being standardized, descendants of AUDIT are set:</p> <p style="padding-left: 40px;">AUDIT("subscript") = ""</p> <p>where "subscript" can be one of the following:</p> <p>AUDIT("FAMILY")  The Family Name starts with ST. (The period and space are removed from the Family Name. For example, the name "ST. JOHN" is converted to "STJOHN".)</p> <p>AUDIT("GIVEN")  Returned if there is no Given Name and the "G" flag isn't passed in.</p> <p>AUDIT("MIDDLE")  Returned if there are three or more names between the first comma and the Suffix(es). (All name parts except the last are assumed to be part of the Given Name. Only the last part is assumed to be the Middle Name.)</p> <p>AUDIT("NM")  Returned if NMI or NMN appears to be used as the Middle Name. (NMI and NMN are removed from the standard name, and the Middle Name component is returned as null.)</p> <p>AUDIT("NOTE")  Returned if the name appears to contain a note or flag that may not actually be part of the name. For example, the name starts with "C-" or "EEE," or has "FEE" at the end.</p> <p>AUDIT("NUMBER")  Returned if a name part (other than a valid numeric Suffix) contains a number.</p> <p>AUDIT("PERIOD")  Returned if periods were removed.</p> <p>AUDIT("PUNC")  Returned if punctuation was removed.</p> <p>AUDIT("SPACE")  Returned if spaces were removed from the Family Name.</p>

	<p>AUDIT("STRIP")  Returned if text in parentheses (), brackets [], or braces {} were removed from the Name. (This is done only if the "P" flag is passed.)</p> <p>AUDIT("SUFFIX")  Returned if:</p> <ul style="list-style-type: none"> <li>• Suffix(es) are found immediately to the left of the 1<sup>st</sup> comma.</li> <li>• I, V, or X, and nothing else except valid suffixes, appear immediately after the Given Name. (It is interpreted as the Middle Name.)</li> <li>• The name immediately after the Given Name appears to be a non-numeric suffix (except I, V, and X), and everything after that also appear to be suffixes. (It is assumed there are a Given Name and Suffix(es), but no Middle Name.)</li> <li>• M.D. or M D is found at the end of the name, or before any valid suffixes at the end of the name. (It is assumed that M and D are initials in the Given or Middle Name rather than a Suffix.)</li> <li>• The name part before any recognizable suffixes is more than one character in length and doesn't contain any vowels or Y. It is interpreted as a suffix.</li> <li>• A Suffix is found between commas immediately after the Family Name.</li> </ul>
--	--

**Details:**

In forming the standard name, the following changes are made:

1. The name is converted to uppercase.
2. In the Family Name:
  - a. Semicolons (;) and colons (:) are converted to hyphens (-).
  - b. Spaces and all other punctuation except hyphens are removed.
3. In the other name parts (Given Name, Middle Name, and Suffix).
  - a. Semicolon, colons, commas (,), and periods (.) are converted to spaces.
  - b. All punctuation except hyphens and spaces are removed.
4. Hyphens and spaces at the beginning and end of the name are removed.
5. Two or more consecutive hyphens/spaces are replaced with a single hyphen/space.
6. Any suffixes immediate preceding the comma are moved to the end.
7. The suffixes indicating birth positions 1st, 2nd, 3rd, ..., 10th are converted to their Roman numeral equivalents I, II, III, ... X.
8. DR immediately after the comma (or if there is no comma, at the beginning of the name), is assumed to be a suffix and moved to the end of the name.
9. Any suffixes between two commas immediate after the Family Name are moved to the end of the name.
10. NMI or NMN used as a Middle Name is deleted.

In forming the component parts of the name, only the following changes are made:

1. The name component is converted to uppercase.
2. In the Family Name, semicolons (;) and colons (:) are converted to hyphens (-).
3. In the other name parts (Given Name, Middle Name, and Suffix), semicolons, colons, and commas (,) are converted to spaces.
4. Hyphens and spaces at the beginning and end of the name are removed.
5. Two or more consecutive hyphens/spaces are replaced with a single hyphen/space.
6. A Middle Name of NMI or NMN is changed to null.
7. Spaces after periods are removed.
8. Accent graves (`) and up-arrows (^) are removed.

In parsing the name into its component parts, if the name contains a comma or the "F" flag is passed, STDNAME^XLFFNAME looks for suffixes immediately to the left of the first comma, and at the very end of the name. The suffixes it recognizes are 1ST through 10TH, JR, SR, DR, MD, ESQ, DDS, RN and Roman numerals I through X. If a name part before any recognizable suffixes is more than one character in length, and contains no vowel or 'Y', it is also assumed to be a suffix. The Name Standardization looks for the DR suffix immediately after the first comma, and for any suffix between two commas immediately after the Family Name. The portion of the name to the left of the comma, less any suffixes, is assumed to be the Family Name.

After STDNAME^XLFFNAME accounts for all Suffixes, it looks at the portion of the name after the comma. It assumes that the first space-delimited piece is the Given Name. If any other pieces are left, the last one (rightmost) is assumed to be the Middle Name, and anything else is appended to the end of the Given Name.

If the name contains no comma, and the "F" flag is not passed, STDNAME^XLFFNAME looks for suffixes at the very end of the name. The last space-delimited piece before any suffixes is assumed to be the Family Name. The first space-delimited piece is assumed to be the Given Name. If any other pieces are left, the last one (rightmost) is assumed to be the Middle Name, and anything else is appended to the end of the Given Name.

### Example:

In this example, the variable MYNAME is set to the name to be standardized. The "C" flag indicates that the name components should be returned in the MYNAME array, and the "P" flag indicates that parenthetical text should be removed from the name. STDNAME^XLFFNAME sets MYAUD to original name passed in and sets nodes in the MYAUD array to flag changes and possible problems.

```
>S MYNAME="NS PROVIDER, JOHN A. B. 2ND (TEST) "
>D STDNAME^XLFFNAME (.MYNAME, "CP", .MYAUD)

>ZW MYNAME
MYNAME=NSPROVIDER, JOHN A B II
MYNAME ("FAMILY")=NS PROVIDER
MYNAME ("GIVEN")=JOHN A.
MYNAME ("MIDDLE")=B.
MYNAME ("SUFFIX")=2ND
```

```
>ZW MYAUD  
MYAUD=NS PROVIDER,JOHN A. B. 2ND (TEST)  
MYAUD("MIDDLE")=""  
MYAUD("PERIOD")=""  
MYAUD("SPACE")=""  
MYAUD("STRIP")=""
```

STDNAME^XLFNAME returned the standard form of the name in MYNAME as NSPROVIDER,JOHN A B II. It interpreted JOHN A. as the given (first) name and B. as the middle name. Since this may not be correct, MYAUD("MIDDLE") is set. Periods were removed and spaces were removed to form the standard name, therefore MYAUD("PERIOD") and MYAUD("SPACE") were set. Finally, since the parenthetical text (TEST) was removed, MYAUD("STRIP") was set.

## New Controlled Subscription Integration Agreements

These are the Kernel Controlled Subscription Integration Agreements for Name Standardization. They contain attributes/functions that must be controlled in their use. Permission to use them is granted by the custodian package (i.e., Kernel) on a one-by-one basis.

### DELCOMP^XLFNNAME2: Delete Name Components Entry

This procedure deletes an entry in the NAME COMPONENTS file, and optionally, the value of the pointer in the source file that points to the name components entry.

(This call is designed to be used in the KILL logic for the MUMPS cross-reference mentioned previously in UPDCOMP^XLFNNAME2.)

#### Format:

```
DELCOMP^XLFNNAME2 (FILE, [ . ]RECORD, FIELD, PTRFIELD)
```

#### Input Parameters:

<b>FILE</b>	(Required) The number of the file or subfile (the "source file") that contains the name.
<b>RECORD</b>	(Required) The IENS or the internal entry number array (that looks like the DA array) of the record in the source file that contains the name.
<b>FIELD</b>	(Required) The number of the field in the source file that contains the name.
<b>PTRFIELD</b>	The number of the pointer field in the source file that points to the NAME COMPONENTS file. Only if this parameter is passed will the value of this pointer field be deleted.

#### Example:

Suppose that you have a NAME COMPONENTS file entry that contains the components of a name stored in File #1000, Record #132, Field #.01. Pointer field #1.1 of that File #1000 is a pointer to the NAME COMPONENTS file. To delete the entry in the NAME COMPONENTS file, and the value of the pointer field, you can do the following:

```
>D DELCOMP^XLFNNAME(1000,132,.01,1.1)
```

## UPDCOMP^XLFFNAME2: Update Name Components Entry

This procedure updates an entry in the NAME COMPONENTS file. Optionally, the pointer in the source file that points to the name components entry is also updated.

This API is designed to be used in the SET logic of a MUMPS cross-reference on the name field in a source file, to keep the name field and the associated name components in sync. For an example of its use, see the ANAME index in the INDEX file (#.11). The ANAME index is a MUMPS cross-reference on the .01 NAME field of the NEW PERSON file. If an entry's NAME field is edited, the ANAME cross-reference updates the associated entry in the NAME COMPONENTS file. Note that existing MUMPS cross-references on the NAME COMPONENTS file already exist to update the associated name field on the source file if the components are edited.

### Format:

UPDCOMP^XLFFNAME2 (FILE, [.]RECORD, FIELD, [.]NAME, PTRFIELD, PTRVAL)

### Input Parameters:

<b>FILE</b>	(Required) The number of the file or subfile (the "source file") that contains the name.
<b>[.]RECORD</b>	(Required) The IENS or the internal entry number array (that looks like the DA array) of the record in the source file that contains the name.
<b>FIELD</b>	(Required) The number of the field in the source file that contains the name.
<b>[.]NAME</b>	<p>(Required) An array that contains the component parts of the name to store in the NAME COMPONENTS file entry:</p> <p style="padding-left: 40px;">NAME("FAMILY") = Family Name (required)  NAME("GIVEN") = Given Name(s) (optional)  NAME("MIDDLE") = Middle Name(s) (optional)  NAME("SUFFIX") = Suffix(es) (optional)  NAME("PREFIX") = Prefix (optional)  NAME("NOTES") = optional free text string</p> <p>Alternatively, a name in standard format can be passed in the NAME input parameter. If the NAME input parameter has no descendents (that is, \$D(NAME)=1), UPDCOMP^XLFFNAME2 will make a call to NAMECOMP^XLFFNAME to build the NAME array for you.</p>
<b>PTRFIELD</b>	The number of the pointer field in the source file that points to the NAME COMPONENTS file. Only if this parameter is passed will the value of this pointer field be updated with the entry number of the record in the NAME COMPONENTS file that was added or edited.
<b>PTRVAL</b>	The current value of the pointer field specified by the PTRFIELD input parameter. This parameter can be used to save processing time. If both PTRFIELD and PTRVAL are passed, the pointer field will be updated only if this value is different from the entry number of the record in the NAME COMPONENTS file that was added or edited.

### Example:

Suppose the .01 field of File #1000 contains a person's name, and the component parts of the name in entry 132 should be updated as follows:



Family (last) name: NS'PROVIDER  
 Given (first) name: JOHN HENRY  
 Middle name: A.  
 Suffix: JR.

Field #1.1 is defined as a pointer to the NAME COMPONENTS file (#20) and has a value of 42, the IEN of a record in the NAME COMPONENTS file. To update the NAME COMPONENTS file with this name, you can do the following:

```
>S MYNAME ("FAMILY")="NS ' PROVIDER"
>S MYNAME ("GIVEN")="JOHN HENRY"
>S MYNAME ("MIDDLE")="A. "
>S MYNAME ("SUFFIX")="JR. "

>D UPDCOMP^XLFNNAME2 (1000,132,.01,.MYNAME,1.1,42)
```

If there is an entry in the NAME COMPONENTS file that corresponds to File #1000, Field #.01, IEN #132, that entry is updated with the name components passed in the MYNAME array. Otherwise a new entry is added to the name components with this information.

If the entry in the name components that was updated or added is record #42, no change is made to the value of the pointer field #1.1, since 42 was passed in the 6th parameter.

MUMPS cross-references on the NAME COMPONENTS file updates the name in the Field #.01 of File #1000 to "NSPROVIDER,JOHN HENRY A JR" if it doesn't already contain that name.

## Modified Kernel API

### \$\$ADD^XUSERNEW: Add New Users

Patch XU\*8.0\*134 modifies this existing Kernel extrinsic function, which can be used to add new entries to the NEW PERSON file. After prompting for the user name, \$\$ADD^XUSERNEW parses the input into its component parts, and then prompts for each name component separately, presenting the parsed input as defaults.

<b>Usage</b>	S X=\$\$ADD^XUSERNEW([dr_string][,keys])	
<b>Input</b>	dr_string:	[optional] Additional fields to ask when adding the new user, in the format for a DR string as used in a standard DIC call (see the VA FileMan documentation for information about DIC).
	keys:	[optional] A comma-delimited string of keys to assign to the newly created user.
<b>Output</b>	return value:	Return value is in a similar format to the value of Y returned from a standard DIC call (see the VA FileMan documentation for information about DIC):
	return value=-1	User neither existed nor could be added.
	return value=N^S	User already exists in the file; N is the internal number of the entry in the file, and S is the value of the .01 field for that entry.
	return value=N^S^1	N and S are defined as above, and the 1 indicates the user has just been added to the file.

Use this extrinsic function to add new entries to the NEW PERSON file. It prompts for the user's name, parses the name that is entered into its component parts, and prompts for each name component separately, presenting the parsed input as defaults. It then prompts for the default identifiers for the NEW PERSON entry, which are Initials (field 1), Sex (field 4), and SSN (field 9). The default identifiers can be locally modified by modifying the New Person Identifiers field in the KERNEL SYSTEM PARAMETERS file.

To prompt for additional fields during this call, you pass a DR string containing the field to prompt for as a parameter to this function. If the person adding the entry up-arrows out before filling in all the identifiers and requested fields, the entry will be removed from the NEW PERSON file, and -1 will be returned.

**Examples**

To add a new user, asking default fields for new entry:

```
>S X=$$ADD^XUSERNEW
```

To add a new user, specifying a key to add:

```
>S X=$$ADD^XUSERNEW ("", "PROVIDER")
```

To add a new user, specifying additional fields to ask, plus two keys to add:

```
>S X=$$ADD^XUSERNEW ("5;13;53", "PSMGR, PSNARC")
```

# Technical Manual Information

This is the Technical Manual section of this supplemental documentation for Patch XU\*8.0\*134. It will be incorporated into the Kernel Technical Manual, Version 8.0 at a later date.

The intended audience for this chapter is Information Resource Management (IRM) and Veterans Affairs Medical Center (VAMC) personnel who will be doing the changes to the system. However, it can also be helpful to others in Technical Service, the Program Office, National *VISTA* Support (NVS), and Technical Integration.

## Implementation and Maintenance

The Name Standardization patch (Patch XU\*8.0\*134) is a Kernel Installation and Distribution System (KIDS) software release.

## Package Requirements

Patch XU\*8.0\*134 requires a standard **VISTA** operating environment in order to function correctly. Check your **VISTA** environment for packages and versions installed.

The minimum **VISTA** packages and patches that are required are listed as follows by:

1. **VISTA** package and current version number,
2. Associated patch designation(s) that need(s) to be installed in addition to the **VISTA** package, and
3. Brief description of the associated patch.

<b>VISTA Package and Version</b>	<b>Associated Patch Designation(s)</b>	<b>Brief Patch Description</b>
New Person, 3/6/16/20 Killer Patch	A4A7*1.01*11	Remove the DD and data for Files #3, #6, #16, and #20, and the x-refs in File #200 that set data into them.
VA FileMan, V. 22.0	DI*22.0*1	Post Verification Fixes.
" " "	DI*22.0*4	Misc. DIC Fixes
" " "	DI*22.0*12	New API to delete an x-ref.
" " "	DI*22.0*16	Allocation errors in lookup (DIC) call
" " "	DI*22.0*17	Miscellaneous bug fixes in DIC lookup.
" " "	DI*22.0*20	VP screens, long .01, dic(p)
" " "	DI*22.0*21	Editing screened LAYGO pointers.
" " "	DI*22.0*29	Undef on lookup to File #200, user enters spaces.
Kernel, V. 8.0	XU*8.0*135	New RAI/MDS file and new Triggers in File #200
" "	XU*8.0*137	KIDs fixes for FM & HL7, MM X-ref, STATE File.

Figure 7: Minimum **VISTA** packages and patches required

## Routines

The following routines are included in this patch. The second line of these routines looks like:

```
<tab>;;8.0;KERNEL;**[patch list]**;Jul 10, 1995
```

```
XLFNAME  
XLFNAME1  
XLFNAME2  
XLFNAME3  
XLFNAME4
```

```
XLFNAME5  
XLFNAME6  
XUSERNEW
```

# File List

This chapter lists the two files sent with Patch XU\*8.0\*134.

VISTA File and File Number	Global Location	Data w/ File?	Description
NAME COMPONENTS File #20	^VA(20,	No	<p>This file, introduced with Name Standardization (Patch XU*8.0*134), holds the component parts of a person's name as follows:</p> <ul style="list-style-type: none"> <li>• FAMILY (LAST) NAME field (#1)</li> <li>• GIVEN (FIRST) NAME field (#2)</li> <li>• MIDDLE NAME field (#3)</li> <li>• PREFIX field (#4)</li> <li>• SUFFIX field (#5)</li> <li>• DEGREE field (#6)</li> </ul> <p>The "source name" that has these components is identified by the following three fields:</p> <ul style="list-style-type: none"> <li>• FILE (field #.01)</li> <li>• FIELD (field #.02)</li> <li>• IENS (field #.03)</li> </ul> <p>The "ANAME" cross-reference on the FAMILY (LAST) NAME, GIVEN (FIRST) NAME, MIDDLE NAME, and SUFFIX field's keeps each component in synchronization with the corresponding source name. In the case of Patch XU*8.0*134, the source name is the .01 field (NAME field) of the NEW PERSON file (#200).</p> <p>The DEGREE and PREFIX fields are not considered part of a standard name, but can be used to build formatted names for display.</p>
NEW PERSON File #200	^VA(200,	No	<p>The full DD for the NEW PERSON file (#200) is NOT being exported with patch XU*8.0*134. However, this patch affects the following fields:</p> <ul style="list-style-type: none"> <li>• The new NAME COMPONENTS pointer field (#200,10.1) is added,</li> <li>• The definition of the NAME field (#200,.01) is modified, and</li> <li>• The definition of the DEGREE field (200,10.6) is modified.</li> </ul> <p>The new field and field definitions are transported in the KIDS transport global and installed at the site.</p>

Figure 8: Exported field definitions and file

## VA FileMan Input Templates

Following is a list of the modified VA FileMan templates exported with Patch XU\*8.0\*134.

Input Template	Option and Menu Text	File and Number	Description
XUNEW USER	[XUSERNEW] Add a New User to the System	NEW PERSON File #200	<p>The XUNEW USER INPUT template used by the option has been modified to allow the user to edit the individual name components stored in the NAME COMPONENTS file (#20):</p> <ul style="list-style-type: none"> <li>• FAMILY (LAST) NAME field (#1)</li> <li>• GIVEN (FIRST) NAME field (#2)</li> <li>• MIDDLE NAME field (#3)</li> <li>• PREFIX field (#4)</li> <li>• SUFFIX field (#5)</li> <li>• DEGREE field (#6)</li> </ul>
XUEXISTING USER	[XUSEREDIT] Edit an Existing User	NEW PERSON File #200	<p>The XUEXISTING USER INPUT template has been modified to allow the user to edit the individual name components stored in the NAME COMPONENTS file:</p> <ul style="list-style-type: none"> <li>• FAMILY (LAST) NAME field (#1)</li> <li>• GIVEN (FIRST) NAME field (#2)</li> <li>• MIDDLE NAME field (#3)</li> <li>• PREFIX field (#4)</li> <li>• SUFFIX field (#5)</li> <li>• DEGREE field (#6)</li> </ul>
XUREACT USER	[XUSERREACT] Reactivate a User	NEW PERSON File #200	<p>The XUREACT USER INPUT template has been modified to allow the user to edit the individual name components stored in the NAME COMPONENTS file:</p> <ul style="list-style-type: none"> <li>• FAMILY (LAST) NAME field (#1)</li> <li>• GIVEN (FIRST) NAME field (#2)</li> <li>• MIDDLE NAME field (#3)</li> <li>• PREFIX field (#4)</li> <li>• SUFFIX field (#5)</li> <li>• DEGREE field (#6)</li> </ul>

Figure 9: Modified INPUT templates exported with this patch

## ScreenMan Forms

Following is a list of the modified ScreenMan forms exported with Patch XU\*8.0\*134.

ScreenMan Form	Option and Menu Text	File and Number	Description
XUNEW USER	[XUSERNEW] Add a New User to the System	NEW PERSON File #200	The XUNEW USER ScreenMan form used by the option has been modified to allow the user to edit the individual name components stored in the NAME COMPONENTS file: <ul style="list-style-type: none"> <li>• FAMILY (LAST) NAME field (#1)</li> <li>• GIVEN (FIRST) NAME field (#2)</li> <li>• MIDDLE NAME field (#3)</li> <li>• PREFIX field (#4)</li> <li>• SUFFIX field (#5).</li> </ul>
XUEXISTING USER	[XUSEREDIT] Edit an Existing User	NEW PERSON File #200	The XUEXISTING USER ScreenMan form has been modified to allow users to edit the individual name components stored in the NAME COMPONENTS file: <ul style="list-style-type: none"> <li>• FAMILY (LAST) NAME field (#1)</li> <li>• GIVEN (FIRST) NAME field (#2)</li> <li>• MIDDLE NAME field (#3)</li> <li>• PREFIX field (#4)</li> <li>• SUFFIX field (#5)</li> </ul>
XUREACT USER	[XUSERREACT] Reactivate a User	NEW PERSON File #200	The XUREACT USER ScreenMan form has been modified to allow users to edit the individual name components stored in the NAME COMPONENTS file: <ul style="list-style-type: none"> <li>• FAMILY (LAST) NAME field (#1)</li> <li>• GIVEN (FIRST) NAME field (#2)</li> <li>• MIDDLE NAME field (#3)</li> <li>• PREFIX field (#4)</li> <li>• SUFFIX field (#5)</li> </ul>

Figure 10: Modified ScreenMan forms exported with this patch



## Kernel Options Affected by the Patch

This chapter describes the Kernel options that are affected by Patch XU\*8.0\*134. These options are NOT exported with Patch XU\*8.0\*134. However, the INPUT template and ScreenMan forms associated with them have been modified, and they are exported with the patch. Figure 11 lists the Kernel options on the right with the modified ScreenMan forms and INPUT templates on the right:

Kernel Option and Menu Text	ScreenMan Form	Input Template
XUSERNEW Add a New User to the System	XUNEW USER	XUNEW USER
XUSEREDIT Edit an Existing User	XUEXISTING USER	XUEXISTING USER
XUSERREACT Reactivate a User	XUREACT USER	XUREACT USER

Figure 11: Kernel options with associated ScreenMan forms and INPUT templates

### INPUT Templates

The INPUT templates used by these Kernel options have been modified to allow you the capability to individually edit the following name components stored in the NAME COMPONENTS file (#20):

- FAMILY (LAST) NAME field (#1),
- GIVEN (FIRST) NAME field (#2),
- MIDDLE NAME field (#3),
- PREFIX field (#4),
- SUFFIX field (#5), and
- DEGREE field (#6).

### ScreenMan Forms

There is a slight difference between the way the DEGREE field (#6) is populated in the ScreenMan Forms versus the INPUT Templates. While the INPUT templates allow you the capability of editing the DEGREE field directly in the NAME COMPONENTS file, the ScreenMan forms already contain the DEGREE field (#10.6) from the NEW PERSON file (#200). So, the DEGREE field (#6) from the NAME COMPONENTS file has not been added to the ScreenMan forms. The "ADEG" cross-reference on the DEGREE field in the NEW PERSON file automatically updates the DEGREE field in the NAME COMPONENTS file.

Hence, the ScreenMan forms used by these Kernel options have been modified to allow you the capability to individually edit the following name components directly in a "pop-up" window, which are stored in the NAME COMPONENTS file (#20):

- FAMILY (LAST) NAME field (#1),

- GIVEN (FIRST) NAME field (#2),
- MIDDLE NAME field (#3),
- PREFIX field (#4),
- SUFFIX field (#5), and

MUMPS code on the ScreenMan form and MUMPS cross references in the NEW PERSON file (#200) and the NAME COMPONENTS file will keep Fields #1, #2, #3, and #5 in the NAME COMPONENTS file synchronized with the NAME field in the NEW PERSON file.

## Kernel Options and Description of Changes

The affected options are located on the Kernel User Management menu as shown in Figure 12:

SYSTEMS MANAGER MENU ...	[EVE]
User Management ...	[XUSER]
Add a New User to the System	[XUSERNEW]
Edit an Existing User	[XUSEREDIT]
Reactivate a User	[XUSERREACT]

Figure 12: Kernel options affected by Name Standardization

### Add a New User to the System

[XUSERNEW]

A change has been made to the way entries are initially added to the NEW PERSON file in the Kernel option Add a New User to the System (XUSERNEW). This option invokes the ScreenMan form XUNEW USER. If for some reason the XUNEW USER form cannot be invoked (because the terminal type cannot handle screen-oriented applications, for example), the XUSERNEW option invokes the XUNEW USER Input template. Both the XUNEW USER form and INPUT template have been modified to allow you to edit the component parts of a person's name.

After the .01 field (NAME field) has been entered, and after the ScreenMan form for editing the properties of the new person is invoked, the individual name components will be displayed in a window so that they can be edited, if needed.

### Edit an Existing User

[XUSEREDIT]

This option invokes the ScreenMan form XUEXISTING USER. If for some reason the XUEXISTING USER form cannot be invoked (because the terminal type cannot handle screen-oriented applications, for example), the XUSEREDIT option invokes the XUEXISTING USER Input template. Both the XUEXISTING USER form and INPUT template have been modified to allow you to edit the component parts of a person's name.

On the first page of the ScreenMan form, the first field is 'Name...'. If you edit the name displayed there, or simply press <RET> there, a window that displays the name components will open. There you can edit the component parts of the name, fields in the NAME COMPONENTS file, separately. From the

components, the standard form of the name is displayed in the lower left side of the window. It is this standard name that is stored in the .01 field of the NEW PERSON file. Though Prefix can be entered and edited, it is never included as part of the standard name.

## Reactivate a User

[XUSERREACT]

This option invokes the ScreenMan form XUREACT USER. If for some reason the XUREACT USER form cannot be invoked (because the terminal type cannot handle screen-oriented applications, for example), the XUSERREACT option invokes the XUREACT USER Input template. Both the XUREACT USER form and INPUT template have been modified to allow you to edit the component parts of a person's name.

The changes to the XUREACT USER form are exactly the same as those to the XUEXISTING USER form described previously.

## New VA FileMan FUNCTION: XLFMTNAME – Name Formatting Function

As part of the installation of Patch XU\*8.0\*134, a new FUNCTION will be installed in the FileMan FUNCTION file (#.5). It is called: XLFMTNAME. This function can be used within the FileMan Print options, or within word-processing windows used to generate forms. The new FUNCTION allows the user to display names in various formats from any *VISTA* file containing a name field.

(For more information about this new VA FileMan FUNCTION: XLFMTNAME, see "VA FileMan Name Formatting Function: XLFMTNAME" in the "User Manual Information" section of this documentation.)

## Menu Diagrams

This is a menu diagram of the Kernel options affected by Patch XU\*8.0\*134. These options are NOT exported with Patch XU\*8.0\*134. However, the INPUT template and ScreenMan forms associated with them have been modified, and they are exported with this patch. (The Kernel options that have NOT been affected by Patch XU\*8.0\*134 are NOT shown in this menu diagram.)

```

User Management (XUSER)
|
|
----- Add a New User to the System
| [XUSERNEW]
|
----- Edit an Existing User
| [XUSEREDIT]
| **ENTRY ACTION:
| S
| DIC=" ^VA(200, ", DIC(0)="AEMQ", D
| IC("S")="I
| $S($P(^ (0),U,11):$P(^ (0),U,11)
| '<DT,1:1)" D ^DIC K DIC Q:Y=-1
| S DA=+Y,DR="[XUEXISTING
| USER]",DIE=" ^VA(200, " D
| XUDIE^XUS5 K D0,DA,DIE,DR
|
----- Reactivate a User [XUSERREACT]

```

## Archiving and Purging

There are no application-specific archiving procedures or recommendations for the Name Standardization patch (Patch XU\*8.0\*134).

# Callable Routines

This chapter lists all the Kernel APIs exported with the Name Standardization patch.

Every callable entry point is described in the "Programmer Manual Information" section of this documentation. Refer to the indicated chapter under "Where to find More Information" in Figure 13 for more details about the calls in this documentation.

## Alphabetized by Entry Point

Entry Point	Brief Description and IA Reference	Where to find More Information
\$\$BLDNAME^XLFNAME	Build Name from Component Parts IA #3065	See the "APIs" chapter of this documentation, under "Supported References."
\$\$CLEANC^XLFNAME	Name Component Standardization Routine IA #3065	See the "APIs" chapter of this documentation, under "Supported References."
\$\$FMNAME^XLFNAME	Convert HL7 Formatted Name to Name IA #3065	See the "APIs" chapter of this documentation, under "Supported References."
\$\$HLNAME^XLFNAME	Convert Name to HL7 Formatted Name IA #3065	See the "APIs" chapter of this documentation, under "Supported References."
NAMECOMP^XLFNAME	Component Parts from Standard Name IA #3065	See the "APIs" chapter of this documentation, under "Supported References."
\$\$NAMEFMT^XLFNAME	Name Formatting for display, HL7 IA #3065	See the "APIs" chapter of this documentation, under "Supported References."
STDNAME^XLFNAME	Name Standardization Routine IA #3065	See the "APIs" chapter of this documentation, under "Supported References."
DELCOMP^XLFNAME2	Delete Name Components Entry IA #3066	See the "APIs" chapter of this documentation, under "Controlled Subscriptions."
UPDCOMP^XLFNAME2	Update Name Components Entry IA #3066	See the "APIs" chapter of this documentation, under "Controlled Subscriptions."
\$\$ADD^XUSERNEW	Add New Users IA #10053	See the "Sign-On/Security: User Interface" chapter of the Kernel Systems Manual, Version 8.0, and the "APIs" chapter of this manual under "Modified Kernel API."

Figure 13: Application Programmer Interfaces (APIs) exported with the Name Standardization patch

## External Interfaces (HL7 Components)

### Current VISTA HL7 APIs for Person Name Conversion

The following HL7 APIs are widely used throughout VISTA for person name conversion. Two new APIs in the Kernel namespace providing a more accurate representation of standard name components are introduced with Patch XU\*8.0\*134. These new Kernel APIs will more precisely reflect the name components resulting in a more accurate representation of a person's name.

As defined in Health Level Seven (HL7) Standards, Version 2.3 (HL7 reference 2.8.28), HL7 currently transmits person names in the following format. The Data Type of PN for Person Name is defined as follows:

<family name (ST)>^<given name (ST)>^<middle initial or name (ST)>^<suffix (e.g., JR or III) (ST)>^<prefix (e.g., DR) (ST)>^<degree (e.g., MD) (ST)>^<name type code (ID)>.

The VISTA Health Level Seven (HL7) package currently supports two APIs:

1. An API to convert a VISTA person name into the HL7 2.3 format (see the entry point \$\$HLNAME^HLFNC in Figure 14).
2. An API to convert a person name from an HL7 2.3 format into a DHCP (or VISTA) format (see the \$\$FMNAME^HLFNC entry point in the next figure).

These HL7 APIs are listed, respectively by entry point and description:

Entry Point	Description
\$\$HLNAME^HLFNC	Converts a name in the format typically used in VISTA (e.g. lastname,firstname) to a name in HL7 format. The variable X is the input variable equal to the DHCP name.
\$\$FMNAME^HLFNC	Converts a name in HL7 format to a name in DHCP format.

Figure 14: VISTA HL7 APIs for converting person names

(For more information about these APIs see the "API Reference" section of the "VISTA HL7 Site Manager & Developer Manual" found here: <http://vista.med.va.gov/openvista> .)

### New Kernel APIs for Person Name Conversion

The previous two HL7 APIs for person name conversion were used as models for two new APIs, written in the Kernel XLF Namespace for Person Name Conversion. They are listed as follows:

1. An API has been written to support converting the NAME COMPONENT fields 1 through 6 into HL7 2.3 format.
2. An API has been written to support converting the HL7 2.3 format to the individual NAME COMPONENT fields 1 through 6 in an output array.

These new Kernel APIs provide a more precise representation of a person's name.

Entry Point	Description
\$\$HLNAME^XLFNAME	Convert NAME COMPONENT fields 1 through 6 to HL7 Formatted Name.
\$\$FMNAME^XLFNAME	Convert HL7 Formatted Name to the individual NAME COMPONENT fields 1 through 6 in an output array.

Figure 15: New Kernel APIs for converting person names

(For more information about these new APIs see the "APIs" chapter in the "Programmer Manual Information" section of this documentation.)

## External Relations

### Package Requirements

Patch XU\*8.0\*134 requires a standard **VISTA** operating environment in order to function correctly. Check your **VISTA** environment for packages and versions installed.

(For more information on the minimum **VISTA** packages and patches that are required by this patch see the "Implementation and Maintenance portion of the "Technical Manual Information" section of this document.)

### Dependencies

A common set of APIs has been developed for standardizing and retrieving person names to assist **VISTA** applications in standardizing person names in their databases.

This project required the coordination between the Information Infrastructure and Patient Management Systems divisions of Technical Services. A common set of APIs has been developed which can be used for standardizing and retrieving person names. To support this effort, each group applied the necessary changes to files under their applications' domain (i.e., the NEW PERSON file [#200], and the PATIENT file [#2], respectively). Name Standardization (Patch XU\*8.0\*134) is the Information Infrastructure patch to standardized names (.01 field) in the NEW PERSON file, a critical step in preparing for National Provider Identification.

### Integration Agreements (IA)

Patch XU\*8.0\*134 is a party in the Integration Agreements (IAs) described on the following pages.

#### Name Standardization Supported References Alphabetized by Entry Point

Entry Point or Global Root	IA #	Type	Name
\$\$ADD^XUSERNEW	#10053	Routine	Add New Person File Entry (in Sign-on/ Security chapter)
\$\$BLDNAME^XLFNAME	#3065	Routine	Build Name from Component Parts
\$\$CLEANC^XLFNAME	#3065	Routine	Name Component Standardization Routine
\$\$FMNAME^XLFNAME	#3065	Routine	Convert HL7 Formatted Name to Name
\$\$HLNAME^XLFNAME	#3065	Routine	Convert Name to HL7 Formatted Name
NAMECOMP^XLFNAME	#3065	Routine	Component Parts from Standard Name
\$\$NAMEFMT^XLFNAME	#3065	Routine	Formatted Name from Name Components
STDNAME^XLFNAME	#3065	Routine	Name Standardization Routine
^VA(200,	#10060	File	NEW PERSON FILE

Figure 16: Patch XU\*8.0\*134 Supported Reference Integration Agreements



**Controlled IAs Where Kernel Is Custodian**

Entry Point or Global Root	IA #	Type	Name
UPDCOMP^XLFFNAME2	#3066	Routine	Update Name Components Entry
DELCOMP^XLFFNAME2	#3066	Routine	Delete Name Components Entry
^VA(20,	#3041	File	Access to File #20 Name Components

Figure 17: Patch XU\*8.0\*134 Controlled Subscription Integration Agreements

## Internal Relations

There are no exported options with this patch. However, the INPUT template and ScreenMan forms associated with the following Kernel options have been modified and they are exported with this release:

- Add a New User to the System [XUNEW USER]
- Edit an Existing User [XUSEREDIT]
- Reactivate a User [XUSERREACT]

(For more information on the modified ScreenMan forms and INPUT templates exported with Patch XU\*8.0\*134, see the chapter "Kernel Options Affected by the Patch" in the "Technical Manual Information" section of this document.)

(For the minimum *VISTA* packages and patches required before installing Patch XU\*8.0\*134, see the chapter "Implementation and Maintenance" in the "Technical Manual Information" section of this document..)

## Namespace

The Name Standardization patch uses the namespace **XLF**, which is part of the Kernel namespace.

## File Numbers

Patch XU\*8.0\*134 file numbers and global locations are listed as follows:

File #	Global
20	^VA(20,
200	^VA(200,



The full Data Dictionary for the NEW PERSON file (#200) is NOT being exported with patch XU\*8.0\*134. However, this patch affects the following fields:

- The new NAME COMPONENTS pointer field (#200,10.1) has been added.
- The definition of the NAME field (#200,.01) has been modified.
- The definition of the DEGREE field (200,10.6) has also been modified.

The new field and changed field definitions will be transported in the KIDS transport global and installed at the site.

## Package-wide Variables

Patch XU\*8.0\*134 contains no package-wide variables.

## Software Product Security

There are no special security issues associated with Patch XU\*8.0\*134.

### Mail Groups

There are no mail groups exported with Patch XU\*8.0\*134.

### Remote Systems

There are no remote systems involved with Patch XU\*8.0\*134

### Archiving/Purging

There are no package-specific archiving procedures or recommendations for Patch XU\*8.0\*134.

### Interfacing

There are no specialized (not VA produced) products (hardware and/or software) embedded within or required by Patch XU\*8.0\*134.

### Electronic Signatures

There are no electronic signatures used in Patch XU\*8.0\*134.

### Menus

There are no options of particular interest to Information Security Officers (ISOs) in Patch XU\*8.0\*134.

### Security Keys

There are no security keys exported with Patch XU\*8.0\*134.

### File Security

File #	File Name	DD	RD	WR	DEL	LAYGO	AUDIT
200	NEW PERSON	@	#	#	#	#	

File #	File Name	DD	RD	WR	DEL	LAYGO	AUDIT
20	NAME COMPONENTS	@	#	#	#	#	

Figure 18: File Security for NAME COMPONENTS (file #200) and NEW PERSON (file #200)

# Glossary

CALLABLE ENTRY POINT	Authorized programmer call that may be used in any <b>VISTA</b> application package. The DBA maintains the list of DBIC-approved entry points.
CONTROLLED SUBSCRIPTION INTEGRATION AGREEMENT	This applies where the IA describes attributes/functions that must 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 <b>VISTA</b> 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.
CROSS REFERENCE	Cross-reference—There are several types of cross-references available. Most generally, a VA FileMan cross-reference 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; 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.
DATA	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.
DATA DICTIONARY (DD)	<p>The Data Dictionary is a global containing a description of what kind of data is stored in the global corresponding to a particular file. The data is used internally by VA FileMan for interpreting and processing files.</p> <p>A <b>Data Dictionary</b> contains the definitions of a file's elements (fields or data attributes); relationships to other files; and structure or design. Users generally review the definitions of a file's elements or data attributes; programmers review the definitions of a file's internal structure.</p>
DBA	<b>Database Administrator</b> , oversees package development with respect to <b>VISTA</b> Standards and Conventions (SAC) such as namespacing. Also, this term refers to the <b>Database Administration</b> function and staff.

DBIA	<b>D</b> atabase <b>I</b> ntegration <b>A</b> greement, a formal understanding between two or more <b>VISTA</b> packages, which describes how data is shared or how packages interact. The DBA maintains a list of DBIAs.
DEFAULT	Response the computer considers the most probable answer to the prompt being given. It is identified by double slash marks (//) immediately following it. This allows you the option of accepting the default answer or entering your own answer. To accept the default you simply press the Enter (or Return) key. To change the default answer, type in your response.
DELIMITER	Special character used to separate a field, record or string. VA FileMan uses the ^ character as the delimiter within strings.
DEPARTMENT OF VETERANS AFFAIRS	The Department of <b>V</b> eterans <b>A</b> ffairs, formerly called the <b>V</b> eterans <b>A</b> dministration.
DHCP	<b>D</b> ecentralized <b>H</b> ospital <b>C</b> omputer <b>P</b> rogram of the Veterans Health Administration (VHA), Department of Veterans Affairs (VA) is the former name for Veterans Health Information Systems and Technology Architecture ( <b>VISTA</b> ). <b>VISTA</b> 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. <b>VISTA</b> is composed of packages that undergo a verification process to ensure conformity with namespacing and other <b>VISTA</b> standards and conventions.
DIRECT MODE UTILITY	A programmer call that is made when working in direct programmer mode. A direct mode utility is entered at the MUMPS prompt (e.g., >D ^XUP). Calls that are documented as direct mode utilities <i>cannot</i> be used in application package code.
ELECTRONIC SIGNATURE CODE	Secret password that some users may need to establish in order to sign documents via the computer.
ENTRY	VA FileMan record. It is uniquely identified by an internal entry number (the .001 field) in a file.
EXTRINSIC FUNCTION	Extrinsic function is an expression that accepts parameters as input and returns a value as output that can be directly assigned.
FIELD	In a record, a specified area used for the value of a data attribute. The data specifications of each VA FileMan field are documented in the file's data dictionary. A field is similar to blanks on forms. It is preceded by words that tell you what information goes in that particular field. The blank, marked by the cursor on your terminal screen, is where you enter the information.

FILE	Set of related records treated as a unit. VA FileMan files maintain a count of the number of entries or records.
FILE MANAGER (VA FILEMAN)	The <b>VISTA</b> 's Database Management System (DBMS). The central component of the Kernel that defines the way standard <b>VISTA</b> files are structured and manipulated.
FORM	See ScreenMan Form.)
FORUM	The central E-mail system within <b>VISTA</b> . It is used by developers to communicate at a national level about programming and other issues. FORUM is located at the CIO Field Office - Washington, DC (162-2).
FREE TEXT	A DATA TYPE that can contain any printable characters.
GLOBAL VARIABLE	Variable that is stored on disk (M usage).
HEALTH LEVEL SEVEN (HL7)	National level standard for data exchange in all healthcare environments regardless of individual computer application systems.
HEALTH LEVEL SEVEN (HL7) <b>VISTA</b>	Messaging system developed as a <b>VISTA</b> software package that follows the HL7 Standard for data exchange.
INPUT TEMPLATE	A pre-defined list of fields that together comprise an editing session.
INTEGRATION AGREEMENTS (IA)  (Formerly known as DATABASE INTEGRATION AGREEMENTS [DBIA])	Integration Agreements define an agreement between two or more <b>VISTA</b> packages to allow access to one development domain by another. Any package developed for use in the <b>VISTA</b> 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 IA defines the attributes and functions that specify access. All IAs are recorded in the Integration Agreement database on FORUM. Content can be viewed using the DBA menu or the Technical Services' web page.
IRM	Information Resource Management. A service at VA medical centers responsible for computer management and system security.
KERNEL	Set of <b>VISTA</b> software routines that function as an intermediary between the host operating system and the <b>VISTA</b> application packages such as Laboratory, Pharmacy, IFCAP, etc. The Kernel provides a standard and consistent user and programmer interface between application packages and the underlying M implementation.
LINK	Non-specific term referring to ways in which files may be related (via pointer links). Files have links into other files.



MENU	List of choices for computing activity. A menu is a type of option designed to identify a series of items (other options) for presentation to the user for selection. When displayed, menu-type options are preceded by the word "Select" and followed by the word "option" as in Select Menu Management option: (the menu's select prompt).
MENU SYSTEM	The overall Menu Manager logic as it functions within the Kernel framework.
MENU TEXT	The descriptive words that appear when a list of option choices is displayed. Specifically, the Menu Text field of the OPTION file (#19). For example, User's Toolbox is the menu text of the XUSERTOOLS option. The option's synonym is TBOX.
NAME COMPONENTS	<p>A person's name is divided into the following component parts:</p> <ul style="list-style-type: none"> <li>• Family (Last) Name</li> <li>• Given (First) Name</li> <li>• Middle Name</li> <li>• Prefix</li> <li>• Suffix</li> <li>• Degree</li> </ul> <p>This is consistent with the ANSI HISPP Message Standards Developers Subcommittee Common Data Types definitions for element labels and formats, and thus compatible with messaging standards such as Health Level Seven (HL7) and X.12.</p>
NAMESPACING	Convention for naming <i>VISTA</i> package elements. The 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.
OPTION	An entry in the OPTION file. As an item on a menu, an option provides an opportunity for users to select it, thereby invoking the associated computing activity. Options may also be scheduled to run in the background, non-interactively, by Task Manager.
OPTION NAME	Name field in the OPTION file (e.g., XUMAINT for the option that has the menu text "Menu Management"). Options are namespaced according to <i>VISTA</i> conventions monitored by the DBA.

PACKAGE	The set of programs, files, documentation, help prompts, and installation procedures required for a given software application. For example, Laboratory, Pharmacy, and PIMS are packages. A <b>VISTA</b> software environment composed of elements specified via the PACKAGE file (#9.4). Elements include files and associated templates, namespaced routines, and namespaced file entries from the OPTION, HELP FRAME, BULLETIN, and FUNCTION files. As public domain software, packages may be requested through the Freedom of Information Act (FOIA).
POINTER	The address at which a data value is stored in computer memory. A relationship between two VA FileMan files, a pointer is a file entry that references another file (forward or backward). Pointers can be an efficient means for applications to access data by referring to the storage location at which the data exists.
PRIMARY KEY	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.
PRIVATE INTEGRATION AGREEMENT	Where only a single application is granted permission to use an attribute/function of another <b>VISTA</b> package. These IAs are granted for special cases, transitional problems between versions, and release coordination. A Private IA is also created by the requesting package based on their examination of the custodian package's features. An example would be where one package distributes a patch from another package to ensure smooth installation.
PROMPT	The computer interacts with the user by issuing questions called prompts, to which the user issues a response.
RECORD	Set of related data treated as a unit. An entry in a VA FileMan file constitutes a record. A collection of data items that refer to a specific entity (e.g., in a name-address-phone number file, each record would contain a collection of data relating to one person).
ROUTINE	Program or a sequence of instructions called by a program that may have some general or frequent use. M (previously referred to as MUMPS) routines are groups of program lines, which are saved, loaded, and called as a single unit via a specific name.
SAC	<b>Standards and Conventions.</b> Through a process of verification, <b>VISTA</b> packages are reviewed with respect to SAC guidelines as set forth by the Standards and Conventions Committee (SACC). Package documentation is similarly reviewed in terms of standards set by the Documentation Standards and Conventions Committee (DSCC).

SACC	<i>VISTA's</i> Standards and Conventions Committee. This Committee is responsible for maintaining the SAC.
SCREEN EDITOR	VA FileMan's Screen-oriented text editor. It can be used to enter data into any WORD-PROCESSING field using full-screen editing instead of line-by-line editing.
SCREENMAN FORMS	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 file (#.403) and exported with a package. Forms are composed of blocks (stored in the BLOCK file [#.404]) and can be regular, full screen pages or smaller, "pop-up" pages.
SCREEN-ORIENTED	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 Scrolling Mode.
SCROLLING MODE	The presentation of the interactive dialogue one line at a time. Compare to Screen-oriented.
STANDARD FORM of a name (also called STANDARD FORMAT)	<p>The definition of standard form (or standard format) is a person's name entirely in uppercase letters, containing no Arabic numerals (i.e., 1, 2, 3, etc.). The Family Name (last name) portion of a standard name appears to the left of the comma and contains no spaces and no punctuation except hyphens (-). The Given Name (first name), Middle Name, and Suffix (the portion to the right of the comma) contain no punctuation except for hyphens and spaces. NMI and NMN are not used for the Middle Name.</p> <p>The standard form of a name is:</p> <p style="text-align: center;">Family_name,Given_name&lt;space&gt;Middle_name&lt;space&gt;Suffix</p>
SUPPORTED REFERENCE INTEGRATION AGREEMENT	This applies where any <i>VISTA</i> application may use the attributes/functions defined by the IA (these are also called " <b>Public</b> "). An example is an IA that describes a standard API such as DIE or VADPT. The package that creates/maintains the Supported Reference must ensure it is recorded as a Supported Reference in the IA database. There is no need for other <i>VISTA</i> packages to request an IA to use these references; they are open to all by default.
TEMPLATE	Means of storing report formats, data entry formats, and sorted entry sequences. A template is a permanent place to store selected fields for use at a later time. Edit sequences are stored in the INPUT TEMPLATE file (#.402), print specifications are stored in the PRINT TEMPLATE file (#.4), and search or sort specifications are stored in the SORT TEMPLATE file (#.401).

TOOLKIT	<p>Toolkit (or Kernel Toolkit) is a robust set of tools developed to aid the Veterans Health Information Systems and Technology Architecture (<b>VISTA</b>) development community, and Information Resources Management (IRM), in writing, testing, and analysis of code. They are a set of generic tools that are used by developers, documenters, verifiers, and packages to support distinct tasks.</p> <p>The Toolkit provides utilities for the management and definition of development projects. Many of these utilities have been used by the CIO Field Office - San Francisco for internal management and have proven valuable. Toolkit also includes tools provided by other CIO Field Offices based on their proven utility.</p>
TRIGGER	<p>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 automatically enter the current date into another field holding the creation date.</p>
VA	<p>The Department of <b>V</b>eterans Affairs, formerly called the <b>V</b>eterans Administration.</p>
VA FILEMAN	<p>Set of programs used to enter, maintain, access, and manipulate a database management system consisting of files. A package of online computer routines written in the M language, which can be used as a stand-alone database system or as a set of application utilities. In either form, such routines can be used to define, enter, edit, and retrieve information from a set of computer stored files.</p>
VARIABLE	<p>Character, or group of characters, that refer to a value. M (previously referred to as MUMPS) recognizes 3 types of variables: local variables, global variables, and special variables. Local variables exist in a partition of main memory and disappear at sign-off. A global variable is 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. A special variable is defined by systems operations (e.g., \$TEST).</p>
VISN	<p><b>V</b>eterans <b>I</b>ntegrated <b>S</b>ervice <b>N</b>etwork</p>
<b>VISTA</b>	<p><b>V</b>eterans <b>H</b>ealth <b>I</b>nformation <b>S</b>ystems and <b>T</b>echnology <b>A</b>rchitecture (<b>VISTA</b>) (formerly the Decentralized Hospital Computer Program [DHCP]) of the Veterans Health Administration (VHA), Department of Veterans Affairs (VA). <b>VISTA</b> 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. <b>VISTA</b> is composed of packages, which undergo a verification process to ensure conformity with namespacing and other <b>VISTA</b> standards and conventions.</p>



# Appendix A (Data Conversion of the New Person File)

As part of the Post-Installation process for Patch XU\*8.0\*134, a data conversion is run in the NEW PERSON file (#200) to convert the .01 field (NAME field) to a standard format. This conversion standardizes names in the NEW PERSON file and parses them into their component parts.

The standard form of a name is:

Family\_name,Given\_name<space>Middle\_name<space>Suffix

As each name in the NAME field of the NEW PERSON file is converted and parsed, a corresponding entry is created in the new NAME COMPONENTS file (#20), exported with this patch. The components of the parsed name are stored in this new file.

This Appendix contains detailed information about the data conversion of the NEW PERSON file. The information contained in the following topics can also be found in the description of Patch XU\*8.0\*134 on the Patch Module.

## **POST^XLFNAME: New Person Name Conversion**

The New Person Name Conversion is run as part of the POST-INSTALL ROUTINE (POST^XLFNAME) of this patch. POST^XLFNAME loops through the entries in the NEW PERSON file and:

1. Converts the name stored in the NAME field (#.01) to the following standard form, if it is not already in standard form:

Family\_name,Given\_name<space>Middle\_name<space>Suffix

In forming the standard name, the conversion routine makes the following changes to each name:

- a. Removes any text within parentheses (), brackets [], or braces {}.

(If any such text is deleted, the original name with the parenthetical text is stored in the NOTES ABOUT NAME field in the NAME COMPONENTS file. After the conversion, you can use the search utility of VA FileMan to print out those entries in the NAME COMPONENTS file where NOTES ABOUT NAME is not null. See the VA FileMan Getting Started manual for more information on VA FileMan's Search utility.)

- b. Converts the name to uppercase.
- c. Moves any suffixes found between two commas immediately after the Family Name to the end of the name.
- d. In the Family Name (the portion to the left of the first comma):

- (1) Converts colons (:) and semicolons (;) to hyphens (-).
  - (2) Removes spaces and all other punctuation except hyphens.
- e. In the other name parts (the portion to the right of the first comma):
- (1) Converts colons, semicolons, commas (,), and periods (.) to spaces.
  - (2) Removes all punctuation except hyphens and spaces.
- f. Removes hyphens and spaces at the beginning and end of the name.
- g. Replaces two or more consecutive hyphens or spaces with a single hyphen or space.
- h. Converts Arabic numeral birth position suffixes 1ST, 2ND, 3RD, ..., 10TH to their Roman numeral equivalents I, II, III, ..., X.
- i. Moves any suffixes immediately to the left of the first comma to the end of the name, before any suffixes already at the end of the name.
- j. Moves DR to the end of the name if it is found immediately to the right of the first comma.
- k. Removes the string "NMI" or "NMN" if it is used as the Middle Name.
- l. If the length of the resulting standard name is greater than the maximum length (35) allowed by the input transform of the NAME field, the following algorithm is performed to shorten the name:
- (1) Truncate Middle Name from the right-most position until only the initial character is left;
  - (2) Drop Suffix;
  - (3) Truncate Given Name from the right-most position until only the initial character is left;
  - (4) Truncate Family Name from the right-most position.

Note that since the standard form of the name is never longer than the name being converted, this truncation would only occur if the name stored in the .01 field is already greater than 35 characters. This is unlikely, since the input transform restricts the name to 35 characters.

2. Parses the name into its component parts: Family (Last) Name, Given (First) Name(s), Middle Name, and Suffix(es). Those name parts are then stored in a new NAME COMPONENTS file (#20) brought in with this patch.

In parsing the name into its component parts, the New Person Name Conversion looks for Suffixes immediately to the left of the first comma, and at the very end of the name. It also looks for the DR suffix immediately after the first comma and looks for any suffix between two commas immediately after the Family Name. The portion of the name to the left of the comma, less any Suffixes, is assumed to be the Family Name.

After the conversion routine accounts for all Suffixes, it looks at the portion of the name after the comma. It assumes that the first space-delimited piece is the Given Name. If any other pieces are left, the last one (rightmost) is assumed to be the Middle Name, and anything else is appended to the end of the Given Name.

3. Makes the following changes to the name components before they are stored in the NAME COMPONENTS file:
  - a. Converts the name component to uppercase.
  - b. In the Family Name, converts semicolons (;) and colons (:) to hyphens (-).
  - c. In the other name parts (Given Name, Middle Name, and Suffix), converts semicolons, colons, and commas (,) to spaces.
  - d. Removes hyphens and spaces at the beginning and end of the name.
  - e. Replaces two or more consecutive hyphens/spaces with a single hyphen/space.
  - f. Removes spaces after periods.
  - g. Removes accent graves (`) and up-arrows (^).
4. Establishes a pointer from each entry in the NEW PERSON file to the corresponding entry in the NAME COMPONENTS file that contains the name parts.
5. Record in ^XTMP all changes that were made, and any potentially incorrect assumptions that were made in standardizing the name or parsing the name into its component parts. The purge date recorded in the 1st ^-piece of ^XTMP("XLFNAME",0) is the current date plus 90 days.
6. Stores in the 4th ^-piece of ^XTMP("XLFNAME",0) the record number of the last record successfully converted. If the New Person Name Conversion is run again if, for example, an aborted installation is restarted, the conversion will continue on from this point, rather than from the beginning of the file.

The PRINT^XLFNAME entry point, described below, can be used to print the information recorded in ^XTMP in step 5 above.

Information stored in ^XTMP by the New Person Name Conversion

```
-----
^XTMP("XLFNAME",0) = date run plus 90 days (purge date)
                    ^date run
                    ^Created by POST~XLFNAME (Post Install Conversion
                    of XU*8.0*134)
                    ^ien of last record successfully converted

^XTMP("XLFNAME",200,.01,rec) = Original Name
                              ^New (Standard) Name
                              ^Given (First) Name
                              ^Middle Name
                              ^Family (Last) Name
```



^Suffix

^XTMP("XLFNAME",200,.01,rec,sub) = ""

where,

rec = The internal entry number of a record in the NEW PERSON file.

sub = Any of the following:

- "DIFFERENT" The standard name is different from the original name.
- "FAMILY" The Family Name started with ST<period>. (The conversion routine removes the period and the space, if any, that immediately follows. For example, the name "NSPROVIDER,JOHN" is converted to "NSPROVIDER,JOHN".)
- "GIVEN" There is no Given Name.
- "MIDDLE" There are three or more names between the comma and the Suffix(es). (The conversion routine assumes that all name parts except the last between the comma and any suffixes are part of the Given Name. Only the last part is assumed to be the Middle Name.)
- "NM" "NMI" or NMN" appears to have been used as the Middle Name. (The conversion routine removes it from the standard name, and sets the Middle Name to null in the NAME COMPONENTS file.)
- "NUMBER" A name part (other than a valid numeric Suffix) contains a number.
- "PERIOD" The name contains periods that were removed to form the standard name.
- "PUNC" The name contains punctuation other than hyphens or spaces that were removed to form the standard name.
- "SPACE" The Family Name contains spaces that were removed to form the standard name.
- "STRIP" The name contained parenthetical text that was stripped out of the name. The conversion routine removes text enclosed in parentheses (), brackets [], and braces {}, and stores the original name with the parenthetical text in the NOTES ABOUT NAME field (#11) in the NAME COMPONENTS file. (For example, the conversion routine changes NSPROVIDER,JOHN (TRM) to NSPROVIDER,JOHN.)
- "SUFFIX" This node is returned if:
- Suffix(es) were found immediately to the left of the 1st comma. (The conversion routine moves them to the end of the name, before any suffixes already there.)

- I, V, or X, and nothing else except valid suffixes appear immediately after the Given Name. (The conversion routine interprets it as a Middle Name.)
- The name immediately after the Given Name appears to be a non-numeric suffix (IV, JR, SR, DR, MD, ESQ, etc.) and everything after that also appears to be suffixes. (The conversion routine assumes that there are a Given Name and Suffixes, but no Middle Name.)
- M.D. or M D is found at the end of the name, or before any valid suffixes at the end of the name. (The conversion routine assumes that M and D are initials in the Given or Middle Name.)
- The name part before any recognizable suffixes is more than one character in length and doesn't contain a "Y" or any vowels. (It is assumed to be a suffix.)
- A suffix was found between commas immediately after the Family Name. (The conversion routines moves it to the end of the name.)

"TRUNCATE" The standard name was truncated because it was longer than the maximum field length.

## **PRINT^XLFFNAME: Print Report in ^XTMP Global**

This is the Kernel Direct Mode Utility for Name Standardization. A Direct Mode Utility is a programmer call (entry point) that is made when working in direct programmer mode. A Direct Mode Utility is entered at the MUMPS prompt (e.g., >D ^XUP). Calls that are documented as direct mode utilities cannot be used in application package code.

This entry point prints a report of the information in the ^XTMP("XLFFNAME") global, which was populated during the New Person Name Conversion. Information in the ^XTMP("XLFFNAME") global is retained for 90 days. The report can be queued.

For each name listed, the report shows what file, field, and record that name came from. The report also shows the original and standardized form of each name. It shows a list of codes for each name, to indicate whether the name is different from its standardized form, and whether the conversion routine had a potential problem in determining the component parts of the name.

PRINT^XLFFNAME first prompts for a list of codes:

Enter a list of codes to print: ALL//

Press <RET> to accept the default "ALL" if you wish to print out all the records in ^XTMP.

If you wish to print out only those records with specific codes, you can enter a list of codes, separated by commas. Only those records with a code in your list will be printed. For example, if you enter 'PU,SP', only those entries where punctuation was removed by the conversion routine and entries where spaces were removed are printed in the report. Enter '??' at the prompt to see a complete listing of codes you can enter:

Explanation of Codes:

```

-----
D  : The standard name is different from the original name.
F  : The Family Name starts with ST<period>. The period and
      following space, if any, were removed.
G  : There is no Given Name.
M  : Assumption: There is more than one Given and only one Middle Name.
NM : NMI or NMN was used as the Middle Name.
NU : A name part contains a number.
PE : Periods were removed.
PU : Punctuation was removed.
SP : Spaces were removed from the Family Name.
ST : Text in parentheses was stripped from the name.
SU : One or more of the following situations was encountered relating
      to suffixes:
      - Suffixes were found immediate to left of the first comma.
      - I, V, or X was interpreted as a Middle Name.
      - A name part was interpreted as a Suffix, not a Middle Name.
      - M.D. or M D was NOT interpreted as a Suffix.
      - A name part with no vowels was interpreted as a Suffix.
      - A Suffix was found between commas immediately after the Family
        Name.
T  : The standard name was truncated.

```

You are then prompted for a list of codes to exclude.

Enter a list of codes to exclude:

Records with any code in the list you enter here will be excluded from the printout. This list overrides the list of codes to include.

## Correcting Names That Were Standardized or Parsed Incorrectly

If the report generated by the PRINT^XLFFNAME entry point described above shows that names were standardized or parsed incorrectly, you can use the "Edit an Existing User" option [XUEXISTING USER] to correct them.

Note that the "Edit an Existing User" option does not let you select users who have been terminated. However, if you wish to correct the name of a terminated user, you can use VA FileMan to edit the entry in the Name Components file that corresponds to that user.

For example, suppose the New Person Name Conversion parsed and stored the name "NS'PROVIDER,MARY ROSE ELIZABETH" in the Name Components file as first name equal

Appendix A

to "MARY ROSE" and middle name equal to "ELIZABETH". However, the name should be stored as first name equal to "MARY" and middle name equal to "ROSE ELIZABETH". If you are unable to use the "Edit An Existing User" option to edit that name because the person has been terminated, you can use VA FileMan's "ENTER OR EDIT FILE ENTRIES" option as follows:

Select OPTION: **ENTER OR EDIT FILE ENTRIES**

INPUT TO WHAT FILE: NAME COMPONENTS//  
EDIT WHICH FIELD: ALL// **<RET>**

Select NAME COMPONENTS FILE: NS'PROVIDER,MARY JANE ELIZABETH      200  
.01 1122, O'  
REILLY,MARY JANE ELIZABETH  
FILE: 200//      (No Editing)  
FIELD: .01//      (No Editing)  
IENS: 1122,//      (No Editing)  
FAMILY (LAST) NAME: NS'PROVIDER// **<RET>**  
GIVEN (FIRST) NAME: MARY ROSE// **MARY**  
MIDDLE NAME: ELIZABETH// **ROSE ELIZABETH**



# Index

## A

ADD^XUSERNEW, 51  
Adding new users  
    ADD^XUSERNEW, 51  
APIs  
    \$\$BLDNAME^XLFNNAME  
        Build Name from Component Parts, 31  
    \$\$CLEANC^XLFNNAME  
        Name Component Standardization Routine, 34  
    \$\$FMNAME^XLFNNAME  
        Convert HL7 Formatted Name to Name, 35  
    \$\$HLNAME^XLFNNAME  
        Convert Name to HL7 Formatted Name, 37  
    \$\$NAMEFMT^XLFNNAME  
        Formatted Name from Name Components, 40  
    DELCOMP^XLFNNAME2  
        Delete Name Components Entry, 48  
    NAMECOMP^XLFNNAME  
        Component Parts from Standard Name, 39  
    STDNAME^XLFNNAME  
        Name Standardization Routine, 43  
    UPDCOMP^XLFNNAME2  
        Update Name Components Entry, 49  
Application entry points, 62

## C

Callable entry points, 62  
Convert HL7 Formatted Name to Name, 35  
Convert HL7 Formatted Name to NAME COMPONENT fields 1 through 6, 64  
Convert NAME COMPONENT fields 1 through 6 to HL7 Formatted Name, 64  
Convert Name to HL7 Format, 37, 63  
Convert name to standard format, 3, 4, 9, 25, 43, 79–83  
Correcting Names That Were Standardized or Parsed Incorrectly, 84

## D

Data Conversion of the New Person File, 3  
Data Dictionary of New Person File, 4, 25, 67  
Document History, iii

## E

Entry points, 62

## F

File Security, 70  
FUNCTION file (#.5), VA FileMan, 19

## G

Guidelines for Entering Person Names in VISTA, 11

## H

How to Use this Manual, ix

## I

INPUT templates, 14  
    XUEXISTING USER, 13, 58, 60, 62  
    XUNEW USER, 13, 58, 59  
    XUREACT USER, 13, 58, 60  
INPUT Templates

## K

Kernel APIs for Name Standardization  
    \$\$BLDNAME^XLFNNAME  
        Build Name from Component Parts, 31  
    \$\$CLEANC^XLFNNAME  
        Name Component Standardization Routine, 34  
    \$\$FMNAME^XLFNNAME  
        Convert HL7 Formatted Name to Name, 35  
    \$\$HLNAME^XLFNNAME  
        Convert Name to HL7 Formatted Name, 37  
    \$\$NAMEFMT^XLFNNAME  
        Formatted Name from Name Components, 40  
    DELCOMP^XLFNNAME2  
        Delete Name Components Entry, 48  
    NAMECOMP^XLFNNAME  
        Component Parts from Standard Name, 39  
    STDNAME^XLFNNAME  
        Name Standardization Routine, 43  
    UPDCOMP^XLFNNAME2  
        Update Name Components Entry, 49  
KERNEL SYSTEM PARAMETERS file, 51

## N

NAME COMPONENTS file (#20)  
    ANSI HISPP Message Standards Developers Subcommittee Common Data Types, 4, 74  
    Convert HL7 Formatted Name to NAME COMPONENT fields 1 through 6, 64  
    Convert NAME COMPONENT fields 1 through 6 to HL7 Formatted Name, 64  
    corresponding entry in NEW PERSON file, 3  
    DEGREE field (#6), 4, 13, 24, 55, 58  
    FAMILY (LAST) NAME field (#1), 4, 10, 13, 23, 55, 58, 59  
    FIELD field (#.02), primary key, 23, 55  
    FILE field (#.01), primary key, 23, 55

GIVEN (FIRST) NAME field (#2), 4, 10, 13, 24, 55, 58, 59  
 hold the component parts of a name, 23  
 IENS field (#.03), primary key, 23, 55  
 MIDDLE NAME field (#3), 4, 10, 13, 24, 55, 58, 59  
 NAME COMPONENTS pointer field (#10.1), 25  
 NOTES ABOUT NAME field (#11), 25  
 pointer to, 25  
 PREFIX field (#4), 4, 13, 24, 55, 58, 59  
 preserves punctuation, 10  
 Primary key, 23  
 SUFFIX field (#5), 4, 10, 13, 24, 55, 58, 59  
 synchronized with NEW PERSON file, 4, 17  
 using the Name Components, 16  
 VA FileMan FUNCTION XLFMTNAME, 5, 19  
 NAME COMPONENTS pointer field (#200,10.1), 55  
 namespace, 67  
 NEW PERSON file (#200)  
 NAME COMPONENTS pointer field (#200,10.1), 55  
 NEW PERSON file (#200)  
 adding a new user to, 15  
 corresponding entry in NAME COMPONENTS file, 10  
 data conversion of, 3  
 Data Dictionary, 4, 25, 55  
 edits to name components saved in, 17  
 PREFIX field NOT stored in the, 16  
 Standard Form, 9  
 synchronized with NAME COMPONENTS file, 17, 25  
 using the Name Components, 16  
 VA FileMan FUNCTION XLFMTNAME, 5, 20  
 NEW PERSON file (#200)  
 NAME field (#200,.01), 55  
 NEW PERSON file (#200)  
 DEGREE field (200,10.6), 55  
 NEW PERSON file (#200)  
 VA FileMan INPUT Templates, 56  
 NEW PERSON file (#200)  
 ScreenMan Forms, 57  
 NEW PERSON file (#200)  
 ScreenMan forms, 59  
 NEW PERSON file (#200)  
 Data Dictionary, 67  
 NEW PERSON file (#200)  
 NAME COMPONENTS pointer field (#200,10.1), 67  
 NEW PERSON file (#200)  
 NAME field (#200,.01), 67  
 NEW PERSON file (#200)  
 DEGREE field (200,10.6), 67  
 NEW PERSON file (#200)  
 data conversion of, 79–83

## O

Orientation, ix

## P

Patch History, iii  
 patient & user names  
 test data, ix

PATIENT file [#2], 5

## R

Revision History, iii

## S

ScreenMan forms, 14  
 XUEXISTING USER, 13  
 XUNEW USER, 13  
 XUREACT USER, 13  
 Social Security Numbers  
 test data, ix  
**Standard Format (also called Standard Form)**, 3, 9, 16, 34, 43

## T

test data  
 patient & user names, ix  
 Social Security Numbers, ix

## V

VISTA, definition of, 78