



**PARAMETER TOOLS
SUPPLEMENT TO PATCH DESCRIPTION**

Patch XT*7.3*26

August 2001

Revised: October 2008

Revision History

Documentation Revisions

The following table displays the revision history for this document. Revisions to the documentation are based on patches and new versions released to the field.

Table i. Documentation revision history

Date	Revision	Description	Author
10/28/08	2.4	Updates: <ul style="list-style-type: none">• Table 1-1 to add "DEV" entity and correct the OE/RR LIST file number from "101.21" to the correct "100.21" file number.• Made general format updates to follow current style guidelines and standards.	Thom Blom Oakland, CA Office of Information field Office (OIFO)
01/03/05	2.3	Reviewed document and edited for the "Data Scrubbing" and the "PDF 508 Compliance" projects. PDF 508 Compliance —The final PDF document was recreated and now supports the minimum requirements to be 508 compliant (i.e., accessibility tags, language selection, alternate text for all images/icons, fully functional Web links, successfully passed Adobe Acrobat Quick Check).	Lauren Gorgoglione, Bay Pines OIFO
07/08/04	2.2	Updated documentation to include examples for all APIs.	Wally Fort, Oakland, CA OIFO; Susan Strack, Oakland, CA OIFO
07/01/04	2.1	Updated documentation based on changes from Patches XT*7.3*79 and XT*7.3*82.	Wally Fort, Oakland, CA OIFO; Thom Blom, Oakland, CA OIFO; Susan Strack, Oakland, CA OIFO
12/12/03	2.0	Reformatted Patch XT*7.3*26 Supplemental Documentation and updated API content.	Wally Fort, Oakland, CA OIFO; Thom Blom, Oakland, CA OIFO
08/--/01	1.0	Initial Patch XT*7.3*26 Supplemental Documentation creation.	Wally Fort, Oakland, CA OIFO; Marcia Insley, Salt Lake City, UT OIFO

Patch Revisions

For a complete list of patches related to this software, please refer to the Patch Module on FORUM.

Revision History

Contents

Revision History	iii
Figures and Tables	vii
Orientation	ix
1. User Manual—Parameter Tools	1-13
Introduction	1-13
Background	1-14
Description	1-15
Definitions	1-15
Entity	1-15
Parameter	1-16
Instance	1-16
Value	1-17
Parameter Template	1-17
Why Would You Use Parameter Tools?	1-18
Example	1-18
2. Programmer—Parameter Tools	2-1
Application Program Interfaces (APIs)—^XPAR Routine	2-1
ADD^XPAR(): Add Parameter Value	2-1
CHG^XPAR(): Change Parameter Value	2-2
DEL^XPAR(): Delete Parameter Value	2-2
EN^XPAR(): Add, Change, Delete Parameters	2-4
ENVAL^XPAR(): Return All Parameter Instances	2-5
\$\$GET^XPAR(): Return an Instance of a Parameter	2-7
GETLST^XPAR(): Return All Instances of a Parameter	2-9
GETWP^XPAR(): Return Word-processing Text	2-10
NDEL^XPAR(): Delete All Instances of a Parameter	2-11
PUT^XPAR(): Add/Update Parameter Instance	2-11
REP^XPAR(): Replace Instance Value	2-12
Application Program Interfaces (APIs)—^XPAREDIT Routine	2-13
BLDLST^XPAREDIT(): Return All Entities of a Parameter	2-13
EDIT^XPAREDIT(): Edit Instance and Value of a Parameter	2-13

Contents

EDITPAR^XPAREEDIT(): Edit Single Parameter2-14
EN^XPAREEDIT(): Parameter Edit Prompt2-15
GETENT^XPAREEDIT(): Prompt for Entity Based on Parameter.....2-15
GETPAR^XPAREEDIT(): Select Parameter Definition File2-16
TED^XPAREEDIT(): Edit Template Parameters (No Dash Dividers).....2-17
TEDH^XPAREEDIT(): Edit Template Parameters (With Dash Dividers).....2-18
IndexIndex-1

Figures and Tables

Figures

Figure 1-1. Setting up the PARAMETER DEFINITION file (#8989.51).....	1-19
Figure 1-2. Use ^XPAREEDIT to enter a value for your new parameter	1-19
Figure 1-3. Get the value of your new parameter for your VistA application	1-19
Figure 1-4. Adding a sample parameter template	1-20

Tables

Table 1-1. Parameter Entities.....	1-15
Table 1-2. Templates—Parameter Tools	1-17

Orientation



How to Use this Manual

Throughout this manual, advice and instructions are offered regarding the use of Kernel Toolkit and Patch XT*7.3*26 software and the functionality it provides for Veterans Health Information Systems and Technology Architecture (VistA) software products.

This manual uses several methods to highlight different aspects of the material:

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

Table ii: Documentation symbol descriptions

Symbol	Description
	NOTE/REF: Used to inform the reader of general information including references to additional reading material.
	CAUTION/DISCLAIMER: Used to caution the reader to take special notice of critical information.

- Descriptive text is presented in a proportional font (as represented by this font).
- Conventions for displaying TEST data in this document are as follows:
 - The first three digits (prefix) of any Social Security Numbers (SSN) 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 and "N" represents the first name as a number spelled out and incremented with each new entry. For example, in Kernel (KRN) test patient and user names would be documented as follows: KRNPATIENT,ONE; KRNPATIENT,TWO; KRNPATIENT,THREE; etc.
- Sample HL7 messages, "snapshots" of computer online displays (i.e., roll-and-scroll screen captures/dialogues) and computer source code, if any, are shown in a *non*-proportional font and enclosed within a box.
- User's responses to online prompts will be boldface.
- References to "<Enter>" within these snapshots indicate that the user should press the **Enter** key on the keyboard. Other special keys are represented within < > angle brackets. For example, pressing the **PF1** key can be represented as pressing <PF1>.
- Author's comments, if any, are displayed in italics or as "callout" boxes.



NOTE: Callout boxes refer to labels or descriptions usually enclosed within a box, which point to specific areas of a displayed image.

- All uppercase is reserved for the representation of M code, variable names, or the formal name of options, field and file names, and security keys (e.g., the XUPROGMODE key).

How to Obtain Technical Information Online

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



NOTE: Methods of obtaining specific technical information online will be indicated where applicable under the appropriate topic.

REF: Please refer to the *Kernel Technical Manual* for further information.

Help at Prompts

VistA M Server-based software provides online help and commonly used system default prompts. Users are encouraged to enter question marks at any response prompt. At the end of the help display, you are immediately returned to the point from which you started. This is an easy way to learn about any aspect of the software.

Obtaining Data Dictionary Listings

Technical information about VistA M Server-based files and the fields in files is stored in data dictionaries (DD). You can use the List File Attributes option on the Data Dictionary Utilities submenu in VA FileMan to print formatted data dictionaries.



REF: For details about obtaining data dictionaries and about the formats available, please refer to the "List File Attributes" chapter in the "File Management" section of the *VA FileMan Advanced User Manual*.

Assumptions About the Reader

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

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

This manual provides an overall explanation of Kernel and the functionality contained in Kernel 8.0. However, no attempt is made to explain how the overall VistA programming system is integrated and maintained. Such methods and procedures are documented elsewhere. We suggest you look at the various VA Internet and Intranet Web pages for a general orientation to VistA. For example, go to the Office of Information and Technology (OI&T) VistA Development Intranet Website:

<http://vista.med.va.gov/>

Reference Materials

Readers who wish to learn more about the Kernel Toolkit software should consult the following:

- *Kernel Toolkit Release Notes*
- *Kernel Toolkit Installation Guide*
- *Kernel Systems Management Guide*
- *Kernel Developer's Guide*
- *Kernel Technical Manual*
- *Kernel Security Tools Manual*
- Kernel Website:

<http://vista.med.va.gov/kernel/index.asp>

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



NOTE: This site contains additional information and documentation.

VistA documentation is made available online in Microsoft Word format and in Adobe Acrobat Portable Document Format (PDF). The PDF documents *must* be read using the Adobe Acrobat Reader, which is freely distributed by Adobe Systems Incorporated at the following Website:

<http://www.adobe.com/>

VistA documentation can be downloaded from the VHA Software Document Library (VDL) Website:

<http://www.va.gov/vdl/>

VistA documentation and software can also be downloaded from the Product Support (PS) anonymous directories:

- Preferred Method download.vista.med.va.gov

This method transmits the files from the first available FTP server.

- Albany OIFO [ftp.fo-albany.med.va.gov](ftp://ftp.fo-albany.med.va.gov)
- Hines OIFO [ftp.fo-hines.med.va.gov](ftp://ftp.fo-hines.med.va.gov)
- Salt Lake City OIFO [ftp.fo-slc.med.va.gov](ftp://ftp.fo-slc.med.va.gov)



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

1. User Manual—Parameter Tools

This is the User Manual section of this supplemental documentation for the Parameter Tools software (i.e., Kernel Toolkit Patch XT*7.3*26).

The intended audience for this chapter is the Information Resource Management (IRM) at a local site. However, it can also be helpful to application developers of Veterans Health Information Systems and Technology Architecture (VistA) software and others in VA Office of Information & Technology (OI&T), and Product Support (PS).

Introduction

This supplemental documentation is intended for use in conjunction with the Parameter Tools patch (XT*7.3*26). This documentation explains the functions available with the use of the Parameter Tools and describes the APIs that are part of the patch. It combines information from the patch description and two Integration Agreements (IAs): 2263 and 2336, as well as providing additional explanatory material and a generic example to illustrate the use of the Parameter Tools.

In brief, the Parameter Tools patch provides a method of managing the definition, assignment, and retrieval of parameters for VistA software applications.

VistA software applications are designed to be used in a variety of ways. Many aspects of hospital activity vary from one hospital to another and thus there are many possible ways software applications can be used that also vary from one institution to another. Each site has its own requirements—its own settings for each software application. IRM staff must modify the software parameters to fit their requirements.

Previously, each software application had its own files and options but no two software applications had the site parameters set up the same way or found in the same place. Thus, when a new software application was released, each site would have to look for the location where the settings were stored for that software. Next, they would have to look to see what settings were available and how to set them. Very little about the parameters was uniform from software to software.

With the Computerized Patient Record System (CPRS) software, the idea was born that a parameter file could be created to export with the software. The CPRS parameter file and parameter utility were subsequently modified to create a generic method of exporting and installing other VistA software applications. Most developers were willing to abandon previous methods and use this tool for software they were developing.

Parameter Tools was designed as a method of managing the definition, assignment, and retrieval of parameters for VistA software. A parameter may be defined for various levels at which you want to allow the parameter described (e.g., software level, system level, division level, location level, user level).

Background

Whenever you have an entity with many attributes that apply to it, you can do either of the following:

1. Make one big relation to represent that entity.
2. Create a "binary" relation to represent the entity. In the latter case, the relation consists of two columns (thus the term binary), one representing the attribute and the other representing the value for that attribute. So each tuple (i.e., a data type/data object containing two or more components) of the relation represents a single attribute and its associated value.



NOTE: This works only when the individual attributes are independent observations (have no dependencies on anything other than the key that identifies the entity). Such a relation tends to look a lot like a Windows INI file.

Most of the VistA parameter files were very long lists of independent values that pertained to a single entity. In most cases, this entity was the site or system on which the software was running [similar to an INI file]. In other cases, however, the parameter files had multiples that made things more complex. These multiples generally allow parameters to be defined at levels more specific than the site (e.g., by divisions or hospital location). It seems best to accommodate this by using both an entity identifier and parameter together to name any given value. This yields a relation with a compound key:

Entity | Parameter = Value

Finally, it seems that multiple-valued parameters (e.g., collection times) occur often enough that it is worthwhile to add a field to identify the parameter instance. So the relation becomes:

Entity | Parameter | Instance = Value

This is the relation that the PARAMETERS file (#8989.5) is intended to represent.

Software parameter files frequently maintain parameters that apply to the site, a division, or a location. In addition, many parameters that apply to individual users are kept in the NEW PERSON file (#200). Also, many parameter values are hard-coded in individual software routines for the case when the site has not set up a value for a given parameter. Entity, then, is implemented as a variable pointer.

A given parameter may occur for a variety of entities. In fact, we frequently need to obtain the value of a parameter by following an entity "chain." For example, the Add Orders menu a CPRS user sees may be defined at various levels. Initially, a site generally creates a custom Add Orders menu. Later, hospital locations may each build a custom menu that more specifically meets their needs. Individual users may also have their own Add Orders menus. If no site configuration has been done, the Add Orders menu exported with OE/RR is used. So, when OE/RR needs to display an Add Orders menu, a chain is followed that looks first to see if the user has their own menu. Next, the current location is checked, followed by the site. Finally, if no values exist, the software default menu is used.

In the PARAMETER DEFINITION file (#8989.51), a multiple lists which entities are valid with a given parameter. These entities are also assigned a precedence, so that it is possible to write functions that will "chain" through entities until a value is found, using the proper sequence.

Description

Patch XT*7.3*26 contains a developer toolset that allows creation of software parameters in a central location. Integration Agreements (IAs) 2263 and 2336 define the supported entry points for this application. Kernel Patch XU*8.0*201 allows KIDS to transport the parameters.

Parameter Tools is a generic method of handling parameter definition, assignment, and retrieval. A parameter can be defined for various entities where an entity is the level at which you want to allow the parameter defined (e.g., software level, system level, division level, location level, user level, etc.). A developer can then determine in which order the values assigned to given entities are interpreted.

Definitions

The following are some basic definitions used by Parameter Tools.

Entity

An entity is a level at which you can define a parameter. The entities allowed are stored in the PARAMETER ENTITY file (#8989.518). Kernel Toolkit patches maintain entries in this file. The list of allowable entries is as follows:

Table 1-1. Parameter Entities

Prefix	Message	Points To File
PKG	Package	PACKAGE (#9.4)
SYS	System	DOMAIN (#4.2)
DIV	Division	INSTITUTION (#4)
SRV	Service	SERVICE/SECTION (#49)
LOC	Location	HOSPITAL LOCATION (#44)
TEA	Team	TEAM (#404.51)
CLS	Class	USR CLASS (#8930)
USR	User	NEW PERSON (#200)
BED	Room-Bed	ROOM-BED (#405.4)
OTL	Team (OE/RR)	OE/RR LIST (#100.21)
DEV	Device	DEVICE (#3.5)

Package (PKG), as an entity, allows the software defaults to be handled the same way as other parameters rather than hard-coded.

System (SYS), Division (DIV), Location (LOC), and User (USR) are frequent entries in existing software parameter files (or additions to the NEW PERSON file [#200]).

Service (SRV), Team (TEA), and Class (CLS) are referenced frequently by parameters that pertain to Notifications.

The process of exporting software using this kind of parameters file involves sending:

- Parameter definitions that belong to the software (entries in the PARAMETER DEFINITION file [#8989.51]).
- Actual parameter instances that point to the software (entries in the PARAMETERS file [#8989.5] that have an entity that matches the software).

All the other entries in the PARAMETERS file (#8989.5 (those that correspond to entities other than package [PKG]) would never be exported, as they are only valid for the system on which they reside.

Parameter

A parameter is the actual name under which values are stored. The name of the parameter must be namespaced and it must be unique and start with two uppercase characters. Parameters can be defined to store the typical software parameter data (e.g., the default add order screen in OE/RR), but they can also be used to store graphical user interface (GUI) application screen settings a user has selected (e.g., font or window width). With each parameter, a more readable display name can also be defined. When a parameter is defined, the entities that may set that parameter are also defined. The definition of parameters is stored in the PARAMETER DEFINITION file (#8989.51).

Instance

An instance is a unique value assigned to an entity/parameter combination. For most parameters, there will only be one instance, that is, instance does not apply and is simply set to "1".

However, a parameter can be multi-valued—it can have more than one instance. More than one value can be assigned to the parameter as it relates to a specific entity. For example, lab collection times at a division. For a single entity (division in this case), multiple collection times may exist. Each collection time would be assigned a unique instance.

A parameter is not considered multi-valued if it can apply to several entities, but for each entity only one value of the parameter exists. For example, "maximum days for a lab order" can be set for every location in the hospital. However, since there is only one value for each location, "maximum days for a lab order" is not multi-valued.

When a parameter that is multi-valued is defined, the instance can be defined as any of the following:

- Numeric
- Date/Time
- Pointer
- Set Of Codes
- Free Text
- Yes/No

The validating logic for an instance is defined the same way as for a value.

Value

A value can be assigned to every parameter for the entities allowed in the parameter definition. Values are stored in the PARAMETERS file (#8989.5). Fields in the PARAMETERS file (#8989.5) map to DIR fields. DIR is used to validate the data. Values can be any of the following:

- Numeric
- Date/Time
- Pointer
- Set Of Codes
- Free Text
- Yes/No
- Word-processing Type

Parameter Template

A Parameter template is similar to an Input template. It contains a list of parameters that can be entered through an input session (e.g., an option). Templates are stored in the PARAMETER TEMPLATE file (#8989.52). Entries in this file must also be namespaced.

There are two Input templates for adding parameter definitions:

Table 1-2. Templates—Parameter Tools

Template	Description
XPAR SINGLE VALUED CREATE	For adding/editing parameters that will be single valued
XPAR MULTI VALUED CREATE	For adding/editing parameters that will be multiple valued

Why Would You Use Parameter Tools?

The reason a developer would use Parameter Tools is to allow a hierarchical designation of a parameter value. Thus, rather than many parameters that exist now, which are just for the system level or just for a particular clinic, Parameter Tools allows you to define:

- Different levels at which the parameter can be set.
- In what priority the values are used.

Take, for example, setting up a default order menu for a person. Each facility may have a default order menu for their primary care clinicians. Each division may have one that is slightly different if their practices vary enough. For each location, they may set up a different order menu so that users working in a cardiology clinic get a different set of possible orders than those in a dermatology clinic. And there may be reasons to give one specific person a different order menu because they are authorized to prescribe additional medications, because they tend to practice in a different flow, or for other reasons. It's one parameter, but it allows the parameter to be set for multiple entities (at multiple levels). Those entities are defined in the IA, but can include package (PKG, which only developers should set—these are default export values), system (SYS, whole medical facility), division (DIV), location (LOC), room-bed (BED), team (TEA), provider, etc.

The PARAMETER DEFINITION file (#8989.51) defines what entities are allowed to be used for a parameter and in which order they are resolved (individual takes precedence over location takes precedence over division takes precedence over system which takes precedence over package). Sometimes you would want to create defaults for your medical center, but allow users in a certain area to customize what they see and do for their particular role.

XPAR finds the appropriate value based on the parameter definitions and settings that may exist. This way, the developer does not need to look at multiple different location or person files to determine how the software should operate.

With integrations, this is even more important because it allows facilities to integrate; however, at the same time, continue some business practices based on parameters set at the division level rather than at the system level.

Example

The following is a simple example of a way you might use the Parameter Tools. Suppose you needed a parameter that could be set as a default for the system (account) and also overridden for a given user. Previously, you had to add a field to a software site file (e.g., the KERNEL SYSTEM PARAMETERS file [#8989.3]) and then add a similar field to the NEW PERSON file (#200). This situation is a perfect use of the Parameter Tools.

1. You need the equivalent to a data dictionary (DD) entry. Figure 1-1 goes into the PARAMETER DEFINITION file (#8989.51). In this case we need a Yes/No Set of Codes. So, this is what you set up:

Figure 1-1. Setting up the PARAMETER DEFINITION file (#8989.51)

```
Name: XUS-XUP VPE
DISPLAY TEXT: Drop into VPE
MULTIPLE VALUED: n <Enter> No
VALUE DATA TYPE: y <Enter> yes/no
VALUE HELP: Should XUP drop the user into the VPE environment?
Description...
PRECEDENCE: 1      ENTITY FILE: USER
PRECEDENCE: 2      ENTITY FILE: SYSTEM
```



NOTE: Figure 1-1 only shows the fields with the data necessary to set up the PARAMETER DEFINITION file (#8989.51).

Figure 1-1 lists the order that values are looked for and/or returned. You want a USER value (File #200) if there is one; otherwise a SYSTEM value (File #4.2). It also gives the entities that are allowed to have values of this data. In the place of SYSTEM, you could have used PACKAGE.

2. You can use ^XPAREDIT to enter a value for your new parameter:

Figure 1-2. Use ^XPAREDIT to enter a value for your new parameter

```
>D ^XPAREDIT

      --- Edit Parameter Values ---

Select PARAMETER DEFINITION NAME: XUS-XUP VPE <Enter>      Drop into VPE

XUS-XUP VPE may be set for the following:

      1  User          USR      [choose from NEW PERSON]
      2  System        SYS      [NXT.KERNEL.ISC-SF.VA.GOV]

Enter selection: 2 <Enter>  System  NXT.KERNEL.ISC-SF.VA.GOV

----- Setting XUS-XUP VPE for System: NXT.KERNEL.ISC-SF.VA.GOV -----
Value: NO
...
```

3. How do you get this value out in your VistA application?

Figure 1-3. Get the value of your new parameter for your VistA application

```
>S X=$$GET^XPAR("USR^SYS","XUS-XUP VPE",1,"Q") ;X will be null, 0 or 1.
```

For the first parameter, you want a value from USR (user / New Person) or SYS (system)
Next, this is the name of the parameter: "XUS-XUP VPE"

Next, in this example you only allow one instance (optional, Defaults to 1 if not passed in).

Last, the format to return: Use "Q" to get the internal value.

Adding the parameter template with VA FileMan, Figure 1-4:

Figure 1-4. Adding a sample parameter template

```
Select PARAMETER DEFINITION NAME: XUS-XUP VPE  <Enter>      Drop into VPE

NAME: XUS-XUP VPE// <Enter>
DISPLAY TEXT: Drop into VPE// <Enter>
MULTIPLE VALUED: No// <Enter>
INSTANCE TERM: <Enter>
VALUE TERM: <Enter>
PROHIBIT EDITING: <Enter>
VALUE DATA TYPE: yes/no// <Enter>
VALUE DOMAIN: <Enter>
VALUE HELP: Should XUP drop the user into the VPE environment.
VALUE VALIDATION CODE: <Enter>
VALUE SCREEN CODE: <Enter>
INSTANCE DATA TYPE: <Enter>
INSTANCE DOMAIN: <Enter>
INSTANCE HELP: <Enter>
INSTANCE VALIDATION CODE: <Enter>
INSTANCE SCREEN CODE: <Enter>
DESCRIPTION:
  1> This parameter controls if a user when exiting XUP is dropped into
  2> VPE or right to the ">" prompt.
EDIT Option: <Enter>
Select PRECEDENCE: 2// <Enter>
  PRECEDENCE: 2// <Enter>
  ENTITY FILE: SYSTEM// <Enter>
Select PRECEDENCE: <Enter>
```

2. Programmer Manual—Parameter Tools


This is the Programmer Manual section of this supplemental documentation for the Parameter Tools software (i.e., Kernel Toolkit Patch XT*7.3*26).

The intended audience for this chapter is the application developers of VistA software. However, it can also be helpful to others in Information Resource Management (IRM), Product Support (PS), and Application Structure and Integration Services (ASIS).

Application Program Interfaces (APIs)—^XPAR Routine

The following is a list of APIs available in the ^XPAR routine.


ADD^XPAR(): Add Parameter Value

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2263
Description	<p>This API can be called to add a new parameter value as an entry to the PARAMETERS file (#8989.5) if the Entity/Parameter/Instance combination does not already exist.</p> <p> REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.</p>
Format	ADD^XPAR(entity,parameter[,instance],value[,.error])
Input/Output Parameters	For the definition of the input and output parameters used in this API, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

Example:

```
>D ADD^XPAR("PKG.KERNEL","XPAR TEST FREE TEXT",,"Today Good",.ERROR)
```


CHG^XPAR(): Change Parameter Value

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2263
Description	<p>This API can be called to change the value assigned to an existing parameter if the Entity/Parameter/Instance combination already exists.</p> <p> REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.</p>
Format	CHG^XPAR(entity,parameter[,instance],value[,.error])
Input/Output Parameters	For the definition of the input and output parameters used in this API, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

Example

```
>D CHG^XPAR("PKG.KERNEL","XPAR TEST FREE TEXT",,"Tomorrow Hot",.ERROR)
```

DEL^XPAR(): Delete Parameter Value

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2263
Description	<p>This API can be called to delete an existing parameter instance if the value assigned is "@".</p> <p> REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.</p>
Format	DEL^XPAR(entity,parameter[,instance][,.error])
Input/Output Parameters	For the definition of the input and output parameters used in this API, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

Example

```
>D DEL^XPAR("PKG.KERNEL","XPAR TEST FREE TEXT",),.ERROR) I ERROR>0 W !.ERROR
```

EN^XPAR(): Add, Change, Delete Parameters

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2263
Description	<p>This API performs any one of the following functions:</p> <ul style="list-style-type: none"> • Adds the value as a new entry to the PARAMETERS file (#8989.5) if the Entity Parameter Instance combination does not already exist. • Changes the value assigned to the parameter in the PARAMETERS file (#8989.5) if the Entity Parameter Instance combination already exists. • Deletes the parameter instance in the PARAMETERS file (#8989.5) if the value assigned is "@".
Format	EN^XPAR(entity,parameter[,instance],value[,.error])
Input Parameters	<p>entity: (required) Entity can be set to the following:</p> <ul style="list-style-type: none"> • Internal variable pointer (nnn;GLO(123,)) • External format of the variable pointer using the three-character prefix (prefix.entryname) • Prefix alone to set the parameter based on the current entity selected. This works for the following entities: <ul style="list-style-type: none"> – "USR"—Uses current value of DUZ. – "DIV"—Uses current value of DUZ(2). – "SYS"—Uses system (domain). – "PKG"—Uses the package to which the parameter belongs. <p>parameter: (required) Can be passed in external or internal format. Identifies the name or internal entry number (IEN) of the parameter as defined in the PARAMETER DEFINITION file (#8989.51).</p> <p>instance: (optional) Defaults to 1 if not passed. Can be passed in external or internal format. Internal format requires that the value be preceded by the grave accent (`) character.</p> <p>value: (required) Can be passed in external or internal format. If using internal format for a pointer type parameter, the value must be preceded by the accent grave (`) character.</p> <p>If the value is being assigned to a word-processing parameter, the text can be passed in the subordinate nodes of Value (e.g., Value(1,0)=Text) and the variable "Value" itself can be</p>

defined as a title or description of the text.

Output Parameter .error: (optional) If used, *must* be passed in by reference. It returns any error condition that may occur:

- 0 (Zero)—If no error occurs.
- #^errortext—If an error does occur.

The "#" is the number in the VA FileMan DIALOG file (#.84) and the "errortext" describes the error.

Example

```
>D EN^XPAR("SYS","XPAR TEST FREE TEXT",0,"Good times",.ERROR)
>D EN^XPAR("SYS","XPAR TEST FREE TEXT",1,"to night",.ERROR)
```

ENVAL^XPAR(): Return All Parameter Instances

Reference Type Supported

Category Toolkit—Parameter Tools

IA # 2263

Description This API can be called to return all parameter instances.



REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.

Format ENVAL^XPAR(.list,parameter,instance[,.error][,gbl])

Input/Output Parameter .list (required) If the gbl parameter is set to 1, then the .list parameter becomes an input and holds the closed root of a global where the GETLST^XPAR(): Return All Instances of a Parameter API should put the output. For example:

```
$NA(^TMP($J,"XPAR"))
```

Input Parameters parameter: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

instance: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

gbl: (optional) If this optional parameter is set to 1, then the parameter ".list" *must* be set before the call to the closed global root where the return data should be put. For example:

```
S LIST=$NA(^TMP($J))  
ENVAL^XPAR(LIST,par,inst,.error,1
```

If this optional variable is set to 1. Then the parameter List must be set before the call to the closed global root where the return data should be put. For example:

```
GETLST^XPAR($NA(^TMP($J)),ent,par,fmt,.error,1)
```

**Output
Parameter**

.error:

(optional) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

\$\$GET^XPAR(): Return an Instance of a Parameter**Reference Type** Supported**Category** Toolkit—Parameter Tools**IA #** 2263**Description** This extrinsic function retrieves the value of a parameter. The value is returned from this call in the format defined by the input parameter named "format."**REF:** For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.**Format** \$\$GET^XPAR(entity,parameter,instance,format)**Input Parameters** entity: (required) Entity is defined as the single entity or group of entities you want to look at in order to retrieve the value. Entities may be passed in internal or external format (e.g., LOC.PULMONARY or LOC.'57 or 57;SC()). The list of entities in this variable may be defined as follows:

- A single entity to look at (e.g., LOC.PULMONARY).
- The word "ALL" which will tell the utility to look for values assigned to the parameter using the entity precedence defined in the PARAMETER DEFINITION file (#8989.51).
- A list of entities you want to search (e.g., "USR^LOC^SYS^PKG"). The list is searched from left to right with the first value found returned.

Items 2 or 3 with specific entity values referenced such as:

- ALL^LOC.PULMONARY—To look at the defined entity precedence, but when looking at location, only look at the PULMONARY location.
- USR^LOC.PULMONARY^SYS^PKG—To look for values for all current user, PULMONARY location, system, or package).



parameter: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

instance: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

format: (required) Format determines how the value is returned. It can be set to the following:

- "I" - Internal, returns internal value.
- "Q" - returns the value in the quickest manner - internal format.
- "E" - returns external value.
- "B" - returns internal^external value.

GETLST^XPAR(): Return All Instances of a Parameter

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2263
Description	<p>This API is similar to the ENVAL^XPAR(): Return All Parameter Instances API; however, it returns <i>all</i> instances of a parameter.</p> <p> REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.</p>
Format	GETLST^XPAR(.list,entity,parameter,format[,.error][,gbl])
Input/Output Parameter	<p>.list: (required) The array passed as List will be returned with all of the possible values assigned to the parameter.</p> <p> REF: To see how this data can be returned, please refer to the "format" parameter description below.</p> <p>If the gbl parameter is set to 1, then the .list parameter becomes an input and holds the closed root of a global where the GETLST^XPAR(): Return All Instances of a Parameter API should put the output [i.e., \$NA(^TMP(\$J,"XPAR"))].</p>
Input Parameters	<p>entity: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.</p> <p>parameter: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.</p> <p>instance: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.</p> <p>format: (required) For a description of this parameter, please refer to the \$\$GET^XPAR(): Return an Instance of a Parameter API.</p> <p>gbl: (optional) If this optional variable is set to 1. Then the parameter ".list" must be set before the call to the closed global root where the return data should be put. For example:</p> <pre style="margin-left: 40px;">GETLST^XPAR(\$NA(^TMP(\$J)),ent,par,fmt,.error,1)</pre>
Output Parameter	<p>.error: (optional) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.</p>

Example:

```
>D GETLST^XPAR(.LIST,"SYS","XPAR TEST MULTI FREE TEXT",,.ERROR)
```

GETWP^XPAR(): Return Word-processing Text

Reference Type Supported

Category Toolkit—Parameter Tools

IA # 2263

Description This API returns word-processing text in the returnedtext parameter. The returnedtext parameter itself contains the value field, which is free text that may contain a title, description, etc. The word-processing text is returned in returnedtext(#,0).



REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.

Format GETWP^XPAR(returnedtext,entity,parameter[,instance][,.error])

Input/Output Parameter .returnedtext (required) This parameter is defined as the name of an array in which you want the text returned. The .returnedtext parameter is set to the title, description, etc. The actual word-processing text will be returned in returnedtext(#,0). For example:

```
>returnedtext="Select Notes Help"
>returnedtext(1,0)="To select a progress note from
the list, "
>returnedtext(2,0)="click on the date/title of the
note."
```

Input Parameters entity: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

parameter: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.


instance: (optional) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

Output Parameter .error (optional) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

Example:

```
>D GETWP^XPAR(.X,"PKG","ORW HELP","1stNotes",.ERROR)
```


NDEL^XPAR(): Delete All Instances of a Parameter

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2263
Description	<p>This API can be called to delete the value for all instances of a parameter for a given entity.</p> <p> REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.</p>
Format	NDEL^XPAR(entity,parameter[,.error])
Input/Output Parameters	For the definition of the input and output parameters used in this API, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

Example

```
>D NDEL^XPAR("SYS","XPAR TEST MULTI FREE TEXT",.ERROR)
```


PUT^XPAR(): Add/Update Parameter Instance

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2263
Description	<p>This API can be called to add or update a parameter instance and bypass the input transforms.</p> <p> REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.</p>
Format	PUT^XPAR(entity,parameter[,instance],value[,.error])
Input/Output Parameters	For the definition of the input and output parameters used in this API, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

Example:

```
>D PUT^XPAR("SYS","XPAR TEST MULTI FREE TEXT",0,"Good times",.ERROR)
```

REP^XPAR(): Replace Instance Value

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2263
Description	<p>This API can be called to replace the value of an instance with another value.</p> <p> REF: For descriptive information about the elements and how they are used in the callable entry points into XPAR, please refer to Chapter 1, "User Manual Information" in this manual.</p>
Format	REP^XPAR(entity,parameter,currentinstance,newinstance[,.error])
Input Parameters	<p>entity: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.</p> <p>parameter: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.</p> <p>currentinstance: (required) The instance for which the value is currently defined.</p> <p>newinstance: (required) The instance to which you want to assign the value that is currently assigned to currentinstance.</p>
Output Parameter	.error: (optional) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

Application Program Interfaces (APIs)—^XPAREEDIT Routine

The following is a list of APIs available in the ^XPAREEDIT routine. The calls are supported for use with the Parameter Tools and are part of the Parameter Tools component of Kernel Toolkit. These calls contain some additional utilities for editing parameters and are covered by IA #2336. (See IA #2263 for the main XPAR entry points to this module.)

BLDLST^XPAREEDIT(): Return All Entities of a Parameter

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2336
Description	This API returns in the array "list" all entities allowed for the input parameter named "parameter."
Format	BLDLST^XPAREEDIT(.list,parameter)
Input Parameters	.list: (required) Name of array to receive output. parameter: (required) Internal Entry Number (IEN) of entry in the PARAMETER DEFINITION file (#8989.51).
Output Parameter	.list: The array passed as "list" is returned with all of the possible values assigned to the parameter. Data is returned in the list(ent,inst)=val format.

EDIT^XPAREEDIT(): Edit Instance and Value of a Parameter


Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2336
Description	This API interactively edits the instance (if multiple instances are allowed) and the value for a parameter associated with a given entity.
Format	EDIT^XPAREEDIT(entity,parameter)

Make sure to perform the following steps before calling this API:

- NEW all variables.
- Set all input variables.

- Call the API.

If you do not follow these steps, the variables could unintentionally assume the values of the variables of the current running task.

Input Parameters	entity:	(required) Identifies the specific entity for which a parameter may be edited. The entity <i>must</i> be in variable pointer format.
	parameter:	(required) Identifies the parameter that should be edited. Parameter should contain two pieces: IEN^DisplayNameOfParameter
Output Parameters	.LIST:	The array passed as "list" is returned with all of the possible values assigned to the parameter.
		 REF: For a description of this parameter, please refer to the "format" parameter in the ENVAL^XPAR(): Return All Parameter Instances API.
	.error	(optional) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.

EDITPAR^XPAREEDIT(): Edit Single Parameter

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2336
Description	This API is used to edit a single parameter.
Format	EDITPAR^XPAREEDIT(parameter)
Input Parameter	parameter: (required) For a description of this parameter, please refer to the EN^XPAR(): Add, Change, Delete Parameters API.
Output	Returns requested parameter.

EN^XPAREEDIT(): Parameter Edit Prompt

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2336
Description	This API is called to prompt the user for a parameter to edit. This is provided as a tool for developers and <i>is not intended for exported calls</i> as it allows editing of <i>any</i> parameter.
Format	EN^XPAREEDIT
Input Parameter	none
Output	none

GETENT^XPAREEDIT(): Prompt for Entity Based on Parameter

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2336
Description	This API interactively prompts for an entity, based on the definition of a parameter.
Format	GETENT^XPAREEDIT(.entity,parameter[,.onlyone?])
Output	.entity (required) Returns the selected entity in variable pointer format.
Input Parameter	parameter: (required) Specifies the parameter for which an entity should be selected. Parameter should contain two pieces: IEN^DisplayNameOfParameter
Output Parameter	onlyone? (optional) Returns "1" if there is only one possible entity for the value. For example: <ul style="list-style-type: none"> • 1—If the parameter can only be set for the system, onlyone? • 0—If the parameter could be set for any location, onlyone?

GETPAR^XPAREEDIT(): Select Parameter Definition File

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2336
Description	This API allows the user to select the PARAMETER DEFINITION file (#8989.51) entry.
Format	GETPAR^XPAREEDIT(.variable)

Make sure to perform the following steps before calling this API:

- NEW all variables.
- Set all input variables.
- Call the API.

If you do not follow these steps, the variables could unintentionally assume the values of the variables of the current running task.

Input Parameter	.variable:	(required) The name of the variable where data is returned.
Output Variable	.OUTPUTVALU:	Returns the value Y in standard DIC lookup format.

TED^XPAREdit(): Edit Template Parameters (No Dash Dividers)

Reference Type	Supported
Category	Toolkit—Parameter Tools
IA #	2336
Description	<p>This API allows editing of parameters defined in a template. The parameters in the template are prompted in VA FileMan style—prompt by prompt. No dashed line dividers are displayed between each parameter.</p> <p>Since the dashed line headers are suppressed, it is important to define the VALUE TERM for each parameter in the template, as this is what is used to prompt for the value.</p>
Format	TED^XPAREdit(template[,reviewflags][,allentities])
Input Parameters	<p>template: (required) The Internal Entry Number (IEN) or NAME of an entry in the PARAMETER TEMPLATE file (#8989.52).</p> <p>reviewflags: (optional) There are two flags (A and B) that can be used individually, together, or not at all:</p> <ul style="list-style-type: none"> • A—Indicates that the new values for the parameters in the template are displayed <i>after</i> the prompting is done. • B—Indicates that the current values of the parameters are displayed <i>before</i> editing. <p>allentities: (optional) This is a variable pointer that should be used as the entity for all parameters in the template. If left blank, prompting for the entity is done as defined in the PARAMETER TEMPLATE file (#8989.52).</p>

TEDH^XPAREEDIT(): Edit Template Parameters (With Dash Dividers)

Reference Type	Supported	
Category	Toolkit—Parameter Tools	
IA #	2336	
Description	<p>This API is similar to the TED^XPAREEDIT(): Edit Template Parameters (No Dash Dividers) API except that the dashed line headers <i>are</i> shown between each parameter.</p> <p>It allows editing of parameters defined in a template. The parameters in the template are prompted in VA FileMan style—prompt by prompt.</p>	
Format	TEDH^XPAREEDIT(template[,reviewflags][,allentities])	
Input Parameters	template	(required) For a description of this parameter, please refer to the TED^XPAREEDIT(): Edit Template Parameters (No Dash Dividers) API.
	reviewflags	(optional) For a description of this parameter, please refer to the TED^XPAREEDIT(): Edit Template Parameters (No Dash Dividers) API.
	allentities	(optional) For a description of this parameter, please refer to the TED^XPAREEDIT(): Edit Template Parameters (No Dash Dividers) API.
Output	none	

Index

\$

\$\$GET^XPAR, 2-7

A

ADD^XPAR, 2-1

Assumptions About the Reader, x

B

Background, 1-14

BLDLST^XPAREDIT, 2-13

C

Callout Boxes, ix

CHG^XPAR, 2-2

Contents, v

D

Data Dictionary

 Data Dictionary Utilities Menu, x

 Listings, x

Definitions, 1-15

DEL^XPAR, 2-2

Description, 1-15

DIALOG File (#.84), 2-5

Documentation

 Revisions, iii

 Symbols, ix

E

EDIT^XPAREDIT, 2-13

EDITPAR^XPAREDIT, 2-14

EN^XPAR, 2-4

EN^XPAREDIT, 2-15

Entity

 Definition, 1-15

ENVAL^XPAR, 2-5

Example, 1-18

F

Figures, vii

Files

August 2001

Revised: October 2008

DIALOG (#.84), 2-5

KERNEL SYSTEM PARAMETERS

 (#8989.3), 1-18

NEW PERSON (#200), 1-14, 1-15, 1-18

PARAMETER DEFINITION (#8989.51), 1-

 14, 1-16, 1-18, 1-19, 2-4, 2-7, 2-13, 2-16

PARAMETER ENTITY (#8989.518), 1-15

PARAMETER TEMPLATE (#8989.52), 1-

 17, 2-17

PARAMETERS (#8989.5), 1-14, 1-16, 1-17,

 2-1, 2-4

G

GETENT^XPAREDIT, 2-15

GETLST^XPAR, 2-9

GETPAR^XPAREDIT, 2-16

GETWP^XPAR, 2-10

H

Help

 At Prompts, x

 Online, x

 Question Marks, x

Home Pages

 Adobe Website, xi

 Kernel Website, xi

 VHA Software Document Library (VDL)

 Website, xi

 VistA Development Website, xi

How to

 Obtain Technical Information Online, x

 Use this Manual, ix

I

Instance

 Definition, 1-16

Introduction, 1-13

K

Kernel

 Website, xi

KERNEL SYSTEM PARAMETERS file

 (#8989.3), 1-18

L

List File Attributes Option, x

M

Menus

 Data Dictionary Utilities, x

N

NDEL^XPAR, 2-11

NEW PERSON File (#200), 1-14, 1-15, 1-18

O

Online

 Documentation, x

 Technical Information, How to Obtain, x

Options

 Data Dictionary Utilities, x

 List File Attributes, x

Orientation, ix

P

Parameter, 1-17

 Definition, 1-16

PARAMETER DEFINITION File (#8989.51),
 1-14, 1-16, 1-18, 1-19, 2-4, 2-7, 2-13, 2-16

PARAMETER ENTITY File (#8989.518), 1-15

PARAMETER TEMPLATE File (#8989.52), 1-
 17, 2-17

Parameter Tools

 Background, 1-14

 Definitions, 1-15

 Description, 1-15

 Entity Definition, 1-15

 Example, 1-18

 Instance Definition, 1-16

 Introduction, 1-13

 Parameter Definition, 1-16

 Template Definition, 1-17

 Value Definition, 1-17

 Why Would You Use?, 1-18

PARAMETERS File (#8989.5), 1-14, 1-16, 1-
 17, 2-1, 2-4

Patches

 Revisions, iii

Programmer Manual Information, 2-1

PS Anonymous Directories, xi

PUT^XPAR, 2-11

Index-2

Q

Question Mark Help, x

R

Reader, Assumptions About the, x

Reference Materials, xi

Reference Type

 Supported

 \$\$GET^XPAR, 2-7

 ADD^XPAR, 2-1

 BLDLST^XPAREDIT, 2-13

 CHG^XPAR, 2-2

 DEL^XPAR, 2-2

 EDIT^XPAREDIT, 2-13

 EDITPAR^XPAREDIT, 2-14

 EN^XPAR, 2-4

 EN^XPAREDIT, 2-15

 ENVAL^XPAR, 2-5

 GETENT^XPAREDIT, 2-15

 GETLST^XPAR, 2-9

 GETPAR^XPAREDIT, 2-16

 GETWP^XPAR, 2-10

 NDEL^XPAR, 2-11

 PUT^XPAR, 2-11

 REP^XPAR, 2-12

 TED^XPAREDIT, 2-17

 TEDH^XPAREDIT, 2-18

REP^XPAR, 2-12

Revision History, iii

 Documentation, iii

 Patches, iii

Routines

 XPAR

 APIs, 2-1

S

Symbols

 Found in the Documentation, ix

T

Table of Contents, v

Tables, vii

TED^XPAREDIT, 2-17

TEDH^XPAREDIT, 2-18

Templates

 Definition, 1-17

Toolkit

 Parameter Tools

\$\$GET^XPAR, 2-7
 ADD^XPAR, 2-1
 BLDLST^XPAREDIT, 2-13
 CHG^XPAR, 2-2
 DEL^XPAR, 2-2
 EDIT^XPAREDIT, 2-13
 EDITPAR^XPAREDIT, 2-14
 EN^XPAR, 2-4
 EN^XPAREDIT, 2-15
 ENVAL^XPAR, 2-5
 GETENT^XPAREDIT, 2-15
 GETLST^XPAR, 2-9
 GETPAR^XPAREDIT, 2-16
 GETWP^XPAR, 2-10
 NDEL^XPAR, 2-11
 PUT^XPAR, 2-11
 REP^XPAR, 2-12
 TED^XPAREDIT, 2-17
 TEDH^XPAREDIT, 2-18

U

URLs

Adobe Website, xi
 Kernel Website, xi
 VHA Software Document Library (VDL)
 Website, xi
 VistA Development Website, xi
 Use this Manual, How to, ix
 User Manual Information, 1-13

V

Value

Definition, 1-17
 VHA Software Document Library (VDL)
 Website, xi

W

Web Pages

Adobe Website, xi
 Kernel Website, xi
 VHA Software Document Library (VDL)
 Website, xi
 VistA Development Website, xi
 Why Would You Use Parameter Tools?, 1-18

X

XPAR

\$\$GET^XPAR, 2-7
 ADD^XPAR, 2-1
 CHG^XPAR, 2-2
 DEL^XPAR, 2-2
 EN^XPAR, 2-4
 ENVAL^XPAR, 2-5
 GETLST^XPAR, 2-9
 GETWP^XPAR, 2-10
 NDEL^XPAR, 2-11
 PUT^XPAR, 2-11
 REP^XPAR, 2-12

Routine

APIs, 2-1

XPAREDIT

BLDLST^XPAREDIT, 2-13
 EDIT^XPAREDIT, 2-13
 EDITPAR^XPAREDIT, 2-14
 EN^XPAREDIT, 2-15
 GETENT^XPAREDIT, 2-15
 GETPAR^XPAREDIT, 2-16
 TED^XPAREDIT, 2-17
 TEDH^XPAREDIT, 2-18

