Indian Health Service
Office of Information Technology
Resource and Patient Management System

Standards and Conventions
Developers' Handbook

DECEMBER 2006

SUMMARY OF CHANGES

Date	Location	Change
11/17/2006	1.8	**Removed** section on using the National Patch Module. No longer required by SAC.
11/17/2006	1.7	**Added** requirement to review Patient Merge Checklist to Developer's Checklist
11/17/2006	1.7.1	**Added** Patient Merge Checklist
11/30/2006	2.8	**Added** information on using ^ATXSTX to send taxonomies with your application

TABLE OF CONTENTS

1.0 Policies and Procedures

1.1 General Development Policies

1.1.1 Policy on SACC Peer Reviews

All RPMS development projects will be coordinated through the SAC Committee (SACC). Development requests, including enhancements, must be reviewed and approved by the SACC prior to submission to contracting or to development by federal staff. This includes all subsequent task orders submitted under a current contract. Developers will submit preliminary database designs to the DBA for review by the SACC. Iterations of working software that are sent to users for testing and feedback, will also be installed in the OIT testing environment for SACC review.

1.1.2 When to use the DBA

- To obtain official name space and number space for your application
- To be assigned a port number for TCP/IP interfaces
- To updates a standard tables, includes creation of new tables
- For final approval of data dictionary structures and any changes
- For final approval of APIs to be used by other RPMS applications
- To submit all integration agreements between your application and others

1.1.3 Policy on using APIs for reads and writes in other applications

To insure the accuracy and integrity of our data, there should be one and only one way to obtain or update that data and that method must be owned by the application that owns the data. See standard #2.2.12.3.1.

1.1.4 Policy on Creating PCC Visits

Two APIs have been approved for creating PCC visits. If your application is involved with patient encounters where the patient may have been checked in or duplicate visits are a problem, use the new GETVISIT^APCDAPI4 call. Otherwise continue to use ^APCDALV. The new APCDAPI4 call requires PIMS version 5.3 patch 1002. See section 2.3.3 Visit Creation API for details.

1.1.5 Policy on using parameters for IHS modifications to VA software

In compliance with the IHS initiative to converge as much as possible with the VA VistA software, all VA applications modified by IHS will use parameters in making those modifications. All parameterized modifications will then be submitted to the VA DBA to see if they can be incorporated into the VA applications. This will eliminate quite a bit of maintenance on the part of IHS. See the section 2.3.9 VA Convergence for details and examples.

1.1.6 Policy on using event drivers

In those RPMS applications where certain events may trigger actions in other applications, we recommend the application publish an "event driver." In those cases where event drivers exist, the proper method for triggering actions in other applications is for those applications to subscribe to the event driver. See section 2.5 Event Drivers for details and examples.

1.2 Principles of RPMS Data Base Design

1.2.1 Objective

The primary objective of data base design for the RPMS is to integrate patient and cost data into a single ADP system, supporting the various information needs of specific disciplines and programs, while at the same time collecting and storing a core set of health and management data that cuts across disciplines and facilities.

This integration of data infers:

- That specific information is collected only once, by the program that generates the data, and made available to all users who need access to it.
- That like information is stored in a single, integrated file, rather than being scattered in a number of discipline-specific files.

An example of the above is the Patient Registration file. Patient data is collected only once, by Health Records staff, and made available to all users who need some form of patient data. The Contract Health Services (CHS) program, Third Party Billing program, Pharmacy system, Laboratory system, Dental system and others do not have to separately collect and maintain a basic patient file, which would result in redundant data collection efforts, redundant files, and errors and inconsistencies in patient data between programs.

Moreover, the data is maintained in a central registration file, not scattered in a number of dissimilar discipline specific files. If the latter were true, each program needing patient data would need to be aware of the existence of all such files and

access all of the individual files to search for patient data. Furthermore, each time a new discipline-specific program was added to the system, all of the old programs needing patient data would need to be rewritten to access the new files.

The same principle applies to diagnoses, operations, laboratory procedures, etc. All of this data should be stored in a core set of centralized files, and not scattered in a number of discipline-specific or program-specific files. Cost accounting, third party billing and patient care programs should be able to obtain all diagnoses, operative procedures, basic visit data, etc., from the core set of files, and not have to access a myriad of different disciplinary files for relevant information. This integration requires each discipline or special program application to make sure that it's system stores core data into the appropriate integrated file. Additional, discipline-specific information, can be stored in discipline-defined files, as needed. The relationship of these sets of files is illustrated in the attached wheel diagram. (Figure 1). At the core is a standard set of IHS/VA tables shared by all systems. These include community tables, facility tables, discipline tables, tribal designations, etc.

Surrounding these tables is a core set of data files, shared by all disciplinary and programmatic groups as appropriate. Each group will feed information from its patient encounters into this core set of data.

The outer ring represents the discipline-specific program and management data, that is not in common with most other disciplines. For Contract Health Services, for instance, the outer ring would contain files on outstanding and completed CHS vouchers, commitment register balance, contract appropriation, etc.

RPMS DICTIONARY RELATIONS

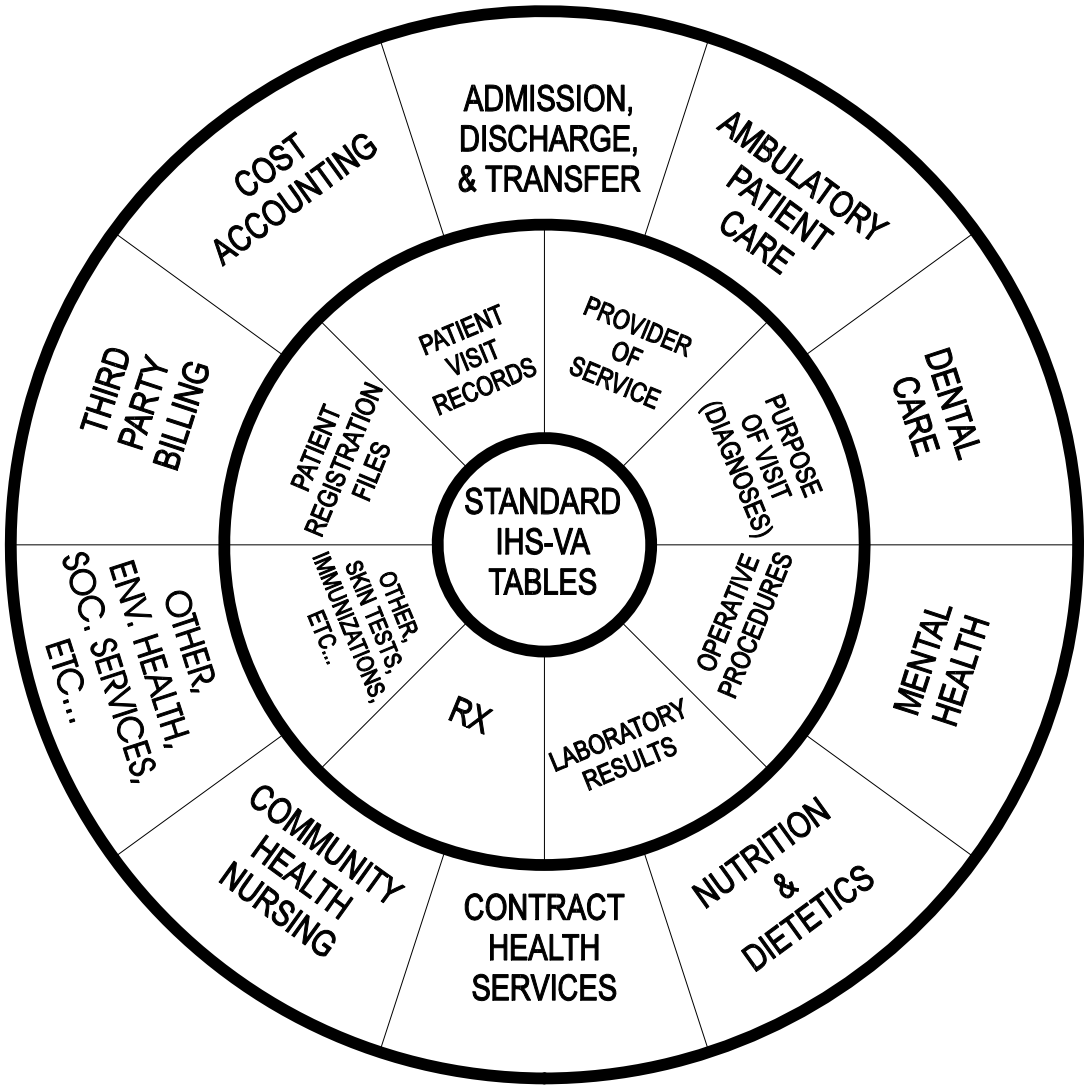


Figure 1-1: RPMS DICTIONARY RELATIONS

1.2.2 Integration with VA Data Dictionary

The IHS is committed to integrating its data dictionary with the Veteran's Administration dictionary, to assure that systems obtained from the VA will run in a compatible mode with RPMS.

In general, where the VA has a dictionary for a specific application, the IHS will utilize that dictionary. Quite often, however, the IHS needs an additional sub-set of data not contained in the VA file. If there are only two or three additional fields needed, the IHS will generally add these fields to the VA file, utilizing a high-order set of field numbers that will never collide with VA fields.

If the number of differences between IHS data needs and VA data needs exceeds three or four fields, the IHS will generally define it's own file for the data set, which will point to the VA file for name and other basic information.

An example of this latter structure is shown in the patient file itself. The IHS has its own patient file (#9000001), which in turn points to the VA Patient file (#2), where the actual patient name, sex and date of birth are stored. These two files are DINUM so that the DFN is always the same for each patient.

The relationship of the RPMS patient visit files to other RPMS and VA files is shown in Figure 1-2.

1.2.3 Use of Standard Tables

Each developer should become familiar with the RPMS standard dictionary before initiating work on a new system, and should utilize tables and files already defined to the greatest extent possible.

The following are examples of some of the tables already developed:

File	File Name	Global
5	STATE	^DIC(5
7	PROVIDER CLASS	^DIC(7
10	RACE	^DIC(10
11	MARITAL STATUS	^DIC(11
12	OCCUPATION	^DIC(12
13	RELIGION	^DIC(13
25	TYPE OF DISCHARGE	^DIC(25
71	RADIOLOGY PROCEDURES	^RAMIS(71
80	ICD DIAGNOSIS	^ICD9(
80.1	ICD OPERATION/PROCEDURE	^ICD0(
80.2	DRG	^ICD(
80.3	MAJOR DIAGNOSTIC CATEGORY	^ICM(
9999999.01	USER CONVERSION	^AUTTUCV(
9999999.02	PATIENT RECORD DISPOSITION	^AUTTDIS(
9999999.03	TRIBE	^AUTTTRI(
9999999.04	IMM MANUFACTURER	^AUTTIMAN(
9999999.05	COMMUNITY	^AUTTCOM(
9999999.06	LOCATION	^AUTTLOC(
9999999.07	MEASUREMENT TYPE	^AUTTMSR(
9999999.08	RECODE ICD/APC	^AUTTRCD(
9999999.081	APC RECODE CATEGORY	^AUTTRCDC(
9999999.09	EDUCATION TOPICS	^AUTTEDT(
9999999.11	VENDOR	^AUTTVNDR(
9999999.12	RECODE INJURY	^AUTTRIJ(

File	File Name	Global
9999999.13	LAB TEST	^AUTTLAB(
9999999.14	IMMUNIZATION	^AUTTIMM(
999999.15	EXAM	^AUTTEXAM(
9999999.16	MEDICATION	^AUTTMED(
9999999.17	TREATMENT	^AUTTTRT(
9999999.18	INSURER	^AUTNINS(
9999999.19	SURVEILLANCE CODE	^AUTTSURC(
9999999.21	AREA	^AUTTAREA(
9999999.22	SERVICE UNIT	^AUTTSU(
9999999.23	COUNTY	^AUTTCTY(
9999999.24	CLINICAL REMINDER CODE	^AUTTCRC(
9999999.25	BENEFICIARY	^AUTTBEN(
9999999.26	SERVICE CATEGORY	^AUTTSC(
9999999.27	PROVIDER NARRATIVE	^AUTNPOV(
9999999.28	SKIN TEST	^AUTTSK(
9999999.29	DISTRICT	^AUTTDST(
9999999.31	ADA CODE	^AUTTADA(
9999999.32	MEDICARE SUFFIX	^AUTTMCS(
9999999.33	RAILROAD PREFIX	^AUTTRRP(
9999999.34	VENDOR TYPE	^AUTTVTYP(
9999999.35	QUANTUM CODE	^AUTTQUAN(
9999999.36	RELATIONSHIP	^AUTTRLSH(
9999999.37	U/R DENIAL REASONS	^AUTTREAS(
9999999.38	ERROR MESSAGES	^AUTTEMSG(
9999999.39	RPMS SITE	^AUTTSITE(
9999999.41	IMMUNIZATION LOT	^AUTTIML(
9999999.42	HOUSEHOLD STATUS	^AUTTHHS(
9999999.43	EMPLOYMENT STATUS	^AUTTEMP(
9999999.44	INCOME SOURCE	^AUTTINC(
9999999.45	RPMS APPLICATION PARAMETERS	^AUTNSYS(
9999999.46	PHYSICAL THERAPY	^AUTTPHTH(
9999999.47	RPMS RESERVATION	^AUTTRES(
9999999.48	BIC ELIGIBILITY	^AUTTBICE(
9999999.49	IHS TASK REPRINT	^AUTTZTSK(
9999999.51	APPROPRIATION NO.	^AUTTPRO(
9999999.52	ALLOWANCE NO.	^AUTTALLW(
9999999.53	NO LONGER USED	
9999999.54	BUDGET ACTIVITY	^AUTTBA(
9999999.55	SUB-ACTIVITY	^AUTTSA(
9999999.56	SUB-SUB-ACTIVITY	^AUTTSSA(
9999999.57	COMMON ACCOUNTING NUMBER	^AUTTCAN(
9999999.58	COST CENTER	^AUTTCCT(
9999999.59	OBJECT/SUB-OBJECT	^AUTTOBJC(

File	File Name	Global
9999999.591	OBJECT CLASS GROUP	^AUTTOCG(
9999999.61	OBJECT CLASS CATEGORY	^AUTTOBCC(
9999999.62	FMS DEPARTMENT/PROGRAM	^AUTTPRG(
9999999.63	REFERENCE CODE	^AUTTDOCR(
9999999.64	HEALTH FACTORS	^AUTTHF(
9999999.65	COVERAGE TYPE	^AUTTPIC(
9999999.66	LOCATION CODE	^AUTTLCOD(
9999999.67	OBJECT CLASS REPORT GRP	^AUTTOCRG(
9999999.68	DIAGNOSTIC PROCEDURE RESULT	^AUTTDXPR(
9999999.69	ACCOUNTING POINT	^AUTTACPT(
9999999.71	IHS COMMUNICATIONS PARAMETERS	^AUTTTEL(
9999999.72	REVENUE CODES	^AUTTREVN(
9999999.73	NO LONGER USED	
9999999.74	CHA ICD RECODE TABLE	^AUTTCHA(
9999999.75	EMPLOYER	^AUTNEMPL(
9999999.76	TYPE OF BUSINESS	^AUTTTOB(
9999999.77	EMPLOYER GROUP INSURANCE	^AUTNEGRP(
9999999.78	SSN STATUS	^AUTTSSN(
9999999.79	GEOGRAPHICAL LOCATION	^AUTTGL(
9999999.81	SIZE OF SMALL BUSINESS	^AUTTSOB(
9999999.82	PERCENTILES	^AUTTPCT(

1.2.4 FileMan Compatible Files

All files utilized in RPMS applications will be FileMan compatible files, and FileMan protocols, (e.g., DIE or IX1^DIK), plus edits will be honored.

One of the main goals of the RPMS is to provide the tools to local facility staff to generate reports and make retrievals to satisfy their unique and constantly changing information needs. Historically, it was necessary to write a new program every time new information needs arose. Under RPMS, the VA FileMan offers a flexible, sophisticated but easy-to-use capability for local report generation.

Using DIE to update the files ensures that all necessary housekeeping and cross-references updates are accomplished. This also allows users to add additional cross-indices to files to speed up retrieval time for specific applications in local use, without having to modify the update logic.

1.2.5 Review of Proposed Data Dictionary

Each developer will define data requirements early in the design layout.

For RPMS applications, it is recommended that this file description be submitted to the DBA early in the system design phase for review and possible suggestions on integration with other RPMS applications.

1.2.6 Relationship of RPMS Patient Visit Files to Other RPMS and VA Files

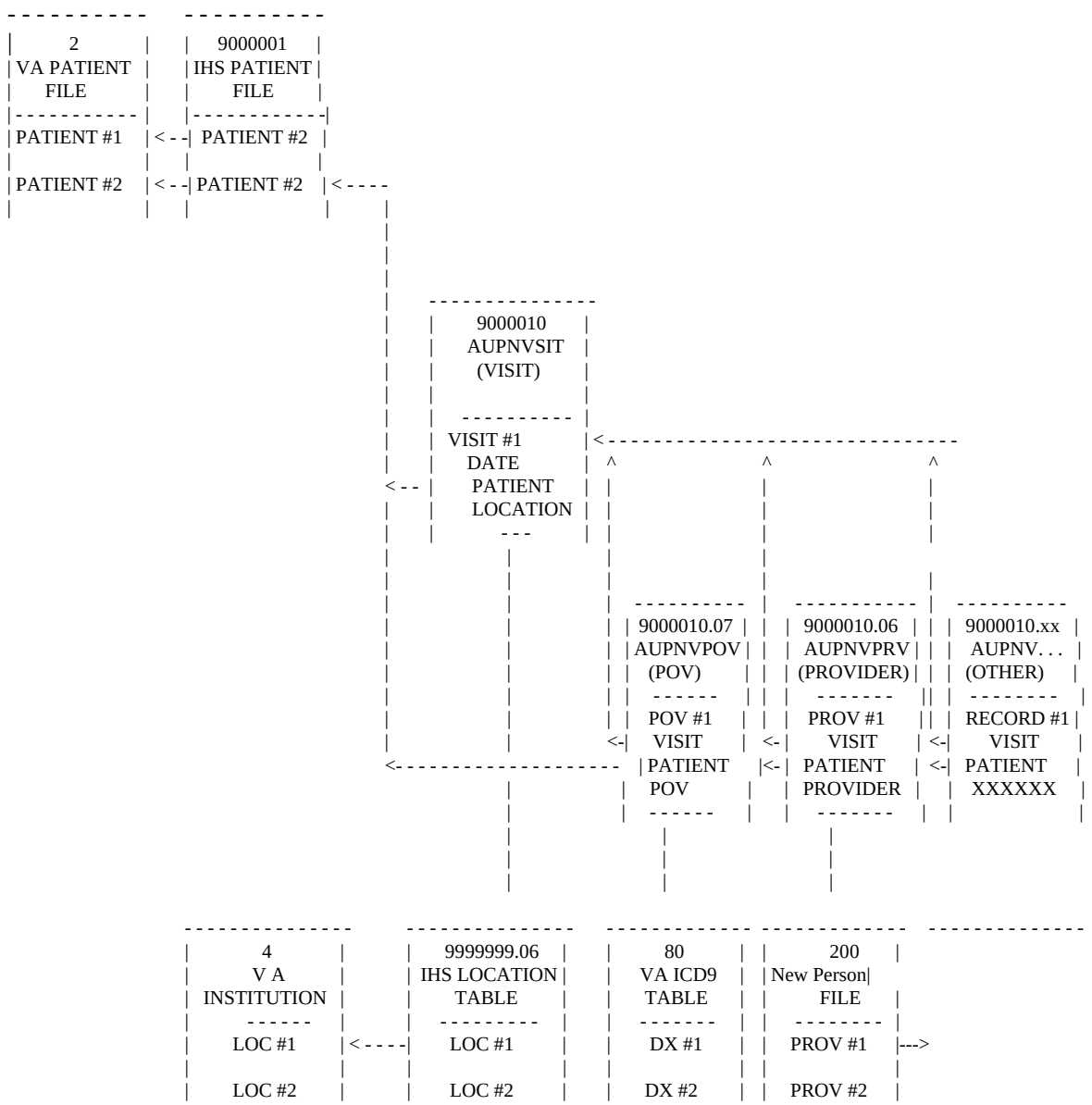


Figure 1-2

1.3 Procedure for Building Application Packages

1.3.1 Creating Packages Using KIDS Build

All RPMS packages will be distributed using the VA Kernel Installation and Distribution System (KIDS) builds. These builds will contain all M components necessary to run an application, including data dictionaries, routines, options, protocols, keys and parameters. Full documentation on how to build and use KIDS is found in the VA Systems Manual.

1.3.2 Procedure for Building Local M Packages

A designer of an RPMS application eventually will need to create an application package of all of the routines and globals to be sent to other computer sites. Package systems to be used by other members of the IHS must be completed in a standard way. The following rules must be followed, and if followed, will allow the recipients of the system to install the package without interfering with other applications running on their computer.

The Area ISC should assign a name and number space to any user that will be creating systems of any kind. Each Area has been assigned a range of numbers that will assure uniqueness and compatibility with other users of RPMS-developed software. This is done to protect systems within the Area even if there is no intent to send the system to other Areas.

1.4 RPMS Software Certification Policy and Guidelines

1.4.1 Purpose

The purpose of this document is to provide policy and guidelines to be followed in the certification of RPMS Software. The purpose of certification is to ensure the functional soundness and technical correctness of RPMS software and documentation.

1.4.2 Overview

1.4.2.1 Requirement for Certification

All IHS RPMS programs and packages which are to be distributed nationally throughout the IHS must first be certified by Software Quality Assurance (SQA) section of OIT/DEPM after a final SAC committee review.

1.4.2.2 Certification Process Components

The certification process consists of four components:

1. **Technical Verification:** This will be performed by a member of the SQA to ensure that the system conforms to RPMS Programming Standards and Conventions.

2. **Functional Verification:** This will be accomplished by the appropriate Professional Specialty Group (PSG), or designated project leader, in conjunction with the alpha/beta test facilities to verify that the application conforms to system requirements, operates correctly and, in general, does what the PSG has specified. Limited functional verification will be done in SQA.

3. **Design Verification:** This will be performed by the DBA and will assure that the packages are integrated with other RPMS dictionaries in accordance with the Principles of Data Base Design as outlined elsewhere in this handbook.

4. **Documentation Verification:** This will be accomplished by SQA in conjunction with the alpha/beta test phase to ensure conformance to the RPMS Documentation Standards as outlined in Appendix F.

1.4.2.3 Non-RPMS Packages

If an RPMS package conforms with the first two requirements but not the third, it will be certified for IHS distribution as a Class II package as defined in Appendix J.

1.4.2.4 Certified RPMS Packages

If a package satisfies all four components, the system will be certified as a Class I system as defined in Appendix J.

1.4.3 RPMS Software Development Process

This section defines the procedures for RPMS software development and certification. The first three phases may be sequential, or concurrent if using a rapid development methodology. The steps involved in producing RPMS software are:

1.4.3.1 Functional Requirements/Design Phase

A defined customer group (may be a PSG or national program office) conceptualizes new functionality and determines high-level conceptual design. The customer group and assigned developer then specify detailed requirements, starting with the highest value features first. Particular attention is focused on building in flexibility for future

enhancements. Statements of Work are drafted and reviewed by the SAC Committee (SACC).

1.4.3.2 Prototype/Development Phase

The developers begin creating those features with the highest value to the customer. By the time, development starts on any feature, both the developers and customers have defined enough detail using language both can understand. Design and development of the software will be in compliance with the defined IHS Standards and Conventions and policies defined in this document. The developer will begin preparing technical documentation.

1.4.3.3 Alpha Testing

The first step in testing is performed on the developer's system to test both the functionality and the installation procedure. The developer will then send the software and installation instructions to the selected alpha test site or sites. The test site will install the software verifying the accuracy of the installation instructions. Then the test site's users will test the functionality of the new features. This is an important time for feedback to the developer to refine the software in compliance with the users' needs. Additional iterations of the software will continue to be distributed to the alpha test sites until the all defined requirements are met.

1.4.3.4 Preliminary Technical Verification

Software Quality Assurance (SQA) will assign a verifier to the package to complete the Preliminary Technical Verification (PTV). Preliminary verification can be run in parallel with the final stages of the alpha test or at the initiation of the beta test phase. The developer is responsible for checking compliance with the SAC by running ^XINDEX and SAC Checker (^AZHLSC) and will correct all items noted by these utilities. The developer will forward the application using a KIDS build and all accompanying documentation to the appropriate SQA directory. Requests for Exemption to the SAC must accompany the files being submitted to SQA for preliminary review. Use only the approved form in Appendix B of the SAC. Review by the DBA and SACC are performed at this time. The PTV should be completed within 30 days from receipt of the system. The reviewer will provide written findings (electronic or letter format) to the developer for action. Upon completion of the PTV, SQA will coordinate any changes required with the developer prior to or during the initiation of the beta test.

1.4.3.5 Beta Testing

The customer group, with the assistance of the developer and OIT, will select at least two, but no more than five beta test sites. Test sites are selected based on adequate

hardware, IT support and users able and willing to test all functionality, balancing size and type of facility. The developer will provide SQA with test site information including site names and points of contact. SQA staff will coordinate the beta testing phase, prepare beta test agreements, supply test sites with a beta test checklist and begin a beta test log on the approved electronic mail system. Signed copies of the test agreement will be maintained in the SQA Verification files for each package. Beta testing will not commence until all signed agreements are received by SQA.

The purpose of the beta test is to confirm the functionality on diverse systems. The developer, with oversight by the customer group and SQA, will make modifications as needed to insure the specified level of service. As adjustments are made, the developer will send updated package files to SQA for distribution to test sites. New functionality will not be added during beta testing. At this time the customer group works with a technical writer to create a user manual while the developer finalizes the technical documentation.

When the package has remained stable for a minimum of 30 days and meets the requirements of the customer group, it is ready for final certification. The customer group will notify SQA using the formal endorsement form that the system has met functional requirements and is ready for release. In addition, all beta test sites will complete the beta test checklist and return it to SQA.

1.4.3.6 Final Verification & Release

Once SQA receives formal endorsement from the customer group, a final technical verification is performed. This is to insure that modifications made during beta testing did not affect SAC compliance. Final verification of technical and user manuals is also performed. Once the application has passed final verification, a release letter is written, signed and released. The software is placed in the appropriate distribution directory and released to the field via Area Offices.

1.4.3.7 Software Maintenance & Future Modifications

Software errors reported by sites will be addressed by the developer and incorporated into future patches. The customer group periodically will define enhancements desired in the certified system. These will be made by the developer and incorporated into either a future patch or a new version of the system. The decision whether to include enhancements into a patch or a new version rests with the customer group and OIT via a SACC review.

1.4.3.8 New Version Certification

The new release will be sent to the SQA for re-certification, where it will follow the same procedures as a system in Beta testing.

1.4.3.9 Patch Certification

Patches follow the same defined development steps as new applications and new versions except for preliminary technical verification and beta testing. The patch testing phase is complete after 2 weeks with no changes or as defined in the Statement of Work. Once an adequate number of test sites have certified that the patch works, the developer submits the software to SQA.

1.4.4 Technical Verification Procedure

1.4.4.1 Objectives of Technical Verification

It is not the purpose of this verification to ensure that the program accomplishes its design objectives, since this will be done by the functional verification of the system. The objectives of technical verification are basically to ensure the RPMS programs distributed throughout IHS meet the following requirements:

1. **Correctness:** That there are no obvious errors in functionality, system design, or programming methodology.

2. **Standards and Conventions:** That appropriate SAC are followed to facilitate maintenance and program readability, and to provide consistency across RPMS programs.

3. **Documentation:** That adequate user and technical documentation has been developed for the system.

1.4.4.2 When Verification is Required

1. **National Distribution:** Verification is required for any RPMS program developed by the IHS that is to be distributed nationally from the SQA, or which is defined as a part of the RPMS "family" of systems.

2. **Local Use:** Verification is not required for programs intended for local use only. However, it would be prudent to follow the RPMS SAC even for local development, since these standards are designed primarily to protect systems that run in the same environment, and to facilitate program maintenance.

    - Local systems may be distributed informally to other Areas without verification, but the receiver of the system should be forewarned that the unverified application may interfere with other approved application systems running on their computer(s) either now or at some time in the future.

    - A locally developed system may be submitted for verification at any time by following the guidelines described in this manual.

3. **VA Software:** VA programs will not be verified by the SQA since they have already undergone a certification by the VA. However, any modules or changes added by the IHS will be verified.

1.4.5 Functional Verification Procedure

1.4.5.1 Objectives of Functional Verification

The objectives of the functional verification of a system are to:

1. Ensure that the system performs according to the design criteria,
2. Ensure that the system is easy to understand and use by an end user,
3. Ensure that the system is documented, and has a readable user manual

1.4.5.2 When Verification is Required

A functional verification of a system is performed by the system designer and designated test sites during an alpha test of the system and again during the beta test of the system.

1.4.5.3 Definition of Alpha/Beta Test

Alpha Test

An alpha test of a system is the first comprehensive check-out of the system by someone other than the development staff. It is usually performed by end users in an operational setting in the developer's Area, but could be a non-operational "laboratory" test by PSG members at the development site. In either case, the test should be coordinated and monitored by the PSG.

Beta Test

A beta test is the second formal review of a system, after all of the problems identified in the alpha test have been corrected. A beta test is definitely in an operational setting, and involves day-to-day use by end users. Preferably, it will be accomplished in an Area outside of the development Area, but this is not mandatory. The PSG is responsible for overseeing the pilot operation, and verifying that the system accomplishes all design objectives in an acceptable fashion.

1.4.5.4 Functional Verification of Program

Every option of the program must be thoroughly and accurately tested using a logical testing sequence to ensure completeness. The PSG will prepare a written test plan before alpha and/or beta test, to follow during the system check-out. Upon completion of the beta test phase, the PSG/test site will complete the formal endorsement form and beta checklist certifying that the package has been fully tested and is ready for release.

Beta testing involves not only a formal check-out as described in this document, but also routine day-to-day operation of the system by end users for some period of time, usually 30 days at a minimum.

Formal Functional Verification Plan
The formal functional verification plan should ensure the following.

1. Each option should function according to PSG design criteria, and must receive PSG endorsement.

2. The software must do what the documentation says it does. There should be no inconsistencies between performance and documentation.

3. If there are nuances in the way the software works, they should be documented, either in the software itself or in the User Manual.

4. If there are assumptions made that are not readily apparent, they should be documented.

5. The style of the package, in screen formats, data input, and option selection, should be consistent and unambiguous.

6. No unwarranted errors should be generated while using any of the options in any possible way.

7. The system should never "hang up" in an error condition, but should display the error message and return the user to an option selection.

8. Response times must be acceptable from the end-user standpoint.

9. "Help" prompts should be meaningful and provide examples when appropriate.

1.4.5.5 Sample Beta Checklist

RPMS BETA TEST CHECKLIST

Please fill in the blanks or circle appropriate response as necessary.

DATE: _____

BETA SITE NAME: _____

COMPUTER: _____ OPERATING SYSTEM: UNIX DOS

PACKAGE NAME & VERSION: _____

SIZE OF DATABASE WHERE INSTALLED: _____ INSTALL TIME: _____

==========

Check ONLY the items that were FUNCTIONALLY reviewed during the beta test.

PROBLEMS OR ERRORS ENCOUNTERED DURING THE BETA TESTING OF ANY PACKAGE MUST BE REPORTED IMMEDIATELY USING THE BETA TEST LOG E-MAIL OR VIA TELEPHONE CONTACT THROUGH THE RESPONSIBLE AREA PERSONNEL TO THE DEVELOPER OF THE PACKAGE.

Review installation notes file before installation	
Ascertain that REQUIREMENTS in Notes file is met on target system	
Review all accompanying manuals	
Execute all menu options	
Review all help prompts for clarity and usefulness	
Execute all field prompts	
Test escapes at all prompts ("^")	
Test jumping capabilities at all menu options, prompts ("^Mnemonic")	
Manipulate all site parameters in various combinations	
Test various access levels of users (use the various security keys)	
Perform all available data entry functions	
Check that all fields are storing the appropriate data	
Test all reporting functionality of the package for accuracy of results and presentation	
Review data extractions for integrity of data	
Ascertain functionality of all bulletins in package, if applicable	
If an audit is included, test if functioning as intended	
Schedule all background tasks and determine if appropriately functioning and correct results	
Check performance of batch transmissions, if applicable	
Review hooks and integration of data with other packages, if applicable	

Test printer functionality where indicated, i.e., slave, queuing, direct printing, screen printing	
Report all problems encountered through proper channels to the developer	
Other items not listed, specify	

CHECKLIST COMPLETED BY:_____DATE:_____

Please provide any comments you may have on any area of testing the noted package.

_____

_____

Thanks for participating in the BETA testing process. Your assistance helps assure quality software for all of IHS.

At the end of the BETA testing phase, please return this form and your BETA test endorsement form to SQA, Headquarters West, 5300 Homestead NE, Albuquerque, NM 87110. A FAX copy is acceptable

GUIDELINES FOR BETA TEST SITES

The following are items that the test site should evaluate while performing the BETA Testing. All concerns should be reported via proper channels to the responsible developer as noted.

Application package validation is complete when the PSG/Program Manager has determined and certifies that the system is correct based on the criteria contained in its requirements document.

- Installation steps - clear and concise, accurate. Pre- and post-init requirements are clear. Can they be simplified for the installer?
- All package requirements are noted.
- Resource requirements are noted, if applicable.
- User Manuals are clear, accurately present the screen displays, are useful and provide clear presentation.
- Technical Manuals are useful to the installer/support personnel.
- Screen displays are logical and well presented.
- Help frames, prompts or on-line documentation are accurate and useful.
- Menu options/prompts perform appropriately.
- Background tasks perform accurately.
- Data integrity is maintained.
- Data transmission performs as required.
- Security is sufficient.
- Devices function appropriately.
- All reports produce the desired results.
- Any system configuration/management concerns are appropriately addressed.
- Global management is clearly defined.

- Unnecessary routines or components are noted for deletion to assist with system management.
- Additional functionality to an existing package is clearly stated.
- Site configurable items are clearly defined.
- All information necessary to install and support the package was provided with package.

1.4.6 Documentation Verification

This process will involve reviewing all documentation for:

- thoroughness,
- accuracy,
- readability,
- functionality of option descriptions and examples, and
- adherence to RPMS Documentation Standards for RPMS Software.

1.4.6.1 Minimum Documentation Required

- User Manual
- Technical Manual
- Release Notes (Required for major revisions of existing systems) identifying all enhancements and changes made to the system since the last verified version
- Installation Manual (or Notes file for patches)

1.4.6.2 Preparation of Documentation

All documentation that is to accompany a RPMS package must be prepared in electronic form as outlined in the RPMS Documentation Standards.

1.4.6.3 Review of User Manual

The User Manual will be evaluated for the following:

- all options are fully described;
- all options should have appropriate examples;
- the format should be readable, complete and easily understood and can be followed by a novice user;
- the manual should be organized in a logical manner;
- all prompts described in the manual should match what is actually on the screen; and
- rules of grammar and correct spelling should be followed.

1.4.6.4 Review of Technical Manual

The Technical Manual will be evaluated for the following:

- that content meets minimum requirements as set forth in the RPMS Documentation Standards
- and that items required to be noted by a RPMS SAC are accurately documented.

1.4.6.5 Review of Documentation

The SQA will complete the documentation review and provide a report of their findings in writing by electronic mail, or by letter to the responsible developer/ preparer of the manual. Unresolved issues will be handled as outlined in the RPMS Documentation Standards.

1.4.6.6 Procedure for Package Installation Guide and Patch Notes

1.4.6.6.1 RPMS Application Installation Guide

All RPMS software must have an Installation Guide prepared as outlined in Appendix F of the 2005 RPMS SAC. This Guide is to be prepared using PDF format. In general, developers are encourage to try NOT to instruct the user to "see the manual" for information, as the manual is generally not available during installations.

1.4.6.6.2 Patch Notes

All patches will have a notes file. When writing the notes file, developers should use the sample template noted in section 1.4.6.6.3. This sample represents the minimum items required in all notes files. In general, developers are encouraged to NOT instruct the user to "see the manual" for information, as the manual is generally not available during installation.

1.4.6.6.3 Sample Patch Notes

```
INSTALLATION NOTES FOR (Application Package Name)
================================================================
PREFIX:   {Prefix as found in the PACKAGE file}
CURRENT VERSION: {Version}   {Patch number if a patch}
================================================================

*****NOTE*****NOTE*****NOTE*****NOTE*****NOTE*****NOTE*****NOTE*
*********************************************************************
*       Read the Entire Notes File Prior to Attempting Any Installation !!!          *
```

****
*****NOTE*****NOTE*****NOTE*****NOTE*****NOTE*****NOTE*****NOTE*

1. GENERAL INFORMATION

a) Print all notes/readme files.

b) It is recommended that the terminal output during the installation be captured using an auxport printer attached to the terminal at which you are performing the software installation. This will assure a printed audit trail if any problems should arise.

2. CONTENTS OF DISTRIBUTION

a) <KIDS_file> - Routines
b) <notes_file> - This file

c) <List any other files>

3. REQUIREMENTS

a) Kernel V 8.0 or later

b) FileMan V 22 or later

c) {List any packages, including version, required to install and/or run this package}

4. INSTALLATION INSTRUCTIONS

In all UCI's running {application name}:

a) Disable logins or ensure all users are off, or disable [prefix]MENU.

b) Do routine save and global save of <old routines/globals>.
(Note: Remove these saves from your system 7-10 days after this installation if no problems noted with new install)

c) {Delete the routines [prefix]*.}

d) Load the Distribution into KIDS using option 1 on the KIDS Installation menu. The distribution was released in a file named <KIDS file name>.

e) Verify the load using option 2 on the KIDS Installation menu.

f) Consider using options 3 and 4 on the KIDS Installation menu to print and compare the Transport global.

g) Install the distribution using option 6 on the Installation menu. {Include any special instructions on answering KIDS questions.}

h) Assign option [prefix]MENU and security key [prefix]ZMENU.

i) {Assign other menus and keys to designated users }

i) Enable logins and/or [prefix]MENU.

5. POINT OF CONTACT

OIT CUSTOMER SUPPORT CENTER
PHONE: (505) 248-4371
(888) 830-7280
FAX: (505) 248-4199
Web: http://www.rpms.ihs.gov/TechSupp.asp
EMAIL: ITSCHelp@ihs.hhs.gov

1.4.7 Design Verification Procedure

1.4.7.1 Responsibility

The DBA will review the design of the system concurrently with Technical Verification to determine whether it is fully integrated with other RPMS data dictionaries. The section on Data Base Design in this handbook discusses data base design criteria for RPMS applications. These requirements are:

- If the application generates patient-specific health data, this data will be integrated into the core set of patient care dictionaries;
- Whether dealing with patient health data or not, the application should integrate with other pre-existing RPMS dictionaries as appropriate;
- The application should use standard IHS tables where appropriate, rather than create new tables.

1.4.7.2 Results of Review

Depending on the results of the design verification, the system will either be certified or returned to the developer for conformance to the RPMS design principles.

1.4.8 Certification and Release of RPMS Software

1.4.8.1 Items Required for Certification

1. Receipt of signed PSG Endorsement Form and Beta Test Checklist
2. Successful completion of final verification
3. Successful completion of beta test phase
4. Relevant data base design criteria are satisfied
5. Final versions of software and documentation

1.4.8.2 National Release

If the software satisfies all outlined requirements, it will be submitted for final certification and national release. If the software does not meet all the outlined requirements it will be returned to the developer for rewrite or redesign. Any software released to users that is not classified as certified will not be supported by or maintained by OIT.

1. **Responsibility:** The SQA is responsible for the coordination and release of RPMS packages and will maintain a central file of all systems that have been certified.

2. **Release Letter:** A release letter signed by appropriate officials will be prepared and distributed through appropriate channels announcing the availability of the newly certified and released software. A copy of the PSG endorsement(s) will be attached to the release letter.

3. **Certified Software:** When all concurrences are in place, the SQA will make the newly certified software available for Area ISC personnel. A national release e-mail message will be generated in addition to the written Release Letter announcing the package release.

1.5 Procedure for RPMS Documentation

1.5.1 Purpose

To establish policy for all IHS RPMS software package documentation. It may also apply to package documentation for non-RPMS automated information systems. This section defines package documentation and management of documentation. RPMS documentation standards (Appendix F) have been established to:

- Provide a basic documentation structure that can be applied to every RPMS national package. For non-RPMS automated information systems software packages, documentation must be available to support the product.

- Provide consistency in all RPMS documentation.
- Provide criteria by which documentation of an RPMS package can be verified.
- Ensure the highest standard of documentation in order to achieve the goal of providing optimal information to the targeted audiences.

1.5.2 Policy

1.5.2.1 RPMS Packages

1. Electronic copy documentation shall be provided for all RPMS national packages.

2. Package documentation shall be provided in sufficient detail for users and local site manager to install, operate, and manage the package. Documentation must provide clear, understandable materials that serve the software package users.

3. Package documentation shall comply with the RPMS documentation standards.

4. Documentation shall reference the availability of on-line tools (e.g., help frames).

5. The SQA, OIT shall review all nationally-developed RPMS software packages for compliance with the documentation standards.

1.6 Classification of RPMS Software

1.6.1 Purpose

The purpose of this appendix is to provide policy and guidelines to be followed in the classification of RPMS software. The purpose of classification is to distinguish, for users, the different levels of software and the level of support they can expect from the OIT.

1.6.2 Overview

Classes: Within the RPMS, there are two classes of software based on whether the application has been certified by the RPMS/CMB, and whether the data base has been integrated into the RPMS Data Dictionary.

Software Classes

	Class I	Class II
Certified by RPMS/CMB	Yes	Yes
Integrated with RPMS Data Dictionary	Yes	No

1.6.3 Certified Attributes

A system certified by the SQA (Class I and Class II) has the following attributes.

Technical Verification: The system has received technical verification. This verification ensures that the system has been written in accordance with RPMS Programming SAC.

Functional Verification: The system has received functional verification by the appropriate PSG, or in the absence of a PSG, by a user group outside of the developer's Area.

Support: The system will be supported by OIT.

Distribution: The system is available for IHS national distribution as part of the RPMS.

1.6.4 RPMS Data Dictionary Integration

A system which has been integrated with the RPMS Data Dictionary is one which meets the following criteria.

Certification: The system is certified by the SQA both functionally and technically.

Standard RPMS Tables: The system uses standard RPMS tables where appropriate, as opposed to local tables. Examples of non-complying applications might be those that use a local provider discipline table, rather than the RPMS Provider Class Table/File; or a local Tribe table rather than the RPMS Tribe Table/File.

Core Data: The system stores RPMS core data in the appropriate RPMS file (i.e., visits are generated in the RPMS Visit File, diagnoses in the RPMS Purpose of Visit File, immunizations in the RPMS Immunization File, etc...)

1.6.5 Classification

Based upon the classification criteria, the two RPMS Classifications for software are further defined as follows.

1.6.5.1 Class I Software

- Verified technically by the SQA
- Verified functionally by a PSG or user group outside of the developer's Area
- Integrated into the RPMS Data Dictionary
- Supported by a national development center
- Available for IHS-wide distribution through
- SQA
- May or may not be mandated for implementation IHS wide

1.6.5.2 Class II Software

- Verified technically by the SQA
- Verified functionally by a PSG or user group
- Not integrated into the RPMS Data Dictionary
- May or may not be supported by a national development center
- Available for distribution through the SQA

1.6.5.3 Designation Documentation

The class designation of all distributed packages will be identified in the package documentation and will be recorded in the Package File which is distributed with the Kernel.

1.6.5.4 Additional Information

For additional information on certification procedures, programming standards, and RPMS data base design criteria, refer to the appropriate appendices of this document.

1.7 Developer's Checklist

Developer's should go over this checklist prior to submitting their packages for verification.

	Have you reviewed the Patient Merge checklist below to determine that your data dictionaries are patient merge compliant?
	Have you run the VA Cross Referencer (^XINDEX) and cleaned up any errors found?

	Have you run the IHS SAC Checker Utility (^AZHLSC) and cleaned up any pertinent errors?
	If this is not a VA package, have you run the XB Developer utilities to set the first 2 lines in your routines and to set version numbers and authorities on your data dictionaries?
	Have you submitted any requests for exemption from the SAC?
	Are the manuals ready to submit with the package?
	Have you named the KIDS build and any other transport files correctly?
	Have you tested the install of this final build?
	Have all references to test versions been removed?
	If this is a patch, have you created a notes file using the template found in section 1.4.6.6.3.

1.7.1 Patient Merge Checklist

IHS Patient Merge requires that all pointers to VA Patient (#2) and Patient (#9000001) files contain a regular full file cross-reference or a special merge routine must be written. If your data dictionaries contain any of the following variations, a special merge routine might be required. Please refer to this checklist when developing your data dictionaries and follow the instructions for each case.

Okay?	DD structure type	Procedure to comply
	Do you have any DD with fields that act like pointers to files 2 or 900001 but are not defined as pointers? (i.e. free text or number)	If so, you will need to write a special merge routine and submit it to the current IHS Patient Merge developer for inclusion before you application is released.
	Do all of your pointers to files 2 and 9000001 have regular cross-references defined and populated?	If not, you need to create them and insure that they are updated as the fields are updated. If this is not feasible, contact the current IHS Patient Merge developer.
	Do you have any variable pointer fields that include pointers to file 2 or 9000001 AND are not on the following	If so, you may need to write a special merge subroutine to be incorporated into the IHS Patient

Okay?	DD structure type	Procedure to comply
	list?	Merge ^BPMXVP routine that handles variable pointers.
	Do you have any data dictionaries with no-standard global references? Such as having DUZ(2) imbedded in the reference?	If so, you may need to write a special merge routine and submit it to the current IHS Patient Merge developer for inclusion before your application is released.
	Do you have a data dictionary that is DINUM'ed to either VA Patient or Patient files AND it contains a word-processing field?	If so, your file and field will have to be included in the special merge code in IHS Patient Merge. Otherwise, the word-processing data will be deleted by the merge process.
	Do you have a data file whose entries rely on a unique sequence number? An example is the Problem file where each problem has a unique sequence number. When merging problems these numbers must be re-sequenced. For example, if both patients have a problem #1, then one of those problems will have to be given a different and unique number.	If you have any doubts, please contact the IHS Patient Merge developer. If your file's data does not merge correctly, this could jeopardize the integrity of the patient's record.

1.8 Procedure for Submitting Patches to Certified RPMS Software (Removed – Use of National Patch Module no longer required)

1.9 Software Maintenance & Support

Once a new version of an RPMS application has been released, fixes and enhancements to the old version will not be made starting 18 months after the release date. Sites will continue to receive emergency support for the old version and will be assisted in upgrading to the current version."

1.10 Local Modifications to Data Dictionaries, Data Elements, and Routines in Class I Packages

The currently released version of an RPMS application including patches will be the standard configuration for that application. Any local modifications to routines, data dictionaries, templates, options, forms or protocols ***may*** result in the site needing to perform a new Certification & Accreditation (C&A). In any case, the minimum result of non-standard configurations would result in the need to do a risk assessment of the potential risks the change has introduced to the security of the application or system.

Modifications to core data dictionaries, or dictionaries associated with Class I software must be restricted to adding new data elements and creating input and output templates to meet specific local needs. In order to assure the capability of installing new releases of the application packages, it is important that any local additions to the database be made in areas that will not conflict with elements contained in the nationally distributed database and should be coordinated with the DBA or be made a sub package so the changes can be reinstalled after a new dictionary release.

When adding new data elements to a data dictionary, the numbering conventions used for creating new files should be used for data elements. That is, a data element number should be entered that is in the numbering range of the Area making the change (or the sub-range of numbers assigned to a specific IHS site). The same numbering convention should be applied to global subscripts for local data add-ons to previously defined globals. Sub-files numbers should be assigned in an analogous fashion, putting the number after the decimal point.

When input or output templates are incorporated into the option file, the options names should be prefixed by the namespace followed by "Z" and the letter assigned to the Area making the addition. For example, a local option called LOG for the PS package made in the Tucson Area would have the option name "PSZSLOG". This will allow a site to readily differentiate between those options developed locally and those associated with the standard package, and will prevent collisions.

Any other types of local data modifications to the core and Class I packages are strongly discouraged. If local modifications are made to existing data elements in the data dictionaries, it will then be the responsibility of the site to maintain those modifications as new versions of the package are installed. Furthermore, should a data element be modified that is used in standard external reports, it will be the responsibility of the facility to ensure that the modification will not affect the accuracy or validity of the data contained in the report.

Class I software carries with it the guarantee that the Office of Information Technology (OIT) will support and maintain that software in conjunction with staff at the sites of installation. Local modifications of Class I routines invalidates that guarantee and transfers maintenance and support responsibility from the OIT to the

modifying site. Debugging local variations of Class I software and assessing the "ripple effects" that such changes may have throughout the module are not the responsibility of the OIT.

2.0 Developers' Tools

2.1 References to VA System Utilities

2.1.1 VA Infrastructure Manuals

All the following VA infrastructure manuals can be found at ftp.va.gov/vista/vistAdocs/Infrastructure/. Several more manuals are available than are listed here.

VA Kernel Systems Manual: All RPMS developers should become very familiar with this manual. It contains documentation on several APIs as well as tools to be used in your development and testing process. Major sections with useful APIs are:

- Device Handler
- Task Manager
- KIDS (Kernel Installation and Distribution System)
- Date, Math and String Functions Library

VA Mailman Programmer Manual: Contains APIs for sending mailman messages and bulletins.

VA List Manager Developer's Guide: Contains information for building and maintaining list templates with attached protocols and accompanying APIs.

Parameter Tools (Kernel Toolkit additional document titled ktk7_3p26sp.pdf): Documents the use of parameterization which is now required for IHS modifications to VA routines.

VA FileMan Programmer Manual: Documents data dictionary structures and well as programmer APIs (both silent calls and those with built-in user interaction)

VA FileMan User Manual : Important documentation on basic FileMan functionality. All RPMS programmers must be very familiar with them.

ScreenMan Tutorial for Developers: Documents how to build and maintain data entry screens using the VA ScreenMan tool.

2.1.2 FileMan Sorting: Crib Sheet

Formatting Codes:	Examples:	Explanations:

Formatting Codes:	Examples:	Explanations:
Sort Options		
C-Column Assignment	SORT BY: Sex;C30	Print Sex sub-header in column 30
S-Skip Lines	SORT BY: Sex;S2	Skip 2 lines before printing the next sex sub-header
L-Left Justify	SORT BY: Nursing Home;L10	Print only the first 10 characters of the Nursing Home as the sub-header
" "-Print Your Header	SORT BY: Sex;"Gender"	Prints Gender: as a sub-header rather than Sex
@-Supress Sub-Header	SORT BY: @SEX	Sorts by the selected field but suppresses the Sub-Header
Sort Functions		
!-Ranking Numbers	SORT BY: !Nursing Home	Items printed under Nursing Home sub-header will have ranking numbers
+-Sub Totals	SORT BY: +Nursing Home	All print fields with !,&,+,or# will be sub-totaled at each new sorted by value
#-Form Feed	SORT BY: #Nursing Home	A form feed will be generated at each new sorted by value
- Reverse Order	SORT BY: -Days Of Care	Will reverse order of print from lowest-highest to highest-lowest order
'-Select Entries	SORT BY: 'Placement Date	Selects items only, rather than selects and sorts
Special Features		
@ at START WITH prompt	SORT BY: Days Absent START WITH DAYS ABSENT: @	Prints all entries with a value in the days of care field first followed by null values for that field
@ at the START WITH and GO TO prompt	SORT BY: Days Absent START WITH DAYS ABSENT: @ GO TO DAYS ABSENT: @	Prints only entries with null values for in the days of care field
Templates		
] forces FileMan to offer a template prompt	SORT BY:] FIRST PRINT FIELD:]	Forces FileMan to offer you a template
[used to call a template	SORT BY: [VINCE FIRST PRINT FIELD: [VINCE	Calls a previously created template sort or print template named Vince
[? will show all templates available to the user	SORT BY: [? FIRST PRINT FIELD: [?	Shows all sort or print templates
^ inserts	THEN PRINT FIELD:SEX//^SSN THEN PRINT FIELD: SEX//	Inserts a field before another field

Formatting Codes:	Examples:	Explanations:
@ deletes Note: Any changes must be re-saved.	THEN PRINT FIELD: SEX//@	Deletes a field
BOOLEAN LOGIC		
= Equal	SORT BY: NAME// SEX="MALE"	Equal finds exact matches
> Greater Than	SORT BY: NAME// DAYS ABSENT>3	Finds all entries with DAYS ABSENT values greater than three
< Less Than	SORT BY: NAME// DAYS ABSENT<3	Finds all entries with DAYS ABSENT values less than three
[Contains	SORT BY: NAME// NAME["AR"	Finds all names having the letters 'AR' in them
] Follows	SORT BY: NAME// NAME]"ST"	Finds all names starting with 'ST' to the end of the alphabet (STEVENS to TURNER)
! OR	SORT BY: NAME// NAME["AR"!(SEX=”MALE”)	Finds entries with 'AR' in the name or the sex is male
& AND	SORT BY: NAME// NAME["AR'&(SEX="MALE")	Finds entries with 'AR' in the name and the sex is male
' Apostrophe	SORT BY: NAME// NAME'["AR"	Negates any condition; Finds all entries with-out AR in the name

2.1.3 FileMan Printing: Crib Sheet

Formatting Codes:	Examples:	Explanations:
Print Options		
C-Column Assignment	FIRST PRINT FIELD: Name;C10 FIRST PRINT FIELD: Name;C-30	Print Name in column 10 Print Name 30 columns from the right margin
S-Skip Lines	FIRST PRINT FIELD: Name;S1	Skip 1 line before printing the next name
L-Left Justify	FIRST PRINT FIELD: Name;L8	Print only the first 8 characters of the name
R-Right Justify	FIRST PRINT FIELD: Name;R30	Right justify the columns from the end of the last value plus 2 column spaces
W-Word Wrap	FIRST PRINT FIELD: Text;W20	Wrap after 20 columns of text but will not split words

Formatting Codes:	Examples:	Explanations:
D-Decimal Points	FIRST PRINT FIELD: Cost;D1	Use only one decimal place
N-No Repeat	FIRST PRINT FIELD: Nursing;N	Will not repeat consecutive occurrence of the same name
Y-Start at Row	FIRST PRINT FIELD: Name;Y10	Start printing 10 rows from the top
@-Suppress Heading	FIRST PRINT FIELD: Name;Y-10	Start printing 10 rows from the bottom
X-Suppress Spacing	HEADING: CONTRA..Replace @	Suppresses the entire Heading
" "-Print Your Header	FIRST PRINT FIELD: .01	Suppress spacing THEN PRINT FIELD : SSN;X between the name and the SSN
_ Concatenate (Join)	FIRST PRINT FIELD: Sex;"S"	Prints S as a column header rather than Sex
other :-Forward Pointing	FIRST PRINT FIELD: City_", "_State FIRST PRINT FIELD: Nursing Home: THEN PRINT CONTRACT NURSING HOME FIELD:	Joins field values with literals or other fields Follows the Nursing Home pointer field to the Nursing Home file
Arithmetic Operators:		
!-Counts Any Field	FIRST PRINT FIELD: !Sex	Counts the entries that have values in the Sex field
&-Totals Numerics	FIRST PRINT FIELD: &Cost	Totals numeric fields
+-Totals,Count&Mean	FIRST PRINT FIELD: +Cost	Totals and Counts fields and provides a mean value
#-Totals,Count,Mean	FIRST PRINT FIELD: #COST	Minimum,Maximum, & Totals and Counts fields and Standard Deviation provides a minimum value and a maximum value found with the deviation
Binary Operators:		
+ Addition	FIRST PRINT FIELD: SPER DIEM RATE+10	Add 10 to the Skilled Per Diem Rate
- Subtract	FIRST PRINT FIELD: SPER DIEM RATE-10	Subtract 10 from the Skilled Diem Rate
* Multiply	FIRST PRINT FIELD: SPER DIEM RATE*30	Multiply Skilled Per Diem Rate by 30
/ Divide	FIRST PRINT FIELD: NURSING HOME COST/DAYS OF CARE	Divide the Total Nursing Home Cost by the Total Days of Care
\ Integer Division	FIRST PRINT FIELD: TODAY-DATE OF BIRTH\365.25	Divide the Age by 365.25 leaving off all remainders

2.1.4 ScreenMan Help: Crib Sheet

Cursor Movement

Move right one character	<Right>
Move left one character	<Left>
Move right one word	<Ctrl-L> or <PF1><Space>
Move left one word	<Ctrl-J>
Move to right of window	<PF1><Right>
Move to left of window	<PF1><Left>
Move to end of field	<PF1><PF1><Right>
Move to beginning of field	<PF1><PF1><Left>

Modes

Insert/Replace toggle	<PF3>
Zoom (invoke multiline editor)	<PF1>Z

Deletions

Character under cursor	<PF2> or <Delete>
Character left of cursor	<Backspace>
From cursor to end of word	<Ctrl-W>
From cursor to end of field	<PF1><PF2>
Toggle null/last edit/default	<PF1>D or <Ctrl-U>

Macro Movement

Field below	<Down>	Next page	<PF1><Down> or <PageDown>
Field above	<Up>	Previous page	<PF1><Up> or <PageUp>
Field to right	<Tab>	Next block	<PF1><PF4>
Field to left	<PF4>	Jump to a field	^caption
Pre-defined order	<Return>	Go to Command Line	^

Go into multiple or word processing field <Return>

Command Line Options (Enter '^' at any field to jump to the command line.)

Command	Shortcut	Description
EXIT	see below	Exit form (asks whether changes should be saved)
CLOSE	<PF1>C	Close window and return to previous level
SAVE	<PF1>S	Save changes
NEXT PAGE	<PF1><Down>	Go to next page
REFRESH	<PF1>R	Repaint screen

Other Shortcut Keys

Exit form and save changes	<PF1>E
Quit form without saving changes	<PF1>Q
Invoke Record Selection Page	<PF1>L

2.1.5 Full Screen Editor: Crib Sheet

Summary of Key Sequences

Navigation

Incremental movement	Arrow keys
One word left and right	<Ctrl-J> and <Ctrl-L>
Next tab stop to the right	<Tab>
Jump left and right	<PF1><Left> and <PF1><Right>
Beginning and end of line	<PF1><PF1><Left> and <PF1><PF1><Right>
Screen up or down	<PF1><Up> and <PF1><Down> or: <PrevScr> and <NextScr> or: <PageUp> and <PageDown>
Top or bottom of document	<PF1>T and <PF1>B
Go to a specific location	<PF1>G

Exiting/Saving

Exit and save text	<PF1>E
Quit without saving	<PF1>Q
Exit, save, and switch editors	<PF1>A
Save without exiting	<PF1>S

Deleting

Character before cursor	<Backspace>
Character at cursor	<PF4> or <Remove> or <Delete>
From cursor to end of word	<Ctrl-W>
From cursor to end of line	<PF1><PF2>
Entire line	<PF1>D

Settings/Modes

Wrap/nowrap mode toggle	<PF2>
Insert/replace mode toggle	<PF3>
Set/clear tab stop	<PF1><Tab>
Set left margin	<PF1>,

Set right margin	<PF1>.
Status line toggle	<PF1>?

Formatting

Join current line to next line	<PF1>J
Reformat paragraph	<PF1>R

Finding

Find text	<PF1>F or <Find>
Find next occurence of text	<PF1>N
Find/Replace text	<PF1>P

Cutting/Copying/Pasting

Select (Mark) text	<PF1>M at beginning and end of text
Deselect (Unmark) text	<PF1><PF1>M
Delete selected text	<Delete> or <Backspace> on selected text
Cut and save to buffer	<PF1>X on selected text
Copy and save to buffer	<PF1>C on selected text
Paste from buffer	<PF1>V
Move text to another location	<PF1>X at new location
Copy text to another location	<PF1>C at new location

2.2 IHS System Utilities (XB)

2.2.1 Summary / Overview

The IHS/VA Utilities are in the XB namespace for routines that are not MUMPS (M) implementation specific. Routines that are implementation specific will be in the ZIB namespace.

Programmer tools are available from programmer mode thru the menu-driver routine XB. There are no files associated with the XB/ZIB package.

2.2.2 Routine Descriptions

2.2.2.1 XB

This routine lists available utilities in the form of a menu with a brief description of what the utility does. New utilities may be added to this routine by adding the

appropriate ";;" entries to the bottom of this routine. See routine XB1 for further documentation and the menu options for the XB/ZIB Utility package.

2.2.2.2 XB1

In this routine each label represents a menu.

2.2.2.3 XBARRAY*

This utility provides a word processing format of free text and local variable references to build an array.

2.2.2.4 XBBPI

This routine builds a pre-init routine for a specified package. The pre-init routine will delete FileMan dictionaries being created by the package. Data globals and templates will be saved.

2.2.2.5 XBCLM

This routine displays a column number header followed by the passed string.

2.2.2.6 XBCLS

This routine clears the screen.

2.2.2.7 XBCNODE

This routine counts unique values in a selected global node.

2.2.2.8 XBCSPC

This routine checks selected fields to see what percent of the time it exists in the entries in a file, and if it should be unique, makes sure it is unique.

2.2.2.9 XBAD0

This routine sets the DA array from D0, D1, etc., or D0, D1, etc., from the DA array. If the variable XBDAD0=2, it sets the DA array, otherwise, it sets D0, D1, etc.

2.2.2.10 XBDATE

This routine limits routines selected by %RSEL to routines edited after some date.

2.2.2.11 XBDBQDOC

This routine contains double queuing shell handler documentation.

2.2.2.12 XBDBQUE

This utility gives the programmer a very friendly way to provide single, double, or no queuing at all to the applications. Report programming is structured into a compute routine, a print routine, and an exit routine. The call to XBDBQUE handles all the devices and host files as necessary.

- %ZIS with "PQM" is called by XBDBQUE
- The user will be asked to queue if queuing has not been selected.
- IO variables as necessary are automatically stored.
- XBDBQUE can be nested. The compute and print phases can call XBDBQUE individually (XBIOP is then required)
- The appropriate %ZTSK node is killed.

Input Variables

(Mandatory)

Either XBRC = Compute Routine
Or XBRP = Print Routine

(Optional)

XBRC = Compute Routine
XBRP = Print Routine
XBRX = Exit Routine that cleans variables (Highly Suggested)
XBNS = Namespace of variables to auto load in ZTSAVE("NS*")=""
= "DG;AUPN;PS;..." ; (will add '*' if missing)

Or

XBNS("xxx")="" Where xxx is structured as in ZTSAVE variable arrays where xxx is as described for ZTSAVE("xxxx")=""

XBFQ = 1 Force Queuing
XBDTH = FM date time of computing/printing
XBIOP = pre-selected printer device constructed with ION ; IOST ; IOSL ; IOM (mandatory if the calling routine is a queued routine itself)
XBPAR = %ZIS("IOPAR") values for host file with XBIOP, if needed

```
EX:   S XBRC="C^AGTEST",XBRP="P^AGTEST"
      S XBRX="END^AGTEST",XBNS="AG"
      D ^XBDBQUE ;handles foreground and tasking
      Q
```

2.2.2.13 XBDIQ0

Documentation routine for XBDIQ1.

2.2.2.14 XBDIQ1

This utility provides an easy pulling of data from the FM data base. The data is returned in an array and format designated by the programmer. (An enhanced EN^DIQ1)

Input Variables

The following variables are the same as for EN^DIQ1 but with friendlier results.

1. Data arrays are returned into the target array, @DIQ, in a variety of formats controlled by the setting of DIQ(0). The default return array is @DIQ(Field Number)= external value of field of the field.
2. Data retrieval is antiseptic! It does not disturb any local variables.
3. The input variable DA can either be an array or a literal of explicit values or a literal of variables.

Entry Points

ENP^XBDIQ1(DIC,DA,DR,DIQ,DIQ(0))
Returns @DIQ(FLD)= data for One Entry for fields indicated in DR

ENPM^XBDIQ1(DIC,DA,DR,DIQ,DIQ(0))
Returns @DIQ(DA,FLD)= data for Multiple Entries
DIC("S") can be set and used for screening entries

For ENPM the lowest level DA must be set to 0 (zero)

$$VAL^XBDIQ1(DIC,DA,DR)
Returns External value of one field.

$$VALI^XBDIQ1(DIC,DA,DR)
Returns Internal value of one field.

$$DIC^XBDIQ1(DIC)
Returns constructed DIC from file/subfile number

Input Variables

DIC, DR, DIQ are defined as in the call to EN^DIQ1

DIQ(0) Format Options

If DIQ(0) is not present the default is set to NULL

0 OR NULL	@DIQ(FLD)	= external data base value
1	@DIQ(DA,FLD)	= ""
2	@DIQ(DA(x),..,DA,FLD)	= ""
(1 or 2)_I	@DIQ(...,FLD,"I")	= internal data base value
(1 or 2)_N	NULL fields are not returned	

DA Can be the array DA or a literal string in descending order. The literal string may be made up of explicit values or variables.

EX: DA = "1,23,45"

Or DA = "1,PATDFN,BLDFN"

Or DA = BARVDA("EOBSUB") :: ="BAFCLDA,BARITDA,BAREDA"

For ENPM the lowest level DA must be set to 0 (zero)

2.2.2.15 XBFORM*

This utility provides two entry points. The first is the editing of a word processing form where free text and markers for variables are placed. The second is for the generation of an array as defined by the form being referenced. Several options in the form definition enhance its flexibility. XBFORM0 contains the XBFORM documentation.

The programmer must supply a file for the forms with the .01 field being the name of the form and another field that is a WP field to hold the form itself.

(Requires XBLM and VALM utilities to be present.)

Entry Points

EDIT^XBFORM(NAME,DIC,FLD)

Edits and displays the form. Place the call to EDIT in the code where the data or variables have been gathered. Typically this is one line previous to

the call to $$GEN^XBFORM. Once the form is designed the EDIT call is commented out. It uses XBLM to display the run time data as it will be structured into the array.

Exiting the editor portion the XBLM display can also provide markers in the document for those that are working with forms.

See FORM DEFINITION OPTIONS for instructions on format options within the form.

Input Variables

NAME Name of form.
DIC File number of the file with the forms
FLD Field number of the WP field holding the form definition.

Y = $$GEN^XBFORM(NAME,DIC,FLD,%Y,FORMAT,OFFSET)

Generates the form into the root array indicated by %Y. The call to $$GEN must have all variables referenced already present in the partition. The return value of $$GEN is equal to the last line set in the array.

Input Variables

NAME Name of form.
DIC File number of the file with the forms.
FLD Field number of the WP field holding the form definition.
NAME Name of the form

%Y The root of the target array to be built. Either a global or a variable root as in the format used for a %XY^%RCR call. (%RCR is actually used)

FORMAT null or zero The array is built %Y(line)="...
1 The array is built %Y(line,0)=".... as used by VALM.

OFFSET The offset is line numbers in building the array. The array will start construction at OFFSET +1. The value of the last line created is returned $$GEN.

WP Format Definition Options

Free Text: Free text is key stricken where desired. Do not use "~" as it is used to mark variables and their placement.

Variables: The reference to a variable is marked with a beginning "~" and a trailing "~". The trailing ~ is always required even if the variable is last item on the line.

Functions

Comment Line Programmers comments can be put into the form and are ignored by the generator.

Mnemonic Variables

This is a short hand for variables.

Output Transform

Mumps code can be input that will transform a selected variable's output.

Functions

Comment Line Begin the line with a ';'

Mnemonic Variable

Namespaced variables can be long. A mnemonic reference is available to make life simple. Mnemonic definitions are place at the top of the form. Mnemonic variables then can be placed anywhere in the form.

Begin each line of definition with a '#'. The mnemonic is separated from it reference by a '|' (vertical bar). Multiple mnemonic references on the same line are separated by '*'

Mnemonic definitions are placed at the top of the form

(M|R) MNEMONIC|REFERENCE

Example: #D|DUZ*V|BARVPT
#I|BARIPT

~D will be interpreted as meaning ~DUZ
~V will be interpreted as meaning ~BARVPT
~I will be interpreted as meaning ~BARIPT

(BARIPT is an array storing IHS Patient Information)
(BARVPT is an array storing VA Patient Information)

MNEMONIC MARKER

The mnemonic markers can be used anywhere in the WP form. It is marked by a beginning and ending pair of '~'s. A vertical bar separates the mnemonic and the value of the subscript.

(M|S) MNEMONIC|SUBSCRIPT

Format ~mnemonic|variable subscript~
Example ; following the mnemonic reference definition
Define #D|DUZ*I|BARIT
~D|~ for DUZ
~D|0~ for DUZ(0)
~I|.01~ for BARIPT(.01)

Output Transform

A MUMPS expression of X. 'X' must be used literally in the function defined.

A simple mumps output transform capability is also provided to aid in form design. A variable or mnemonic indicated will have its output transformed prior to being put into the form. The definition line is placed at the top of the form. It is started with a '*', followed by the variable reference, followed by a ':', and then the function of x 'f(x)' to be performed. Multiple lines can be used and multiple outputs defined on a line separated by a '*'.

Setup

*var1:mumps code1*var2:mumps code2
*mnemonic3:mumps code3*mnemonic4:mumps code4

Ex: *DUZ(2):$J(X,10,2) will transform ~DUZ(2)~ to $J(DUZ(2),10,2)
*D|2:$J(X,10,2) mnemonic notation of same

Special Output Transforms provided by XBFORM

$$MDY(X) Returns a date format of MM/DD/YY

Many times only a mm/dd/yy is desired. This function automatically converts any external date to mm/dd/yy. (An external form of date is returned by XBDIQ1.

*M|S:$$MDY(X) a literal ~"NOW"~ or variable ~IT|9~
ex: *"NOW":$$MDY(X) or *IT|9:$$MDY(X)
returns mm/dd/yy

$$WP("X") Word Processing field printing

Word Processing fields are handled by this output transform.

*M|S:$$WP("X") for a word processing field array ~M|S~
NOTE: "X" THE QUOTES ARE ABSOLUTELY NECESSARY!
The variable array must have the form
VAR(subscript,n) where n = 1:1

$$FL(X) Fill Lines

Sometimes it is necessary to jump to a specific line

Blank lines are used to fill from the present line through the line indicated.
*19:$$FL(X) ~19~ fill lines through 19 with a " "

TIPS for VALM Users

The compilation of the form resides in the ^TMP($J,"XBFORM","Form Name", nodes. Those subscripts are organized ..."Form Name",Line,Column)=. Lines and columns are straight forward for text and variables start at their column +.5 to indicate the expression has to be evaluated. If the $$FL output transform has been used, the programmer will have to calculate the new line numbering offsets manually.

Adding video attributes with VALM calls needs to be done within the INIT lines of the VALM program being called.

Examples of XBFORM Calls

```
BARFORM0 ; IHS/ADC/PDW - FORMS FOR XBFORM ; [ 07/06/95 11:03 AM ]
    ;;1.0c4;IHS ACCOUNTS RECEIVABLE;;JUN 21, 1995
TEST ;;
    ;** set up variables
    D ENP^XBDIQ1(200,DUZ,".01:.116","BARU(")
    ;** setup a word processing field
    F I=1:1:5 S BARWP(101,I)=" LINE "_I_" has the value of "_I
    ;** setup form name
    S BARFORM="PW TEST"
    ;** call form editor
    D EDIT^XBFORM(BARFORM,90053.01,1000)
    ;** call array generator
    S LASTLINE=$$GEN^XBFORM(BARFORM,90053.01,1000,"BARFM(",0)
    Q
```

Editor Form Example

```
;------ Mnemonic References Definitions
#G|BARU*D|DUZ
#W|BARWP|
;------  Output Transforms - Mumps Expressions - $$MDY - $$WP("X") - $$FL(X)
*"TODAY":$$MDY(X)
*DUZ(2):$J(X,10,2)
*W|101:$$WP("X")
*39:$$FL(X)
;----------------------------------START OF FORM---------------------
        DUZ      DUZ(2)      DUZ(0)    DT

 Variable Reference   ~DUZ~    *~DUZ(2)~     ~DUZ(0)~  ~DT~
 Mnemonic|Subscript   ~D|~      ~D|2~        ~D|0~
               * DUZ(2) has a $J(X,10,2) output transform
  /----------------------------------\
  |   NAME   ~G|.01~   |
  |   ADD2   ~G|.112~  |
  |   ADD3    ~G|.113~ |
  |   CITY    ~G|.114~ |
  |   STATE  ~G|.115~ |
  |   ZIP       ~G|.116~ |
  \----------------------------------/


    TERMINAL CHARACTERISTICS
    TYPE               ~IOST~
    R MAR              ~IOM~
    FORM LENGTH ~IOSL~

Word processing example  W|101 with an output transform W|101:$$WP("X")
=============================================================
=
~W|101~|
=============================================================
=
SKIP TO LINE 40
~39~
LINE 40
/////////
END OF FRAME

ARRAY GENERATION EXAMPLE

[DEV,DSD]>ZW BARFM
BARFM(1)="                 DUZ      DUZ(2)      DUZ(0)    DT"
BARFM(2)=" "
```

```
BARFM(3)="  Variable Reference   60      *   1546.00   @        2950706"
BARFM(4)="  Mnemonic|Subscript   60      1546        @"
BARFM(5)="                          * DUZ(2) has a $J(X,10,2) output transform"
BARFM(6)=" "
BARFM(7)="            /----------------------------------\"
BARFM(8)="            |  NAME   WESLEY,PAUL |"
BARFM(9)="            |  ADD1   PO BOX 958 |"
BARFM(10)="           |  ADD2   'CRAVEN ELMS MOBILE HOME |"
BARFM(11)="           |  ADD3   ' #5 |"
BARFM(12)="           |  CITY   EDGEWOOD |"
BARFM(13)="           |  STATE  NEW MEXICO |"
BARFM(14)="           |  ZIP    87015  |"
BARFM(15)="           \----------------------------------/"
BARFM(16)=" "
BARFM(17)=" "
BARFM(18)="              TERMINAL CHARACTERISTICS"
BARFM(19)="              TYPE       C-VT100"
BARFM(20)="              R Mar      80"
BARFM(21)="              FORM LENGTH 24"
BARFM(22)=" "
BARFM(23)="Word processing example      W|101 with an output transform
W|101!$$WP("X")"
BARFM(24)="=================================================
="
BARFM(25)="   LINE 1 has the value of 1"
BARFM(26)="   LINE 2 has the value of 2"
BARFM(27)="   LINE 3 has the value of 3"
BARFM(28)="   LINE 4 has the value of 4"
BARFM(29)="   LINE 5 has the value of 5"
BARFM(30)="=================================================
=
BARFM(31)="SKIP TO LINE 40"
BARFM(32)=" "
BARFM(33)=" "
BARFM(34)=" "
BARFM(35)=" "
BARFM(36)=" "
BARFM(37)=" "
BARFM(38)=" "
BARFM(39)=" "
BARFM(40)="LINE 40"
```

2.2.2.16 XBLM

XBLM provides simplified utility entry points for the programmer to utilize the VALM utility without having to design their own browser. It has an interface already built in to access the host file systems so that hard coded or FM generated displays can be loaded into the browser. It also has an array entry point.

This utilizes and requires the presence of the VALM software as distributed by the VA and/or IHS.

Requirement

The VALM software must be installed and inited.

The DEFAULT HOST FILE as identified in FILE(1) by $$PWD^%ZISH(.FILE) must have its permission for WR and group set for RPMS users.

D ^XBONIT	Installs the XBLM Protocol
D ^XBL	Installs the XBLM List Manager Template.

Entry Points

FILE^XBLM("Directory","File Name")

Displays file indicated

Directory	Host file directory
File Name	File Name to be displayed

SFILE^XBLM — Manual selection of host file for display. Allows the user real time access to host files. Wildcarding of the file during selection is allowed.

VIEWR^XBLM("TAG^ROUTINE","Header")

Displays printout of the routine. (non - FM, using IO)

VIEWD^XBLM("TAG^ROUTINE","Header")

Displays printout of the routine. (FM - using EN1^DIP)

DIQ^XBLM("DIC","DA")

Displays EN1^DIQ for the DIC,DA

ARRAY^XBLM("ARRAY(","Header")

Displays the array(...,n,0) as necessary in VALM format.

2.2.2.17 XBNEW

This routine provides the programmer with a SACC-approved manner of performing an exclusive NEW preserving the required Kernel variables at the same time. It also includes wild carding.

Entry Point
EN^XBNEW("TAG^ROUTINE","Variable List")

Input Variables
"TAG^ROUTINE" The routine to be executed within the exclusive newed environment.

"Variable List" The list of variables to be carried into the exclusive newed environment.

EX: "AGDFN;AGINS;AGP*" Wild card allowed.

Required Kernel variables are automatically added to the list.

2.2.2.18 XBDH*

XBDH is the Header Editor main routine. XBDHD sets basic info about file and fields. XBDHD1 compiles header line. XBDHD2 works with special choices. XBDHDF gets field information for header line editor. XBDHDF1 checks jump syntax. XBDHDIP is an overlay of DIP2 for Auto FileMan. XBDHDSV compiles header info for auto entry into DIP. XBDHDSP puts spaces between headers.

2.2.2.19 XBDHNTEG

XB integrity checker.

2.2.2.20 XBDICV

This routine sets FileMan dictionary version numbers.

2.2.2.21 XBDIE

Use this routine to nest DIE calls.

2.2.2.22 XBDIFF

The difference between two dates/times is returned with this routine.

2.2.2.23 XBDINUM

Use to convert a non-dinum file to a dinum file.

2.2.2.24 XBDIR

The purpose of this routine is to provide an interface methodology for a call to ^DIR, to ensure correct handling of variables, and to provide for the expressiveness of an extrinsic function.

2.2.2.25 XBDR*

This routine builds a string which sets variables DIR and its descendants for use in a routine. The string is stored in the variable "%", and in the "Temp" storage area for the screen editor for the current device.

2.2.2.26 XBDSET

This routine selects FileMan dictionaries individually, by a range, or for a specific package. This routine can be called from another routine by setting the variables XBDSLO, XBDSHI, and then D EN1^XBDSET.

2.2.2.27 XBENHANC

This routine prints enhancements to a package from the entry in the package file. Entry point EN^XBENHANC(ns) is used with the caller providing the namespace of the package.

2.2.2.28 XBFCMP

This routine compares FileMan files in two UCIs.

2.2.2.29 XBFDINFO

Given a file/subfile number, a field number, and an array root, this routine will return information about the specified field. The information will be returned as a subscripted variable from the root passed by the caller.

2.2.2.30 XBFIXL1

This routine asks the user to select a set of routines, for the programmer information, and standardizes the format of the first line of each routine.

2.2.2.31 XBFIXPT

This routine fixes all "PT" nodes for files 1 through highest file number in the current UCI.

2.2.2.32 XBFLD*

This routine lists dictionaries which may be selected individually or by a range of dictionary numbers. XBFLD0 prints field triggers.

2.2.2.33 XBFMK

This routine kills variables left by FileMan.

2.2.2.34 XBCDIC*

This routine cleans up ^DIC and ^DD. XBCDIC2 checks dictionary names and data globals. XBCDIC3 checks ^DD. XBCDICD deletes bad files.

2.2.2.35 XBFIX

This routine counts entries in FileMan files and fixes.

2.2.2.36 XBCFXREF

Use this routine to check and fix cross references.

2.2.2.37 XBCOUNT

This routine counts entries in a FileMan file.

2.2.2.38 XBFRESET

Routine is used to reset file globals.

2.2.2.39 XBFUNC*

These routines make up the Function Library.

2.2.2.40 XBGC

Use to copy a global at any level.

2.2.2.41 XBGCMP

Routine compares two different globals. XBCMP2 contains help for XBCMP.

2.2.2.42 XBGLDFN

Routine to get last DFN.

2.2.2.43 XBGSAVE

See separate section in this manual for further guidance. Generic global save for transmission globals.

2.2.2.44 XBGTI

Use to restore globals saved in DSM %GTO format.

2.2.2.45 XBGTOT

Use for fast save to tape.

2.2.2.46 XBGXFR

Routine is used to transfer global trees.

2.2.2.47 XBGXREFS

Use to get cross references for one field in one file.

2.2.2.48 XBHEDD*

Contains the components for the Electronic Data Dictionary. The EDD provides a more user friendly method of reviewing data dictionary structures and global structures. It contains on-line documentation.

2.2.2.49 XBHELP

Use to display help text from a routine.

2.2.2.50 XBHFMAN*

These routines are for the help frame manual.

2.2.2.51 XBKD*

Use to kill DICs and globals.

2.2.2.52 XBKERCLN

Use this routine to clean out Kernel namespace items prior to install.

2.2.2.53 XBKSET

This routine sets minimal Kernel variables.

2.2.2.54 XBKTMP

Use to clean ^TMP nodes for the current job.

2.2.2.55 XBKVAR

This routine sets minimal Kernel variables.

2.2.2.56 XBL

This routine is used for a list template exporter.

2.2.2.57 XBLCALL

Use to provide a list of callable subroutines.

2.2.2.58 XBLML

Use to enter or reset XB display in List Template File for List Manager.

2.2.2.59 XBLUTL

This routine lists all entries in the ^UTILITY global for the current $J where $J is the first or second subscript. This is most useful from programmer mode. If used thru the XB menu, ^UTILITY($J) is killed in ^XBKSET before this routine is run.

2.2.2.60 XBLZRO

This routines lists the 0th nodes of FileMan files.

2.2.2.61 XBMAIL

This utility generates a mail message to everyone on the local machine that holds a security key according to the namespace, range, or single key provided in the parameter. The text of the mail messages must be provided by the developer, and passed to the utility as a line reference. The utility uses the first line after the line reference as the mail message subject, and subsequent lines as the body of the message, until a null string is encountered. This places an implicit limit on your mail messages to the maximum size of a routine. Suggested text may be used to inform the users that a patch has been installed, and to describe any changes in displays or functionality, or problems addressed, and provide a contact number for questions,

e.g:

Please direct your questions or comments about RPMS software to:
OIT
Albuquerque NM 87110
505-837-4189

2.2.2.62 XBNODEL

This routine sets FileMan dictionaries so users cannot delete entries. Protection is provided by SET'ing the "DEL" node of the .01 fields in the selected dd's to "I 1".

2.2.2.63 XBOFF

Use to set reverse video off.

2.2.2.64 XBON

Use to set reverse video on.

2.2.2.65 XBPATSE

Use to search routines for patched versions.

2.2.2.66 XBPFTV

This routine is used to return pointer field terminal value.

NOTE TO PROGRAMMERS; Use entry point PFTV. Do not use the first line of this routine, as pending initiatives in MDC might make a formal list on the first line of a routine invalid. Given a file number, file entry number, and variable name into which the results will be placed, return the terminal value after following the pointer chain. U must exist and have a value of "^".

Formal list
1) F = file number (call by value)
2) E = file entry number (call by value)
3) V = variable for results (call by reference)
Scratch vars:
D = Flag, 1 = Done, 0 = continue
G = Global for file F

2.2.2.67 XBPKDEL

Programmers can use this routine to remove options, input, sort, print templates, help frames, bulletins, functions, and if indicated, security keys for a package.

XBPKNSP must be set to the namespace, e.g., "AICD", if this routine is called from a preinit. If you want security keys deleted, set XBPKEY=1 if this routine is called from a preinit. Call LIST^XBPKDEL to list all namespaced options, templates, etc. Call RUN^XBPKDEL to delete all namespaced options, templates, etc. The RUN and LIST entry points are for programmer use and are not to be called from a preinit. Preinit calls XBPKDEL directly with variables set as indicated above.

2.2.2.68 XBPOST

XB/ZIB installation postinit.

2.2.2.69 XBPRE

Preinit that checks requirements, etc.

2.2.2.70 XBRESID

This routine deletes residual entries in ^DD by a range of dictionary numbers. A residual entry is one that has no parent. The process is reiterative, so an entry that has a parent in ^DD, and the parent is deleted because it has no parent, will also be deleted. The parent of an entry in ^DD is defined as another entry in ^DD for sub-files, and an entry in ^DIC for primary files.

The range of dictionary numbers is inclusive but residual entries for the high file number will not be deleted at the sub-file level. This is because sub-files are numbered with the primary file number with decimal numbers appended. The terminating check is ^DD entry greater than high file number specified, so by definition all sub-files for the high number are greater than the high number.

This routine can be called by another routine by setting XBRLO and XBRHI and then D EN1^XBRESID.

2.2.2.71 XBRESTL1

Routine to restore first line of routines from a save file.

2.2.2.72 XBRLL

This routine lists a single routine line by line noting the length of the line plus the cumulative character count.

2.2.2.73 XBRPRTBD

This functionality has been moved to ZIBRPTRD because of the use of non-standard $Z special variables. The GO is provided for backwards compatibility.

2.2.2.74 XBRPTL

This routine prints the selected routine down to the first line label.

2.2.2.75 XBRSBD

This routine saves selected routines edited after a given date.

2.2.2.76 XBRSELM

Routine selector.

2.2.2.77 XBRSIZ

List routine names and sizes with overall total.

2.2.2.78 XBRSRCH*

Search data dictionary (DD) for called routines, common check logic, search input transform for routines, search output transform for routines, search cross references for routines, and search miscellaneous for routines.

2.2.2.79 XBRXREF*

This routine re-cross references selected cross references(xrefs) for a file. The xrefs are killed at the highest level and then reset. This is very different from what FileMan does when you RE-INDEX a field. FileMan does a logical kill and then sets the new xrefs. The reason for this is multiple fields may set the same xref so one would want to kill only the ones set by the field being RE-INDEXed. One must re-xref all fields that set any one of the xrefs being killed and reset, unless the xref is set the same from

multiple fields. This is very hard to explain. Therefore, if you do not understand the problem, you probably should not be running this routine.

This routine executes an entry point in ^DIK to build the xref logic for all xrefs on the file. It then deletes the logic for all xrefs not selected, and executes another entry point in ^DIK to actually xref the file.

TRIGGERS are very complex animals which do not have a xref to kill and may be conditional and may have no affect.

2.2.2.80 XBSAUD

This routine sets 'audit' on at the file level for selected files.

2.2.2.81 XBSAUTH

This routine sets FileMan dictionary authorities:

"AUDIT" "DD" "DEL" "LAYGO" "RD" "WR".

2.2.2.82 XBSFGBL

This routine returns a subfile global reference.

NOTE TO PROGRAMMERS: Use entry point EN. Do not use the first line of this routine, as pending initiatives in MDC might make a formal list on the first line of a routine invalid. Given a file or subfile number and global reference form, this routine will return the global reference in the form specified.

F (form) is optional but if passed should equal 1 or 2.
If F is not passed the default form will be 1.

F = 1 will be in the form ^GLOBAL(DA(2),11,DA(1),11,DA,
F = 2 will be in the form ^GLOBAL(D0,11,D1,11,D2,

Formal list:
1) S = subfile number (call by value)
2) G = global reference (call by reference)

2.2.2.83 XBSITE

Routine to set DUZ(2).

2.2.2.84 XBSUMBLD

This routine requests the user to select a set of routines and generates an integrity checking routine for the selected routines. The user is asked to enter the name of the generated routine.

The VA's equivalent routine is XTSUMBLD, which also creates integrity checking routine(s).

2.2.2.85 XBTM*

This routine, and subsequent routines in the XBTM* namespace, produce a technical manual from information contained in the package. The manual is approximately 80 pages. All, or individual chapters can be printed.

2.2.2.86 XBUPCASE

Call to convert to uppercase.

2.2.2.87 XBVCH*

Routine(s) to intelligently change variable names.

2.2.2.88 XBVCHV

Use to pull in variables and routines from a %INDEX.

2.2.2.89 XBVIDEO

Set various video attributes. $X is saved and the cursor is returned to it's original position thru X IOXY (except certain attributes). In addition to the attributes supported by ENDR^%ZISS, some color attributes are supported, and other mnemonics are provided for backward compatibility.

2.2.2.90 XBVK

This is the front end for killing local variables in the namespaced parameter. Implementation specific routines are called from this routine which is in the ZIBVK* namespace.

This routine is intended to be called by applications that are done executing in order to KILL any remaining namespaced local variables. E.g., D EN^XBVK("AG") will KILL any local variables that exist in the AG namespace.

Notice that if called in background, and the OS is not supported, the routine will quit, unpleasantly. If your implementation is other than what is supported, and your vendor has implemented all Type A extensions to the 1990 ANSI M standard, you can safely remove the two lines that check for OS, and use the existing call to the MSM-specific routine.

2.2.2.91 XBVL

This is the front end for listing local variables. Implementation specific routines are called from this routine. Therefore, it has been moved to the ZIBVL* namespace.

2.2.2.92 XBVLINE

This routine asks the user to select a set of routines, asks for the version number, package, date, and sets the second line of each routine.

The form of the version line will be as follows:

```
;;n;package name;patch level;date    E.G.
;;1.1;PCC DATA ENTRY;**1,2**;Sep 9, 1989
```

2.2.2.93 XBXTSS

Use for extract and table subscripts.

2.2.2.94 XBPATC

This routine will $Order through the patient and 3rd party globals looking for missing entries. It will compare IHS/VA Patient files to define unequal DFN's. It will also look for a null pointer value in Medicaid global.

2.2.2.95 ZIBCKPKG

This routine checks UCI for package content.

2.2.2.96 ZIBCLU*

This is a general purpose clean up utility global and driver to get UCI. This routine will initiate a job running ^%ZIBCLU0 in each UCI and then wait 5 seconds to elapse before getting the next UCI, skip the UCI this task is in, and then run ^%ZIBCLU0 here. %ZIBCLU0 will remove all dangling ^UTILITY, ^XUTL, ^ZUT entries. This routine is usually started via TaskMan by scheduling the -ZIBCLU- option which runs this routine.

DSM ONLY - $ZU(ZIBI) returns <NOUCi> error at end of UCI list

MSM ONLY - $ZU(ZIBI) returns -NULL- value at end of UCI list

2.2.2.97 ZIBER*

MSM error report routines.

2.2.2.98 ZIBFIND

MSM-specific utility for finding blocks which contain a specific GBL.

ZIBCC=common count, ZIBUC=unique count
ZIBCHAR=string of characters

2.2.2.99 ZIBFMD

Routine to display FileMan installation data.

2.2.2.100 ZIBFR

Given a routine name, this routine searches all UCIs and reports the first line of the selected routine to the user.

2.2.2.101 ZIBGCHAR

This routine contains non-interactive modifications of global characteristics.

Not all capabilities of the implementation-specific global characteristics routines are reflected in this routine. The argument for each entry point is the unsubscripted name of the global whose characteristics you want to change with the circumflex present. If the call is successful, 0 is returned. If the call is not successful, a positive integer is returned, and the cause can be retrieved at the ERR() entry point.

E.C.'s:
S %=$$NOJOURN^ZIBGCHAR("^AUTTSITE")
I % W !,$$ERR^ZIBGCHAR(%)

2.2.2.102 ZIBGCHR

Routine to search for control characters in globals.

2.2.2.103 ZIBGD

This routine displays a selected range or subset of the global directory.

2.2.2.104 ZIBGSVED

Routine to save global(s) to tape - DSM specific.

2.2.2.105 ZIBGSVEM

Routine to save global to MSM Unix.

2.2.2.106 ZIBGSVEP

Routine to save global to DOS media.

2.2.2.107 ZIBGTOT

This routine creates a global save in the same format as ^%GTO. The difference is it uses $ZO which is much faster. Because tests showed no significant difference in speed between CDT and CAVL4 this routine accepts CDT only. It would be difficult, although not impossible, to allow partial globals. Therefore, this routine will save complete globals only. This routine always rewinds the tape before saving the globals. That means only one file per tape but that one file may use multiple volumes.

The primary purpose of this routine is to move globals from DSM to MSM. Therefore, the limitation is use of tape only.

2.2.2.108 ZIBJRNI

This utility is used to initialize all except the active journal areas of a system. The STU entry point is used by the automatic partition reference for a particular configuration. This was developed for the PC Network Configuration and has only been tested using MSM-PC/386.

2.2.2.109 ZIBNSSV

Use to return non-standard ($Z) special variables, e.g., W $$Z^ZIBNSSV("ERROR") will write the contents of the error message most recently produced by the OS. These are the variables supported:

ERROR : Text of error message most recently produced.
LEVEL : Number of the current nesting level.
NAME : Name of routine currently loaded in memory.
ORDER : Data value of the next global node that follows he current global reference.
TRAP : Line label and routine name of the program that s to receive control when an error occurs.

VERSION : Name and release of M implementation.

2.2.2.110 ZIBPKGF

Use this routine to obtain an installation status report.

2.2.2.111 ZIBPKGP

Use this routine to process implementation status files.

2.2.2.112 ZIBRD

Use this routine to display a MSM directory of selected routines .

2.2.2.113 ZIBRER*

This routine provides remote error reporting. It $ORDERs through the ERROR LOG, extracting errors executed since the last run. It then WRITEs the errors to a file, and SENDs the file to the identified destination(s) according to the parameter (ENTRY ACTION of option). It removes errors in the ERROR LOG that are more than 180 days old. This routine is non-interactive. It is designed to run in the background from TaskMan only.

Entry point OPT is used to set an option into the OPTION file which is scheduled for every 6 days at 9 PM. The process begins at START.

2.2.2.114 ZIBRERP

ZIB remote error file processing routine. An option will be placed in the OPTION file for daily processing of files sent to this machine by the Remote Error Reporting utility beginning the next night at 10:30 PM. The user can change the frequency/time of scheduling by using the TaskMan option thru the Kernel.

2.2.2.115 ZIBRNSPC

Use this routine to namespace previously written routines.

2.2.2.116 ZIBRPI*

This utility creates an entry in the OPTION file which is scheduled to run daily in TaskMan. It searches for files matching the naming conventions for patch files (specified in the SAC) in the directory indicated by the user. If the package version currently installed on the system and the patch version match, the routines are restored from the file, an entry is made in the VERSION multiple of the PACKAGE

file entry, and a report file is sent to the systems indicated by the user. If an action routine (A9 or B9) is detected during the ZLOAD, and the user has the indicated permission to run action routines, the action routine is called after all routines have been restored.

> NOTE: Use the same entry point, OPT^ZIBRPI, to edit any changes to the parameters. If the user unschedules the option, the TaskMan options must be used to reschedule it.

2.2.2.117 ZIBRPRTD

This routine lists routines edited after given date.

2.2.2.118 ZIBRSEL

This is a non-interactive routine select which returns the number of selected routines set into the indicated variable.

E.g.:
I '$$RSEL^ZIBRSEL("B-BZZZZZZZ","ARRAY(") W "NONE SELECTED" Q

If ro2utines exists in the list or range, their names will be returned as the last subscript of indicated variable in the 2nd parameter. The default is ^TMP("ZIBRSEL",$J,.
If routine B exists, then node ^TMP("ZIBRSEL",$J,"B") will be null. It is the programmer's responsibility to ensure the name of the array is correctly formed.

Variables used:

X = String indicating list or range of routines.
Y = String indicating variable into which to set the selected routines. Default = ^TMP("ZIBRSEL",$J,
F = First routine, if range.
L = Last routine, if range.
N = Number of routines returned.
Q = Quote character.

2.2.2.119 ZIBRUN

Use to check for active routine in a specific UCI.

2.2.2.120 ZIBSSD

This utility is used as the nightly shutdown routine for MSM PC and is initiated by the scheduled option AZSJ SHUTDOWN. It was developed for the PC Network configuration.

2.2.2.121 ZIBTCP

TCP Print Test - This routine must be DONE from the CLOSE EXECUTE when printing to a TCP printer. See below for further documentation.

H = Host IP address
P = Port number
I = Counter

Technical Notes:

MSM TCP uses the "!" to clear the TCP buffer. FileMan (RPMS) uses "!" for a carriage return, line feed. Further, TCP does not recognize "?30" as 30 spaces from left margin. To circumvent these problems, write to a temporary host file, which formats the document, and then read it back into the TMP global. Once it's in the TMP global, $O through the global and write each line with a $C(10) and $C(13) concatenated to the string. This process handles the CR/LF problem at the remote end.

Port 2501 is the assigned port from the vendor for the Net Que.

As of 3Jan95, this has only been tested on the Unix platform using MSM. It should work in a DOS environment using FTP Software's TCP, but needs to be tested.

Below is an inquiry of the Device file and Terminal Type file.

```
OUTPUT FROM WHAT FILE: DEVICE//
NAME: P-TCP TEST PRINTER            $I: 51
ASK DEVICE: YES                 ASK PARAMETERS: NO
VOLUME SET(CPU): TUC                    SIGN-ON/SYSTEM
DEVICE: NO
FORCED QUEUING: N0
LOCATION OF TERMINAL: MAT PARKENSON PRINTER
ASK HOST FILE: NO               MARGIN WIDTH: 255
FORM FEED: #                    PAGE LENGTH: 256
BACK SPACE: $C(8)   OPEN PARAMETERS:
                        ("XM"_DUZ_$G(ZIBH)_".DAT":"M")
SUBTYPE: P-TCP PRINTER    TYPE: HOST FILE SERVER

Select TERMINAL TYPE NAME: P-TCP PRINTER
NAME: P-TCP PRINTER          SELECTABLE AT SIGN-ON: NO
RIGHT MARGIN: 255              FORM FEED: #
PAGE LENGTH: 256               BACK SPACE: $C(8)
OPEN EXECUTE: S XMREC="R X#255:1"      CLOSE EXECUTE: D
^ZIBTCP Q
DESCRIPTION: Special Terminal Type used only for P-TCP Printer
Device.
```

2.2.2.122 ZIBVCHV

Use to read variables and routines from a %Index.

2.2.2.123 ZIBVGE

This routine changes the Volume Group Name in ^SYS(and ^%ZOSF. It is used for the rapid change of the volume group names in the PC Network Configuration. It has only been tested using MSM-PC/386.

2.2.2.124 ZIBVKIL

Use to build a namespace variable killer routine in ^.ns.KVAR. Select a %INDEX host file summary on which to build the routine. Select a namespace for the variables and the routine to be built. Enter any package-wide variables. Add D ^.ns.VKL0 to all menu exit actions where package variables are to remain. Add D KILL^XUSCLEAN to the exit action of all other menus.

2.2.2.125 ZIBVKMSM

This routine kills variables in the namespace of the variable passed in the parameter and is accessed thru the front end routine XBVK.

2.2.2.126 ZIBVLMSM

This routine lists variables that begin with the string entered by the user. Selection of variables is case sensitive. This routine is specific to Micronetics. It will work with any M implementation that has all Type A extensions to the 1990 M ANSI standard implemented. The front end routine, XBVL, stops if any other than an MSM implementation is encountered.

2.2.2.127 ZIBZUCI

Swap UCI between volume sets for MSM-UNIX - Save this routine as %ZUCI in the MGR UCI.

This utility permits switching between UCIs and Volume Groups when run in programmer mode. D ^%ZUCI. If switching to a UCI in a Volume Group other than the System Volume Group (0), enter either the Volume Group Number or Volume Group Name along with the UCI Number or Name. A 'help' display identifies all UCIs and Volume Groups that are currently mounted. Use a '?' for 'help'.

A routine may be tied to the UCI,VOL switch. This will be called immediately after the UCI,VOL switch occurs.

2.2.3 XBGSAVE - Generic Global Save

XBGSAVE is a utility to save globals onto a peripheral medium. XBGSAVE determines the operating system (OS) in use, and will invoke OS-specific routines to perform the actual saves. Historically, MSM and DSM routines had been distributed with XBGSAVE. Other OS-specific routines must be provided by the user or submitted to the developer for incorporation into XBGSAVE. Current capabilities are to save MSM globals to cartridge tape, floppy disk, Unix file or 9-track tape, and to save DSM globals to cartridge tape or 9-track tape.

2.2.4 XBGSAVE Routine

XBGSAVE is a parameter-driven routine that: (1) verifies that the global to be saved exists, (2) verifies parameters, (3) determines the operating system under which it is running, then calls the appropriate OS-specific routine to process the global.

2.2.5 ZIBGSV* Routines

These OS-specific routines saves the global entries to the output device specified.

2.2.6 Technical Notes

MSM Write Protect
The MSM operating system cannot test for write protection on tape, therefore, write enable the tape before processing.

Kernel Environment Assumed
XBGSAVE assumes it's running under the KERNEL with appropriate variables set. If called from other than the Kernel environment, the user must ensure a defined environment prior to calling XBGSAVE.

File Name
A global saved to the Unix file will be named AAAAFFFFFF.JJJ.
"AAAA" is the global namespace.
"FFFFFF" is the six digit numeric facility code.
"JJJ" is the Julian date.

Device numbers
DSM - 47 for cartridge; 48 for 9-track
MSM - 51 - 54 for all devices

The tape format for MSM is in the %GS format.

2.2.7 Programmer Notes

Input:

In addition to the global to be saved, other parameters may be used to customize XBGSAVE. Only the global name parameter, XBGL, is required. For non-KERNEL applications, also specify DT for date, DUZ(2) for location and %ZOSF("OS") for operating system type.

Process:

1. Verify global to be saved exists.
2. Verify parameters.
3. Call OS-specific ZIBGS* routine.

Output:

Local variable XBFLG will contain the result of the call to XBGSAVE. If the save was successful, XBFLG will be 0 (zero). If the save was unsuccessful, local variable XBFLG(1) will have a narrative that can be displayed to the user indicating the cause of failure.

2.2.8 XBGSAVE Input Parameters

Name	Description/ Default/ Format/ Use
XBDT	Date of Global save. Default: NOW. Format: Enter date in FileMan format, leading zeros not required. Ex.: S XBDT=7991231 Use: Header dump comment.
XBE	Ending first-level numeric subscript Default: End of file. Format: Canonic number.
XBF	Beginning first-level numeric subscript, seed for $ORDER. Default: beginning of file. Format: Canonic number.
XBFLT	S= 1, saves as flat file. Default: Save as a subscripted global. Format: 1 or none. Ex.: S XBFLT=1 Use: If 1, just the data will be saved, in flat file format.
XBFN	Output file name. Default: "<ns><asufac>.<Julian date>"
XBGL	Global name to be saved. Default: None. This is required. Format: Global name with no "^". Ex.: S XBGL="ATAGLOB".
XBIO	IO device parameter. Default: 51 - HFS for MSM and 47 - cartridge for DSM.

Name	Description/ Default/ Format/ Use
	Format: one to three digit number specifying device.
XBMED	Media used for output to global save. Default: If not defined, the user is asked to select the media. Format: "C" for cartridge tape "D" for diskette "F" for Unix file "T" for 9-track tape
XBNAR	Narrative for Operator display if help needed. Default: None. Format: Free text Ex.: S XBNAR="APC Service Unit". Use: Customizing HELP display. "This option saves the "_XBNAR_" "_XBGL_" transaction file to tape or diskette...".
XBPAR	DSM IO parameter. Default: open parameter from DEVICE file for specified device. Format: "V" or undefined Use: for DSM operating systems only, ignored by a MSM routine.
XBQ	Y/N, to place file in uucp q. Default: "Y" Format: "Y" or "N". Ex.: S XBQ="N" Use: If "Y", will place file in uucp q to send to Area Office.
XBQTO	'sendto' destination. Default: Area Office sysid in RPMS SITE file
XBTLE	Global dump comment. Default: DUZ(2) location name. Format: free text. Ex.: S XBTLE="APC's October trans" Use: Header dump comment for off-line media. XBTLE will be appended with DUZ(2) name for header dump comment.
XBUF	Full Path name for file (MSM only). Default: /user/spool/uucppublic for UNIX, C:\EXPORT for DOS.

2.2.9 Sample Set-up of Routine Call

To save a global to an operator-specified output device:

```
S XBGL="gloname",XBTLE="T&A for FY87"  D ^XBGSAVE
```

To save a global to a MSM cartridge:

```
S XBGL="gloname",XBMED="C"   D ^XBGSAVE
```

To save a global to a DSM cartridge:

```
S XBGL="gloname",XBMED="C"   D ^XBGSAVE
```

2.2.10 Error Codes

Various types of errors are produced:

Parameter errors
Operator cancellation errors
I/O errors
Tape test errors

If errors are detected, XBFLG is set to "-1", the error narrative is stored in XBFLG(1) and XBGSAVE returns to the calling program WITHOUT SAVING THE FILE.

If no errors are detected, XBFLG is set to "0".

Below are some of the possible error messages returned:

"Abort at drive select"
"Device Not Available During Tape Testing"
"Facility Number 'DUZ(2)' is not defined"
"Job Aborted by Operator during Tape Test"
"Job Aborted by Operator During Floppy Mount"
"Job Terminated by Operator at Device Select"
"Job Terminated By Operator at Mount Message"
"Job Aborted by Operator During Tape Mount"
"Job Aborted by Operator During Tape Test"
"Job Aborted, Tape not Ready"
"Job Aborted by Operator During Floppy Mount"
"Media Type '"_XBMED_"' is incorrect"
"Operating system is not 'MSM' or 'DSM'"
"Queue of File to uucp Failed"
"Tape not rewound"
"Tape Test Failed During Testing"
"The variable 'XBGL' must contain the name of the global you wish to save."
"The ^%ZOSF(""OS"") node does not exist"
"Transaction File does not exist"
XBERRMSG_" Not Available"
XBFLG(1)," After 6 Minutes"
XBMSG_" Drive Not Available"

2.3 Approved IHS APIs

2.3.1 Demographic Data

2.3.1.1 Routine - AUPNPAT

Data for the following function calls come from the PATIENT file, file 9000001, and from the Medicare Eligible, Medicaid Eligible, and Private Insurance Eligible files.

SEX(p) Returns SEX of patient p

arguments

p - patient ien (DFN)

examples

W $$SEX^AUPNPAT(234) => F

DOB(p,f) Returns DATE OF BIRTH of patient p in format f

arguments

p - patient ien (DFN)

f - optional format; if null, returns internal fileman format of DOB

E - external written-out format (MAR 05, 1995)

examples

W $$DOB^AUPNPAT(1234) => 2950305

W $$DOB^AUPNPAT(1234,"E") => MAR 05,1995

SSN(p) Returns SSN of patient

arguments

p - patient ien (DFN)

examples

W $$SSN^AUPNPAT(234) => 123456789

AGE(p,d,f) Returns AGE of patient p on date d in format f

arguments

p - patient ien (DFN)

d - optional date in internal fileman format; if null, will default to DT

f - optional format; if null, returns age in years

null - age in years

R - age in readable format

examples

W $$AGE^AUPNPAT(1234) => 32

W $$AGE^AUPNPAT(1234,"R") => 32 YRS

DOD(p,f) Returns DATE OF DEATH of patient p in format f

arguments

p - patient ien (DFN)

f - optional format; if null, returns internal fileman format of DOD

E - external written-out format (MAR 05, 1995)

examples

W $$DOD^AUPNPAT(1234) => 2950305
W $$DOD^AUPNPAT(1234,"E") => MAR 05,1995

TRIBE(p,f) Returns TRIBE OF MEMBERSHIP of patient p in format f

arguments

p - patient ien (DFN)
f - optional format; if null, returns tribe code
I - internal format of tribe (tribe ien)
E - external written-out format of tribe
C - tribe code

examples

W $$TRIBE^AUPNPAT(1234,"I") => 31
W $$TRIBE^AUPNPAT(1234,"E") => CHOCTAW NATION OF OK
W $$TRIBE^AUPNPAT(1234,"C") => 031

COMMRES(p,f) Returns COMMUNITY OF RESIDENCE of patient p in format f

arguments

p - patient ien (DFN)
f - optional format; if null, returns STCTYCOM COMMUNITY code
I - internal format of COMMUNITY (community ien)
E - external written-out format of COMMUNITY
C - STCTYCOM code

examples

W $$COMMRES^AUPNPAT(1234,"I") => 31
W $$COMMRES^AUPNPAT(1234,"E") => PRINCESS BAY
W $$COMMRES^AUPNPAT(1234,"C") => 0210019

HRN(p,l,f) Returns HEALTH RECORD NUMBER of patient p at location l in format f

arguments

p - patient ien (DFN)
l - must be valid ien of location
f - optional, 2–HRN will have prefix of site abbreviation

examples

W $$HRN^AUPNPAT(1234,4585) => 3456
W $$HRN^AUPNPAT(1234,4585,2) => SE3456

ELIGSTAT(p,f) Returns ELIGIBILITY STATUS of patient p in format f

arguments

p - patient ien (DFN)
f - optional format; if null, returns internal format
I - internal format of eligibility status (set of codes)
E - external written-out format of eligibility status

examples

W $$ELIGSTAT^AUPNPAT(1234,"I") => D
W $$ELIGSTAT^AUPNPAT(1234,"E") => DIRECT ONLY

BEN(p,f) Returns CLASSIFICATION/BENEFICIARY of patient p in format f

arguments

p - patient ien (DFN)
f - optional format; if null, returns classification/beneficiary code
I - internal format of classification/beneficiary (pointer value)
E - external written-out format of classification/beneficiary
C - classification/beneficiary code

examples

W $$BEN^AUPNPAT(1234,"I") => 1
W $$BEN^AUPNPAT(1234,"E") => INDIAN/ALASKA NATIVE
W $$BEN^AUPNPAT(1234,"C") => 01

MCR(p,d) Returns 1 or 0: Is Patient p eligible for Medicare on date d?

arguments

p - patient ien (DFN)
d - required date in internal fileman format

examples

W $$MCR^AUPNPAT(1234,2950601) => 1
Is patient 1234 eligible for Medicare on 6/1/95? => yes

PI(p,d) Returns 1 or 0: Is Patient p eligible for private insurance on date d?

arguments

p - patient ien (DFN)
d - required date in internal fileman format

examples

W $$PI^AUPNPAT(1234,2950601) => 1
Is patient 1234 eligible for private insurance on 6/1/95? => yes

MCD(p,d) Returns 1 or 0: Is Patient p eligible for Medicaid on date d?

arguments

p-patient ien (DFN)
d-required date in internal fileman format

examples

W $$MCD^AUPNPAT (1234,2950601) =>1
Is patient 1234 eligible for Medicaid on 6/1/95? =>yes

MCDPN(p,d,f) Returns Medicaid plan name for patient p on date d in format f

arguments

p - patient ien (DFN)
d - required date in internal fileman format
f - format, optional; if null, returns internal ien of insurer I

examples

W $$MCDPN^AUPNPAT(1234,2950601,"I") => 1
W $$MCDPN^AUPNPAT(1234,2950601,"E") => CARONDELET

PIN(p,d,f) Returns private insurance plan name for patient p on date d in format f

arguments

p - patient ien (DFN)
d - required date in internal fileman format
f - format, optional; if null, returns internal ien of insurer

examples

W $$PIN^AUPNPAT(1234,2950601,"I") => 1
W $$PIN^AUPNPAT(1234,2950601,"E") => BLUE CROSS/BLUE SHIELD

CDEATH(p,f) Returns CAUSE OF DEATH of patient p in format f

arguments

p - patient ien (DFN)
f - optional format; if null, returns Cause of Death ICD9 code
I - internal format ICD9 ien
E - external written-out format (ICD9 TEXT)
C - ICD9 code

examples

W $$CDEATH^AUPNPAT(1234,"I") => 31
W $$CDEATH^AUPNPAT(1234,"E") => DIABETES MELLITUS
W $$CDEATH^AUPNPAT(1234,"C") => 250.00

ENC(p) Returns an encrypted patient identifier 12 bytes long. The entry-point DEC reverses the process and returns the decoded output in a 27-byte-long string.

arguments

p - patient ien (DFN)

examples

W $$ENC^AUPNPAT(1) => V46332UMH763

DEC(p) Reverses the process of ENC^AUPNPAT and returns the decoded output in a 27-byte-long string.

arguments

p - patient ien (DFN)

examples

W $$DEC^AUPNPAT(V46332UMH763) =>[THA,B__JAN 01,1933_0001]

2.3.1.2 Routine - AUPNPAT1

Data for the following functions comes from the PATIENT file, file 9000001.

BEN(p) Returns Beneficiary/Non-Beneficiary Status

arguments

p - patient ien (DFN)

examples

W $$BEN^AUPNPAT1(1) =>1

OUTPUT:

1 = yes

0 = no

-1 = no/old tribe or unable

2.3.1.3 Generic VA PIMS APIs

The VA uses calls to VADPT to pull various patient data items. The following was borrowed from the VA Technical Manual.

2.3.1.3.1 INP^VADPT

This entry point will return data related to an inpatient episode.

Input Variables:

DFN Internal entry number in the patient file; Required

VAHOW This optional variable can be set to a requested format for the output array. If this variable is not defined or does not contain one of the following values, the output array will be returned with numeric subscripts.

1 -- return the output array with alpha subscripts - see details at end of this API

(e.g., VADM(1) would be VADM("NM"))

2 -- return the output in the ^UTILITY global with numeric subscripts

(e.g., ^UTILITY("VADM",$J,1))

12 -- return the output in the ^UTILITY global with alpha subscripts

(e.g., ^UTILITY("VADM",$J,"NM"))

VAROOT This optional variable can be set to a local variable or global name in which to return the output.(e.g., VAROOT="DGDEM")

VAINDT This optional variable may be set to a past date/time for which the programmer wishes to know the patient's inpatient status. This must be passed as an internal VA FileMan date/time format. If time is not passed, it will assume anytime during that day. If this variable is not defined, it will assume NOW as the date/time.

Output Variables:

VAIN(1) The INTERNAL NUMBER of the admission if one was found for the date/time requested. If no inpatient episode was found for the date/time passed, then all variables in the VAIN array will be returned as null. (e.g., 123044)

VAIN(2) The PRIMARY CARE PHYSICIAN [PROVIDER] assigned to the patient at the date/time requested in internal^external format. (e.g., 3^SMITH,JOSEPH L.)

VAIN(3) The TREATING SPECIALTY assigned to the patient at the date/time requested in internal^external format. (e.g., 19^GERIATRICS)

VAIN(4) The WARD LOCATION to which the patient was assigned at the date/time requested in internal^external format. (e.g., 27^IBSICU)

VAIN(5) The ROOM-BED to which the patient was assigned at the date/time requested in external format. (e.g., 123-B)

VAIN(6) This will return a "1" in the first piece if the patient is in a bed status; otherwise, a "0" will be returned. A non-bed status is made based on the last transfer type to a non-bed status, (i.e., authorized absence, unauthorized absence, etc.) The second piece will contain the name of the last transfer type should one exist. (e.g., 1^FROM AUTHORIZED ABSENCE)

VAIN(7) The ADMISSION DATE/TIME for the patient in internal^external format. (e.g., 2870213.0915^FEB 13,1987@09:15)

VAIN(8) The ADMISSION TYPE for the patient in internal^external format. (e.g., 3^DIRECT)

VAIN(9) The ADMITTING DIAGNOSIS for the patient. (e.g., PSYCHOSIS)

VAIN(10) The internal entry number of the PTF record corresponding to this admission. (e.g., 2032)

VAIN(11) The ATTENDING PHYSICIAN in internal^external format. (e.g., 25^SMITH,JOHN)

VAERR The error flag will have one of the following values.
0 -- no errors encountered
1 -- error encountered - DFN or ^DPT(DFN,0) is not defined

INP^VADPT	VAIN(1)	VAIN("AN")
	VAIN(2)	VAIN("DR")
	VAIN(3)	VAIN("TS")
	VAIN(4)	VAIN("WL")
	VAIN(5)	VAIN("RB")
	VAIN(6)	VAIN("BS")
	VAIN(7)	VAIN("AD")
	VAIN(8)	VAIN("AT")
	VAIN(9)	VAIN("AF")
	VAIN(10)	VAIN("PT")

	VAIN(11)	VAIN("AP")

2.3.1.3.2 IN5^VADPT

This entry point will return data related to an inpatient episode.

Input Variables:

DFN See INP^VADPT for details.

VAHOW See INP^VADPT for details.

VAROOT See INP^VADPT for details.

VAIP("D") This optional variable can be defined as follows:

VAIP("D") =VA FileMan date in internal format. If the patient was an inpatient at the date/time passed, movement data pertaining to that date/time will be returned.

VAIP("D") ="LAST" Movement data pertaining to the last movement on file, regardless if patient is a current inpatient.

VAIP("D") =valid date without time Will return movement data if patient was an inpatient at any time during the day on the date that was passed in.

VAIP("D") If not passed, will return movement data if the patient is inpatient now.

VAIP("L") This optional variable, when passed, will include lodgers movements in the data.

VAIP("V") Can be defined as the variable used instead of VAIP(.(e.g., VAIP("V")="SD")

VAIP("E") This optional variable is defined as the internal file number of a specific movement. If this is defined, VAIP("D") is ignored.(e.g., VAIP("E")=123445)

VAIP("M") This optional variable can be passed as a "1" or a "0" (or null).

VAIP("M")":

0 - The array returned will be based on the admission movement associated with the movement date/time passed.

1 - The array returned will be based on the last movement associated with the date/time passed.

Output Variables:

VAIP(1) The INTERNAL FILE NUMBER [IFN] of the movement found for the specified date/time. (e.g., 231009)

VAIP(2) The TRANSACTION TYPE of the movement in internal^external format where:

1=admission
2=transfer
3=discharge
4=check-in lodger
5=check-out lodger
6=specialty transfer (e.g., 3^DISCHARGE)

VAIP(3) The MOVEMENT DATE/TIME in internal^external date format. (e.g., 2880305.09^MAR 5,1988@09:00)

VAIP(4) The TYPE OF MOVEMENT in internal^external format. (e.g., 4^INTERWARD TRANSFER)

VAIP(5) The WARD LOCATION to which patient was assigned with that movement in internal^external format. (e.g., 32^1B-SURG)

VAIP(6) The ROOM-BED to which the patient was assigned with that movement in internal^external format. (e.g., 88^201-01)

VAIP(7) The PRIMARY CARE PHYSICIAN assigned to the patient in internal^external format. (e.g., 3^SMITH,JACOB J.)

VAIP(8) The TREATING SPECIALTY assigned with that movement in internal^external format. (e.g., 98^OPTOMETRY)

VAIP(9) The DIAGNOSIS assigned with that movement. (e.g., UPPER GI BLEEDING)

VAIP(10) This will return a "1" in the first piece if the patient is in a bed status; otherwise, a "0" will be returned. A non-bed status is made based on the last transfer type, if one exists, and a transfer to a non-bed status, (i.e., authorized absence, unauthorized absence, etc.) The second piece will contain the name of the last transfer type should one exist. (e.g., 1^FROM AUTHORIZED ABSENCE)

VAIP(11) If patient is in an absence status on the movement date/time, this will return the EXPECTED RETURN DATE from absence in internal^external format. (e.g., 2880911^SEP 11,1988)

VAIP(12) The internal entry number of the PTF record corresponding to this admission. (e.g., 2032)

VAIP(13) The INTERNAL FILE NUMBER of the admission associated with this movement. (e.g., 200312)

VAIP(13,1) The MOVEMENT DATE/TIME in internal^external format. (e.g., 2881116.08^NOV 16,1988@08:00)

VAIP(13,2) The TRANSACTION TYPE in internal^external format. (e.g., 1^ADMISSION)

VAIP(13,3) The MOVEMENT TYPE in internal^external format. (e.g., 15^DIRECT)

VAIP(13,4) The WARD LOCATION associated with this patient with this movement in internal^external format. (e.g., 5^7BSCI)

VAIP(13,5) The PRIMARY CARE PHYSICIAN assigned to the patient for this move-ment in internal^external format. (e.g., 16^JONES, CHARLES C)

VAIP(13,6) The TREATING SPECIALTY for the patient for this movement in internal^external format. (e.g., 3^NEUROLOGY)

VAIP(14) The INTERNAL FILE NUMBER of the last movement associated with this movement. (e.g., 187612)

VAIP(14,1) The MOVEMENT DATE/TIME in internal^external format. (e.g., 2881116.08^NOV 16,1988@08:00)

VAIP(14,2) The TRANSACTION TYPE in internal^external format. (e.g., 2^TRANSFER)

VAIP(14,3) The MOVEMENT TYPE in internal^ external format. (e.g., 4^INTERWARD TRANSFER)

VAIP(14,4) The WARD LOCATION associated with this patient with this movement in internal^external format. (e.g., 5^7BSCI)

VAIP(14,5) The PRIMARY CARE PHYSICIAN assigned to the patient for this movement in internal^external format. (e.g., 16^JONES, CHARLES C)

VAIP(14,6) The TREATING SPECIALTY for the patient for this movement in internal^external format. (e.g., 3^NEUROLOGY)

VAIP(15) The INTERNAL FILE NUMBER of the movement which occurred immediately prior to this one, if one exists. (e.g., 153201)

VAIP(15,1) The MOVEMENT DATE/TIME in internal^external format. (e.g., 2881116.08^NOV 16,1988@08:00)

VAIP(15,2) The TRANSACTION TYPE in internal^external format. (e.g., 2^TRANSFER)

VAIP(15,3) The MOVEMENT TYPE in internal^ external format. (e.g., 4^INTERWARD TRANSFER)

VAIP(15,4) The WARD LOCATION associated with this patient with this movement in internal^external format. (e.g., 5^7BSCI)

VAIP(15,5) The PRIMARY CARE PHYSICIAN assigned to the patient for this movement in internal^external format. (e.g., 16^JONES, CHARLES C)

VAIP(15,6) The TREATING SPECIALTY for the patient for this movement in internal^external format. (e.g., 3^NEUROLOGY)

VAIP(16) The INTERNAL FILE NUMBER of the movement which occurred immediately following this one, if one exists. (e.g., 146609)

VAIP(16,1) The MOVEMENT DATE/TIME in internal^external format. (e.g., 2881116.08^NOV 16,1988@08:00)

VAIP(16,2) The TRANSACTION TYPE in internal^external format. (e.g., 2^TRANSFER)

VAIP(16,3) The MOVEMENT TYPE in internal^ external format. (e.g., 4^INTERWARD TRANSFER)

VAIP(16,4) The WARD LOCATION associated with this patient with this movement in internal^external format. (e.g., 5^7BSCI)

VAIP(16,5) The PRIMARY CARE PHYSICIAN assigned to the patient for this movement in internal^external format. (e.g., 16^JONES, CHARLES C)

VAIP(16,6) The TREATING SPECIALTY for the patient for this movement in internal^external format. (e.g., 3^NEUROLOGY)

VAIP(17) The INTERNAL FILE NUMBER of the discharge associated with this movement. (e.g., 1902212)

VAIP(17,1) The MOVEMENT DATE/TIME in internal^external format. (e.g., 2881116.08^NOV 16,1988@08:00)

VAIP(17,2) The TRANSACTION TYPE in internal^external format.(e.g., 3^DISCHARGE)

VAIP(17,3) The MOVEMENT TYPE in internal^external format. (e.g., 16^REGULAR)

VAIP(17,4) The WARD LOCATION associated with this patient for this movement in internal^external format. (e.g., 5^7BSCI)

VAIP(17,5) The PRIMARY CARE PHYSICIAN assigned to the patient for this movement in internal^external format. (e.g., 16^JONES, CHARLES C)

VAIP(17,6) The TREATING SPECIALTY for the patient for this movement in internal^external format. (e.g., 3^NEUROLOGY)

VAIP(18) The ATTENDING PHYSICIAN assigned to the patient for this movement in internal^external format. (e.g., 25^SMITH,JOHN)

VAERR The error flag will have one of the following values.

0 -- no errors encountered
1 -- error encountered - DFN or ^DPT(DFN,0) is not defined

VASD("C",Clinic IFN) Can be set up to contain only those internal file entries from the Hospital Location file for clinics which you would like to see appointments for this particular patient. You may define this array with just one clinic or with many. If you do not define this variable, it will be assumed that you want appointments for this patient in all clinics returned.

IN5^VADPT	VAIP(1)	VAIP("MN")
	VAIP(2)	VAIP("TT")
	VAIP(3)	VAIP("MD")
	VAIP(4)	VAIP("MT")
	VAIP(5)	VAIP("WL")
	VAIP(6)	VAIP("RB")
	VAIP(7)	VAIP("DR")
	VAIP(8)	VAIP("TS")
	VAIP(9)	VAIP("MF")
	VAIP(10)	VAIP("BS")
	VAIP(11)	VAIP("RD")
	VAIP(12)	VAIP("PT")
	VAIP(13)	VAIP("AN")
	VAIP(13,#)	VAIP("AN",#)
	VAIP(14)	VAIP("LN")
	VAIP(14,#)	VAIP("LN",#)
	VAIP(15)	VAIP("PN")
	VAIP(15,#)	VAIP("PT",#)
	VAIP(16)	VAIP("NN")
	VAIP(16,#)	VAIP("NN",#)
	VAIP(17)	VAIP("DN")
	VAIP(17,#)	VAIP("DN",#")

	VAIP(18)	VAIP("AP")

2.3.1.3.3 ADM^VADPT2

This returns the internal file number of the admission movement. If VAINDT is not defined, this will use "NOW" for the date/time.

Input Variables:

DFN This required variable is the internal entry number in the patient file

VAINDT This optional variable may be set to a past date/time for which the programmer wishes to know the patient's inpatient status. This must be passed as an internal VA FileMan date/time format. (e.g., 2880101.08)

Output Variables:

VADMVT Returns the internal file number of the admission movement.

VAERR The error flag will have one of the following values.

0 -- no errors encountered
1 -- error encountered - DFN or ^DPT(DFN,0) is not defined

2.3.1.3.4 KVAR^VADPT

This call is used to remove all variables defined by the VADPT routine. The programmer should elect to utilize this call to remove the arrays which were returned by VADPT.

2.3.1.3.5 KVA^VADPT

This call is used as above and will also kill the VA("BID") and VA("PID") variables.

2.3.2 PCC Data

Function Calls Used to Retrieve Data Items from PCC/Patient Files

This section describes the many function calls available to developers for retrieving data from PCC or the Patient file. The PCC files include the Visit file and all "V" files. If you know the visit IEN, you may use any of these calls to retrieve a particular PCC data item. To retrieve patient demographic information, you only need to know the patient DFN. This document describes published entry points from both the AUPN - IHS DICTIONARIES (PATIENT) package and from the APCL - PCC MANAGEMENT REPORTS package.

2.3.2.1 Visit File Functions

2.3.2.1.1 Routine - APCLV

For each of the function calls in APCLV, the user is required to pass the visit ien as the first parameter. Data for the following functions come from the VISIT file, file 9000010.

VD(v,f) Returns VISIT DATE for visit v in format f

arguments

v - visit ien

f - optional format; if null, returns internal FileMan format of visit date

I - internal FileMan format

E - external written-out format (MAR 05, 1995)

S - slash format (03/05/95)

examples

W $$VD^APCLV(1234,"I") => 2950305

W $$VD^APCLV(1234,"E") => MAR 05,1995

W $$VD^APCLV(1234,"S") => 03/05/95

VDTM(v,f) Returns VISIT DATE AND TIME for visit v in format f.

arguments

v - visit ien

f - optional format; if null, returns internal FileMan format of visit date

I - internal FileMan format

S - slash format (03/05/95)

E - external written-out format (MAR 05, 1995)

examples

W $$VDTM^APCLV(1234,"I") => 2950305.1300

W $$VDTM^APCLV(1234,"E") => MAR 05,1995 1:00pm

W $$VDTM^APCLV(1234,"S") => 03/05/95 1:00pm

TIME(v,f) Returns TIME of visit v in format f

arguments

v - visit ien

f - optional format; if null, returns internal FileMan format of visit time

I - internal FileMan format

P - am/pm

E - external format

examples

W $$TIME^APCLV(1234, "I") => 1300

W $$TIME^APCLV(1234, "P") => 1:00pm

W $$TIME^APCLV(1234, "E") => 1:00

DOW(v,f) Returns DAY OF WEEK for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of visit date
I - internal number of day 0-6 for Sunday to Saturday
E - external written-out format

examples

W $$DOW^APCLV(1234,"I") => 1
W $$DOW^APCLV(1234,"E") => Monday

TYPE(v,f) Returns TYPE OF VISIT for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of type of visit
I - internal FileMan format
E - external format

examples

W $$TYPE^APCLV(1234,"I") => I
W $$TYPE^APCLV(1234,"E") => IHS

PATIENT(v,f) Returns PATIENT for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of PATIENT entry (DFN)
I - internal FileMan format
E - external format
C - chart number (asufac_chart number)

examples

W $$PATIENT^APCLV(1234,"I") => 34
W $$PATIENT^APCLV(1234,"E") => JONES,LORI
W $$PATIENT^APCLV(1234,"C") => 000101000987

COMM(v,f) Returns COMMUNITY OF RESIDENCE of the patient for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of community
I - internal FileMan format
E - external format
C - STCTYCOM code

examples

W $$COMM^APCLV(1234,"I") => 1234
W $$COMM^APCLV(1234,"E") => TUCSON
W $$COMM^APCLV(1234,"C") => 04023876

CHART(v) Returns ASUFAC_HRN of patient for visit v

arguments

v - visit ien
will return ASUFAC_HRN for the HRN at the location of the visit. If no chart exists at that location, the ASUFAC_HRN will be for DUZ(2). Otherwise, a blank will be returned. The HRN is left zero filled to 6 digits.

examples

W $$CHART^APCLV(1234) =>000101003457

LOCENC(v,f) Returns LOCATION OF ENCOUNTER for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of location entry (ien)
I - internal FileMan format
E - external format
C - IHS ASUFAC

examples

W $$LOCENC^APCLV(1234,"I") => 4585
W $$LOCENC^APCLV(1234,"E") => SELLS HOSPITAL
W $$LOCENC^APCLV(1234,"C") => 000101

SC(v,f) Returns SERVICE CATEGORY for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of service category
I - internal FileMan format
E - external format

examples

W $$SC^APCLV(1234,"I") => A
W $$SC^APCLV(1234,"E") => AMBULATORY

CLINIC(v,f) Returns CLINIC for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of clinic entry (ien)
I - internal FileMan format
E - external format
C - IHS Clinic Code

examples

W $$CLINIC^APCLV(1234,"I") => 1
W $$CLINIC^APCLV(1234,"E") => GENERAL
W $$CLINIC^APCLV(1234,"C") => 01

DLM(v,f) Returns DATE LAST MODIFIED for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of visit date
I - internal FileMan format
E - external written-out format (MAR 05, 1995)
S - slash format (03/05/95)

examples

W $$DLM^APCLV(1234,"I") => 2950305
W $$DLM^APCLV(1234,"E") => MAR 05,1995
W $$DLM^APCLV(1234,"S") => 03/05/95

DVEX(v,f) Returns DATE VISIT EXPORTED for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of visit date
I - internal FileMan format
E - external written-out format (MAR 05, 1995)
S - slash format (03/05/95)

examples

W $$DVEX^APCLV(1234,"I") => 2950305
W $$DVEX^APCLV(1234,"E") => MAR 05,1995
W $$DVEX^APCLV(1234,"S") => 03/05/95

APWI(v,f)Returns APPT/WALK IN code from visit file for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of visit date
I - internal FileMan format
E - external written-out format (APPOINTMENT)

examples

W $$APWI^APCLV(1234,"I") => A
W $$APWI^APCLV(1234,"E") => APPOINTMENT

EM(v,f) Returns EVALUATION AND MANAGEMENT CPT for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of CPT CODE (ien)
I - internal FileMan format
E - external format, description of code
C - CPT Code

examples

W $$EM^APCLV(1234,"I") => 99211
W $$EM^APCLV(1234,"E") => OFFICE VISIT, EXTENDED
W $$EM^APCLV(1234,"C") => 99211

CODT(v,f) Returns CHECK OUT DATE AND TIME for visit v in format f

arguments

v - visit ien

f - optional format; if null, returns internal FileMan format of visit date

I - internal FileMan format

E - external written-out format (MAR 05, 1995)

S - slash format (03/05/95)

examples

W $$CODT^APCLV(1234,"I") => 2950305

W $$CODT^APCLV(1234,"E") => MAR 05,1995

W $$CODT^APCLV(1234,"S") => 03/05/95

LS(v,f)Returns LEVEL OF SERVICE for visit v in format f

arguments

v - visit ien

f - optional format; if null, returns internal FileMan format of type of visit

I - internal FileMan format

E - external format

examples

W $$LS^APCLV(1234,"I") => B

W $$LS^APCLV(1234,"E") => BRIEF

APDT(v,f)Returns APPT DATE AND TIME from visit file for visit v in format f

arguments

v - visit ien

f - optional format; if null, returns internal FileMan format of visit date

I - internal FileMan format

E - external written-out format (MAR 05, 1995)

S - slash format (03/05/95)

examples

W $$APDT^APCLV(1234,"I") => 2950305

W $$APDT^APCLV(1234,"E") => MAR 05,1995

W $$APDT^APCLV(1234,"S") => 03/05/95

OUTSL(v) Returns OUTSIDE LOCATION from visit file for visit v.

arguments

v - visit ien

examples

W $$OUTSL^APCLV(1234) => WALGREEN'S PHARMACY

2.3.2.2 Inpatient Information

The following data items are extracted from the V HOSPITALIZATION file and will be present only if the visit is a Hospitalization (service category = H).

2.3.2.2.1 Routine - APCLV

ADMSERV(v,f) Returns ADMITTING SERVICE for visit v in format f

arguments

v - visit ien, must be a hospitalization

f - optional format; if null, returns internal FileMan format of treating specialty entry (ien)

I - internal FileMan format

E - external format

C - IHS Code

examples

W $$ADMSERV^APCLV(1234,"I") => 3

W $$ADMSERV^APCLV(1234,"E") => PEDIATRICS

W $$ADMSERV^APCLV(1234,"C") => 15

DSCHSERV(v,f) Returns DISCHARGE SERVICE for visit v in format f

arguments

v - visit ien

f - optional format; if null, returns internal FileMan format of treating specialty entry (ien)

I - internal FileMan format

E - external format

C - IHS Code

examples

W $$DSCHSERV^APCLV(1234,"I") => 3

W $$DSCHSERV^APCLV(1234,"E") => PEDIATRICS

W $$DSCHSERV^APCLV(1234,"C") => 15

ADMTYPE(v,f) Returns ADMISSION TYPE for visit v in format f

arguments

v - visit ien, must be a hospitalization

f - optional format; if null, returns internal FileMan format of admission type entry (ien)

I - internal FileMan format

E - external format

C - IHS Code

examples

W $$ADMTYPE^APCLV(1234,"I") => 3

W $$ADMTYPE^APCLV(1234,"E") => DIRECT

W $$ADMTYPE^APCLV(1234,"C") => 02

DSCHTYPE(v,f) Returns DISCHARGE TYPE for visit v in format f

arguments

v - visit ien, must be hospitalization

f - optional format; if null, returns internal FileMan format of treating specialty entry (ien)

I - internal FileMan format

E - external format

C - IHS Code

examples

W $$DSCHTYPE^APCLV(1234,"I") => 2
W $$DSCHTYPE^APCLV(1234,"E") => AWOL
W $$DSCHTYPE^APCLV(1234,"C") => 03

DSCHDATE(v,f) Returns DISCHARGE DATE for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of visit date
I - internal FileMan format
E - external written-out format (MAR 05, 1995)
S - slash format (03/05/95)

examples

W $$DSCHDATE^APCLV(1234,"I") => 2950305
W $$DSCHDATE^APCLV(1234,"E") => MAR 05,1995
W $$DSCHDATE^APCLV(1234,"S") => 03/05/95

CONSULTS(v) Returns NUMBER OF CONSULTS for hospitalization for visit v

arguments

v - visit ien

examples

W $$CONSULTS^APCLV(1234) => 2

LOS(v) Returns LOS for visit hospitalization visit v

arguments

v - visit ien

examples

W $$LOS^APCLV(1234) => 4

FACTX(v,f) Returns FACILITY TRANSFERRED TO for visit v in format f

arguments

v - visit ien, must be a hospitalization
f - optional format; if null, returns internal FileMan format of facility
I - internal FileMan format
E - external format
C - IHS ASUFAC if an IHS facility

examples

W $$FACTX^APCLV(1234,"I") => DIC(4;123)
W $$FACTX^APCLV(1234,"E") => SELLS HOSPITAL
W $$FACTX^APCLV(1234,"C") => 000101

ATTPHY(v,f) Returns ATTENDING PHYSICIAN for visit v in format f

arguments

v - visit ien, must be a hospitalization

f - optional format; if null, returns internal FileMan format of provider entry (ien)
I - internal FileMan format
T - provider's initials
A - affiliation, internal format; e.g., 1
B - affiliation, external format; e.g., IHS
C - provider code; e.g., JDS
D - provider's discipline code; e.g., 01
E - provider's discipline, external format; e.g., PHYSICIAN
F - provider's discipline, internal format; e.g., 1 (ien of provider class)
N - provider's name; e.g., SMITH,JOHN DAVID
O - provider's affiliation_discipline; e.g., 101
P - provider's affiliation_discipline_code; e.g., 101JDS

examples
W $$ATTPHY^APCLV(1234,"P") => 101JDS
W $$ATTPHY^APCLV(1234,"E") => PHYSICIAN

ADMDX(v,f) Returns ADMITTING DIAGNOSIS for visit v in format f

arguments
v - visit ien, must be a hospitalization
f - optional format; if null, returns ICD Diagnosis code
I - internal FileMan format (ICD9 ien)
E - external of ICD9 code (HYPERTENSION)
C - ICD Code; e.g., 250.00

examples
W $$ADMDX^APCLV(1234,"I") => 2477
W $$ADMDX^APCLV(1234,"E") => HYPERTENSION
W $$ADMDX^APCLV(1234,"C") => 250.00

2.3.2.3 Visit Measurement Information

2.3.2.3.1 Routine – APCLV

The following data is extracted from the V Measurement file, file 9000010.01. One or more measurements may have been taken on any one visit; therefore, an array is passed back that contains all of the measurements taken on that visit.

Call to Pass Back an Array of Data from V Measurement

FORMAT: S E=$$PCCVF^APCLV(v,t,f)

An array (APCLV) will be passed back that contains a list of all measurements that were taken on visit v. The information passed back will be in format f. If an error was encountered, E will be passed back as one of the following error codes:

1 - no value for t, type of data wanted
2 - no value for f, format of data wanted

3 - no value for v, visit
4 - invalid visit ien passed
5 - invalid type of data passed in t

It is the caller's responsibility to kill array APCLV before and after the call to PCCVF^APCLV.

arguments

v - visit ien

t - is defined as the type of V File you want. For measurements, it must be the word MEASUREMENT.

f - is defined as the format of the data you want and it will depend on the V file being looked at. You may specify several numbers in the string: 7;8 or 1;3;7;8, for example. If you specify several items in f, they will be returned in the corresponding "^ piece of APCLV. In the case of 1;3;7;8, the value for 1 will be in the first piece, the value for 3 in the second piece, the value for 7 in the third piece, and the value for 8 in the fourth piece. Each value of f is described below.

1 - visit date and time in internal FileMan format; e.g., 2950402
2 - visit date and time in external FileMan format, e.g., 04/02/95
3 - internal IEN of patient; e.g., 234
4 - external name of patient; e.g., SMITH,JOHN
5 - internal value of measurement type; e.g., 4
6 - external value of measurement type; e.g., BLOOD PRESSURE
7 - coded value of measurement type; e.g., BP
8 - value of measurement; e.g., 120/90
9 - CPT Code for measurement type (if available)
09 - each of the above items in a "^" pieced string where each piece is equal to the number above; e.g., 2950402^04/02/95^23^SMITH,JOHN^1^ WEIGHT^WT^175^^^^^^"

examples

S E= $$PCCVF^APCLV(1234,"MEASUREMENT","7")
Will return:
APCLV(1)=BP
APCLV(2)=WT
APCLV(3)=HT

S E=PCCVF^APCLV(1234,"MEASUREMENT","7;8")
Will return:
APCLV(1)=BP^120/80
APCLV(2)=WT^125
APCLV(3)=HT^65

S E=PCCVF^APCLV(1234,"MEASUREMENT","99")
Will return:

APCLV(1)=2950402^04/02/95^123^SMITH,JOHN^4^BL
OOD PRESSURE ^BP^120/90
APCLV(2)=2950402^04/02/95^123^SMITH,JOHN^2^WE
IGHT^WT^125

NMSR(v) Returns NUMBER OF MEASUREMENTS taken on visit v

arguments

v - visit ien

examples

W $$NMSR^APCLV(1234) => 3

2.3.2.4 Visit Provider Information

2.3.2.4.1 Routine - APCLV

The following data is extracted from the V PROVIDER file, file 9000010.06. One or more providers may be listed for any one visit; therefore, an array is passed back that contains all of the providers for that visit.

Call to Pass Back an Array of Data from V Provider

FORMAT: S E=$$PCCVF^APCLV(v,t,f)

You will be passed back an array (APCLV) that contains a list of all providers that were seen on visit v. The information passed back will be in format f. If an error is encountered, E will be passed back as one of the following error codes:

1 - no value for t, type of data wanted
2 - no value for f, format of data wanted
3 - no value for v, visit
4 - invalid visit ien passed
5 - invalid type of data passed in t

It is the caller's responsibility to kill array APCLV before and after the call to PCCVF^APCLV.

arguments

v - visit ien

t - is defined as the type of V file you want. For providers, it must be the word PROVIDER.

f - is defined as the format of the data you want and it will depend on the V file being looked at. You may specify several numbers in the string: 7;8 or 1;3;7;8, for example. If you specify several items in f, they will be returned in the corresponding "^ piece of APCLV. In the case of 1;3;7;8, the value for 1 will be in the first piece, the value for 3 in the second piece, the value for 7 in the third piece, and the value for 8 in the fourth piece. Each value of f is described below.

1 - visit date and time in internal FileMan format; e.g., 2950402
2 - visit date and time in external FileMan format; e.g., 04/02/95
3 - internal ien of patient; e.g., 234
4 - external name of patient; e.g., SMITH,JOHN
5- internal value of provider entry; e.g., 4
6 - provider's initials; e.g., JDS
7 - affiliation, internal format; e.g., 1
8 - affiliation, external format; e.g., IHS
9 - provider code; e.g., JDS
10 - provider's discipline code; e.g., 01
11 - provider's discipline, external format; e.g., PHYSICIAN
12 - provider's discipline, internal format; e.g., 1 (ien of provider class)
13- provider's name; e.g., SMITH,JOHN DAVID
14 - provider's affiliation_discipline; e.g., 101
15 - provider's affiliation_discipline_code; e.g., 101JDS
16 - primary or secondary; e.g., P
17 - primary or secondary, external; e.g., PRIMARY
18 - operating/attending, internal; e.g., O
19 - operating/attending, external; e.g., Operating
99 - each of the above items in a "^" pieced string where each piece is equal to the number above; e.g., 2950402^04/02/95^23^JONES,MARY ^34^JS^1^IHS^JS^01^PHYSICIAN^1^SMITH,JOHN^101^101JS

examples

S E=PCCVF^APCLV(1234,"PROVIDER","7")
Will return:
APCLV(1)=1
APCLV(2)=1
APCLV(3)=1

S E=PCCVF^APCLV(1234,"PROVIDER","7;8")
Will return:
APCLV(1)=1^IHS
APCLV(2)=1^IHS

S E=PCCVF^APCLV(1234,"PROVIDER","99")
Will return:
APCLV(1)="2960125^1/25/96^50^SMITH,JIMALLEN^ 618^LAB^3^TRIBAL^LAB^53^COMMUNITY HEALTH REP.^47^BUTCHER,LORI ANN^353^353LAB^P^PRIMARY^^"

PRIMPROV(v,f) Returns PRIMARY PROVIDER for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of provider entry (ien)
I - internal FileMan format

T - provider's initials
A - affiliation, internal format; e.g., 1
B - affiliation, external format; e.g., IHS
C - provider code; e.g., JDS
D - provider's discipline code; e.g., 01
E - provider's discipline, external format; e.g., PHYSICIAN
F - provider's discipline, internal format; e.g., 1 (ien of provider class)
N - provider's name; e.g., SMITH,JOHN DAVID
O - provider's affiliation_discipline; e.g., 101
P - provider's affiliation_discipline_code; e.g., 101JDS

examples

W $$PRIMPROV^APCLV(1234,"P") => 101JDS
W $$PRIMPROV^APCLV(1234,"E") => PHYSICIAN

SECPROV(v,f,n) Returns the nth SECONDARY PROVIDER for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of provider entry (ien)
I - internal FileMan format
T - provider's initials
A - affiliation, internal format; e.g., 1
B - affiliation, external format; e.g., IHS
C - provider code; e.g., JDS
D - provider's discipline code; e.g., 01
E - provider's discipline, external format; e.g., PHYSICIAN
F - provider's discipline, internal format; e.g., 1 (ien of provider class)
N - provider's name; e.g., SMITH,JOHN DAVID
O - provider's affiliation_discipline; e.g., 101
P - provider's affiliation_discipline_code; e.g., 101JDS

examples

W $$SECPROV^APCLV(1234,"P") => 101JDS
W $$SECPROV^APCLV(1234,"E") => PHYSICIAN

MIDWIFE(v) Returns 1 if one of the providers is a midwife

arguments

v - visit ien

examples

W $$MIDWIFE^APCLV(1234) => 1

2.3.2.5 Purpose of Visit Information

2.3.2.5.1 Routine - APCLV

The following data is extracted from the V POV file, file 9000010.07. There may be one or more POVs for any one visit; therefore, an array is passed back that contains all of the POVs for that visit.

Call to Pass Back an Array of Data from V Pov

FORMAT: S E=$$PCCVF^APCLV(v,t,f)

The user will be passed back an array (APCLV) that contains a list of all POVs that were seen on visit v. The information passed backed will be in format f. If an error was encountered, E will be passed back as one of the following error codes:

1 - no value for t, type of data wanted
2 - no value for f, format of data wanted
3 - no value for v, visit
4 - invalid visit ien passed
5 - invalid type of data passed in t

It is the caller's responsibility to kill array APCLV before and after the call to PCCVF^APCLV.

arguments

v - visit ien

t - is defined as the type of V File you want; for POVs, it must be the word POV.

f - is defined as the format of the data you want and will depend on the V file being looked at. You may specify several numbers in the string: 7;8 or 1;3;7;8, for example. If you specify several items in f, they will be returned in the corresponding "^ piece of APCLV. In the case of 1;3;7;8, the value for 1 will be in the first piece, the value for 3 in the second piece, the value for 7 in the third piece, and the value for 8 in the fourth piece. Each value of f is described below.

1 - visit date and time in internal FileMan format; e.g., 2950402
2 - visit date and time in external FileMan format; e.g., 04/02/95
3 - internal IEN of patient; e.g., 234
4 - external name of patient; e.g., SMITH,JOHN
5 - internal value of POV entry; e.g., 4
6 - external ICD code text; e.g., HYPERTENSION
7 - ICD code; e.g., 250.00
8 - APC Recode; e.g., 020
9 - Cause of DX, internal; e.g., 1
10 - Cause of DX, external; e.g., Alcohol-related
11 - Cause of Injury; e.g., E900.2
12 - Place of Injury, internal; e.g., A
13 - Place of Injury, external; e.g., HOME-INSIDE
14 - Provider Narrative

15 - Primary/Secondary code, internal; e.g., P
16 - primary or secondary; e.g., PRIMARY
17 - Date of injury; e.g., 09/01/95
18 - Stage; e.g., 4
19 - Modifier, internal; e.g., 1
20 - Modifier, external; e.g., Rule Out
99 - each of the above items in a "^" pieced string where each piece is equal to the number above.

examples

S E=$$PCCVF^APCLV(72555,"POV",99)
APCLV(1)="2960125^1/25/96^50^SMITH,JAMES A^101531^DM UNCOMPL/T-II/NIDDM,NS UNCON^250.00^080^^^^^^DIABETES MELLITUS:PATIENT CARE - CHR^^^^^^"

PRIMPOV(v,f) Returns PRIMARY POV for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of POV entry (ien)
I - internal FileMan format
E - ICD9 text
C - ICD9 code
A - APC RECODE
D - Cause of Dx (internal)
J - Cause of Injury code
P - Place of Injury (code)
N - Provider Narrative

examples

W $$PRIMPOV^APCLV(1234,"P") => A
W $$PRIMPOV^APCLV(1234,"E") => 250.00

SECPOV(v,f,n) Returns the nth SECONDARY POV for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of POV entry (ien)
I - internal FileMan format
E - ICD9 text
C - ICD9 code
A - APC RECODE
D - Cause of Dx (internal)
J - Cause of Injury code
P - Place of Injury (code)
N - Provider Narrative

examples

W $$SECPOV^APCLV(1234,"P") => A

W $$SECPOV^APCLV(1234,"E") => 311

2.3.2.6 Miscellaneous PCC Calls

Information for these calls is taken from various V files.

NLAB(v) Returns NUMBER OF LABS done on visit v

arguments

v - visit ien

examples

W $$NLAB^APCLV(1234) => 12

NRX(v) Returns NUMBER OF MEDICATIONS prescribed on visit v

arguments

v - visit ien

examples

W $$NRX^APCLV(1234) => 3

ACTTIME(v) Returns ACTIVITY TIME for visit v

arguments

v - visit ien

examples

W $$ACTTIME^APCLV(1234) => 10

TRAVTIME(v) Returns TRAVEL TIME for visit v

arguments

v - visit ien

examples

W $$TRAVTIME^APCLV(1234) => 25

CHSCOST(v,f) Returns TOTAL COST from V CHS file

arguments

v - visit ien

examples

W $$CHSCOST^APCLV(1234) => 12,000

PROC(v,f,n) Returns the nth PROCEDURE for visit v in format f

arguments

v - visit ien

f - optional format; if null, returns internal FileMan format of PROCEDURE entry (ien)

I - ien of ICD0 code

E - external of ICD0 code; e.g., appendectomy

C - ICD0 code; e.g., 44.10

P - CPT CODE

T - CPT internal ien

D - date of procedure/internal FileMan format
G - date of procedure/external format
F - infection Y/N
R - operating provider affl_disc_code
X- dx done for
N - provider's narrative

examples

W $$PROC^APCLV(1234,"R",1)=> 101JDS
W $$PROC^APCLV(1234,"E",1)=> APPENDECTOMY

IMM(v,f,n) Returns the nth IMMUNIZATION for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of IMMUNIZATION entry (ien)
I - ien of immunization entry
E - immunization external format - OPV
C - immunization code - 06
P - CPT CODE
S - series

examples

W $$IMM^APCLV(1234,"C",1) => 06
W $$IMM^APCLV(1234,"E",1) => OPV

DENT(v,f,n) Returns the nth V DENTAL ENTRY for visit v in format f

arguments

v - visit ien
f - optional format; if null, returns internal FileMan format of ADA CODE entry (ien)
I - ien of ADA entry
E - ADA CODE external format
C - ADA code - 0110

examples

W $$DENT^APCLV(1234,"C",1) => 0110

2.3.3 Visit Creation API

2.3.3.1 New API for Creating/Selecting PCC Visit

Background:

Prior to this new API, several RPMS packages made calls to PCC or VA Visit Tracking to create visits. In some cases, creating a new visit was not appropriate since one already existed. Some applications always created the main patient visit, while others created an "ancillary" visit to be merged with the main visit at a later time. In other cases, the same application would create duplicate visits, even when it

had created the previous one. With EHR, providers were confused as to whether they were supposed to create a new visit or use one already there.

We have designed and created a new API to be used by those applications needing to create a PCC visit where one of the above mentioned problems exist. This new API can be used to create main visits as well as "ancillary" ones. It will also return any existing visits that meet the criteria, so duplicates can be avoided. Each calling application decides what to do with the results it receives from the API. This API, by itself, does not solve the various problems. With its use by various RPMS packages, improvements will occur.

Primary call is made to a routine in the PCC namespace. It checks to make sure PIMS v5.3 is running and the BSDAPI4 routine exists (released with patch 1002). Then it simply calls the PIMS routine. This keeps all visit creation calls inside PCC but keeps support of the Scheduling logic added in this new API inside PIMS.

Logic Flow:

1. If sending application says force adding a new visit, just create visit and quit with returning array.

2. Attempt to find visits that match incoming data.
    a. Match on date and optional time range (i.e. within 60 minutes)
    b. Match on location of encounter (Visit field .06)
    c. Match on visit type (Visit field .03)
    d. Match on service category (Visit field .07)
    e. If provider is sent, match on provider
    f. If not called in ancillary mode, match clinic code or hospital location if sent; if matches on clinic code & existing visit is for "triage" clinic, it's a match
    g. Return # of visits that match and array with visit internal entry numbers

3. If calling application sent appointment date & wants a visit added if match not found (Never Add not set), drops to check-in code, step 9.

4. If only one visit was found, quit with returning array.

5. If more than one visit found, quit with returning array. Calling application decides what to do next. Next step might be to call API again with "force add" set.

6. If no visit found, quit if in "Never Add" mode".

7. If calling in ancillary mode, either find another ancillary visit already there or create one with a time of noon, then quit with returning array.

8. If no appointment date/time sent, just create a visit and quit with array.

9. If patient already has appointment at that date/time, call check-in code which will also create visit then quit with returning array.

10. Otherwise, create walk-in appointment which is checked in and visit created. Quit with returning array.

Public Entry Points:

Called using **D GETVISIT^APCDAPI4(.IN,.OUT)** which calls **GETVISIT^BSDAPI4**

Where IN array holds all incoming variables as detailed below

Where OUT array is returned with # of visits found and visit internal entry numbers

Optional API, for use by ancillary applications in roll & scroll mode and is called by using

S VISIT=$$EN^BSDAPI3(patient IEN, default clinic, default reason, item). This interactive API finds all appointments patient has that day. It lists them for the user to select with "walk-in to ancillary department" as last choice. This API then calls the main one (BSDAPI4) to check in patient and create visit.

Inputs and Outputs for GETVISIT^APCDAPI4: (documented in routine)

Special Incoming Variables: OPTIONAL

IN("FORCE ADD") = 1 ; no matter what, create new visit
IN("NEVER ADD") = 1 ; never add visit, just try to find one or more
IN("ANCILLARY") = 1 ; for ancillary packages to create noon visit if no match found

Incoming Variables used in Matching: REQUIRED

IN("PAT") = patient IEN (file 2 or 9000001)
IN("VISIT DATE") = visit date & time (same as check-in date & time)
IN("SITE") = location of encounter IEN (file 4 or 9999999.06)
IN("VISIT TYPE") = internal value for field .03 in Visit file
IN("SRV CAT") = internal value for service category
IN("TIME RANGE") = range in minutes for matching on visit time;
REQUIRED unless FORCE ADD set
zero=exact matches only; -1=don't match on time

These are Used in Matching, if Sent: OPTIONAL

IN("PROVIDER") = IEN for provider to match from file 200
IN("CLINIC CODE") = IEN of clinic stop code (file 40.7)
IN("HOS LOC") = IEN of hospital location (file 44)

Incoming Variables Used in Creating Appt and Visit
IN("USR") = user IEN in file 200; REQUIRED
IN("APPT DATE") = appt date & time; OPTIONAL
IN("OPT") = name for Option Used To Create field, for check-in only ; OPTIONAL
IN("OI") = reason for appointment; used for walk-ins; OPTIONAL

Incoming PCC Variables for Adding Additional Info to Visit: OPTIONAL
IN("APCDTPB") = Third Party Billed (#.04)
IN("APCDPVL") = Parent Visit Link (#.12)
IN("APCDAPPT") = Walk-In/Appt (#.16)
IN("APCDEVM") = Evaluation and Management Code (#.17)
IN("APCDCODT") = Check Out Date & Time (#.18)
IN("APCDLS") = Level of Service -PCC Form (#.19).
IN("APCDVELG") = Eligibility (#.21)
IN("APCDPROT") = Protocol (#.25).
IN("APCDOPT") = Option Used To Create (IEN) (#.24)

Output Array: (Calling application uses it to evaluate next step)
OUT(0) always set; if = 0 none found and may have error message in 2nd piece
if = 1 and OUT(visit IEN)="ADD" new visit just created
if = 1 and OUT(visit IEN)= # (time difference in minutes)
if >1, OUT(visit IEN) for each entry; = # of minutes difference

Inputs and Outputs for EN^BSDAPI3: (documented in routine)

Incoming Variables:
DFN - Patient IEN
BSDCLD - Clinic Default IEN for ancillary walk-in visits
BSDREAS - Default reason for appt ("lab draw", "radiology walk-in", etc.)
BSDITEM - Item name ("test(s)", "exam", "prescription", "order")
Phrasing up to calling routine

Returning Value:
Returns string - first piece is Visit IEN or zero if error occurred
second piece is error message

Logic Notes:

Calling ancillary package must check the INTERACTIVE LINK field of the PACKAGE multiple in the PCC MASTER CONTROL file to see if the facility wants to use this interactive mode.

The default clinic for ancillary walk-ins, is stored in the DEFAULT HOSPITAL LOCATION filed under the PACKAGE multiple of the PCC MASTER CONTROL file.

To alert clinics that an appointment was checked in by an ancillary service, the status on the Scheduling display flashes. The Scheduling check-in process has been modified to allow updating visit clinic code and provider in addition to check-in time. Once updated, the status no longer flashes.

Programming Hints:

Main API can be called multiple times. Depending on results of first call, application can call it again with different variables set.

By using "ancillary mode", applications will either link with the one visit that matches OR create a noon visit for merging later by PCC. If ordering provider is sent, it will attempt to match on provider. No match on clinic code or hospital location will occur in "ancillary mode".

Optional matching variables (provider, clinic code and hospital location) need to be used carefully to prevent duplicate visits from still being added.

Time ranges do span midnight when looking for matching visits.

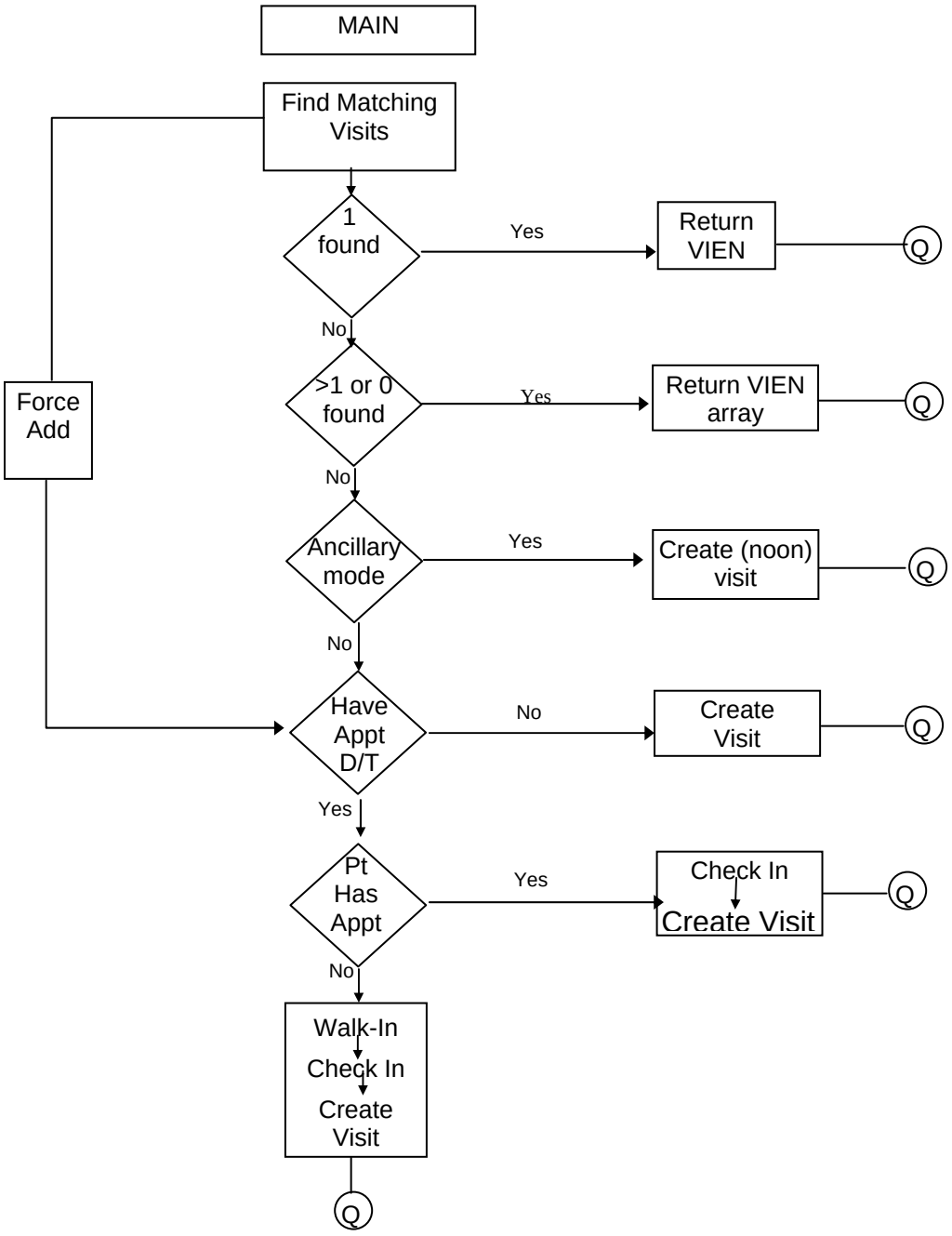
Results returning multiple matches, need to be handled by calling application. Either a user interface needs to be created to ask user to make a choice OR application may just call API again with "force add" set, if user unlikely to be able to make that choice.

Silent call to update Scheduling check-in data, must include appointment date/time and hospital location and must **not** include force add, never add or ancillary mode variables. Send provider if you want provider added to visit. The old call to CHECKIN^BSDAPI is no longer a public entry point.

Check-in code now allows more than one appointment to point to the same PCC visit. The primary provider will be the provider on the last appointment. The hospital location will also be for the last appointment.

Check-in will match on hospital location and provider, if sent. It will also match if existing visit was to a "triage" clinic and matches on clinic code. "Triage" clinic is defined by new field under Set Up A Clinic in Scheduling (ScreenMan page 4 under option).

Visit Creation logic flow – basic decision points



2.3.3.2 Previous API for Creating/Selecting PCC Visit

Call is EN^APCDALV with the following variables set:

Input variables:

APCDALVR("APCDADD")="" Forces the creation of a new visit if set

APCDALVR("APCDADF")="" Used only with user interaction. If set, the default will be for adding a new visit entry.

APCDALVR("APCDAUTO")="" If this variable exists, a visit is searched for at the exact date and time passed in variable APCDDATE. If no visit exists at that date/time then one is created.

APCDALVR("AUPNTALK")="" This variable must exist to prevent subsequent interaction with the user.

APCDALVR("APCDANE")="" This variable must exist to prevent FileMan from echoing information back to the screen.

Now variables that correspond to data fields:

APCDALVR("APCDDATE")= Date and time of visit in internal FileMan format. If time is NOT passed, 12 noon will be appended. (#.01)

APCDALVR("APCDTYPE")= Type of visit (internal format; #.03). See Visit file for choices. If omitted, it will default to "I" for "IHS".

APCDALVR("APCDPAT")=Patient internal entry number (#.05).

APCDALVR("APCDLOC")= Location of visit – facility (#.06). Must be a valid pointer to the Location file (#9999999.06).

APCDALVR("APCDCAT")= Service Category of Visit (internal format). See field #.07 for choices. If omitted, it will default to "A" for "Ambulatory".

APCDALVR("APCDCLN")= Clinic code (#.08). Must be a valid pointer to the Clinic Stop file. If omitted it will default to null if visit is created.

APCDALVR("APCDHL")= Hospital Location pointer (#.22).

Other lesser used variables:

APCDALVR("APCDTPB") Third Party Billed (#.04)
APCDALVR("APCDPVL") Parent Visit Link (#.12)
APCDALVR("APCDAPPT") WalkIn/Appt (#.16)

APCDALVR("APCDEVM")	Evaluation and Management Code (#.17)
APCDALVR("APCDCODT")	Check Out Date & Time (#.18)
APCDALVR("APCDLS")	Level of Service -PCC Form (#.19).
APCDALVR("APCDVELG")	Eligibility (#.21)
APCDALVR("APCDOPT")	Option Used to Create (#.24).
APCDALVR("APCDPROT")	Protocol (#.25).
APCDALVR("APCDAPDT")	Appt Date & Time (.26).

Coding sequence:

```
K APCDALVR
Set APCDALVR array.  See details below.
D EN^APCDALV
If APCDALVR("APCDAFLG") exists, error has occurred and needs to be
processed
Else, set variable in your namespace to APCDALVR("APCDVSIT")
K APCDALVR
```

Output variables:

APCDALVR("APCDVSIT") This variable will contain a pointer to the visit entry just selected or created. Or it will be set to null if no visit entry was selected or created.

APCDALVR("APCDVSIT("NEW")") This variable will exist if a new visit was created..

APCDALVR("APCDAFLG") If this variable exists, it is an indication of on of the following:

Set to 1: user typed "?" and failed to select a visit
Set to 2: Visit failed FileMan edits

Example when not forcing add and allowing user interaction, and there is another visit for the same patient, same date (time could be different) and clinic code :

```
K APCDALVR
S APCDALVR("APCDPAT")=30
S APCDALVR("APCDDATE")=3030307.1201
S APCDALVR("APCDLOC")=DUZ(2)
S APCDALVR("APCDTYPE")="I"
S APCDALVR("APCDCAT")="A"
S APCDALVR("APCDCLN")=28
D EN^APCDALV
```

PATIENT: BOSH,DARRELL W has VISITs, same date, location.

1 Create New VISIT
2 Exit without selecting VISIT
3 Display one of the existing VISITs

Or select one of the following existing VISITs:

4 TIME: 14:30 TYPE: I CATEGORY: A CLINIC: FAMILY PRA DEC: 2

Choose one: (1-4): 4//

2.3.4 Other PCC Data Entry APIs

Other Callable Entry Points in PCC Data Entry: (details on following pages)

EN^APCDVDSP	Interactive Visit Display.
EN^APCDCHKJ	Link In-Hospital Visits to Hospitalizations.
EN^APCDALVR	Called to create a V File entry in PCC.
EN^APCDVDLT	Called to delete a visit in PCC.
EN^APCDEIN	Sets up APCD environment variables.
START^APCDCVDT	Changes a visit date and time.
EN^APCDEA3	Process one mnemonic in PCC Data Entry
EN1^APCDEKL	Kills APCD variables.
EN1^APCDPL	Interactive Problem List updating (DFN = patient).
ADDPROB^APCDALV2	Non-interactive add a problem to problem list.
DELPROB^APCDALV2	Non-interactive delete a problem from problem list.
APCDVD	Display a visit. (pass in visit ien)
GETVISIT^APCDDISP	Get a visit for display.
APCDVLK	Lookup visit.
EN^APCDEFL	Edit Visit List template call

2.3.4.1 Interactive Visit Display: EN^APCDVDSP

Set APCDVDSP = visit internal entry number then call EN^APCDVDSP

Link In-Hospital Visits to Hospitalizations: EN^APCDCHKJ

Called by Billing and any other application that needs to insure all In-Hospital visits are linked to their respective Hospitalization visits prior to performing another function. Simply call EN^APCDCHKJ.

2.3.4.2 Create entries in Visit-related files: EN^APCDALVR

The caller must set the appropriate variables required by the input template. Most of the input templates will be attached to the Visit file. If an ancillary system wants to put data in a non-Visit related file, the variable APCDAFLE must be set. Except for visit and patient, all other variables are input template specific.

Call this routine for each file in which you want to create an entry and for each entry in the file. One call, one file, one entry.

Coding sequence:

K APCDALVR
Set APCDALVR array. See details below.
D EN^APCDALVR
If APCDALVR("APCDAFLG") exists, error has occurred and needs to be processed
Else, set variable in your namespace to APCDALVR("APCDADFN")
K APCDALVR

Input variables:

APCDALVR("APCDPAT") Patient internal entry number

APCDALVR("APCDVSIT") Visit internal entry number

APCDALVR("APCDATMP") Name of input template including the brackets. Example: "[APCDALVR 9000010.14 (ADD)]" where 9000010.14 is the V-file
number and (ADD) or (MOD) signify if you are adding a new entry or modifying one. These templates are created by the PCC Data Entry developer upon request.

APCDAFLE("APCDAFLE") Number of file to which input template is attached. 90000010 is assumed if not set.

APCDALVR("XXX") Each input template has other variables to set, required and optional. See the following pages for details.

Output variables:

APCDALVR("APCDAFLG") Will exist if an error occurred. The meaning is as follows:

Set to 1 means the input template either was not bracketed by [ ] or it did not exist for the file specified.

Set to 2 means one or more of the fields failed FileMan edits or security checks.

APCDALVR("APCDADFN") Set to internal entry number of entry in V-file or set to null if none selected.

NOTE: ALL DATA SHOULD PASS THE INPUT TRANSFORM BEFORE BEING PASSED TO PCC. THE CALLER IS RESPONSIBLE FOR DEALING WITH ANY V FILE FAILURES. FOR PROVIDER FIELDS, PLEASE CHECK PCC DATA DICTIONARY TO KNOW WHICH FILE POINTER TO SEND (6 OR 200).

V MEASUREMENT entries:
Input template: [APCDALVR 9000010.01 (ADD)]

Field	Name	Variable	Data description
.01	Measurement	APCDTTYP	Pass as a "`" concatenated with the measurement type IEN or pass the measurement code.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Value	APCDTVAL	Measurement value; see input transform for details.
.05	Percentile	APCDTPCT	
1201	Event Date/Time	APCDTCDT	Date/time measurement was taken

1202	Ordering Provider field	APCDTPRV	Provider who ordered measurement be taken. Caller must check to see if this points to file 6 or 200 to identify which IEN to send.
1203	Clinic	APCDTCLN	Points to Clinic Stop file
1204	Encounter Provider		APCDTEPR Provider who took measurement. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
1208	Parent	APCDTPNT	Points to another V Measurement entry
1209	External Key	APCDTEXK	
1210	Outside Provider	APCDTOPR	Free text name

V Hospitalization entries: ONLY USE BY ADT IN ADD MODE!!!!
Input template: [APCDALVR 9000010.02 (ADD)]

Field	Field Name	Variable	Data description
.01	Discharge Date	APCDLOOK	Discharge Date and Time in FileMan format.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Admtng Service for	APCDTADS	Pass as a "`" concatenated with the IEN the admission service from file 45.7 (Facility Treating Specialty) Required
.05	Dischrge Service for	APCDTDCS	Pass as a "`" concatenated with the IEN the discharge service from file 45.7 (Facility Treating Specialty) Required
.06	Discharge Type for	APCDTDT	Pass as a "`" concatenated with the IEN the discharge movement type from file 405.1 (Facility Movement Type) Required
.07	Admission Type for	APCDTAT	Pass as a "`" concatenated with the IEN the admission movement type from file

			405.1 (Facility Movement Type) Required
.08	No. of Consults	APCDTNC	Number
.09	Transferred To	APCDTTT	Variable pointer. Send as "VA/IHS.`"_IEN from File 4 (Institution file) or "VENDOR.`"_IEN from file 9999999.11 (Vendor file) **Changed in PIMS v5.3!!!!** Now a pointer to Transfer Facility file (9999999.91). Pass as a "`" concatenated with the IEN.
.11	Medicare Release	APCDTMCR	Y for YES or N for NO
.12	Admitting Dx.	APCDTADX	Pass as a "`" concatenated with the IEN from the ICD9 file 80. Required Field
.13	Assignment of Benefits	APCDTAOB	Y for YES or N for NO
6101	Admission Type-UB92	APCDTATU	Set of codes; pass internal format
6102	Admission Source-OB92	APCDTASU	Pass as a "`" concatenated with the IEN from the Admission Source file (9999999.53).
6103	Discharge Status-UB92	APCDTDTU	Set of codes; pass internal format

V Provider entries:
Input template: [APCDALVR 9000010.06 (ADD)]

Field	Field Name	Variable	Data description
.01	Provider	APCDTPRO	Pass as a "`" concatenated with the Provider file (6 or 200) IEN. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Primary/Sec	APCDTPS	P for primary; S for Secondary. Required Field. There must be one and only one Primary Provider per visit.
.05	Provider Status	APCDTOA	Required for Hospitalization visits. O for

			Operating; A for Attending; C for Consulting. For other types of visits, null is sufficient.
1201	Event Date/Time	APCDTCDT	Date/time provider saw patient
1202	Ordering Provider	APCDTPRV	Not used.
1203	Clinic	APCDTCLN	Points to Clinic Stop file
1204	Encounter Provider	APCDTEPR	Not used.
1208	Parent	APCDTPNT	Points to another V Provider entry
1209	External Key	APCDTEXK	
1210	Outside Provider	APCDTOPR	Free text name

V POV entries:
Input template: [APCDALVR 9000010.07 (ADD)]

Field	Field Name	Variable	Data description
.01	POV	ACDTPOV	Pass as a "`" concatenated with pointer to the ICD Diagnosis file or simply pass the ICD code itself. There is a screen on this field and it would be wise to execute the Input Transform on this field prior to passing the data.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Prov. Narrative	APCDTNQ	The provider's narrative. Must be 2-80 characters long. Input transform should be executed. Required.
.05	Stage	APCDSTG	Pass a number between 0 and 9. Not required.
.06	Modifier	APCDTMOD	Not required. See data dictionary for definition.
.07	Cause of DX	APCDTCD	Not Required. See data dictionary for definition.

.08	First/Revisit	APCDTFR	Not Required. See data dictionary for definition.
.09	Cause of Injury diagnosis	APCDTCI	Required if this is marked as a FIRST visit in field #.08 and if the ICD code is between 800 and 999 (injury codes). See data dictionary for definition.
.11	Place of Accident diagnosis	APCDTPA	Required if this is marked as a FIRST visit in field #.08 and if the ICD code is between 800 and 999 (injury codes). See data dictionary for definition.
.12	Primary/Secondary		APCDTPS Required for Hospitalizations. For other visits, the first POV entered for a visit is considered the primary one.
.13	Date of Injury	APCDTDI	Date in FileMan format. Optional.
.14	Override/Accept	APCDTACC	If an "ACCEPT" command is required for this POV, this variable must exist and be set to DUZ for a valid user.
.15	Clinical Term	APCDTCT	This field is the clinical lexicon term which most closely represents the provider narrative of the problem treated. If VA Lexicon Utility is installed, pass the "`"_concatenated with the IEN from the Expressions file (757.01).
.16	Problem List Entry		APCDTPLE Pass the "`"_concatenated with the IEN from the Problem List file.
.17	Date of Onset	APCDTDOO	Date in FileMan format. Optional.
1201	Event Date/Time	APCDTCDT	Date/time diagnosis was made
1202	Ordering Provider	APCDTPRV	Not used.
1203	Clinic	APCDTCLN	Points to Clinic Stop file
1204	Encounter Provider	APCDTEPR	Provider who made diagnosis. Caller must check to see if this field points to file 6 or 200 to identify which IEN to

			send.
1208	Parent	APCDTPNT	Points to another V POV entry
1209	External Key	APCDTEXK	
1210	Outside Provider	APCDTOPR	Free text name

V Procedure entries:
Input template: [APCDALVR 9000010.08 (ADD)]

Field	Field Name	Variable	Data description
.01	Procedure	ACDTPRC	Pass as a "`" concatenated with pointer to the ICD Operation file or simply pass the ICD code itself. There is a screen on this field and it would be wise to execute the Input Transform on this field prior to passing the data.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Prov. Narrative	APCDTNQ	The provider's narrative. Must be 2-80 characters long. Input transform should be executed. Required.
.05	Diagnosis	APCDTDX	Entered only for Hospitalizations. Should be the ICD diagnosis code for the operation done. Diagnosis must also exist in the V POV file for this visit.
.06	Procedure Date	APCDTPD	Will be the same as the visit date unless visit is a hospitalization. Date in FileMan format. Required.
.07	Principal Procedure		APCDTPP Used with hospitalizations only. Y for YES; N for NO.
.08	Infection	APCDTINF	Used with hospitalizations only. Y for YES; N for NO.
.09	Override/Accept	APCDTACC	If an "ACCEPT" command is required for this POV, this variable must exist and be set to DUZ for a valid user.

.11	Operating Provider		APCDTOP Provider who performed this procedure. Pass as a "`" concatenated with the Provider file (6 or 200) IEN. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
.12	Anesthesiologist	APCDTAN	Anesthesiologist for this procedure. Pass as a "`" concatenated with the Provider file (6 or 200) IEN. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
.13	Elapsed Time(Anesthesia)	APCDTET	Number
.14	Anesthesia Administered	APCDTAA	1 for YES; 0 for NO
.15	ASA/PS Class	APCDTAPC	ASA-PS CLASS is the American Society of Anesthesiologists-Physical Status classification system based on the presence and severity of disease.
.16	CPT Code	APCDTCPT	Pass as a "`" concatenated with pointer to the CPT file or simply pass the CPT code itself. There is a screen on this field and it would be wise to execute the Input Transform on this field prior to passing the data.
1201	Event Date/Time	APCDTCDT	Date/time procedure was performed
1202	Ordering Provider	APCDTPRV	Provider who ordered procedure. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
1203	Clinic	APCDTCLN	Points to Clinic Stop file
1204	Encounter Provider		APCDTEPR Provider who performed procedure. Not used. Field #.11 Operating Provider used instead.
1208	Parent	APCDTPNT	Points to another V Procedure entry
1209	External Key	APCDTEXK	

1210	Outside Provider	APCDTOPR	Free text name

V Laboratory entries:
Input template: [APCDALVR 9000010.09 (ADD)]

Field	Field Name	Variable	Data description
.01	Lab Test	ACDTLAB	Pass as a "`" concatenated with the Laboratory Test file (60) IEN or the lab test name.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Results	APCDTRES	Free Text field. Field has input transform which should be executed prior to sending data.
.05	Abnormal	APCDTABN	Must be deleted if result value changes.
.06	LR Accession No.	APCDTACC	Free Text. Required.
1101	Units	APCDTUNI	Units of measure of this particular test.
1102	Order	APCDTORD	Number
1103	Site	APCDTSTE	Site the specimen was taken from for this lab test. Points to Topography Field file (61). There is a screen on this field and it would be wise to execute the Input Transform on this field prior to passing the data.
1104	Reference Low	APCDTRFL	Free Text.
1105	Reference High	APCDTRFH	Free Text.
1106	Therapeutic Low	APCDTTHL	Free Text.
1107	Therapeutic High	APCDTTHH	Free Text.
1108	Source of Data Input	APCDTSDI	1 for Lab; 0 for Non-Lab. Entered by software only.

1109	Current Status Flag	APCDTCSF	Set of codes. See data dictionary for details.
1110	Lab Test Cost	APCDTCOS	Dollar amount.
1111	Billable Item	APCDTBIL	1 for YES
1112	Lab POV	APCDTLPV	Pass as a "`" concatenated with pointer to the ICD Diagnosis file or simply pass the ICD code itself.
1113	LOINC Code	APCDTLNC	Pass as a "`" concatenated with pointer to the Lab LOINC file (95.3).
1114	Collection Sample	APCDTCLS	Pass as a "`" concatenated with pointer to the Collection Sample file (62).
1201	Collection Date/Time	APCDTCDT	Date/time lab specimen collected
1202	Ordering Provider	APCDTPRV	Provider who ordered lab test. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
1203	Clinic	APCDTCLN	Points to Clinic Stop file
1204	Encounter Provider	APCDTEPR	Provider who performed lab test. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
1208	Parent	APCDTPNT	Points to another V Lab entry
1209	External Key	APCDTEXK	
1210	Outside Provider	APCDTOPR	Free text name
1211	Ordering Date	APCDTODT	Date in FileMan format. Used for merging to patient's main visit along with ordering provider.
1212	Result Date & Time	APCDTRDT	Date in FileMan format.
1301	Comment 1	APCDTLC1	Free Text.
1302	Comment 2	APCDTLC2	Free Text.

1303	Comment 3	APCDTLC3	Free Text.
1401	CPT pointer	APCDTCPT	Pointer to IHS LAB CPT Code file.
1402	CPT - Billable Items	APCDTCPS	Free Text.
1601	Lab POV	APCDTCPS	Free Text.

V Immunization entries:
Input template: [APCDALVR 9000010.11 (ADD)]

Field	Field Name	Variable	Data description
.01	Immunization	ACDTIMM	Pass as a "`" concatenated with the Pointer to Immunization File (Vaccine).
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Series	APCDTSER	Dose # (Series #). See data dictionary for set of codes.
.05	Lot	APCDTLOT	Lot #, Pointer to Immunization Lot File
.06	Reaction	APCDTREC	See data dictionary for set of codes.
.12	VIS Date	APCDTVSD	Vaccine Information Statement Date This is the release date or revision date of the Statement--NOT the date of visit.)
1201	Event Date/Time	APCDTCDT	Date/time immunization was given
1202	Ordering Provider	APCDTPRV	Provider who ordered immunization. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
1203	Clinic	APCDTCLN	Points to Clinic Stop file
1204	Encounter Provider	APCDTEPR	Provider who gave immunization. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.

1208	Parent	APCDTPNT	Points to another V Immunization entry
1209	External Key	APCDTEXK	
1210	Outside Provider	APCDTOPR	Free text name

V Skin Test entries:
Input template: [APCDALVR 9000010.12 (ADD)]

Field	Field Name	Variable	Data description
.01	Skin Test	APCDTTYP	Pass as a "`" concatenated with the Skin Test file IEN.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Results	APCDTRES	See data dictionary for set of codes. Has input transform that should be executed before passing data.
.05	Reading	APCDTREA	The value representing the reading of the skin test. Has input transform that should be executed before passing data.
.06	Date Read	APCDTDR	Date in FileMan format. Has input transform that should be executed before passing data.
1201	Event Date/Time	APCDTCDT	Date/time skin test was performed.
1202	Ordering Provider	APCDTPRV	Provider who ordered skin test. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
1203	Clinic	APCDTCLN	Points to Clinic Stop file
1204	Encounter Provider	APCDTEPR	Provider who took skin test or who read it. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
1208	Parent	APCDTPNT	Points to another V Skin Test entry

1209	External Key	APCDTEXK	
1210	Outside Provider	APCDTOPR	Free text name

V Medication entries:
Input template: [APCDALVR 9000010.14 (ADD)]

Field	Field Name	Variable	Data description
.01	Medication	APCDTRX	Pass as a "`" concatenated with the Drug file IEN.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.04	Name of Non-Table Drug	APCDTNTD	Free Text.
.05	SIG	APCDTSIG	Medication instructions for this prescription.
.06	Quantity	APCDTQTY	Number
.07	Days Prescribed	APCDTDAY	Number
.08	Date Discontinued	APCDTDIS	Date in FileMan format
1101	Comment	APCDTCOM	Free Text
1102	Prescription #	APCDTRXN	Free Text
1201	Event Date/Time	APCDTCDT	Date/time medication was filled.
1202	Ordering Provider	APCDTPRV	Provider who ordered medication. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send. Required to merge with patient's main visit for that day.
1203	Clinic	APCDTCLN	Points to Clinic Stop file
1204	Encounter Provider	APCDTEPR	Provider who filled prescription. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.

1208	Parent	APCDTPNT	Points to another V Medication entry
1209	External Key	APCDTEXK	
1210	Outside Provider	APCDTOPR	Free text name
1211	Ordering Date	APCDTODT	Date physician wrote prescription. Required to merge with patient's main visit for that day.
1212	Alternate Drug Name	APCDTALT	Free text name

V Radiology entries:
Input template: [APCDALVR 9000010.22 (ADD)]

Field	Field Name	Variable	Data description
.01	Radiology Procedure	APCDTRAD	Pass as a "`" concatenated with the Rad/Nuc Med Procedures file (#71) IEN.
.02	Patient Name	APCDPAT	Patient IEN (Required)
.03	Visit	APCDVSIT	Visit IEN (Required)
.05	Abnormal	APCDTABN	1 for Abnormal; 0 for Normal
.06	Diagnostic Code	APCDTDC	Pass as a "`" concatenated with the pointer to Diagnostic Codes file (#78.3)
.07	Modifier	APCDTMOD	Pass as a "`" concatenated with the pointer to the CPT Modifier file.
.08	Modifier 2	APCDTMD2	Pass as a "`" concatenated with the pointer to the CPT Modifier file.
1101	Impression	APCDTIMP	Free Text (up to 245 characters)
1201	Event Date/Time	APCDTCDT	Date/time medication was filled.
1202	Ordering Provider	APCDTPRV	Provider who ordered x-ray. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send. Required to merge with patient's main visit for that day.
1203	Clinic	APCDTCLN	Points to Clinic Stop file

1204	Encounter Provider	APCDTEPR	Radiologist who read the film. Caller must check to see if this field points to file 6 or 200 to identify which IEN to send.
1208	Parent	APCDTPNT	Points to another V Radiology entry
1209	External Key	APCDTEXK	
1210	Outside Provider	APCDTOPR	Free text name
1211	Ordering Date	APCDTODT	Date physician ordered x-ray. Required to merge with patient's main visit for that day.

Other V file entries:
Look at the input templates for the other V files for the variables needed to create entries.

Other V files:

9000010.03	V CHS	^AUPNVCHS(
9000010.04	V EYE GLASS	^AUPNVEYE(
9000010.05	V DENTAL	^AUPNVDEN(
9000010.13	V EXAM	^AUPNVXAM(
9000010.15	V TREATMENT	^AUPNVTRT(
9000010.16	V PATIENT ED	^AUPNVPED(
9000010.17	V PHYSICAL THERAPY	^AUPNVPT(
9000010.18	V CPT	^AUPNVCPT(
9000010.19	V ACTIVITY TIME	^AUPNVTM(
9000010.21	V DIAGNOSTIC PROCEDU	^AUPNVDXP(
9000010.23	V HEALTH FACTORS	^AUPNVHF(
9000010.24	V PATHOLOGY	^AUPNVPTH(
9000010.25	V MICROBIOLOGY	^AUPNVMIC(
9000010.27	ZV MICROBIOLOGY (SF)	^AUPNVMIS(
9000010.28	V NOTE	^AUPNVNOT(
9000010.29	V EMERGENCY VISIT RE	^AUPNVER(
9000010.31	V BLOOD BANK	^AUPNVBB(
9000010.32	V PHN	^AUPNVPHN(
9000010.33	V TRANSACTION CODES	^AUPNVTC(
9000010.34	V NARRATIVE TEXT	^AUPNVNT(
9000010.35	V ELDER CARE	^AUPNVELD(
9000010.37	V TRANSACTION CHARGE	^AUPNVTRC(
9000010.38	V UNHF	^AUPNVUNH(
9000010.39	V TREATMENT CONTRACT	^AUPNVTXC(
9000010.41	V ASTHMA	^AUPNVAST(

9000010.99 V LINE ITEM (GOODS&S ^AUPNVLI(

2.3.4.3 Delete a visit in PCC: EN^APCDVDLT

Set variable APCDVDLT equal to visit internal entry number and call EN^APCDVDLT.

2.3.4.4 Sets up APCD environment variables: EN^APCDEIN

Used when calling PCC data entry in interactive mode.

2.3.4.5 Changes a visit date and time: START^APCDCVDT

WARNING: This routine executes the cross-references on the .03 field (except the "AD") in order to reset the "AA" cross-reference. Very dangerous assumptions here. For one, if the date of the VISIT was used on any other field it would not be reset.

Array APCDCVDT must be passed as follows:

APCDCVDT("VISIT DFN")	DFN of VISIT entry being changed.
APCDCVDT("VISIT DATE/TIME") internal	Date and time to be changed to in FileMan form.
APCDCVDT("TALK")	Any value including NULL

If APCDCVDT("TALK") exists a dot (.) will be printed for each V FILE entry processed during both passes.

Upon exit APCDCVDT("ERROR FLAG") will exist if an error was detected.

It is the callers responsibility to KILL APCDCVDT.

2.3.4.6 Process one mnemonic in PCC Data Entry: EN^APCDEA3

Used to have user perform an add or modify on a visit from outside PCC.

Coding sequence:

```
D ^APCDEIN
Set variables; see following list.
D EN^APCDEA3
D EN^APCDEKL
```

Input Variables:

APCDVSIT	Visit IEN
APCDPAT	Patient IEN
APCDCAT	Service Category
APCDTYPE	Visit Type (IHS, 638, VA, etc.)
APCDVLK	Visit IEN
APCDLOC	Location of Encounter
APCDMODE	A for Add; M for Modify
APCDMNE	IEN for mnemonic (file 9001001)
APCDMNE("NAME")	Mnemonic name

2.3.4.7 Kills APCD variables: EN1^APCDEKL

Used when calling PCC data entry in interactive mode.

2.3.4.8 Interactive Problem List updating: EN1^APCDPL

DFN must be set to patient internal entry number.

2.3.4.9 Non-interactive add to problem list : ADDPROB^APCDALV2

$$ADDPROB^APCDALV2 with the following parameters:
(APCDDX,APCDP,APCDDLM,APCDCLS,APCDN,APCDFAC,APCDDTE,APCDSTAT,APCDDOO)

APCDDX is the diagnosis - pass in "`"_IEN format or pass code (required)
APCDP is the patient DFN (required)
APCDDLM is the date last modified, if null I will stuff DT, PASS IN EXTERNAL FORMAT
APCDCLS is the class (not required)
APCDN - provider narrative pass either "`"_IEN of provider narrative or pass narrative text
APCDFAC - facility IEN, if null will use DUZ(2)
APCDDTE - date entered, if null will use DT , PASS IN EXTERNAL FORMAT
APCDSTAT - status I or A WILL DEFAULT TO A IF NONE PASSED
APCDDOO - date of onset (pass in EXTERNAL format please) (not required)

ENTERED BY (field 1.03) is stuffed with DUZ

Error codes will be passed back

1 = invalid dx, either not a valid ien, inactive code, E code
2 = invalid patient dfn, either not a valid dfn or patient merged
3 = invalid class code
4 = error creating entry with FILE^DICN
5 = invalid date last modified
6 = invalid provider narrative
7 = invalid date entered
8 = invalid facility
9 = invalid status
10 = invalid date of onset

2.3.4.10 Non-interactive delete a problem : DELPROB^APCDALV2

$$DELPROB^APCDALV2(problem IEN)

Result will equal -1 if entry passed is not a valid problem.

2.3.4.11 Display a visit : APCDVD

Set APCDVSIT equal to visit internal entry number and call ^APCDVD. Visit is displayed in List Manager mode.

2.3.4.12 Get a visit for display : GETVISIT^APCDDISP

Set APCDPAT equal to patient internal entry number and call GETVISIT^APCDDISP. Interactive mode where user is asked for visit date and if more than one visit exists, is asked to choose among them. Returns visit internal entry number in APCDVSIT.

2.3.4.13 Lookup visit : APCDVLK

Caller can pass the visit date in APCDVLDT or this routine will prompt for the visit date/time.
Variable APCDPAT must contain the patient DFN.
User will be returned the following variables:
APCDVSIT-IEN of visit
APCDCAT-service category of visit
APCDTYPE-type of visit
APCDDATE-date of visit
APCDCLN-clinic of visit

APCDLOC-location of visit
APCDPAT-patient DFN
Caller is responsible for killing these variables.

2.3.4.14 Edit Visit List Template: EN^APCDEFL

Calling routine must pass visit IEN in APCDVSIT
Calling routine responsible for killing APCDVSIT
Calling routine responsible for killing APCD variables - D EN1^APCDEKL.

2.3.5 Sensitive Patient Tracking APIs

Requires PIMS version 5.3
Used in Patient Lookup calls if FileMan not used

They are to be used by any software NOT using FileMan, which utilizes the patient lookup routine ^APCDLK. This would most likely be GUI applications. Without using these calls, there would be holes in your security system as far as tracking access to sensitive patients.

The main call is PTSEC^DGSEC4, which determines patient sensitivity level and user access permissions. Based on the results, your application will allow access to that patient (with or without presenting warnings), warn the user that they are about to access a sensitive patient and ask if they was to continue, or deny access all together.

The second call, NOTICE^DGSEC4, is used only if the user is attempting to access a sensitive patient, the user does not have the appropriate security key and the user has said "Yes, continue anyway". This call then tracks the access AND sends a bulletin to the security officer.

PTSEC^DGSEC4 – DG Sensitive Record Access
Type: Remote Procedure Call (DG SENSITIVE RECORD ACCESS)
Return Value Type: ARRAY

Description:

This Remote Procedure Call (RPC) will:

1. Verify user is not accessing his/her own Patient file record
   If the Restrict Patient Record Access (#1201) field in the MAS Parameters (#43) file is set to yes
   And the user does not hold the DG RECORD ACCESS security key.
   If parameter set to yes and user is not a key holder, a social security number must be defined in the New Person file for the user to access ANY Patient file record.

2. Determine if user accessing a Sensitive record.

   If the user holds the DG SENSITIVITY security key, access will be tracked and a short warning message is returned for use by the calling application.

   If the user does NOT hold the key, a long warning message is returned to application. Calling application is responsible for asking user if he/she wants to continue and access the patient anyway. If so, the NOTICE^DGSEC4 call must be made to track the access and send a bulletin to the security officer mail group.

3. Record access if facility is tracking patient as non-sensitive. This occur either if the patient has been entered into the DG SECURITY LOG file as a non-sensitive patient. OR if the facility is tracking ALL patients, which will automatically add patients as non-sensitive to the file. No warning message is issued to user that they are being tracked.

Call: D PTSEC^DGSEC4(.RESULT,DFN,DGMSG,DGOPT)

Input Parameter: DFN Parameter Type: LITERAL
Required: YES
Description: Patient (#2) file IEN.

Input Parameter: DGMSG Parameter Type: LITERAL
Maximum Data Length: 1
Required: NO
Description: 1 = generate message if SSN undefined
0 = message will not be generated
If not defined, defaults to 1.

Input Parameter: DGOPT Parameter Type: LITERAL
Required: NO
Description: Option name ^ Menu text (for storing which option user was accessing when patient was selected)

Return Parameter Description:

RESULT(1) =

-1 RPC/API failed Required variable not defined

0 No display/action required. Non-sensitive patient and not accessing own patient record (if access to own record is restricted). If site is tracking all

patients or patient already defined as non-sensitive, access information was stored in file.

1 Short warning message returned. Accessing Sensitive patient who is inpatient OR user holds DG SENSITIVTY key. Access information was stored in file.

2 Long warning message returned. Requires calling application to ask OK to continue. If user continues, make call to DG SENSITIVE RECORD BULLETIN RPC (see details below) to update DG Security Log file and generate Sensitive Record Access mail message. Means patient is defined as Sensitive, is not an inpatient and user does NOT hold DG SENSITIVITY key.

3 Access to record denied. Accessing own Patient file record AND site restricts such access and user does not hold DG RECORD ACCESS key.

4 Access to all Patient file (#2) records denied. SSN not defined for user in New Person file. Cannot determine is accessing own record or not.

RESULT(2-n) = error message or warning/Privacy Act message. Error and warning messages will begin in RESULT(2) array. The Privacy Act message is the longest message and will utilize RESULT(2)- RESULT(9).

NOTICE^DGSEC4 – DG Sensitive Record Bulletin
Type: Remote Procedure Call (DG SENSITIVE RECORD BULLETIN)
Return Value Type: SINGLE VALUE
Description:

This Remote Procedure Call (RPC) will add an entry to the DG Security Log (#38.1) file and/or generate the sensitive record access bulletin depending on the value in Action input parameter. If Action parameter not defined, defaults to update DG Security Log file AND generate Sensitive Record Access mail message.

Call: D NOTICE^DGSEC4(.RESULT,DFN,DGOPT,ACTION)

Input Parameter: ACTION Parameter Type: LITERAL
Maximum Data Length: 1
Required: NO
Description: 1 = Set DG Security Log entry
2 = Generate bulletin
3 = Both

Input Parameter: DFN Parameter Type: Literal
Required: Yes
Description: Patient (#2) file IEN

Input Parameter: DGOPT Parameter Type: LITERAL
Required: NO
Description: DGOPT = Option Name ^ Menu test
If not defined, OP^XQCHK identifies option or defaults to UNKNOWN.

Return Parameter Description:

RESULT=

1 - successfully added entry and/or generated sensitive record access bulletin

0 - unsuccessful

2.3.6 Inpatient Data

2.3.6.1 Inpatient Data Update API's

ADD^BDGAPI – Add ADT event (admission, transfer or discharge)
Parameters: incoming array can be changed but is not killed; passed by reference

ALWAYS REQUIRED:
ARRAY("PAT") = patient ien
ARRAY("TRAN") = transaction type (1=admit, 2=ward transfer, 3=discharge, 4=check-in lodger, 5=check-out lodger, 6=service transfer)
ARRAY("DATE") = date/time for movement, in FM or external format
ARRAY("USER") = user who entered movement

CONDITIONALLY REQUIRED:
if admission -
ARRAY("UBAS") = 1-digit UB92 admit source code, valid 1-9 & A
ARRAY("ADMT") = 1-digit IHS admission code, created from UBAS
ARRAY("ADX") = admitting dx, free text to 30 characters, no ";"
ARRAY("ACCT") = external account # - to be passed to PCC on add

if ADMT=2 or 3 on admission or DSCT=2 on discharge
ARRAY("TFAC") = transfer facility (in or out), name or IEN

if admission or ward transfer
ARRAY("WARD") = ward location, name or IEN

if admission or service transfer
ARRAY("SRV") = treating specialty, 2-digit IHS code (file 45.7)
ARRAY("ADMD") = admitting physician, IHS ADC code or name
ARRAY("PRMD") = primary provider, IHS ADC code or name; if not sent, will be stuffed with attending

ARRAY("ATMD") = attending provider, IHS ADC or code

if discharge
ARRAY("DSCT") = internal entry number in file 405.1

OPTIONAL:
if admission
ARRAY("UBAT") = 1-digit UB92 admission code, valid values 1-4
ARRAY("REFP") = referring provider, free text, up to 30 characters

if admission or ward transfer
ARRAY("ROOM") = room/bed, formatted free text (room-bed)

if discharge
ARRAY("UBDS") = 1-2 digit UB92 discharge status code, valid 1-7,10,20,30

Returns status: ="" means all went well
=1^MESSAGE means event stored but one or more required fields were not filed; original value of those fields in error message
=2^MESSAGE means event was NOT stored; one or more required fields could not be filed

New variable set and passed back: ARRAY("VIEN") = visit IEN

Example: S ERR=$$ADD^BDGAPI(.ARRAY)

CANCEL^BDGAPI1 – Delete ADT Event (admission, transfer or discharge)

Parameters: See details under ADD^BDGAPI

REQUIRED: ARRAY("ACCT") = outside account number for linking to visit

Returns: See details under ADD^BDGAPI

Example: S ERR=$$CANCEL^BDGAPI(.ARRAY)

EDIT^BDGAPI2 – Modify ADT Event (admission, transfer or discharge)

Parameters: See details under ADD^BDGAPI

REQUIRED: ARRAY("ACCT") = outside account number for linking to visit

Returns: See details under ADD^BDGAPI

Example: S ERR=$$EDIT^BDGAPI(.ARRAY)

2.3.6.2 Inpatient Data View API's

2.3.6.2.1 IHS system calls:

VAR^BDGVAR – Sets ADT system-wide variables
Parameters: None

Returns: None

Example: D ^BDGVAR (Used if calling option separate from ADT Menus)

CHECK^BDGVAR – Status of link between ADT and PCC.
Parameters: TALK = 1 means display mini message to screen
TALK = 2 means display full message to screen
TALK = 0 means no display; just return status

Returns: 1 if ADT set up but PCC link is off
2 if ADT set up and PCC link is on
0 if ADT not set up - do not continue
If TALK not = 2 and ADT not set up, XQUIT set to 1

Example: S STATUS=$$CHECK^BDGVAR(TALK)

EXIT^BDGVAR – Cleans up ADT system-wide variables
Parameters: None

Returns: None

Example: D EXIT^BDGVAR

2.3.6.2.2 IHS calls to return patient data:

ADMPRV^BDGF1 – Inpatient's provider's name
Parameters:

ADM = admission IEN
PAT = patient DFN
TYPE ="ADM" for admitting, "PRM" for primary, "ATT" for attending
MODE="" for external format or ="I" for internal format

Returns: Provider name or provider IEN in file 200, depending on MODE set

Example: S X=$$ADMPRV^BDGF1(ADM,PAT,TYPE,MODE)

ADMPRVS^BDGF1 – Inpatient's provider's service/section

Parameters:

ADM = admission IEN
PAT = patient DFN
TYPE ="ADM" for admitting, "PRM" for primary, "ATT" for attending
MODE="" for external format or ="I" for internal format

Returns: Service/Section field in file 200 for provider type requested

Example: S X=$$ADMPRVS^BDGF1(ADM,PAT,TYPE,MODE)

ADMSRV^BDGF1 – Admitting treating specialty name

Parameters:

ADM = admission IEN
PAT = patient DFN

Returns: Admitting treating specialty name

Example: S X=$$ADMSRV^BDGF1(ADM,PAT)

$$ADMSRVC^BDGF1 – Admitting treating specialty abbreviation & code

Parameters:

ADM = admission IEN
PAT = patient DFN

Returns: Admitting treating specialty abbreviation and associated IHS code separated by a space.

Example: S X=$$ADMSRVC^BDGF1(ADM,PAT)

ADMSRVN^BDGF1 – Admitting treating specialty internal number

Parameters:

ADM = admission IEN
PAT = patient DFN

Returns: Admitting treating specialty internal entry number

Example: S X=$$ADMSRVN^BDGF1(ADM,PAT)

ADMTXN^BDGF1 – Admitting treating specialty transfer

Parameters:

ADM = admission IEN
PAT = patient DFN

Returns: Internal entry number in Patient Movement file for admitting service transfer associated with the admission entry. Every admission creates 2 entries – admission entry with ward and visit pointers and a service transfer entry with providers.

Example: S X=$$ADMTXN^BDGF1(ADM,PAT)

ADMTYP^BDGF1 – IHS admit type and code for admission

Parameters: ADM = admission IEN

Returns: IHS code number & IHS admission type name separated by a space.

Example: S X=$$ADMTYP^BDGF1(ADM)

CURDX^BDGF1 – Admitting diagnosis for current inpatient

Parameters: PAT = patient DFN

Returns: Admitting diagnosis (free text) for a current inpatient. If patient is not a current inpatient, "??" is returned.

Example: S X=$$CURDX^BDGF1(PAT)

CURLOS^BDGF1 – Length of stay for current inpatient or observation patient

Parameters: PAT= patient DFN
MODE =1 return LOS in hours

Returns: Length of stay in days for a current inpatient; length of stay in hours if MODE=1; "??" if patient is not a current inpatient

Example: S X=$$CURLOS^BDGF1(PAT,1)

CURPRV^BDGF1 – Current attending provider for patient

Parameters: PAT = patient DFN

LENGTH = number of characters to return; if not set, 20 is used

Returns: Name of attending provider for a current inpatient.

Example: S X=$$CURPRV^BDGF1(PAT,30)

DSADM^BDGF1– Was patient admitted after day surgery within site parameter limit?

Parameters:

ADM = admission IEN
PAT = patient DFN

Returns: "1_^_date of last day surgery" OR "0" if last day surgery not within time range. Time range determined by site parameter.

Example: S X=$$SDADM^BDGF1(ADM,PAT)

INPT1^BDGF1 – Admission date if patient was an inpatient on date sent

Parameters:
PAT = patient DFN
DATE = date in FileMan format

Returns: Readable admission date or NULL if patient was not inpatient on that date

Example: S X=$$INPT1^BDGF1(PAT,DATE)

LASTPRV^BDGF1 – Last attending provider

Parameters:
ADM = admission IEN
PAT = patient DFN
MODE="" for external format or ="I" for internal format

Returns: Last recorded attending provider name or IEN for admission

Example: S X=$$LASTPRV^BDGF1(ADM,PAT,MODE)

LASTPRVC^BDGF1 – Last attending provider's PCC code

Parameters:

ADM = admission IEN

PAT = patient DFN

Returns: Last recorded attending provider's PCC code

Example: S X=$$LASTPRVC^BDGF1(ADM,PAT)

LASTPRVS^BDGF1 – Last attending provider's service

Parameters:

ADM = admission IEN

PAT = patient DFN

MODE="" for external format or ="I" for internal format

Returns: Last recorded attending provider's service/section name or IEN

Example: S X=$$LASTPRVS^BDGF1(ADM,PAT,MODE)

LASTSRVN^BDGF1– Discharge treating specialty name

Parameters:

ADM = admission IEN

PAT = patient DFN

Returns: Last recorded treating specialty name for admission

Example: S X=$$LASTSRVN^BDGF1(ADM,PAT)

LASTSRVC^BDGF1 – Discharge treating specialty abbreviation & code.

Parameters:

ADM = admission IEN

PAT = patient DFN

Returns: Last recorded treating specialty abbreviation and IHS code for admission separated by a space

Example: S X=$$LASTSRVC^BDGF1(ADM,PAT)

LASTTXN^BDGF1 – Discharge treating specialty movement

Parameters:

ADM = admission IEN

PAT = patient DFN

Returns: Last recorded treating specialty transfer IEN for admission and the treating specialty IEN itself, separated by a "^".

Example: S X=$$LASTTXN^BDGF1(ADM,PAT)

LOSHRS^BDGF1 – Length of stay in hours between a specific date/time & admission date/time.

Parameters:

CA = admission IEN

DATE = date/time used for calculation, in FileMan format

PAT = patient DFN

Returns: Number of hours between admission date/time and date/time sent in call.

Example: S X=$$LOSHRS^BDGF1(CA,DATE,PAT)

PRIORTXN^BDGF1 – Last treating specialty transfer prior to a date/time

Parameters:

DATE = date/time used for calculation, in FileMan format

CA = admission IEN

PAT = patient DFN

Returns: Last treating specialty transfer prior to date/time sent within this admission

Example: S X=$$PRIORTXN^BDGF1(DATE,CA,PAT)

PRIORMVT^BDGF1 – Last ward transfer prior to a date/time

Parameters:

DATE = date/time used for calculation, in FileMan format

CA = admission IEN

PAT = patient DFN

Returns: Returns last ward transfer IEN within this admission prior to date/time in call. May return admission itself.

Example: S X=$$PRIORMVT^BDGF1(DATE,CA,PAT)

READM^BDGF1 – Was patient readmitted within date range?

Parameters:

ADM = admission IEN

PAT = patient DFN

LIMIT = # of days to use in calculating; if not set will use site parameter

Returns: "1_^_last discharge date" if patient was admitted within time limit in call or site parameter. Returns 0 (zero) if not.

Example: S X=$$READM^BDGF1(ADM,PAT,LIMIT)

STATUS^BDGF2 – Patient's current inpatient status

Parameters: PAT = patient DFN

Returns: Free text phrase describing patient's inpatient status. Choices are:

"Patient Died on (specific date)"
"Outpatient"
"Pt currently an inpatient on (ward name here)"
"Pt currently an observation patient on (ward name here)"
"Active Day Surgery Patient"
"Active Incomplete Chart"
"Active Day Surgery Incomplete Chart"

Example: S X=$$STATUS^BDGF2(PAT)

VISIT^BDGF1 – Visit linked to admission

Parameters:

PAT = patient DFN

DATE = admission date in FileMan format

Returns: Internal entry number of PCC visit linked to admission on this date

Example: S X=$$VISIT^BDGF1(PAT,DATE)

WRDABRV^BDGF1– Current inpatient's ward

Parameters: PAT = patient DFN

Returns: Abbreviation for patient's current ward location

Example: S X=$$WRDABRV^BDGF1(PAT)

WRDABRV2^BDGF1 – Admitting ward abbreviation

Parameters: N = admission entry IEN

Returns: Abbreviation for admitting ward for this admission

Example: S X=$$WRDABRV2^BDGF1(N)

2.3.7 Appointment Data APIs

2.3.7.1 Appointment Update API's

MAKE^BSDAPI – Make scheduled or walk-in appointment

Input parameter:

ARRAY("PAT") = IEN of patient in file 2

ARRAY("CLN") = IEN of clinic in file 44

ARRAY("TYP") = 3 for scheduled appts, 4 for walk-ins

ARRAY("ADT") = appointment date and time

ARRAY("LEN") = appointment length in minutes (5-120)

ARRAY("OI") = reason for appt - up to 150 characters

ARRAY("USR") = user who made appt

Returns: error status and message

= 0 or null: everything okay

= 1^message: error and reason

Example: S ERR=$$MAKE^BSDAPI(.ARRAY)

CANCEL^BSDAPI – Cancel appointment

Input Parameter:

ARRAY("PAT") = IEN of patient in file 2

ARRAY("CLN") = IEN of clinic in file 44

ARRAY("TYP") = C for canceled by clinic; PC for patient canceled

ARRAY("ADT") = appointment date and time

ARRAY("CDT") = cancel date and time

ARRAY("USR") = user who canceled appt

ARRAY("CR") = cancel reason - pointer to file 409.2

ARRAY("NOT") = cancel remarks - optional notes to 160 characters

Returns: error status and message

= 0 or null: everything okay

= 1^message: error and reason

Example: S ERR=$$CANCEL^BSDAPI(.ARRAY)

CHECK-IN function: See the Visit Creation APIs section.

2.3.7.2 Appointment View Data APIs

PCP^BSDU1 – Returns patient's primary care provider (PCP)

Input Parameters: PAT = patient DFN

Returned array

ARRAY(1)=PCP name/team name/PCP IEN/team IEN

ARRAY(1,0)=date last updated/user who updated/reason

ARRAY(2)=women's health PCP name/WH team name/WH PCP IEN/team IEN

ARRAY(2,0)=date last updated/user who updated/reason

ARRAY(3)=mental health provider name/MH team name/MH PCP IEN/MH team IEN/MH medication mgr name/MH med mgr IEN

ARRAY(3,0)=date last updated/user who updated/reason

ARRAY(3) only used if site is running Cimarron MH Provider menu

Example: S ARRAY="ABC" D PCP^BSDU1(PAT,.ARRAY)

PCPDISP^BSDU1 – Returns array of PCP info with captions

Input Parameters: PAT = patient DFN

Returned array

ARRAY(1)= "Primary Care Provider/Team: "_PCP name/team name

ARRAY(2)= "Women's Health PCP/Team: "_ women's health PCP name/WH team name

ARRAY(3)= "Mental Health Providers/Team: "_ mental health provider name/MH team name

Example: D PCPDISP^BSDU1(PAT,.ARRAY)

PEND^BSDU2 – Display pending appointments

Input Parameters:

PAT = patient IEN
TALK =1 means display results to current device
TALK =0 means be silent and return

Returned Array: ARRAY(#)=date^clinic name^other info

Example: D PEND^BSDU2(PAT, TALK,.ARRAY)

ACTV^BSDU – Is clinic active on this date?

Input Parameters:
CLINIC = IEN from Hospital Location file
DATE = Date in FileMan format

Returns: 1 if clinic is active; otherwise zero.

Example: S X=$$ACTV^BSDU(CLINIC,DATE)

APPTYP^BSDU2 – Type of appt (scheduled or walk-in)

Input Parameters:
PAT = patient DFN
DATE = appointment date in FileMan format

Returns: "SCHED" or "WALK-IN" or "??"

Example: S X=$$APPTYP^BSDU2(PAT,DATE)

CI^BSDU2 – Is appointment already checked-in?

Input Parameters:
PAT = patient DFN
CLINIC = hospital location file IEN
DATE = appointment date in FileMan format
SDIEN = IEN under Appointment multiple in Hospital Location file
(See SCIEN^BSDU2 for how to obtain this IEN)

Returns: 1 if checked in; or zero if not

Example: S X=$$CI^BSDU2(PAT,CLINIC,DATE,SDIEN)

CLNCODE^BSDU – Clinic code number and name

Input Parameters:
CLINIC = hospital location file IEN

Returns: 2 digit code_" – "_clinic name

Example: S X=$$CLNCODE^BSDU(CLINIC)

CO^BSDU2 – Is appointment already checked-out?

Input Parameters:
PAT = patient DFN
CLINIC = hospital location file IEN
DATE = appointment date in FileMan format
SDIEN = IEN under Appointment multiple in Hospital Location file
(See SCIEN^BSDU2 for how to obtain this IEN)

Returns: 1 if checked out; or zero if not

Example: S X=$$CO^BSDU2(PAT,CLINIC,DATE,SDIEN)

INACTVDT^BSDU – Date clinic was inactivated

Input Parameters:
CLINIC = hospital location file IEN

Returns: Inactivation date in readable format

Example: S X=$$INACTVDT^BSDU(CLINIC)

OI^BSDU2 – Other info (comments) for patient's appointment

Input Parameters:
PAT = patient DFN
CLINIC = hospital location file IEN
DATE = appointment date in FileMan format

Returns: free text reason for appointment

Example: S X=$$OI^BSDU2(PAT,CLINIC,DATE)

PC^BSDU – Clinic's principal clinic IEN

Input Parameters:
CLINIC = hospital location file IEN

Returns: IEN of principle clinic linked to this hospital location

Example: S X=$$PC^BSDU(CLINIC)

PCLINE^BSDU1 – Single line of Primary Care Provider info

Input Parameters:
PAT = patient DFN

Returns: "Pcp/Team:" Provider name/Team Name
If patient has women's health provider assigned, it is expanded:
"Pcp/Team /WH Pcp/Team:" Provider name/Team name/…

Example: S X=$$PCLINE^BSDU1(PAT)

PRIN^BSDU – Clinic's principal clinic name

Input Parameters:
CLINIC = hospital location file IEN

Returns: Name of principle clinic linked to this hospital location

Example: S X=$$PRIN^BSDU(CLINIC)

SCIEN^BSDU2 – Internal entry number for appointment in ^SC

Input Parameters:
PAT = patient DFN
CLINIC = hospital location file IEN
DATE = appointment date in FileMan format

Returns: IEN for appointment under multiple for date/time for clinic

Example: S X=$$SCIEN^BSDU2(PAT,CLINIC,DATE)

WALKIN^BSDU2 – Is appointment a walk-in?

Input Parameters:
PAT = patient DFN
DATE = appointment date in FileMan format

Returns: 1 if walk-in; otherwise a zero

Example: S X=$$WALKIN^BSDU2(PAT,DATE)

GETVST^BSDU2 – Visit IEN linked to checked in appointment

Input Parameters:
PAT = patient DFN
DATE = appointment date in FileMan format

Returns: if appointment checked in and visit created and linked, that visit IEN

Example: S X=$$GETVST^BSDU2(PAT,DATE)

LIST^BSDAPI2 – List of patient's appointments for date and clinic

Input Parameters:
DATE = Appointment date/time in FileMan format
TYPE = Contains "W" to include walk-ins; contains "C" to include cancelled appts or set to null for neither
CLINIC = "ALL" for all clinics with appts on date OR array of clinic internal entry numbers; passed by reference

- If you set CLINIC=ALL and you have clinics from multiple facilities, you must set CLINIC("DEV") equal to the internal entry number of Medical Center Division you want used.

- Principal clinics can be passed and they will expand to all clinics under them

ARRAY = Array name where you want list returned; can be local or global array

- Send array ending in (or , such as S ARRAY="XYZ(" or S ARRAY="^ABC(""XYZ"","

Returns: Array subscripted by simple number count (XYZ(1), XYZ(2) or ^ABC("XYZ",1), ^ABC("XYZ",2))
Each line contains: patient DFN ^ Clinic IEN ^ Appt Date/Time ^ Type ^ Length of Appt ^ Other Info

Example: D LIST^BSDAPI2(DATE,TYPE,.CLINIC,ARRAY)

2.3.8 Generic VA Scheduling APIs

2.3.8.1 SDA^VADPT

Returns APPOINTMENT DATE/TIME data for a patient.

Input Variables:

DFN This required variable is the internal entry number in the PATIENT file.

VASD("T") Can be defined as the "to" date for which registrations are desired. This must be passed as a valid VA File-Manager date. If neither VARP("F") nor VARP("T") are defined, all appointments will be returned.

VASD("F") Can be defined as the "from" date for which appointments are desired. This must be passed as a valid VA File-Manager date.

VASD("W") Can be passed as the specific STATUS desired in the following format. If not passed, only those appointments which are still scheduled (or kept in the event of a past date) for both inpatients and outpatients will be returned.

VASD("W") Contains these appts. are returned

1 Active/Kept
2 Inpatient appts. only
3 No-shows
4 No-shows, auto-rebook
5 Cancelled by Clinic
6 Cancelled by Clinic, auto rebook
7 Cancelled by Patient
8 Cancelled by Patient, auto rebook

9 No action taken

VASD("C",Clinic IFN) Can be set up to contain only those internal file entries from the Hospital Location file for clinics which you would like to see appointments for this particular patient. You may define this array with just one clinic or with many. If you do not define this variable, it will be assumed that you want appointments for this patient in all clinics returned.

Output Variables:

^UTILITY("VASD",$J,#,"I") Internal format

^UTILITY("VASD",$J,#,"E") External format

Piece 1 Date/Time of Appointment
Piece 2 Clinic
Piece 3 Status
Piece 4 Appointment Type

VAERR The error flag will have one of the following values.
0 -- no errors encountered
1 -- error encountered - DFN or ^DPT(DFN,0) is not defined

2.3.9 Adverse Reaction Tracking Package Callable Routines

GMRADPT

GMRADPT extracts data from the Patient Allergies (120.8) file for a specified patient based on the criteria specified in the GMRA input parameter. The data will be returned in a local array.

Input:

DFN The internal entry number in the Patient file for the patient whose allergy data needs to be extracted.

GMRA This is an optional three-piece variable that will determine which kinds of allergy data will be returned by the extract. The default values which will be used are shown in the discussion of each piece. Consider the variable GMRA with the format P1^P2^P3 where

P1 can have the value 0, 1, or 2 where

0 means extract all allergies and adverse reactions

1 means extract allergies only

2 means extract adverse reactions only

A record stored in the Patient Allergies file is either an adverse drug reaction, or it is a true allergy. **Every allergy is an adverse reaction, but not every adverse**

reaction is an allergy. This determination is made by the verifier of the allergy data. The default value for this piece is 0.

P2 can have the value 0, 1, or 2 where

0 means extract all verified and non-verified records

1 means extract verified records only

2 means extract non-verified records only

A record can either be verified by some allergy verifier, or it has not yet been verified. In the case that the site is using autoverification, the record is automatically verified at the time the originator of the record signs off (completes) on it. No record can be extracted before it has been signed off (completed) by the originator, as it is not part of the medical record. The default value for this piece is 0.

P3 is a three-character string, where each character can have the value of 0 or 1. Consider P3 represented as XYZ where X, Y and Z are the three different characters. Then the following is what each of these characters represents:

X determines whether to extract records with the type of Other. If X=0 then records with the type Other will not be extracted, and if X=1 then they will be extracted.

Y determines whether to extract records with the type of Food. If Y=0 then records with the type Food will not be extracted, and if Y=1 then they will be extracted.

Z determines whether to extract records with the type of Drug. If Z=0 then records with the type Drug will not be extracted, and if Z=1 then they will be extracted.

A record has a type associated with it. The three types are Food, Drug and Other. This variable will help to determine which of these types of records will be extracted, and which types will not be extracted. The default value for this piece is 111.

Output:

GMRAL This variable is an array of the patient's data extracted by this utility based on the criteria specified in the optional GMRA variable. The format of this variable is:

GMRAL=(1,0,NULL) GMRAL(DA)=A^B^C^D^E^F^G^H
GMRAL(DA,"S",COUNT)=I

Where

GMRAL is 1 if patient has Adverse Reaction. is 0 if patient has no known Adverse Reaction. null if patient has not been asked about Adverse Reaction.

DA is the internal entry number of the record in the Patient Allergies (120.8) file.

A is the patient's DFN (from input variables).

B is the name of the allergen.

C is the type of the allergen where D=Drug, F=Food, and O=Other.

D is a flag denoting if the allergy has been verified where 1=verified and 0=non-verified.

E is a flag denoting whether the allergy is a true allergy, or if it is an adverse reaction where 1=adverse reaction and 0=true allergy.

F is both the external and internal representation of the allergy Mechanism. It is stored in the format External";"Internal.

Mechanism

External	Internal
Allergy	0
Pharmacologic	2
Unknown	U

G is the type of the reaction in the form of "F", "D", or "0" or a combination of the three types.

Types

Internal	External format
D	is a drug reaction.
DF	is a drug/food/other reaction
DFO	is a drug/food reaction
DO	is a drug/other reaction
F	is a food reaction
FO	is a food/other reaction
O	is a other reaction

H is both the external and internal representation of the Adverse Reaction Mechanism. It is stored in the format External";"Internal.

Mechanism

External	Internal
Allergy	A

Pharmacologic	P
Unknown	U

I is both the external and internal representation of the allergy Signs/Symptoms. Each of the Signs/Symptoms will be stored on it's own "S" node in the following format.

External";"Internal pointer to Signs/Symptoms file (120.83). If the pointer equals the "OTHER REACTION" then the free text stored in the patient file will be stored in the external representation.

COUNT is the order which the Signs/Symptoms are stored in the GMRAL(DA,"S",COUNT) Array. Count is a positive whole number.

2.4 VA/IHS Convergence: Parameterization Details and Examples

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

Below is an example used for an IHS mod to the VA routine VADPT6:

First, use FileMan to create a PARAMETER DEFINITION:

NAME: **DG IHS CHART ID** DISPLAY TEXT: **USE IHS CHART ID?**
MULTIPLE VALUED: **No** VALUE DATA TYPE: **yes/no**
VALUE DOMAIN: "**Y:yes;N:no**"
VALUE HELP: **SHOULD SYSTEM USE IHS CHART ID?**
PRECEDENCE: **1** ENTITY FILE: **SYSTEM**

Then, use XPAR MENU TOOLS General Parameter Tools to Edit Parameter Values (this creates the PARAMETER)

XPAR EDIT PARAMETER **Edit Parameter Values**

Select PARAMETER DEFINITION NAME: **DG IHS CHART ID** USE IHS CHART ID?

----- Setting DG IHS CHART ID for System: B-SYSTEM.DSM.IHS.GOV ------

Value: ?

SHOULD SYSTEM USE IHS CHART ID?.

Value: YES

This creates the PARAMETER:

ENTITY: B-SYSTEM.DSM.IHS.GOV
PARAMETER: DG IHS CHART ID
INSTANCE: 1 VALUE: YES

A PARAMETER DEFINITION, 'DG IHS CHART ID' was created using Fileman, then the 'Edit Parameter Values' option was used to create the parameter and to enter the value of 'yes'.

PARAMETER DEFINITION:
NAME: DG IHS CHART ID DISPLAY TEXT: USE IHS CHART ID?
MULTIPLE VALUED: No VALUE DATA TYPE: yes/no
VALUE HELP: SHOULD SYSTEM USE IHS CHART ID?
PRECEDENCE: 1 ENTITY FILE: SYSTEM

PARAMETER:
ENTITY: B-SYSTEM.DSM.IHS.GOV PARAMETER: DG IHS CHART ID
INSTANCE: 1 VALUE: YES

The routine, VADPT6 was modified with the addition of the following lines:

```
I DUZ("AG")="I"!(DUZ("AG")="E")&$$GET^XPAR("SYS","DG
IHS CHART ID")&(L="") D
.S (L,B)=$$HRCN^BDGF2(DFN,+$G(DUZ(2))) S:L=""
(L,B)="??"  ;IHS/ITSC/CLS 01/11/2005
```

A BUILD called 'REGISTRATION - IHS VADPT 5.3' was created that contained the routine VADPT6 and the parameter definition, 'DG IHS CHART ID', which was exported to a kids file, ihsvadpt6.k.

The kids file was successfully loaded and installed on another system. Then the 'Edit Parameter Values' option was used to create the parameter for the system and assign its value.

It works as advertised (if your agency is set to 'I' or 'E').

Please reference PARAMETER TOOLS Supplement to Patch XT*7.3*26 for more information.

2.5 Event Drivers

An 'Extended Action' type of protocol entry in the PROTOCOL file #101 is called from an event point in a package routine. This type of protocol executes the entry action, if present, plus all sub-items. Any sub-items are designated 'Action' type which executes its entry action then its exit action. The order sub-items are called depends on the sequence field.

In the examples for Radiology and PIMS below, the 'Extended Action' protocol is listed first, followed by a selection of sub-item protocols. Then there is a line or two of code from the calling routine.

The Radiology event driver, 'RA EVSEND OR', is called whenever a radiology request is created or changed. The 'RA EVSEND OR' protocol, in turn, calls the 'Action' type protocol, 'RA IHS HOOK' that executes its entry action, '**D ^BRAPRAD**', the IHS PCC hook for Radiology.

NAME: RA EVSEND OR **TYPE: extended action**

DESCRIPTION: Invoked when a request is created or changed by the Radiology/Nuclear Medicine package (the "backdoor") and the data is passed to the Order Entry package, Version 3.0 or greater.

ITEM: RA IHS HOOK

EXIT ACTION: K:$L($G(RAVARBLE)) @RAVARBLE,RAVARBLE

NAME: RA IHS HOOK **TYPE: action**

DESCRIPTION: Place this protocol on the RA EVSEND OR extended action protocol.

ENTRY ACTION: **D ^BRAPRAD**

Example of call to ^RAO7UTL which invokes the event driver:

```
^RAO7CH
D MSG^RAO7UTL("RA EVSEND OR",.@RAVARBLE)

MSG(RAPROTO,RAMSG) ; ship HL7 messages to CPRS from this entry point
         ; input: RAPROTO - protocol to execute
         ;          RAMSG - message (in HL7 format)
         D MSG^XQOR(RAPROTO,.RAMSG)
       Q
```

The PIMS ADT event driver 'BDGPM MOVEMENT EVENTS' is more complex. It has multiple sub-items that are used to not only trigger PIMS routines, but other applications as well.

For instance, when a patient is admitted, the protocol 'BDGPM MOVEMENT EVENTS' is evoked. This protocol then starts down its sub-items and calls them in sequence order. The description field on the event driver protocol contains documentation on what variables are set for the calling protocols.

In PIMS, there is a second event driver for Scheduling events called BSDAM APPOINTMENT EVENTS. See it's description field for more details.

NAME: BDGPM MOVEMENT EVENTS TYPE: extended action

DESCRIPTION: At the completion of a patient movement, a series of events take place through this option. If your site adds a new software application that must be notified when an ADT movement occurs, add the appropriate protocol from that software to this list.

The first item (sequence 1) called MUST be the BDGPM VISIT UPDATE as it will create a PCC visit for the admission and place the visit pointer in the Patient Movement file admission entry. This will take place ONLY if the PCC link is turned on. So if your protocol requires a visit pointer, make sure to quit if there isn't one.

The protocols sent with the release of this version are in the following sequence:

10-99: Prints a form or report OR creates an entry in another PIMS module.

100-199: Informs various applications, if installed, that a patient movement has occurred.

200-299: Protocols that send mail messages and bulletins.

300-399: Reserved for local protocols.

Required Variables: (variables sent by event driver, not to be killed)

DFN = Patient's IFN
DGPMT = Type of Movement (1=admission, 2=ward transfer, 3=discharge, 4=check-in lodger, 5=check-out lodger, 6=service transfer)
DGPMDA = Movement's IFN
DGPMCA = Admission IFN
DGPMP = 0 Node of Primary Movement PRIOR to Add/Edit/Delete
DGPMA = 0 Node of Primary Movement AFTER Add/Edit/Delete

DGQUIET = If $G(DGQUIET) then the read/writes should not occur. Any code called by the event driver MUST check for DGQUIET!!!!

If you need the internal entry number for the PCC visit associated with this admission, use this code: $$GET1^DIQ(405,+DGPMCA,.27,"I")

ITEM: DGOERR NOTE SEQUENCE: 110
ITEM: SD APPT STATUS SEQUENCE: 115
ITEM: SC PCMM INPATIENT ACTIVITY SEQUENCE: 201
ITEM: BDGPM A SHEET SEQUENCE: 20
ITEM: BDGPM BULLETINS SEQUENCE: 205
ITEM: BDGPM INCOMPLETE CHART UPDATE SEQUENCE: 50
ITEM: BDGPM LOCATOR SEQUENCE: 25
ITEM: BDGPM VISIT UPDATE SEQUENCE: 1
ITEM: BDGPM REQUEST CHART SEQUENCE: 27
ITEM: ORU PATIENT MOVMT SEQUENCE: 101
ITEM: ORU AUTOLIST SEQUENCE: 105
ITEM: PSJ OR PAT ADT SEQUENCE: 120
ITEM: GMRADGPM MARK CHART SEQUENCE: 210
ITEM: AQAL ADT EVENT SEQUENCE: 150
ITEM: AMCO ADT EVENT SEQUENCE: 130

NAME: BDGPM A SHEET
ITEM TEXT: CLINICAL RECORD BRIEF TYPE: action

DESCRIPTION: Protocol to print Clinical Record Brief for admission. If DGQUIET variable is set will only print if other required variables sent with call. See routine for details.

```
ENTRY ACTION: D
PAT^BDGCRB(+$G(DFN),+$G(DGPMCA),$G(BDGFIN),

$G(DGPMT),$G(BDGHALF),$G(DGQUIET),$G(BDGCOP),$G(BDGDEV)
)
```

NAME: BDGPM INCOMPLETE CHART UPDATE
ITEM TEXT: UPDATES INCOMPLETE CHART FILE TYPE: action

DESCRIPTION: Protocol to create entry in Incomplete Chart file, if parameter is turned on. If admission is changed after discharge first entered, IC file will be updated unless admission or discharge were deleted. In that case a bulletin will be sent to appropriate people.

ENTRY ACTION: **D ADD^BDGICEVT**

NAME: ORU PATIENT MOVMT
ITEM TEXT: Review Orders on Patient Movement TYPE: action

DESCRIPTION: This is the option used to review orders when a patient is discharged or transferred.

ENTRY ACTION: **D REV^ORF4**

NAME: PSJ OR PAT ADT
ITEM TEXT: Inpatient Medications Actions on Patient ADT TYPE: protocol

DESCRIPTION: This is the actions taken on a patient's Inpatient Medication orders whenever the patient is Admitted, Discharged, or Transferred (ADT). This is an action protocol to be used within the DG MOVEMENT EVENTS protocol.

FILE LINK: PSJ OR PAT ADT ENTRY ACTION: **D ^PSJADT**

Event Driver invoked by ^DGPMEVT:

```
N OROLD D INP^VADPT
S X=$O(^ORD(101,"B","BDGPM MOVEMENT EVENTS",0)) _";ORD(101," D EN1^XQOR:X
K VAIN,X
```

2.6 Using the PCC Visit Merge Pointer Update

If your application stores pointers to PCC visits, you will need to link to the PCC Merge Utility so when visits are merged, your pointer is updated. You will need to add your application to the MODULE PCC LINK CONTROL file with the code to execute when visits are merged. The following is an example of adding your application to the file:

```
PCCLNK ;EP -- add TIU to PCC Visit Merge Utility

     D BMES^XPDUTL("Adding TIU to PCC Visit Merge
     Utility . . .")
     Q:$D(^APCDLINK("B","TEXT INTEGRATION UTILITY"))
     ;already exists
     NEW DD,DO,DIC,DLAYGO,X,Y
     S DIC="^APCDLINK(",DIC(0)="LE",DLAYGO=9001002
     S DIC("DR")="1///I $L($T(MRG^BTIULINK)) D
     MRG^BTIULINK"
     S DIC("DR")=DIC("DR")_";.02///TIU"
     S X="TEXT INTEGRATION UTILITY" D FILE^DICN
     Q
```

Here is an example of the code to use to update the pointers:

```
BTIULINK ; IHS/ITSC/LJF - UPDATE TIU DOC UPON VISIT
MERGE ;

     ;;1.0;TEXT INTEGRATION UTILITIES;;NOV 04, 2004
     ;
     ;This routine is called by the PCC Visit Merge
     Utility.
     ;The input variables are: APCDVMF - Merge from
     visit ifn
     ; APCDVMT - Merge to visit ifn
     ;
     ;This routine finds the patient involved, scans
     for this merged visit
     ;among the occurrences for this patient, and
     updates the visit.
     ;

MRG ;PEP >> PRIVATE ENTRY POINT between TIU and PCC

     N DIE,DA,DR,TIUN,X,Y
     Q:'$D(APCDVMF) Q:'$D(APCDVMT)
     S TIUN=0
     F S TIUN=$O(^TIU(8925,"V",APCDVMF,TIUN)) Q:TIUN=""
     D
     .S DR=".03///"_APCDVMT,DA=TIUN,DIE="^TIU(8925," D
     ^DIE
     ;

EXIT Q
```

2.7 Unix Tools

2.7.1 VI - UNIX Editor: Crib Sheet

Invoking vi

vi	invoke vi, load a new file
vi *file*	invoke vi, load a specified file
vi +n *file*	..... at line *n*
vi + *file*	..... at end of *file*

Character/Word Positioning

h or →	forward one character
l or ←	back one character
0	beginning of line
$	end of line

vi + */s file* ..... at *s*(string)

vi -r *file* recover *file*

view *file* read only mode

Quitting/Saving vi

:wq write (save) and quit
:w write (save) no quit

:q quit without saving

(only if no changes)

:x write (save)

(only if changes made)
:q! quit without saving
ZZ write (save) file and quit vi

Scrolling

<ctrl>f forward one screen

<ctrl>bback one screen

<ctrl>ddown half screen
<ctrl>uup half screen

Line Positioning

H first line on screen
L last line on screen
M middle lline on screen

G go to end of file

*n*G go to line *n*
j or ↓ down one line (same position)

<spacebar> forward one character
w forward to next word
e ... end of next word
b back to previous word

Insert & Replace

a append after cursor
A append at end of line
i insert before cursor
I insert at end of line
o open line below
O open line above

Deleting

dd delete current line
*n*dd delete *n* lines
D ... from cursor to line end
d} ... rest of paragraph
d0 ... to left margin
dw delete word
x delete current character

Search/Replace

/*string* search forward for *string*
?*string* search backwards
n repeat last search
:*#,#s*/*s1*/*s2* replace first occurrence of

k or ↑	up one line (same position)
<ret>	down one line

Changing/Undoing

r*x*	replace character w/ *x*
R	replace characters
cc	change line
*n*cc	change next *n* lines
cw	change word
C	Change to end of line
u	undo last change
U	undo current line changes
yy	yank line to buffer
*n*yy	... *n* lines to buffer
p	put lines at cursor
P	... before cursor
c)	change sentence starting at cursor to new text
J	join next line down to line with cursor

	s1 on each line in range (#-#)
	w/ *s2*
:*#,#*s/*s1/s2*/gc	...but global and confirm

Other Helpful Commands

<ctrl> g	displays line status
<ctrl> l	redraw screen
:se nu	set line numbering
:se nonu	unset ...
:!*command*	execute shell command
:#	go to line #
:w *file*	write to *file*
:r!spell	spell check
:n,kw file2	write lines n-k into another file
:n,kw >> file2	append lines n-k to another file

NOTE: To obtain any command that is preceded by a ":", you press ESC key first then type in the colon.

2.7.2 Some Useful Unix Commands or Unix Scripts

gflist *filename*	provides a listing of globals contained in *filename*
rflist *filename*	provides a listing of routines contained in *filename*
sendto -l dev:dev cmbsyb *filename*	use this to send files to SQA. This will place your files in the /usr/spool/uucppublic/VERIFY directory on cmbsyb.

sendto	send file to a certain location. Example: sendto tuclcl *filename*
getfrom	script to get files from some system. Ex.: getfrom -i -r DIST/96cert cmbsyb aum_9610.tar.gz *This example would immediately get the aum* file from cmbsyb which resides in the /usr/spool/uucppublic/DIST/96cert subdirectory*
gzip -v -9 *filename*	compresses *filename*, filename will have a .gz appended to the end (-9 is the best compression but is the slowest speed - the default is -6)
gunzip -v *filename*	decompresses *filename*
man *command*	to get manual help on *command*
uname -a	will give the version of UNIX install
mu	a UNIX function that will normally take the user to the MUMPS login
pub	a UNIX function to cd (change directories) to /usr/spool/uucppublic
u	... to /usr/lib/uucp
m	... to /usr/mumps
lpstat -t	will display status of the lp (printer) spooler
cancel *device name* -number	will free up hung printer jobs

FTP is another way of transferring files from system to system. Please see ftp help for further instructions on its use. Non-IHS facilities must use FTP instead of any of the unix scripts/commands described above.

2.8 Sending Taxonomies with your Application

The routine ^ATXSTX can be used to create a post init routine for your application to send taxonomies. Just run ^ATXSTX in programmer mode and follow the instructions.

The name of the primary routine to be generated will be the package prefix followed by TX. For each taxonomy being sent there will be one routine with the same name followed by a letter A-Z. For large taxonomies there will be additional routines with the same letter A-Z followed by a letter A-Z.

3.0 Contact Information

If you have any questions or comments regarding this distribution, please contact the OIT Service Center by:

Phone: (505) 248-4371 or
(888) 830-7280

Fax: (505) 248-4363

Web: http://www.rpms.ihs.gov/TechSupp.asp

Email: ITSCHelp@mail.ihs.gov