



Indian Health Service  
Office of Information Technology



# **RPMS Programming Standards and Conventions**

JULY 2009

# Table of Contents

<b>Record of Changes .....</b>	<b>vi</b>
<b>1.0 Purpose, Policy, &amp; Standards and Conventions Committee .....</b>	<b>1</b>
1.1 Purpose .....	1
1.2 Policy .....	1
1.2.1 Conformance .....	1
1.2.2 Quality Control .....	1
1.2.3 Definition/Appeal .....	1
1.2.4 Development .....	1
1.3 Standards and Conventions Committee Charter .....	2
1.3.1 Purpose .....	2
1.3.2 Background .....	2
1.3.3 Mission .....	2
1.3.4 Authority .....	3
1.3.5 Responsibilities .....	3
1.3.6 Meetings .....	4
1.3.7 Membership .....	4
1.3.8 Chairperson .....	4
1.3.9 Agenda Setting .....	5
1.3.10 Charter Review .....	5
1.4 Standards and Conventions Committee Procedures .....	5
1.4.1 Proposals for Changes .....	5
1.4.2 New Programming Standards and Conventions .....	6
1.4.3 Conformance .....	7
1.4.4 Requests for Exemptions from Programming Standards and Conventions .....	7
1.4.5 Publication of Current Programming SAC .....	8
<b>2.0 Programming Standards and Conventions .....</b>	<b>9</b>
2.1 General Programming Standards and Conventions .....	9
2.1.1 Document Definitions .....	9
2.1.2 Files .....	12
2.2 M Language Programming Standards and Conventions .....	12
2.2.1 ANSI Standards .....	13
2.2.2 Routines (Routine Structure and Format) .....	15
2.2.3 Variables .....	20
2.2.4 Commands .....	25
2.2.5 Functions .....	27
2.2.6 Name Requirements .....	28
2.2.7 Options (Option file entries) .....	29
2.2.8 Device Handling .....	30
2.2.9 Date Processing .....	30
2.2.10 Type A Extensions .....	30

2.2.11	Miscellaneous Standards .....	31
2.2.12	Conventions.....	32
<b>3.0</b>	<b>Interface Programming Standards and Conventions.....</b>	<b>34</b>
3.1.1	User Interface Standards for Scroll and Screen Modes.....	34
3.1.2	User Interface Conventions for Scroll and Screen Modes .....	35
3.1.3	User Interface Conventions for GUI Mode.....	36
3.2	Operating System Programs, Scripts, and Files Standards.....	37
3.2.1	Purpose .....	37
3.2.2	Standards .....	37
<b>4.0</b>	<b>Appendix A: RPMS Namespace Assignments.....</b>	<b>39</b>
4.1	Purpose .....	39
4.1.1	Responsibility .....	39
4.2	General Standards.....	39
4.2.1	Naming Assignments.....	39
4.2.2	Other Standards .....	40
4.3	File Numbering Conventions .....	42
4.3.1	Reserved File Numbers .....	42
4.3.2	Multiple Files.....	42
4.3.3	Common Pointer Files .....	43
4.3.4	Area ISCs Files.....	43
4.3.5	Locally Developed Application.....	43
4.3.6	File Manager File Numbers .....	43
<b>5.0</b>	<b>Appendix B: UNIX/Windows File Naming Standards .....</b>	<b>44</b>
5.1	Purpose .....	44
5.1.1	Distributed Applications .....	44
5.2	Standards for RPMS Application Distribution Files .....	44
5.2.1	Positions 1-4.....	44
5.2.2	Positions 5-6.....	44
5.2.3	Positions 7-8.....	44
5.2.4	Position 9.....	45
5.2.5	Position 10.....	45
5.2.6	Position 11.....	45
5.2.7	Position 12.....	46
5.3	Standards for RPMS Application Patch Files.....	46
5.3.1	Position 10.....	46
5.3.2	Position 11.....	46
5.3.3	Position 12.....	46
5.4	Standards for Host Operating Shell Programs.....	47
5.5	Standards for Documentation Files.....	47
5.5.1	Position 8.....	47
5.5.2	Positions 10-12.....	47
5.6	Standard for Final Distribution File.....	48
<b>6.0</b>	<b>Appendix C: Version Numbering for RPMS Systems.....</b>	<b>49</b>

6.1	Purpose .....	49
6.2	Definitions .....	49
6.2.1	Distribution Version .....	49
6.2.2	Target Version .....	49
6.2.3	Iteration .....	49
6.2.4	Sequence Number.....	49
6.2.5	Format of UNIX file name .....	50
6.2.6	Format of M Routines .....	50
<b>7.0</b>	<b>Appendix D: Standards for Submission of Data to NPIRS.....</b>	<b>51</b>
7.1	Overview.....	51
7.2	Facility Procedure .....	51
7.2.1	Creation of Transaction Global .....	51
7.2.2	Transmission of Global to Area .....	54
7.3	Area Procedure.....	55
7.4	NPIRS Procedure .....	56
7.4.1	IHS Subsystems .....	56
7.4.2	Post Updating .....	57
7.5	Formats of Globals Created at Facility.....	58
<b>8.0</b>	<b>Appendix E: RPMS Documentation Standards.....</b>	<b>59</b>
8.1	Purpose .....	59
8.2	General Standards.....	59
8.2.1	Definitions.....	59
8.2.2	Mandatory Documentation Components .....	60
8.2.3	Preparation .....	60
8.2.4	Reference .....	60
8.2.5	Installation Guide and Release Notes .....	60
8.2.6	Contents .....	61
8.2.7	Technical Manual .....	62
8.2.8	Contents .....	62
8.2.9	User Manual .....	64
8.2.10	Contents .....	64
8.2.11	Security Manual.....	66
8.2.12	Contents .....	66
8.2.13	SAC Developers' Handbook.....	70
8.2.14	Patch Documentation Requirements .....	70
8.3	Documentation Style Standards .....	71
8.3.1	Using the RPMS Templates .....	71
8.3.2	Using Styles.....	72
8.3.3	Cover Page Layout.....	72
8.3.4	Headers and Footers.....	73
8.3.5	Page Numbering.....	75
8.3.6	Page Breaks .....	75
8.3.7	Font .....	75
8.3.8	Italic type .....	76

8.3.9	Underlined type .....	76
8.3.10	Headings and Subheadings .....	76
8.3.11	Figure Numbering.....	76
8.3.12	Item/List Numbering .....	76
8.3.13	Appendix .....	77
8.3.14	Figures .....	77
8.3.15	Screen Captures.....	80
8.3.16	Referring to Package Names.....	82
8.3.17	Introducing Procedures .....	82
8.3.18	Referring to System Prompts.....	82
8.3.19	User Input.....	83
8.3.20	Computer Output.....	84
8.3.21	Placement of Tables.....	84
8.3.22	Referring to Acronyms.....	85
8.3.23	Capitalization.....	85
8.3.24	Punctuation as Command, Navigation Response .....	86
8.3.25	Keys .....	86
8.3.26	Enter/Return Key.....	86
8.3.27	Caret (^).....	87
8.3.28	General Word Usage Guidelines .....	87
<b>9.0</b>	<b>Appendix F: IHS RPMS GUI Standards.....</b>	<b>89</b>
9.1	Purpose .....	89
9.2	General Specifications and Guidelines .....	89
9.3	IHS Programming Naming Conventions .....	89
9.4	File Systems .....	90
9.5	Installation .....	91
9.6	Namespacing.....	91
9.6.1	COM Objects.....	91
9.6.2	.Net Objects.....	91
9.6.3	RPMS Kids Files.....	92
9.6.4	Client Naming Conventions .....	92
9.7	Security.....	92
9.7.1	Data Storage on Client .....	92
9.7.2	Broker Interface.....	93
<b>10.0</b>	<b>Appendix G: Data Dictionary .....</b>	<b>94</b>
10.1	Field Numbering and Data Placement Conventions .....	94

## Record of Changes

Date	Section	Change
05/18/05	2.2.1.7	Added “The version number incorporated into any patch name must be exactly the same as the application’s version number. No extra trailing zeros will be allowed.”
	2.2.1.3	Changed “who is currently responsible for maintenance and development of the package” to “(author) of the routine”
	2.2.1.10	Added new section “Use of Sensitive Patient Tracking with Patient Lookup”
	2.2.7.2	Added “with menu options”
	2.2.12.5	Added new section “Assignment of Port Number for Interfaces”
	2.2.13.7	Added new section “Data Dictionary Field Numbering Conventions”
	2.2.19	Changed “APCDALV routine” to “current visit creation API” Added “Documentation on this API can be found in the Developer Tools document.”
06/02/05	10.0	Added new Appendix H
06/08/05	2.2.11.3	Replaced wording with “Any read from or write to data in another RPMS application will be made with a published API.”
07/2005	7.0	Revised Appendix F to update style standards, and added security section in technical manual.
	All	Revised entire document to new format
07/26/05	2.2.2.23	Added new section “Routine Source Code”
07/29/05	9.0	Replaced Appendix G with revision from Horace and FJ.
08/05	8.0	Revised Appendix F
08/29/05	2.2.2.23	Revised to read “All RPMS applications must submit all source code for technical verification”
09/07/05	N/A	Revised document date to September 2005
	1.0	Changed references from “SET” to “DIT”

Date	Section	Change
	2.2.1.9	<p>Changed “Developer Tools” to “Developers’ Handbook”</p> <p>Deleted term DBIC</p> <p>Changed “are accepted and documented by the DBIC and listed on the DBA menu on IHS Mail.” to “are accepted by the DBA.”</p> <p>Added “VistA stands for Veterans Health Information Systems and Technology Architecture.”</p> <p>Replaced membership list with “Membership consists of all Federal RPMS programmers in OIT, as approved by their Division Director, one representative from Software Quality Assurance and optionally one representative from the Area ISC Committee. The number of members of the SACC can vary.”</p>
	5.0	Changed Appendix C title from “Unix/DOS” to “Unix/Windows”
	7.0	Added to Appendix E “IHS is converting from NPIRS to a National Data Warehouse and the most recent documentation for exporting to the NDW is on our web page at: <a href="http://www.ndw.ihs.gov">www.ndw.ihs.gov</a> .”
02/01/06	2.2.1.11	Added new standard: Patient Merge Compatibility
03/04/06	1.3	Added new SAC Charter
05/24/06	2.2.9.3	Added new standard: Reports in Array and Browse Mode
	2.2.11.3	Added new standard: API Backwards Compatibility
06/14/06	4.0	Changed Request for Exemption form to reflect new organization.
	6.3.1.1	Changed reference to routine in file name to reference to KIDS file
	6.3.1.2	Changed reference to routine in file name to reference to KIDS file
7/21/06	8.2.4	Removed section 8.2.4 Documentation Descriptions & Standards and upgraded subheading for each required manual so they show on the Table of Contents
08/09/06	2.2.1.5	Added to end of standard to include in the KIDS build, the NTEG routine with the checksums for all routines in the application.
	8.2.4.5	Added Installation Instructions standard, the requirement to have users run Verify Checksums in Transport Global and to include routines with checksums in the instructions.
	8.2.5.8	Replaced current Package Security section for Technical Manual with new template.
	8.2.6.7	Added new standard to User Manual for Rules of Behavior section.
	8.2.6.8	Added to User Manual’s Package Management section the requirement to describe how any security, event, or audit logs are to be reviewed.
	8.2.9	Added to Patch Documentation standard that notes file must list routines and their checksums.

Date	Section	Change
08/23/06	2.2.3.17	Changed standard to reflect VA standard by eliminating the restriction against using the intrinsic variables: \$SEC[ODE], \$ES[TACK], \$ET[RAP], Added VA convention on how to use \$ETRAP.
10/18/06	2.1.1.16	Added new definition – Standard Configuration
11/08/06	Various	Removed all references to SACC Web Board. SACC information can be found on the RPMS SAC page of the RPMS Home Page.
	2.4.2.6, 8.2.9	Removed standard requiring that patches be entered into the IHS Patch Module of IHS MailMan.
5/16/07	8.2.2	Added Security Manual to list of mandatory documentation components.
	8.2.5.8	Changed security section in Technical manual to a separate Security Manual that will be kept internal to OIT and is detailed in section 8.2.7.
June 2008	All	Annual Review
03/20/09	Various	Removed Heading 5 and re-tagged those headings with a Subhead style. Valid heading levels are 1 through 4 (e.g., 2.0, 2.1., 2.1.1, 2.1.1.1).
	1.4.1	Correct Propose a Change form. The form will be available at the IHS RPMS web site on the RPMS SAC page, and so noted in this section.
	1.4.4	Example of “Request for Exemption” form removed (Appendix B). Form will be available at the IHS RPMS web site on the RPMS SAC page, and so noted in this section
	2.2.8	Correct references for OPEN, CLOSE commands
	4.0	Appendix B, Request for Exemption to RPMS Programming Standards form <b>removed</b> . Direct reference to form on RPMS web site (Section 1.4.4)  <b>Note:</b> Removal resulted in renumbered Sections and appendices, where section 5.0 Appendix C through Section 10.0 Appendix H are now Section 4.0 Appendix C through Section 9.0 Appendix H.
	7.2.6.3	Move Required Resources (Mandatory) section to appear after Release Notes section (formerly 7.2.6.7)
	7.2.8.3	Move Preface section of Technical Manual to appear after TOC (per <i>The Chicago Manual of Style</i> 15th edition), and change to optional (formerly 8.2.5.3).
	7.2.10.3	Move Preface section of User Manual to appear after TOC (per <i>The Chicago Manual of Style</i> 15th edition), and change to optional (formerly 8.2.6.3).



<b>Date</b>	<b>Section</b>	<b>Change</b>
	7.2.10.8	Specify location of “Rules of Behavior” (Mandatory) section in User Manual.
	7.2.13	Replace “Users Guide to Computing” section/reference with “SAC Developers’ Handbook” section/reference.
	7.3	Documentation Style Standards rewritten to reflect changes required to comply with 508 accessibility compliance requirements (formerly 8.3).

## **1.0 Purpose, Policy, & Standards and Conventions Committee**

### **1.1 Purpose**

Programming Standards and Conventions (SAC) mandate functional soundness and technical correctness of Resource & Patient Management System (RPMS) programs, which are written in MUMPS (M) and other programming environments, and all other programs that interface with RPMS. The purpose of this document is to provide a cornerstone for all national software distributed throughout the Indian Health Service (IHS).

### **1.2 Policy**

#### **1.2.1 Conformance**

All RPMS software developed for IHS will conform to the Programming Standards and Conventions. In areas where specific IHS standards have yet to be adopted, software development will conform to accepted industry standards.

#### **1.2.2 Quality Control**

The IHS Programming Standards and Conventions have been established as a key aspect of quality control under the Division of Information Technology (DIT), Office of Information Technology (OIT).

#### **1.2.3 Definition/Appeal**

The IHS Standards and Conventions Committee (SACC) defines the Programming Standards and Conventions and serves as the source of appeal when issues of conformity with Programming Standards and Conventions arise.

#### **1.2.4 Development**

Standards and Conventions Committee members and other technical experts within IHS will participate in the development of Programming Standards and Conventions, in collaboration with public and private subject experts and with national and international standards organizations.

## 1.3 Standards and Conventions Committee Charter

### 1.3.1 Purpose

The purpose of the Standards and Conventions Committee (SACC) is to define and review national standards for Resource and Patient Management System (RPMS) software developed within the Indian Health Service (IHS).

### 1.3.2 Background

The RPMS environment evolved from different development efforts at different sites. Now, individual applications and their integration are very complex. The *RPMS Programming Standards and Conventions* document, developed and maintained by the Standards and Conventions Committee, contain policies, procedures, and notes to guide maintenance and development.

Two extensions to the *Programming Standards and Conventions* document provide additional guidance:

- *Standards and Conventions Developers' Handbook* (resource for the development of RPMS applications, including developers' tools)
- *Standards and Conventions Documentation Style Guide* (resource for guidance when writing RPMS related documentation)

### 1.3.3 Mission

The mission of the Standards and Conventions Committee is to

- Provide a cornerstone of functional soundness and technical correctness for all national software developed and distributed within IHS.
- Perform technical reviews of major software development projects as requested by the project officer or OIT management.
- Provide core utilities and techniques for uniformity.
- Rule on disputes about the interpretations of the Programming Standards and Conventions (SAC).
- Rule on requests for exemptions to a particular section of the Programming Standards and Conventions.
- Sponsor task forces to address cross-application issues, such as messaging and integration.

### 1.3.4 Authority

The Standards and Conventions Committee (SACC), as a whole, will report to the IHS Chief Information Officer (CIO), through the Director, Division of Information Technology (DIT), Office of Information Technology (OIT), in accordance with the Department of Health and Human Services IT Consolidation Metrics memorandum dated June 3, 2003, which states:

“All IT infrastructure organizations and staff within each large OPDIV must report to the OPDIV CIO.”

### 1.3.5 Responsibilities

The Standards and Conventions Committee (SACC) will carry out the responsibilities and authorities as provided in this charter or delegated to it in writing by the IHS CIO and the Director, DIT. The SACC is responsible for the following:

- a. Promoting consistency in development standards by authoring the programming Standards and Conventions to include Graphical User Interface Standards and Conventions (GUI SAC) and a Developers' Handbook.
- b. Serving as an arbitrator when issues of conformance to Standards and Conventions arise. Requests for exemptions from existing RPMS standards shall be submitted by the development community to the SACC via written or electronic media. A simple majority decision by the Committee sends the recommendation, regarding interpretation or waivers of the standards, to the Director, DIT for consideration.
- c. Reviewing and updating the Standards and Conventions document at a minimum of every two years, or as needed. Distribution will be made to all RPMS developers, including those under contract to IHS.
- d. Calling upon individuals outside of its membership on an ad hoc basis as necessary, to serve on focused work groups involved in SACC activities.
- e. Participating with other technical experts in IHS on the development of Standards and Conventions, in collaboration with public and private subject experts and with the national and international standards organizations.
- f. Reporting directly to and forwarding recommendations to the Director, DIT.

### 1.3.6 Meetings

The Standards and Conventions Committee meetings will be held at least once a month via teleconference. These meetings shall focus on

- Updating the SAC documents, processing requests for exemptions
- Identifying cross-application issues
- Coordinating software technical reviews
- Issuing interpretations to clarify the SAC documents

Other ad-hoc meetings to discuss specific assignments may be held as needed or at the request of the Chair or Director, DIT.

### 1.3.7 Membership

Membership consists of interested staff members of OIT Divisions and the Area Information Systems Coordinator community. A quorum for voting consists of at least one member each from the DIT, Division of Enterprise Project Management (DEPM), Division of Program Management and Budget, Division of Information Security (DIS), and the OIT Software Quality Assurance (SQA) team.

### 1.3.8 Chairperson

- Election.** Immediately following the approval of the Standards and Conventions Charter, the Standards and Conventions Committee (SACC) shall hold a meeting to select one Chairperson. The Chairperson will be selected by majority vote to serve for a two-year term. After the two-year term expires, the SACC will meet to select a new Chairperson or to continue the term of the current Chair by majority vote.
- Absence.** An alternate SACC member or representative from the DIT (previously designated by the SACC Chairperson) will chair the meeting and/or conference call in the event the Chairperson is unavailable to perform his/her responsibilities for a particular meeting.
- Resignation/Inability to Continue.** In the event the Chairperson is unavailable to continue to perform his/her responsibilities and/or submits a letter of resignation, a new chairperson may be selected to finish the term by a majority vote.
- Chair Responsibilities.** The SACC Chairperson is responsible for the following:
  - Organizing agendas, publishing meeting minutes, and tracking action items
  - Transmitting all programming SAC changes to RPMS programmers via approved electronic media (e.g. email service, web pages)
  - Establishing new standards as needed

- Coordinating standards revisions
- Coordinating waivers and/or exemption requests
- Coordinating software technical reviews
- Tracking progress of subcommittee task forces dealing with cross-application issues
- Reporting SACC decisions to the RPMS programming and OIT management communities

### 1.3.9 Agenda Setting

The Standards and Conventions Committee (SACC) and the SACC Chairperson will jointly establish meeting agendas. Agendas will be distributed to all SACC members at least five (5) working days prior to meetings.

### 1.3.10 Charter Review

The Standards and Conventions Committee Charter shall be reviewed as needed to evaluate its effectiveness and to incorporate any improvements. Changes to the SACC Charter must be approved by the Director, DIT, and the IHS CIO.

This Charter was signed on March 24, 2006 by Keith Longie, IHS CIO and became effective on that date.

## 1.4 Standards and Conventions Committee Procedures

### 1.4.1 Proposals for Changes

If an individual identifies the need for a change in the Programming Standards and Conventions, a request must be made to the Standards and Conventions Committee via written notification, using the “Request for Change to RPMS Programming Standards and Conventions” form.

To obtain a copy of this form, go to the RPMS SAC page of the IHS RPMS internet web site:

<http://www.ihs.gov/Cio/RPMS/index.cfm?module=home&option=index>

#### **1.4.1.1 Proposal request requirements**

Proposal requests should include the following information:

- Current section and wording
- Recommended change
- Justification and comments

#### **1.4.1.2 SACC review**

After a review of the need and impact of the proposed change, the Standards and Conventions Committee Chairperson will call for a vote on the request. A simple majority decision by the SACC shall be required to grant a change to the Programming Standards and Conventions.

#### **1.4.1.3 Decision notification**

Notification of the decision of the Standards and Conventions Committee will be made to the individual requesting the change and will be forwarded to the Team Lead of OIT, where it will be processed as a new Programming SAC.

### **1.4.2 New Programming Standards and Conventions**

#### **1.4.2.1 Publication of changes**

The Standards and Conventions Committee (SACC) will publish the proposed Programming SAC changes (either individual paragraph or the entire document) on the RPMS SAC page, of the IHS RPMS internet web site,

<http://www.ihs.gov/Cio/RPMS/index.cfm?module=home&option=index>

#### **1.4.2.2 Approval**

After the SACC votes to accept the proposed Programming SAC, the SACC will forward it to the Director, DIT for approval.

- If the Director, DIT approves the proposed Programming SAC document within the 30 days, it is considered effective immediately.
- If the Director, DIT disapproves the proposed Programming SAC document, the SACC will make the appropriate changes and resubmit it to the Director, DIT, repeating the process until the Director, DIT approves the SAC document.
- If the Director, DIT does not respond with approval/disapproval within 30 days from the day the proposed Programming SAC document is submitted to him/her, the Programming SAC is considered approved.

### 1.4.2.3 Effective date

The new Programming SAC becomes effective on the date of approval by the Director, DIT. After approval, the new Programming SAC will be announced and made available on the RPMS SAC page of the IHS RPMS internet web site:

<http://www.ihs.gov/Cio/RPMS/index.cfm?module=home&option=index>

### 1.4.3 Conformance

Following the effective date of the new Programming SAC, newly submitted RPMS packages must adhere to the new Programming SAC unless otherwise exempted, as presented in this document.

### 1.4.4 Requests for Exemptions from Programming Standards and Conventions

If an individual identifies the need for an exemption from the Programming Standards and Conventions for a given package, a request must be made to the Standards and Conventions Committee via written notification, using the “Request for Exemption to RPMS Programming Standards” form.

To obtain a copy of this form, go to the RPMS SAC page of the IHS RPMS internet web site:

<http://www.ihs.gov/Cio/RPMS/index.cfm?module=home&option=index>

<p><b>Note:</b> Exemption requests should be submitted as early as possible in the development cycle.</p>
---

#### 1.4.4.1 Exemption request requirements

Exemption requests should include the following information:

- Package name and version number
- Type (Permanent or temporary)
- Violated Standard
- Justification and comments

#### 1.4.4.2 SACC review

After reviewing the need and impact of the proposed exemption, the Standards and Conventions Committee Chairperson will call for a vote on the request. A simple majority decision by the SACC shall be required to recommend a Programming Standards and Conventions exemption to the Director, DIT.



#### **1.4.4.3 Decision notification**

Notification of the decision of the Director, DIT will be made to the requestor via email.

#### **1.4.5 Publication of Current Programming SAC**

To promote rapid dissemination, the current version of the *RPMS Programming Standards and Conventions* will be published on the RPMS SAC page of the IHS RPMS internet web site for all users to access.

<http://www.ihs.gov/Cio/RPMS/index.cfm?module=home&option=index>

All developers should recognize the need for periodic review of this site.

## **2.0 Programming Standards and Conventions**

This version of the Programming Standards and Conventions (SAC) was adopted by IHS on January 14, 2003, revised between May and September 2005, and again revised between January and November 2006.

### **2.1 General Programming Standards and Conventions**

The definitions, standards, and conventions within this section apply to all programming and user environments used in the Resource and Patient Management System (RPMS). This includes all applications that use commercial software products as an integral part of RPMS (but not to the commercial application itself) and to all development environments, including but not limited to, programming languages such as MUMPS (M) or Pascal, and graphical user interface (GUI) environments such as Visual Basic, Delphi, and others.

#### **2.1.1 Document Definitions**

##### **2.1.1.1 Application Programmer Interface (API)**

See Section 2.1.1.13, "Published Entry Point (PEP)."

##### **2.1.1.2 Conventions**

Programming guidelines that are designed to promote consistency and safety across RPMS applications. Exemptions from conventions are not required, but developers are strongly encouraged to follow them.

##### **2.1.1.3 DBA**

Database Administrator.

##### **2.1.1.4 DHCP**

Decentralized Hospital Computer Program. Now called VistA, this is the Department of Veterans Affairs (VA) software equivalent to RPMS.

##### **2.1.1.5 Entry Point**

A line label within a routine, other than the first line of the routine, that is referenced by a "DO" or "GOTO" command from another routine in the same package to which the routine belongs.

#### **2.1.1.6 Exemption**

Authority granted by the Standards and Conventions Committee to a specific VistA/RPMS application that allows that application to not comply with a particular section of the Programming SAC for a specified timeframe.

#### **2.1.1.7 Kernel**

The Kernel is a set of VA software utilities that form the foundation of VistA/RPMS, which includes elements that start with the namespaces XG\*, XLF\*, XPD\*, XQ\*, XT\*, XU\*, XV\*, ZI\*, ZO\*, ZT\*, ZU\*.

#### **2.1.1.8 MailMan**

MailMan is a set of VA software utilities that form the foundation of VistA/RPMS electronic mail and communications, which includes elements that start with the namespace XM\*.

#### **2.1.1.9 Namespace**

A unique set of alpha characters assigned by the DBA, commonly used to uniquely identify package components.

#### **2.1.1.10 Numberspace**

A unique set of numbers assigned by the DBA, commonly used for assigning FileMan files.

#### **2.1.1.11 Package**

A set of routines, files, options, templates, security keys, screens, bulletins, functions, help frames, forms, blocks, objects, protocols, dialogues, list templates, windows, and such, namespaced according to DBA requirements, that function as a unit.

#### **2.1.1.12 Programmer Mail Group**

The Programmer mail group (G. Programmer) is established on the IHS MailMan system for discussion of technical issues. This mail group is used to disseminate information to the IHS RPMS development community.

<p><b>Note:</b> Subsequent changes to the Programming SAC will indicate a migration from the use of MailMan to the use of WebBoard.</p>
---

#### **2.1.1.13 Published Entry Point (PEP)**

A line label in a routine, other than the first line of a routine, which has been documented internally as a published entry point, and documented in the Technical manual as a published entry point, that is, available to be referenced by a DO or GOTO command from a routine external to the package to which the routine belongs. This is also known as an Application Programmer Interface (API).

#### **2.1.1.14 SACC Web Site**

A documentation and conversation facilitator, based on Internet technology, for issues related to the IHS Standards and Conventions, which is available at:

<http://www.ihs.gov/Cio/RPMS/index.cfm?module=home&option=SAC>

#### **2.1.1.15 Standard**

A rule that all VistA/RPMS software must follow.

#### **2.1.1.16 Standard Configuration**

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.

#### **2.1.1.17 Supported Reference**

Routine labels, extrinsic functions, files, or global nodes that are accepted and documented by the DBA and listed on the DBA menu on IHS Mail.

#### **2.1.1.18 User**

A person interacting with computer applications.

#### **2.1.1.19 Utility**

A callable routine line tag or function, such as a universal routine like XB\*, ZIB\*, or AUPN\*, which is usable by anyone.

#### **2.1.1.20 Up-hat**

Caret symbol ( ^ ), also known as a “hat,” which is denoted by pressing Shift+6 on the keyboard. It is used as a piece delimiter in a global.

### **2.1.1.21 VA FileMan**

The Database Management System for VistA/ RPMS, with namespaces DD\*, DI\*, and DM\*.

### **2.1.1.22 VistA**

Department of Veterans Affairs (VA) software equivalent to the RPMS. VistA stands for Veterans Health Information Systems and Technology Architecture.

## **2.1.2 Files**

### **Naming requirements for files used by VistA/RPMS packages**

#### **2.1.2.1 VA FileMan**

All VA FileMan files in the MUMPS (M) language environment must be numberspaced and namespaced in the spaces assigned to the package by the DBA. For more information, see Section 4.0, “Appendix A: RPMS Namespace Assignments.”

#### **2.1.2.2 DOS, VMS, UNIX or Other Host Files**

All DOS, VMS, UNIX, or other host files created or exported as part of a VistA/RPMS application must be namespaced in the namespace assigned by the DBA. For more information, see Section 5.0, “UNIX/Windows File Naming Standards.”

#### **2.1.2.3 Exporting Script Files**

Packages exporting script files should provide script files for the variety of terminal emulation packages commonly used with the VistA/RPMS applications.

#### **2.1.2.4 Exporting Spreadsheets**

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

## **2.2 M Language Programming Standards and Conventions**

All MUMPS (M) based VistA/RPMS/ software will meet the following standards and comply with the spirit of the conventions.

## 2.2.1 ANSI Standards

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

### 2.2.1.1 Standard Dictionary releases

All development will be produced using the most recent standard dictionary releases, as distributed by the OIT.

### 2.2.1.2 Fields within a string

Fields within a string will be separated by the caret (up-hat) symbol, except when the data within the string must contain the caret symbol itself.

### 2.2.1.3 Re-indexing of File Manager MUMPS cross-references

Re-indexing of individual File Manager MUMPS cross-references must not result in an error. This affects both SET and KILL logic.

```
I $D(XYZ),XYZ="" S ^GBL("AC",XYZ)=""
```

### 2.2.1.4 FileMan entry points

Only standard documented FileMan entry points will be called by IHS routines.

### 2.2.1.5 Restrictions when creating a package for distribution

All packages submitted for certification for release to Indian/Tribal/Urban (I/T/U) sites will be in a KIDS build, will contain only those components that belong to the package, and will not contain global data except as allowed by KIDS. Components that are part of one package will not normally be distributed with another package. Exceptions are existing authorized applications that are related but require separate KIDS builds, which can be placed in a composite transport. Every build will include the current NTEG routine that stores the checksums of all routines in this application.

### 2.2.1.6 File Manager file access code security

All packages delivered to SQA for verification and distribution will come with complete File Manager file access code security in the data dictionary. All files will have programmer only data dictionary access. This assures that security will be present initially, if dictionaries are deleted prior to installation. The only exception to having file access security codes is in the READ field in a DD, in which case it can be without access or blank. If developers need file access codes for their packages, they will coordinate this with the DBA.

### **2.2.1.7 Version information**

Complete version information will be set in the package file (DIC(9.4)), and all files for a package will have the version number in DD(file#,"VR"). The version number incorporated into any patch name must be exactly the same as the application's version number. No extra trailing zeros are allowed.

### **2.2.1.8 LAYGO restriction**

Application programs will not allow LAYGO entry to the New Person file. Packages which need to be able to add to this file will have separate locked options that come with the package, but that are not assigned to any of the package menus as distributed. Application programs will not allow LAYGO to standard tables.

### **2.2.1.9 Entry in file 9000010**

Packages external to the Patient Care Component (PCC) that are passing data to the Visit and V-files must use the current visit creation API to select or create a visit entry in file 9000010. The APCDALVR will be used to append a V-file entry to a visit file entry. Documentation on this API can be found in the *Standards and Conventions Developers' Handbook*.

### **2.2.1.10 Use of Sensitive Patient Tracking with patient lookup**

If an application does not use the standard FileMan lookup for patient selection, then the API to check for Sensitive Patient Tracking status must be called and business rules followed. Documentation on this API can be found in the *Standards and Conventions Developers' Handbook*.

### **2.2.1.11 Patient merge compatibility**

All FileMan files with pointers to either the VA Patient (#2) or Patient (#9000001) files must have a *regular* cross-reference defined and populated for each patient pointer field with no triggers or MUMPS cross-references that change other fields, to be Patient Merge compliant.

If a file is structured so that the generic merge utilities cannot successfully merge any part of the data contained in that file, then that file is not Patient Merge compliant. This includes files with variable pointers, DINUMed files with word-processing fields, sequencing issues, and any non-standard global references. If any file in an application is not compliant, then a special merge routine must be distributed and placed in the Affects Record Merge multiple of the Package (#9.4) file. For full details, see the detailed documentation in the *Standards and Conventions Developers' Handbook*.

## 2.2.2 Routines (Routine Structure and Format)

### 2.2.2.1 First line

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

The first line of a routine has the following format:

(routine-name) ;(Agency)/(site)/(programmer) - (brief-description); (date-of-edit)

Where

(Agency)/(site)/(programmer) identifies the developing Agency and site, and the programmer (author) of the routine.

(date-of-edit) - The use of time in date-of-edit is optional

Example:

```
AGPAT ;IHS/OKC/LD - Patient registration driver; JUL 20, 1986
```

**Note:** Routines generated by VA FileMan or Kernel (e.g., INIT, ONIT, NTEG, and compiled routines) and other compiled routines used in exporting a package, need not comply with this standard.

### 2.2.2.2 Second line

The second line of a routine must be in the following format:

LABEL-optional<ls>;version number;package name,\*\*pm,...pn\*\*;version date

Where:

*version number* must be the same on all of the package namespaced routines, including INIT, ONIT, and others.

*patch numbers* (\*\*pm,...pn\*\*) are the applied patch numbers, separated by commas. If there are no patches, this “;” piece is null.

*version date* must be the same on all of the package namespaced routines, including INIT, ONIT, and others.

**Note:** **Compiled routines** from templates, cross-referenced, etc., by VA FileMan during or after package installation are exempt from the second line requirement.



### **2.2.2.3 Third line**

If local modifications to a routine are restricted or prohibited by policy or directive, the third line should contain an appropriate notice; for example:

"Per VHA Directive 10-92-142, this routine should not be modified"

#### **Labels**

Labels are limited to eight (8) characters and may not contain lower case characters.

#### **Label+offset**

Label+offset references will not be used except for \$TEXT references.

#### **Lines referenced by \$TEXT**

Lines referenced by \$TEXT for use other than to check for the existence of the routine must be in the following format:

(LABEL-optional)<ls>;text or M code

### **2.2.2.4 Linebody**

The linebody must contain at least 1 character, must not exceed 245 characters in length, and must contain only the ASCII characters values 32-126. Commands, functions, local and global variable names, system variables, SSVNs, etc., must be uppercase.

### **2.2.2.5 Routine names**

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

- NAMESPACE\_I\* (Exceptions are Kernel, VA FileMan, and routines created to support the INIT process.)
- NAMESPACE\_NTE\* (Exception are package integrity routines.)

### **2.2.2.6 Routine size**

The maximum routine size, as determined by executing `^%ZOSF("SIZE")`, is 15,000 characters. The combination of routine and symbol table must run in the partition size specified in the appropriate VistA/RPMS operating system manual/cookbook.

### **2.2.2.7 Vendor specific subroutines**

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

### 2.2.2.8 TaskMan globals

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

### 2.2.2.9 Naked references

Naked references must be either preceded by the full reference defining it or be documented, where

- An appropriate preceding full reference is one that is on the same physical routine line as the naked reference and has no code between it and the naked reference that branches in any manner to other lines of code or executables.
- Those naked references requiring documentation must be documented within the routine in the immediate vicinity of the naked reference. Those naked references that are preceded by a full reference that is outside of the routine where the naked reference is used must have documentation in both the routine containing the full reference and the routine containing the naked reference. This documentation must be in the immediate vicinity of the appropriate reference.

In **called utilities**, naked references are exempt. For example,

```
S DIC=200,DIC(0)="AEQ",DIC("S")="I $L($P($G(^1)), "^",9))" D ^DIC
```

is a legitimate use of the naked reference.

### 2.2.2.10 % routines

No application will distribute % routines. (Exemptions are the Kernel and VA FileMan).

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

No routine that will be resident in the Library (MGR) account will use VIEW commands using variables as arguments that could be set by a programmer prior to executing the code.

### 2.2.2.11 Z routines

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

When creating local routines to be added to an existing national package, the routine name begins with the namespace followed by the letter "Z."

### 2.2.2.12 Extended reference syntax

Routines may not be invoked using the extended reference syntax, `D ^|"VAH"|TAG^ROUTINE`, which is illegal.

### 2.2.2.13 Entry points

Lines that are entry points (referenced by a DO or GOTO in other routines) will consist only of a label and a semi-colon followed by “entry point” or the abbreviation EP. An optional comment that describes the entry point may follow the entry point comment. For example:

```
CLEAR ; EP - CLEARS SCREEN  
CLEAR ; ENTRY POINT - CLEARS SCREEN
```

### 2.2.2.14 Published entry points

Lines in a routine, other than the first line, that are published entry points (PEP) must consist of a label (and, optionally, the formal parameter list) and a semicolon, followed by “Published Entry Point” or the abbreviation PEP, followed by a comment.

### 2.2.2.15 Changed lines

Lines changed from the standard release will be marked with a semicolon and comment. This comment will contain the Agency and site identification, the programmer's initials, the date of the change, and reason for the change. Duplicate and comment out the original line of code. Additional comment lines may be used. For example:

```
CHGLINE ;  
;S ZXXX=3  
S AXXX=3 ; IHS/ABQD/FBD 04/01/96  
;This was changed to conform to IHS SAC
```

### 2.2.2.16 IHS changes to VA packages

IHS changes to VA packages will be incorporated into separate routines or templates using a proper IHS namespace, rather than embedding the changes in the middle of VA programs. (A DO statement will be used to exit to the IHS routines, if needed). When it is not possible to use an external IHS routine when modifying VA packages, add changes in the format outlined in Section 2.2.2.15.

### 2.2.2.17 Comments

Comments will be provided at the start of each independently callable routine, specifying the following items:

Item	Description
<b>Purpose or function</b>	Overall purpose or functions of the routine.
<b>Input Variables</b>	Input variables (variables set up by other routines that are used by this routine).
<b>Output variables</b>	Output variables (variables set by this routine).

### 2.2.2.18 XB/ZIB prefixed routines

AU/AZ prefixed utility routines have been re-namespaced XB/ZIB respectively, where

- All XB prefixed utility routines will refer to utility routines that are totally universal from machine to machine regardless of operating system or hardware.
- All ZIB prefixed utility routines will refer to utility programs that are operating system or hardware dependent.

All program calls to previous AU/AZ prefixed utility routines must be changed to make the program calls to the XB/ZIB equivalent utility routines.

### 2.2.2.19 Facility or Area specific information

Facility or Area specific information needed by a routine will be obtained from tables or input parameters, rather than being hard-coded in the routine.

### 2.2.2.20 Approved exemptions

Approved exemptions to standards must be documented in the routine with a separate comment line adjacent to the line containing the exemption in the form; for example,

```
;IHS exemption approved on DATE
```

### 2.2.2.21 'Namespace' VAR routine

Packages that need to set local variables prior to entering a package will use a routine that is named in the form 'namespace' VAR.

### 2.2.2.22 Data conversion processes

All data conversion processes should be restartable at the point of interruption in the execution.

**2.2.2.23 Syntactically correct lines**

All lines must be syntactically correct. Blanks will not appear at the end of lines.

**2.2.2.24 Routine source code**

All RPMS applications must submit all source code for technical verification.

**2.2.3 Variables**

**2.2.3.1 Local variables**

Lowercase characters in local variable names are prohibited.

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

Example of subscripted variable:

```

NAME(sub1,sub2,...,subn)
(1.)  +$L(NAME)+3
(2.)  +$L(sub1) + $L(sub2) + ... +$L(subn)
(3.)  + 2 * number of subscripts n
(4.)  +15
    
```

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

**2.2.3.2 System-wide variables**

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

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

### 2.2.3.3 Assumed variables

The DT\*, DUZ\*, IO\*, and U variables, referenced elsewhere in this document, are set by the Kernel during sign-on or by VA FileMan, and can be assumed to exist by all VistA/RPMS applications. Refer to the *Kernel Systems Manual* or the *VA FileMan Technical Manual* for their definitions.

- **DUZ or DUZ-Array**

VistA/RPMS packages are not allowed to KILL, NEW, SET, MERGE, READ (into) or otherwise modify the variable DUZ or any DUZ array element, with the exception of DUZ(2) (Exemptions are Kernel and VA FileMan). In order to allow programs to run stand alone or for transport to non-Kernel environments, DUZ and its descendants can be set after confirming that they do not already exist.

- **DUZ(2)**

The VA local variable DUZ(2) will be set for all users upon logon by the Kernel, using the FACILITY field in the New Person. A user will be able to log on to only those sites listed under the FACILITY field in his or her New Person file entry. Any application allowing a user to switch facilities in a package will use this multiple field in the New Person file, when selecting a valid site for the user to switch to.

If a package modifies DUZ(2) after the original set, it will be reset to its original value before exiting the package. A value of 0 in DUZ(2) or the existence of the local variable AUPNLK("ALL") will allow complete lookup ability for all facilities in the Patient File.

### 2.2.3.4 Special variables

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

- **DT**

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

- **DTIME**

The DTIME variable may be changed only to a value less than the existing value of DTIME, but must be restored to its original value before exiting the option. All routines where DTIME is changed but not restored prior to exiting the routine must be documented in the package Technical Manual, including the location where DTIME is restored.

- **U**

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

### **2.2.3.5 IO variables**

VistA/ RPMS packages are not allowed to KILL, SET, MERGE, READ (into) or otherwise modify IO namespaced variables and any of their array elements, except those documented as modifiable in the *Kernel System Manual*. (Exemptions are Kernel, MailMan, and VA FileMan.) The routine XBKVAR can be invoked to set these variables.

### **2.2.3.6 % variables**

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

### **2.2.3.7 Namespaced variables**

A VistA/ RPMS package may declare namespaced, local variables as package-wide. The variables and all of their array elements must be described in the package Technical Manual. A VistA/ RPMS package may not kill or change another VistA/ RPMS package's package-wide variables.

### **2.2.3.8 Required documentation**

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

### **2.2.3.9 Lock tables/local symbol tables**

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

- Documented input and output variables (including globals).
- Supported reference namespaced variables may be changed or killed. For example, the VA FileMan ^DIC call kills the variable DIE, which may exist in the symbol table prior to the call.
- Documented side effects, such as lock table changes, and changes to files.
- The variable %.

These supported references must be documented in the package Technical Manual with a descriptive list of ALL input and resulting output variables.

### 2.2.3.10 Variables passed between packages

Naming requirements for variables passed between packages:

- **Actual List**

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

**Legal:** D BLD^DIALOG(3500010, " ", .IBDATA, "IBX")

**Illegal:** D BLD^DIALOG(3500010, " ", .Y, "IBX")

- **Input Variables**

Input variables that represent local variables into which data will be exchanged must represent a data location that is package namespaced.

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

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

### 2.2.3.11 Global variables

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

The full, evaluated length of a global reference must not exceed 511 characters. The evaluated length is calculated as follows.

Example subscripted variable:

```

^NAME(sub1,sub2,...,subn)
(1.)  +$L(NAME)+3
(2.)  +$L(sub1) + $L(sub2) + ... +$L(subn)
(3.)  + 2 * number of subscripts n
(4.)  +15
    
```

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

### 2.2.3.12 Unsubscripted globals

The KILLing of unsubscripted globals is prohibited. (The VA FileMan EN^DIU2 utility allows the deletion of files stored in unsubscripted globals, and therefore, allows the killing of unsubscripted globals. Application developers must document when calls to EN^DIU2 are made to delete files stored in unsubscripted globals.)



### 2.2.3.13 % globals

READING, KILLing, SETting or MERGing ^% globals is prohibited. (Exemption: Kernel) %Globals will not be created without approval from the SET.

### 2.2.3.14 Global exemptions

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

- **^TMP**

The global ^TMP will be used as a scratch global within a session. The first subscript shall be \$J, or the first two subscripts shall be a package namespaced subscript followed by \$J. The ^UTILITY global will not be used unless necessary to retrieve output from a Kernel or FileMan utility.

- **^XTMP**

The global ^XTMP will be translated, with one copy for the entire RPMS production system at each site. The structure of each top node shall follow the format,

^XTMP(namespaced- subscript,0)=purge date^create date^optional  
descriptive information

and both dates will be in VA FileMan internal date format.

### 2.2.3.15 Executable code

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

- **DD global**

Sets and Kills to the DD global are prohibited.

- **Global variables executed by % routines**

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

### 2.2.3.16 Referencing global variables

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

### 2.2.3.17 Intrinsic (System) variables

Lowercase intrinsic variables are prohibited.

No VistA/RPMS package may use the following intrinsic (system) variables unless they are accessed using the Kernel or VA FileMan supported references:

```
$D[EVICE]
$I[O]
$K[EY]
$P[RINCIPAL]
$Q[UIT]
$ST[ACK]
$SY[STEM],
$Z*
```

To use the new error trapping in a routine,

```
NEW $ESTACK,$ETRAP S $ETRAP="D tag^routine"
```

### 2.2.3.18 Structured System Variables (SSVNS)

Lowercase SSVNS are prohibited.

The following structured system variables may be used only by the Kernel or VA FileMan, or through their supported references:

```
^$CHARACTER
^$DEVICE
^$DISPLAY
^$EVENT
^$GLOBAL
^$JOB
^$LOCK
^$ROUTINE
^$SYSTEM
^$Z*
^$WINDOW
```

## 2.2.4 Commands

<b>Note:</b> Lowercase commands are prohibited.
---

### 2.2.4.1 BREAK

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

#### **2.2.4.2 CLOSE**

Direct use of the CLOSE command is prohibited. Use the routine ^%ZISC.  
(Exemptions are the Kernel, MailMan and VA FileMan.)

#### **2.2.4.3 HALT**

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

#### **2.2.4.4 JOB**

Direct use of the JOB command is prohibited. Use the Kernel Task Manager supported calls to create jobs. (Exemptions are the Kernel and MailMan.)

#### **2.2.4.5 KILL**

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

The **exclusive** form of the KILL command is prohibited. (Exemptions are the Kernel and VA FileMan.)

#### **2.2.4.6 LOCK**

All LOCKs shall be of the incremental or decremental form. All routines will lock nodes to be created or updated. Appropriate error recovery action will be taken if the lock is not obtained. (Exemption is the Kernel.)

All **incremental** LOCKS must have a timeout.

#### **2.2.4.7 NEW**

The **argumentless** form of the NEW command is prohibited.

The **exclusive** form of the NEW command is prohibited (VA Only).

#### **2.2.4.8 OPEN**

The use of the OPEN command is prohibited. (Exemptions are the Kernel, MailMan and VA FileMan.)

#### **2.2.4.9 READ**

All READ commands shall read into local variables, or one of the non-VA FileMan compatible globals, ^TMP and ^XTMP.

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

All user input READ commands shall be terminated by a carriage return. (Exemptions are the Kernel and VA FileMan.)

Developers desiring to implement escape processing (function keys, arrow keys, etc.) must use Kernel-supplied supported references [XGF].

Use of the READ command in a data dictionary is prohibited.

#### **2.2.4.10 USE**

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

#### **2.2.4.11 VIEW**

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

#### **2.2.4.12 MWAPI commands**

No Vista/RPMS package may use the MWAPI Commands, ESTART, ESTOP, and ETRIGGER. (Exemption is the Kernel.)

#### **2.2.4.13 WRITE**

Use of the WRITE command in a data dictionary is prohibited. The call to EN^DDIOL will be used. (Exemption is VA FileMan.)

#### **2.2.4.14 Z\* commands**

The use of Z\* commands is prohibited. (Exemptions are the Kernel and VA FileMan.)

### **2.2.5 Functions**

<b>Note:</b> Lowercase functions are prohibited
---

### 2.2.5.1 Intrinsic Functions

- **\$NEXT**  
Use of the \$NEXT function is prohibited. All RPMS packages should remove all occurrences of \$NEXT. This includes occurrences generated by VA FileMan.
- **\$VIEW**  
The use of the \$VIEW function is prohibited. (Exemptions are the Kernel and VA FileMan.)
- **\$Z\***  
The use of \$Z\* functions are prohibited. (Exemptions are the Kernel and VA FileMan.)
- **\$ORDER**  
Reverse \$ORDER through the local symbol table when the variables are unsubscripted is not supported by Cache and is prohibited. Reverse \$ORDER through subscripted local variables is supported and allowed.

### 2.2.5.2 Extrinsic Functions

- **Parameters**  
Supported References that use parameters will document the elements of the formal list internally within the routine and in the package Technical Manual. Documentation will specify which elements of the formal list are required and which are optional, if any, and those elements that must be passed by reference.
- **Supported extrinsic special variables**  
Extrinsic functions with an empty formal list will be documented within the routine and in the package Technical Manual.

## 2.2.6 Name Requirements

### 2.2.6.1 Package Namespace

Unless approved by the DBA, routine, global, security key, option, template, bulletin, function, screen, help frame, protocol, form, block, list templates, objects, dialogues, remote procedures, etc., names must be consistent with the assigned DBA namespace for the package. The namespace of all IHS-developed packages assigned after January 1, 1994 must begin with B.

### 2.2.6.2 "Namespace" Menu

All IHS packages with menu options will have a top menu option in the form of *NAMESPACEMENU* with a lock on that menu option that is of the form *NAMESPACEZMENU*.

When entering this menu, the current version of the package will be displayed above the options along with the location determined by the current DUZ(2) setting.

## 2.2.7 Options (Option file entries)

### 2.2.7.1 Selection

Option selection must be made through Menu Manager or by using the VA FileMan DIR utility. The DIR utility may only be used at the lowest menu tree level. Hardcoded menu management systems are not allowed.

### 2.2.7.2 Path independence

All options in a package must be path independent, once the steps described in the package Technical Manual for creating and killing package wide variables have been taken.

### 2.2.7.3 After Exit

The following must not exist after exiting an option:

- **Output Variables**  
All documented output variables created by a called supported reference.
- **Locks**  
All documented locks created by a called supported reference.
- **Global Nodes**  
All documented temporary scratch global nodes (e.g., ^TMP and ^UTILITY) created by a called supported reference, with the exception of ^XTMP global data.
- **Variables/Locks/Globals**  
All local variables, locks, and scratch global nodes (except ^XTMP, or other scratch globals designed to be passed between parts of a package) created by the application.

#### **2.2.7.4 Menu options**

All menu options in a package will utilize alpha synonyms, not numeric. This is to allow use of the Kernel menu jumping capability.

### **2.2.8 Device Handling**

All device selection and closing will be made through the use of the Kernel supported references. For specific information about the OPEN and CLOSE commands, see the *Kernel Developer's Guide*, Version 8.0, Section 5.

#### **2.2.8.1 Output queuing**

All output to a hard copy device (e.g., printer) must allow for queuing or use of an auxport printer. Internal device selection must be by device name rather than by \$I. Developers should bear in mind that either IO or \$I variables may be non-numeric. Forced queuing, which is sometimes desirable, will be local site parameter driven.

#### **2.2.8.2 Outputs**

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

#### **2.2.8.3 Reports in array and browse mode**

All new reports must be available to browse on the computer screen (via Browser device, List Manager, or GUI window) and be accessible as an array with all IO controls removed. This allows easy viewing by users and easy conversion to GUI screens.

### **2.2.9 Date Processing**

#### **2.2.9.1 Manipulation**

Any date processing will accommodate the century.

#### **2.2.9.2 Date display**

Date display will include the century where there is a potential for ambiguity.

### **2.2.10 Type A Extensions**

No Type A extensions to the M Language standard are currently approved for use.

## 2.2.11 Miscellaneous Standards

### 2.2.11.1 Implementation specific functions

Application software must use documented Kernel-supported references to perform all MUMPS operating system specific functions. (Exemptions are the Kernel and VA FileMan.) In addition, %ZISH is to be used for Host File functions.

Application software must use documented VA FileMan standards, such as defaults, editing text, date/time format, spacebar recall, help prompts, deleting stored values, control of logic flow, escapes, and such (see *VA FileMan Users Guide*). Developers are encouraged to use documented FileMan calls to the fullest extent possible (see *VA FileMan Programmers Guide*).

### 2.2.11.2 Data element interpreted as number

Any data element that may be interpreted as a number must contain no more than 15 significant digits.

### 2.2.11.3 Published Entry Points (PEP or API)

- **SET or KILL**

A SET or KILL to another application's data must be done through an API supplied by that application. All other retrieval of information should be done through an API or standard FileMan calls.

- **IO**

If the published entry point has IO, the PEP will have a parameter to allow silent (background) processing.

- **Phase out**

Packages may phase out supported references (as callable from outside the application and documented in the package technical manual) by providing a minimum 18-month notice to the PROGRAMMER and ISC mail groups on IHS Mail.

- **Utilization**

No application will make a call to another application except through documented published entry points or supported FileMan/Kernel calls.

- **API backwards compatibility**

The parameter list or result of a previously published API may not be changed in such a way as to break an application using that call.



#### **2.2.11.4 Character set**

All globals and routines will use only the M character set profile.

#### **2.2.11.5 Assignment of port number for interfaces**

Any RPMS application that requires the use of a port number for interfaces will request said number from the DBA.

### **2.2.12 Conventions**

#### **2.2.12.1 ^UTILITY**

Only the Kernel and VA FileMan and existing Supported References should use ^UTILITY.

#### **2.2.12.2 Deleting tasks**

Tasks should be deleted from Task Manager list by setting the variable ZTREQ equal to "@" just prior to the application QUITting.

#### **2.2.12.3 Routine documentation**

- **Entry Points**

Routine line tags referenced from outside the routine should be documented before, on, or after the line tag. Documentation should include a description of function.

- **Supported references/routines**

All supported references or routines invoked initially from an option or protocol should contain documentation explaining the functionality and any required, passed input and output variables.

#### **2.2.12.4 Device a CRT**

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

#### **2.2.12.5 ^XTMP**

Developers are encouraged to include other descriptive information on the third piece of the 0 node of the XTMP global, such as task description and creator DUZ.

#### **2.2.12.6 Location name**

All packages will display the location name determined by the current DUZ(2) setting above all menus of the package.

### **2.2.12.7 Data Dictionary field numbering conventions**

Developers are encouraged to follow the field number conventions as documented in Section 10.0,

“Appendix G: Data Dictionary

Field Numbering and Data Placement Conventions” and in the XB utilities.

## 3.0 Interface Programming Standards and Conventions

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

### 3.1.1 User Interface Standards for Scroll and Screen Modes

#### 3.1.1.1 Deletion

Deletion of a data value, if permissible, must be initiated by the user entering the “at” symbol, @.

#### 3.1.1.2 READ

Escapes from user input READs, if permissible, must be initiated by the user entering a caret symbol, ^.

#### 3.1.1.3 Help

All prompts requesting user input must provide additional help when the user enters a question mark (?). Any unrecognized or inappropriate response must be handled properly; that is, at minimum in a manner similar to the way VA FileMan handles responses (see the *VA FileMan User's Manual*). Responses to READs that are in no way evaluated by the application are excluded from this requirement.

#### 3.1.1.4 Defaults

In scrolling mode, defaults must be indicated with a double slash (//) or “replace,” indicating that “replace/with” editing is allowed. The null response (pressing only the Enter key) shall select the default.

#### 3.1.1.5 READ timeouts

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

### 3.1.2 User Interface Conventions for Scroll and Screen Modes

#### 3.1.2.1 Terminology

Developers are encouraged to use the following terminology.

Term	Description
EXIT	Exit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. If information has been changed, the application may save the information automatically, or prompt the user to save or discard the information.
QUIT	Like Exit, Quit ends a function or application and removes from the screen all windows and pop-ups associated with that function or application. If information has been changed, the application may discard the information automatically, or prompt the user to save or discard the information.
CANCEL	<p>Cancel allows users to back out of a function or application, one pop-up at a time, until they reach the highest-level window. At that point, another Cancel request has the same effect as an Exit action.</p> <p>When users Cancel a pop-up, the application can decide whether to discard or retain the information in that pop-up, depending on how the application wants to establish the default values the next time the pop-up is displayed.</p> <ul style="list-style-type: none"> <li>• If the information is discarded and the pop-up is later re-displayed, the pop-up contains the default values set by the application.</li> <li>• If the information is retained and the pop-up is later re-displayed, the pop-up contains the same values as it did when the user canceled the pop-up.</li> </ul>
CLOSE	Synonymous with Cancel.

### 3.1.2.2 Key Assignments

Developers are encouraged to use the following key assignments.

Key Assignment	Description
PF1 Key - Gold	May result in different actions based on the next key selected.
PF2 Key - Context-sensitive Help	Provides context-sensitive help about a specific item, field, or window.
PF3 Key - Exit	Exit is defined in Section <b>Error! Reference source not found.</b>
PF4 Key - Backtab	Moves the cursor to the previous entry field. The cursor moves from right-to-left, bottom-to-top.
F10 Key - Menu Bar	Moves the cursor to the menu bar, if one is available, at the top of the window or pop-up currently in focus.
F12 Key - Cancel	Cancel is defined in Section 3.1.2.1.
TAB Key - Tab	Moves the cursor to the next entry field. The cursor moves from left-to-right, top-to-bottom.
PF1, H Key Sequence	Application Help. Provides information about the particular segment of the application being used

### 3.1.2.3 Lock

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

### 3.1.3 User Interface Conventions for GUI Mode

VistA/ RPMS packages should follow the guidelines for GUI applications set forward in the VistA/ RPMS Standards document (see Section 9.0, “Appendix F: IHS RPMS GUI Standards”).

## 3.2 Operating System Programs, Scripts, and Files Standards

### 3.2.1 Purpose

The purpose of this section is to provide guidance in certifying and distributing operating system (OS) programs, scripts, and files that support the RPMS MUMPS applications. Operating systems include UNIX, Windows NT, and Windows 2000.

### 3.2.2 Standards

#### 3.2.2.1 Namespacing

All OS programs, scripts, and files must be namespaced with the assigned namespace (e.g., aibxxxx).

#### 3.2.2.2 Directory

All application-related Shell programs and files shall reside in a unique subdirectory under a variable, parameter-driven directory, depending on the operating system and user input. The subdirectory will be named using the assigned namespace (e.g., /usr/aib or D:\usr\aib).

#### 3.2.2.3 Version

The first line of all OS programs, scripts, and files shall contain the name of the Shell program or file, and the version number of the application. The version number shall be the same as the associated MUMPS application version number.

#### 3.2.2.4 Install shell program

An Install program shall be included in all distributions. This Install program shall perform the following:

- Determine the **Operating System (OS) type** to insure that correct files are loaded during the installation
- **Obtain variables/parameters** based on user input, to drive the actual installation
- Ensure that the **ownership** of the OS programs, scripts, and files are properly set and that they are correct for the operating system.
- Ensure that the **group membership** of the OS programs, scripts, and files are properly set and that they are correct for the operating system.
- Ensure that the **modes** of the OS programs, scripts, and files are properly set.
- Ensure that all OS programs, scripts, and files reside in the **proper subdirectories** based on variable, parameter driven input.

### **3.2.2.5 Operating System commands/utility files**

The RPMS application packages submitted for certification should not include standard OS commands and utility files (e.g., sendto command).

### **3.2.2.6 Patches**

All corrections of errors identified in the production release should be treated similar to corrections in the associated MUMPS application. Each patch should be documented in the OS Shell program, script, or File, beginning with the second line.

### **3.2.2.7 Enhancements/Modifications**

All future enhancements and modifications of the application should be coordinated with the person, who is responsible for developing or maintaining those applications affected.

## **4.0 Appendix A: RPMS Namespace Assignments**

### **4.1 Purpose**

Systems developed for the RPMS are identified by a set of program and file names, and are assigned a range of file numbers with the computer. Standards must be followed in assigning these names and numbers to ensure that they are unique and do not overlap between systems. The purpose of this documentation is to define these standards.

#### **4.1.1 Responsibility**

##### **4.1.1.1 Program/File Name Assignments**

The DBA will maintain a master list of all RPMS name assignments and will assign and/or approve of new names for new systems.

##### **4.1.1.2 File Numbers**

Blocks of file numbers have been allocated to RPMS core systems, the Division of Data Processing Services (DDPS), and to each Area, as indicated in Figure 4-1.

The DBA will be responsible for maintaining and assigning file numbers for RPMS applications; the Area Information System Coordinators (ISCs) will maintain and assign file numbers for local systems.

### **4.2 General Standards**

The general standards to be followed in assigning names are as follows.

#### **4.2.1 Naming Assignments**

##### **4.2.1.1 Namespace position 1**

The namespace of all IHS-developed packages assigned after May 5, 1993 must begin with B.

##### **4.2.1.2 Namespace positions 2-3**

The next two-to-three characters will be the system prefix, and will be used to identify the application system itself (e.g., CHS for Contract Health Services; CA for Cost Accounting).



#### **4.2.1.3 Remaining namespace positions**

The remaining characters will be used to identify names used within the application.

### **4.2.2 Other Standards**

#### **4.2.2.1 All names**

The use of Z as the fourth or fifth character, immediately after the system prefix, designates locks, to distinguish locks from menus and options.

#### **4.2.2.2 XB/ZIB - utilities**

XB/ZIB will be assigned to utilities that have applicability for multiple systems.

Initially, there will be no central assignment of XB/ZIB names. A developer will only need to select a name not already being used. If this should become a problem later, the DBA will coordinate assignments of XB/ZIB names. Routines that only apply to a particular application should carry the name prefix of that application.

#### **4.2.2.3 BZ - local Area routines**

BZ will be reserved for local program development, when there is no immediate plan or intention to distribute the system outside of the Area. ISCs should notify everyone within their Area, who may be writing programs to use BZ as the first two characters of their application name. The Information Systems Coordinator (ISC) should monitor and/or assign names with the BZ format to assure there is no duplication within the Area.

It is recommended that Areas use the third digit to identify the Area in which the system was developed. In this way, you can ultimately share the program with another Area with the assurance that it will not interfere with the local programs developed by the other Area. A list of Area codes to be used for this purpose is shown in Figure 4-1. BZ should be used when there is no intention to share the system outside of the Area. In all other cases, the ISC should contact the DBA for a global name assignment.

#### **4.2.2.4 Universal globals**

The DBA will coordinate the name assignments of globals that have applicability for multiple systems. Globals that only apply to a particular application should carry the namespace of that application.

#### 4.2.2.5 Standard table globals

The first two characters of the IHS Standard Tables will begin with AU. The third character will indicate the type of global, where

- T = Table
- P = Patient File
- A = Administrative File

The fourth character will indicate whether the global supports UCI translation (i.e., can be accessible from multiple UCIs). These codes are:

- T = Translation required
- N = Not required

Thus, a table supporting UCI translation, would have as its first four characters

AUTT \_ \_ \_ \_

Care should be taken not to misuse the AU name. Routines and globals that only apply to a particular system should carry the name prefix of that system.

#### 4.2.2.6 Use of prefix AVA

This prefix is reserved for IHS changes to VA files, such as the New Person file. Its use should be coordinated through the DBA.

Codes for Area Use in Local Program Development	
-----	
A	Alaska
B	Billings
C	Aberdeen
D	Bemidji
H	OIT
L	California
N	Navajo
O	Oklahoma
P	Portland
Q	Albuquerque
S	Tucson
U	Nashville
X	Phoenix

Figure 4-1: Codes for Area Use in Local Program Development

## 4.3 File Numbering Conventions

### 4.3.1 Reserved File Numbers

Blocks of file numbers have been reserved for each Area, the DDPS, and for RPMS core systems as shown in the following figure.

Reserved File Numbers	
File Number Range	Reserved For
-----	-----
0-9999	VA Vista Supported Systems
90000-99999	IHS RPMS Systems (After 5/5/93)
8000000-8999999	SET, DSD, DTM, and DDPS
9000000-9999999	IHS RPMS Systems (Pre-1993)
1000000-1099999	Area 10, Site 00-99, File 000-999- Aberdeen
1100000-1199999	Area 11, Site 00-99, File 000-999- Alaska
1200000-1299999	Area 12, Site 00-99, File 000-999- Albuquerque
1300000-1399999	Area 13, Site 00-99, File 000-999- Bemidji
1400000-1499999	Area 14, Site 00-99, File 000-999- Billings
1500000-1599999	Area 15, Site 00-99, File 000-999- California
1600000-1699999	Area 16, Site 00-99, File 000-999- Nashville
1700000-1799999	Area 17, Site 00-99, File 000-999- Navajo
1800000-1899999	Area 18, Site 00-99, File 000-999- Oklahoma
1900000-1999999	Area 19, Site 00-99, File 000-999- Phoenix
2000000-2099999	Area 20, Site 00-99, File 000-999- Portland
2100000-2199999	Area 21, Site 00-99, File 000-999- Tucson

Figure 4-2: Reserved File Numbers

### 4.3.2 Multiple Files

When applications are developed that involve multiple files, the same integer may be used for all files directly related to the package, and decimal numbers used for members of the group.

For example, a Personnel Package developed in Area 10, Site 05, might have the file number 1005007 and the group of files might be numbered as follows:

Personnel	1005007.1
Title	1005007.2
Wage Scale	1005007.3

### 4.3.3 Common Pointer Files

Common pointer files that are pointed to by various applications (e.g., LOCATION file) should be numbered in a different manner, to indicate these files are not unique to a single application. For example, the common pointer files in the IHS RPMS Systems will be numbered 9999999.n, where n is the next available sequential canonic number.

If you have more than nine files in any single group, you should append two digit numbers if you want them to sort out properly (e.g., 01, 02; 11 sorts lower than 2).

### 4.3.4 Area ISCs Files

The Area Information Systems Coordinators (ISCs) will assign file number ranges to their various sites, keeping in mind that there may be more than one system at a site. A range of numbers should also be retained for the Area office. The allocation of numbers will vary from Area to Area, depending on circumstances; not that one may typically want to assign different site numbers to FileMan globals in PRD and PVT UCIs.

### 4.3.5 Locally Developed Application

When a locally developed application is to be incorporated into the IHS Core System, the files will be renumbered with the range 9000000 - 9999999. If one site wants to incorporate another site's application, the receiving site may want to renumber the files, or it may choose to run the application with the original site's file numbers. When each site conforms to these conventions, the transfer of applications from site to site should not be difficult.

### 4.3.6 File Manager File Numbers

File numbers in File Manager must be canonic; that is, must not have leading zeros or trailing zeros following a decimal. This includes sub-files generated by multi-valued fields.

## 5.0 Appendix B: UNIX/Windows File Naming Standards

### 5.1 Purpose

Applications developed for RPMS are distributed in sets of files, the names of which must conform to the specifications of the host operating system. A defined operating system-independent file name format simplifies the activities of staff responsible for manipulation of these files (Area support staff, facility site managers, etc.) and provide a structured paradigm that can be used for automated manipulation of these files by future applications. UNIX/Windows file naming standards are being established for this purpose.

#### 5.1.1 Distributed Applications

This standard applies only to files of distributed applications. It is not necessarily binding upon filenames for test and verification copies of an application, due to their inordinately long version numbers (see Section 6.0, “Appendix C: Version Numbering for RPMS Systems”).

### 5.2 Standards for RPMS Application Distribution Files

#### 5.2.1 Positions 1-4

Positions 1-4 are reserved for the namespace of the RPMS application. If the application namespace is less than four characters, the remainder of the character positions must be padded with underline ( \_ ) character(s).

##### 5.2.1.1 Namespace case

Namespace and alphabetic codes use lower case characters for distribution filenames.

#### 5.2.2 Positions 5-6

Positions 5-6 are reserved for the major portion of the RPMS application version number (the portion preceding the decimal point). If less than two digits, this value should be right justified and padded with one or more zero (0) digits.

#### 5.2.3 Positions 7-8

Positions 7-8 are reserved for the minor portion of the RPMS application version number (the portion following the decimal point). Position 8 will always be a zero (0), except when annotating a .pdf file as outlined elsewhere.

### 5.2.4 Position 9

Position 9 is a period or decimal point (.).

### 5.2.5 Position 10

Position 10 is reserved for one of the following alphabetic codes, which indicate the type of distribution file.

Code	Description
g	Distribution Globals
u	Unix Files. Archived Unix Files
k	A file containing a KIDS transport distribution
z	Zipped/ Windows-related file

For example, apch0190.k indicates a production UCI KIDS distribution for Version 1.9 of the PCC Health Summary package.

### 5.2.6 Position 11

Position 11 is a flexible field. Possible values for this field are:

Code	Description
m	Routines or globals to be installed in MGR UCI
s	Required if using "u" in position 10 to denote Unix scripts/files

For example, xu\_0710.rm indicates VA Kernel Version 7.1 routines to be installed in the manager UCI.

#### 5.2.6.1 Multiple Files

A single digit, numeric value used to denote one of multiple files to be installed in a production UCI.

For example, ade\_0520.g1 would denote the first set of globals to be considered in an installation of Version 5.20 of the IHS Dental package.

## 5.2.7 Position 12

Position 12 is an optional numeric field. Possible values are as follows.

### 5.2.7.1 Multiple MGR/script files

A single digit value used to denote one of multiple files to be installed in the manager UCI or multiple UNIX script files.

For example, xu\_0710.gm2 would denote the second set of manager UCI globals to be considered in an installation of Version 7.1 of the VA Kernel.

## 5.3 Standards for RPMS Application Patch Files

### 5.3.1 Position 10

Position 10 is reserved for a numeric. The numeric is to be the “tens” position of a two-digit patch number. If there is no “tens” digit, a zero is used to pad this space.

For example, xu\_\_0710.10k would denote routines file for patch number 10 of Kernel Version 7.1; xu\_\_0710.02k would denote routines file for patch number 2 of Kernel Version 7.

### 5.3.2 Position 11

Position 11 is reserved for a numeric. The numeric is the “ones” position of a two-digit patch number.

For example, xu\_\_0710.01k would denote a routines file for patch number 1 of Kernel Version 7.1.

### 5.3.3 Position 12

Position 12 is reserved for one of the following:

Code	Description
b	Patch Globals
n	Patch Notes
k	File containing a KIDS transport

For example, bw\_\_0200.01n would denote Notes for patch 2 of Women’s Health Version 2.0.

## 5.4 Standards for Host Operating Shell Programs

Shell programs shall be in the form as outlined the previous sections through Position 9. Position 10-16 will contain “install.” For example,

aib\_0300.install

## 5.5 Standards for Documentation Files

### 5.5.1 Position 8

Character position 8 in a documentation file will contain one of the following codes.

Code	Description
u	User Manual
t	Technical Manual
s	Security Manual
r	Readme File
i	Installation Guide
n	Release Notes
o	Other

### 5.5.2 Positions 10-12

Character positions 10-12 will contain “pdf” to designate that the manual was prepared in PDF format. For example,

bw\_\_020u.pdf - User Manual in PDF format for the Women’s Health Version 2.0

bw\_\_020t.pdf - Technical Manual in PDF format for the Women’s Health Version 2.0

#### 5.5.2.1 Documentation Distribution File

All documentation files will be zipped prior to final distribution. The zipped file will contain those files as outlined in the previous sections. For example, bw\_\_0200.zip will contain the User, Technical, and Readme files for Women’s Health Version 2.0.



## 5.6 Standard for Final Distribution File

The final distribution file will be a compressed tar file, which will be named using the general standards for Positions 1-4 and position 9 as outlined in the previous sections. The file will contain all the files necessary for the application, that is., routines, globals, notes, Readme, all documentation files, and any other files specified.

For example, bw\_\_0200.tar.gz contains the following files:

- bw\_\_0200.k - KIDS transport file
- bw\_\_0200.g - Globals
- bw\_\_0200.zip - Archived files (Windows)
- bw\_\_0200.us - Archived Host Operating System Files
- bw\_\_0200.install - Host Operating System Installation Program Files

## **6.0 Appendix C: Version Numbering for RPMS Systems**

### **6.1 Purpose**

Applications developed for RPMS go through three primary phases of evolution:

- (1) Testing (both Alpha and Beta)
- (2) Verification
- (3) Distribution

A single version of an application can go through multiple iterations of the first two phases before being distributed; with each iteration differing from the previous one. This standard establishes a standard format whereby multiple iterations of an application can be easily managed by developers, testers, and verifiers.

### **6.2 Definitions**

#### **6.2.1 Distribution Version**

The version number assigned to the application when released for IHS-wide installation.

#### **6.2.2 Target Version**

The intended distribution version number assigned to the application, while going through the testing and verification phases.

#### **6.2.3 Iteration**

A single set of files containing all routines, globals, and notes necessary to install the application.

#### **6.2.4 Sequence Number**

The number identifying the current iteration of the application in either testing or verification phases. Testing sequence numbers and verification sequence numbers are not consecutive.

## 6.2.5 Format of UNIX file name

### 6.2.5.1 Testing

Versions of an application submitted for alpha or beta testing will be signified by the lowercase letter “t” immediately following the target version. Immediately following the “t” will be the sequence number of the test version. There is no distinction between alpha and beta test iterations.

For example, bw\_\_0350.t3k would identify the third test iteration of version 3.5 KIDS file of Women’s Health.

### 6.2.5.2 Distribution

Applications that have passed verification will assume the target version of the last verification iteration.

For example, if bw\_\_0350.t3k passes verification, the application will be distributed as version 3.5 and will be annotated in the Unix file as bw\_\_0350.k.

## 6.2.6 Format of M Routines

### 6.2.6.1 Second routine line

The second line of all package M routines will reflect the test iterations.

For example, ABCTest; IHS/ABDEV/MMM,

```
11/4/95; ;1.0t3;Test;*0*;11/4/95
```

### 6.2.6.2 Package file

The Package File will reflect the test iterations in the current version field. When the package passes verification, the Package File will be cleaned to include only the final distributed version number.

## 7.0 Appendix D: Standards for Submission of Data to NPIRS

### 7.1 Overview

Many applications being developed to run on facility computers include a requirement to generate and transmit data into an IHS-wide reporting system maintained at the National Patient Information and Reports System (NPIRS), in Albuquerque, New Mexico. Examples are the PCC Data Entry System, Patient Registration, Dental Data System, and the Admission, Discharge and Transfer (ADT) System.

For these systems, the data flows from the facility to the Area, where data is consolidated for all facilities for each system, and then forwarded periodically via the IHS wide-area network for each system to NPIRS. NPIRS does not accept data directly from individual facilities. Standards and procedures have been developed to facilitate this transfer, and are described in this document.

For more information, go to the IHS National Data Warehouse (NDW) web site:

<http://www.ihs.gov/CIO/DataQuality/warehouse/>

### 7.2 Facility Procedure

#### 7.2.1 Creation of Transaction Global

A programmer or analyst developing a system with IHS-wide reporting requirements will need to determine the criteria by which data will be selected from the facility database for transmission to the Area/NPIRS.

##### 7.2.1.1 Process options

There are a number of ways this can be done, including:

- Selection based on the transaction encounter date
- Selection based on the facility posting date
- A special flag to indicate whether data has been transmitted
- Creation of a special global, identifying records that have been created or modified since the last transmittal

However the data is identified, a program will be required to extract the data from the database and generate a global that can be written to the host operating system (or transmitted directly) to the Area.

### 7.2.1.2 Global creation standards

The standards to be used in creating this global are as follows.

#### Global Name

The name of the global will be the 4-digit namespace assigned to the system, such as AAPC, ACHS, BCHR; concatenated with the word "DATA." If the namespace has less than 4 digits assigned, characters should be added to fill out the four spaces. Examples of these names are:

- ^AGTXDATA - Patient Registration
- ^AAPCDATA - PCC Data Entry
- ^ADENDATA - Dental Data System

#### 0 Node

The global will have a 0 node, followed by a series of data records subscripted from 1 to n for that transmission. For example:

- ^AGTXDATA(0)=
- ^AGTXDATA(1)=
- ^AGTXDATA(2)=

**0 Node format**

<b>Note:</b> Pieces 1-5 and 7 are required.
---

<b>Piece</b>	<b>Description</b>
<b>1. Facility Code</b>	This is the standard IHS 6 digit code identifying the facility.
<b>2. Facility Name</b>	Narrative name of the facility.
<b>3. Date of Run</b>	The date that the global was created in the format YYYYMMDD, where YYYY is the number of years since 1700 (e.g., 1988 would be 288), MM is the month (01-12), DD is the day of the month.
<b>4. Beginning Date</b>	This field is required, but the definition of the field is program specific. For many programs, data will be selected by a range of dates (i.e., posting dates), and this field will be the earliest date in the date range. If this date is not important or is used by the program concerned for data extraction, the date can be the date of the run.
<b>5. Ending Date</b>	See “Beginning Date.” If not important to the data extraction, the date can be the date of the run.
<b>6. Last Record Transmitted</b>	This field is optional, and was intended for use in cases where data was extracted based on some type of sequential number.
<b>7. Number of Records</b>	This is a count of the total number of records contained in the global (not including the 0 node), and is required.
<b>8. Cartridge Number</b>	The number of the cartridge on which the global will be written. This is for facility audit purposes. Not required.
<b>9. Date Transmitted</b>	The date the data file was transmitted. This field is normally not used, since this date and the date of the run are usually the same.

For an example of the global 0 Node format, see Section 7.5.

## Data Node Format

The format of data in each data node is as follows:

Globalname(n)=xxx^data string (fields separated by a ^)

where,

*n* is the next sequential subscript for the transmission,

*xxx* is the transaction type, i.e., RG1, RG2, IR1, etc.,

*data string* is the actual data being transmitted. Each field is separated by the ^ symbol. A piece must exist for each field in the transaction, regardless of whether there is any data for that piece, and all pieces must be in the same sequence as the fixed length transaction required at NPIRS.

Care must be taken to assure that the total length of the data in the node will never exceed 511 characters in length. If this should ever happen, the data needs to be broken down into two record types (e.g., RG1 and RG2).

An example of a global created for a particular application might be as follows:

```

^AGTXDATA(0)=508201^CARL ALBERT
HOSPITAL^2860804^2860701^2860731^^^3^^^
^ATGXDATA(1)=RG1^508201^336^SMITH^JAMIE^H^01^0608908^
^AGTXDATA(2)=RG2^TOAHTY^MARY^^^Y^223098761A^
^AGTXDATA(3)=RG1^508201^947^BEGAY^JOHNATHON^L^01^0214893^
^AGTXDATA(4)=RG2^ADAMS^EVA^^^N^123456789A^
    
```

The AIB Record Consolidation System at the Area will eventually receive the data and process it for the central computer. If there are two record types, as in the previous example, they will be combined into a single record at NPIRS, with the data from the second record (RG2) added onto the end of the data from the first (RG1).

### 7.2.2 Transmission of Global to Area

A general-purpose utility routine XBGSAVE has been developed to read a global as described previously, and copy it to a tape cartridge. This utility requires that the name of the global be placed in the variable XBGZL prior to execution. For example,

```

S XBGL="AGTXDATA:
D ^XBGSAVE
    
```

Other variables can be set to select various IO options. For more information on XBGSAVE, see the *IHS/VA Utilities (XB/ZIB) Technical Manual*.

Execution of this routine can be initiated automatically by incorporating the routine at the end of the program that creates the global, or it can be designed as a separate option to be selected from the program's main menu.

### **For MSM Systems:**

The XBGSAVE routine will create global save format to a UNIX text file.

When the copy operation has been completed, the original global needs to be deleted before additional data is extracted and posted from the facility database. If an operator fails to do this, new data will be merged with the data already sent to the Area, and this will all be re-forwarded to NPIRS on the next transmission. This could cause problems, depending on the system concerned. A programmer may want to consider automatically killing the global after successfully copying to the host operating system.

The process can be done at whatever frequency is agreed upon between the Area and the facility.

## **7.3 Area Procedure**

When facility data files are received at the Area for a particular application, they are merged into a file containing similar data from other facilities with the AIB utility routines.

The AIB consolidate routines will determine the name of the file to be updated from the name on the incoming file (e.g., AGTXDATA). If the file already exists on disk (e.g., AGTXBLOB), the AIB consolidation routines will merge the data from the incoming data file into the global file. If the file does not exist, it will create the file.

At periodic intervals, normally once or twice a month, the Area will create a transmission file for all information in the global files using the AIB routines. This will delete the global files on the Area disk, and the cycle will start over when the next data file is received from a facility.

These convert programs refer to a table that identifies the fixed length transaction to be created from the incoming data records. The pieces in the data string must be in the same sequence as the transaction to be created. The table identifies the column in which each field starts, and the length of the field. For example,

```
;1;3; RECORD TYPE (starts in col.1;3 characters long)
;4;6; IHS FACILITY CODE (starts in col.4;6 characters long)
;10;2; TYPE OF ACTIVITY
;12;5; REFERRAL CODE
```

If an incoming field is longer than the field in the fixed length transaction, it will be truncated.

If more than one record type is contained in the incoming global, for example., RG1 and RG2, the convert program will combine the two records into a single transaction by adding the data from record two (RG2) onto the end of record one (RG1).



The fixed length transactions will be written out to a transaction tape at NPIRS or stored in a transaction file, and processed on the next update of the application.

Any programmer/analyst designing a system with reporting requirements to NPIRS will need to notify the SET with as much lead-time as possible, to allow the convert program to be developed by SET staff.

Each Area will be required to transmit monthly data sets to the NPIRS computer. Users can submit their data using the Area Data Consolidation System (AIB).

## 7.4 NPIRS Procedure

The process for updating the master database will revolve around a program called “incoming” that begins by identifying file types and processing order of files received, updating each subsystem in the master database, activating a report generator, and electronically informing the submitting area of the status of its data.

### 7.4.1 IHS Subsystems

The master database is comprised of several systems that include Patient Registration (GTX), Patient Eligibility (ELG), Health Record Add (GHA), Inpatient Care (APC), Outpatient Care (INP), Inpatient and Outpatient Contract Health Services (CHS), and Patient Merge (GDM). The systems are listed in the order in which they are processed.

#### 7.4.1.1 Patient Registration (GTX)

This subsystem is the first to be processed since it adds patient records that are updated by the rest of the systems listed. The master database is searched for an existing facility code and health record number. If an entry is found, the health record is scanned for the associated patient record link. If no link is found, this indicates that this record was inserted by the health record add (GHA) subsystem and needs to be updated with the rest of the patient’s personal information. If an entry is not found, the patient and associated personal information is inserted.

#### 7.4.1.2 Patient Eligibility (ELG)

This subsystem is the second to be processed due to the remaining subsystems requiring updated eligibility information. This system updates an existing patient’s eligibility record or creates a new eligibility record, depending on whether a starting or ending data is included in the incoming record.

#### **7.4.1.3 Health Record Add (ELG)**

This subsystem is the third to be processed because the remaining subsystems update records according to facility code and health record number. This system inserts a new record into the chart table with only a health record number and an associated facility code. A flag is set in the new chart record to signal the GTX subsystem to connect this health record with a person. If the associated patient record exists, the GTX system updates the patient record with GTX information and the flag is removed.

#### **7.4.1.4 Ambulatory Patient Care (APC)**

This subsystem, which inserts an outpatient record, is the fourth to be processed.

#### **7.4.1.5 Inpatient Care (INP)**

This subsystem, which inserts an inpatient record, is the fifth to be processed.

#### **7.4.1.6 Contract Health Services (CHS)**

This subsystem, which inserts a Contract Health Services inpatient or outpatient record, is the sixth to be processed. The source of this information comes not only from the Areas but also from Blue Cross/Blue Shield (BCBS).

#### **7.4.1.7 Patient Merge (GDM)**

Formerly Delete/Merge, this subsystem is the last to be processed. This subsystem closes out old health record numbers by setting an ending date in the record; and creating a new health record using the updated information provided, and populating the rest of the new record with the most recent information available.

### **7.4.2 Post Updating**

To close out a month's processing, a table containing each Area's submission information is updated with the status of each system's results. If an Area has not submitted its data, the table will contain NULL values that will flag the operator and send a message to the Area contact and the appropriate NPIRS individual(s).

A log file is also created for every file processed and resides in the same directory as the processed file that contains a full account of the processing and a summary of the essential totals. These totals are inserted into the Area's submit table that informs NPIRS that an Area has submitted its data. This process also allows for a previous month's count to be compared to the current month's count. If the current counts are within a preset percentage, an error is generated and sent to the Area contact and NPIRS individuals.

## 7.5 Formats of Globals Created at Facility

```

AGTXDATA(0)=508201^CARL ALBERT^2860804^2860701^2860731^ ^45^ ^2860805
-----
|                                     |Date Mailed
|                                     |Cartridge Number
|                                     |Number of Records
|                                     |Last Record Posted
|                                     |Ending Date
|                                     |Beginning Date
|                                     |Date of Run
|                                     |Facility Name
|                                     |Facility Code (ASUFAC)
|Global Name
-----

```

Figure 7-1: Annotated example of a global 0 node

```

AGTXDATA(1)=XXX^DATA STRING
-----
|                                     |Transaction data, with fields separated by a ^
|                                     |Transaction type
|Global Name
-----

```

Figure 7-2: Global data node format

## 8.0 Appendix E: RPMS Documentation Standards

### 8.1 Purpose

The purpose of these documentation standards is to provide

- Basic documentation structure that can be applied to every software package
- Consistency in all documentation
- Criteria by which documentation of a national package can be verified

To meet these documentation standards, all documents must follow and meet the style guidelines outlined here and in the *Standards and Conventions Documentation Style Guide*.

### 8.2 General Standards

#### 8.2.1 Definitions

##### 8.2.1.1 RPMS Package

An RPMS package is RPMS software intended for IHS and Tribal distribution and implementation that is fully supported by the Division of Information Technology (DIT). Each package is considered a component of RPMS and is assigned by the Director, DIT, to a development center for development and maintenance.

##### 8.2.1.2 Non-RPMS Automated Information Systems Package

A software package not developed by an approved development center, and not supported by DIT, but may be for use within IHS.

##### 8.2.1.3 Package Documentation

Package documentation is the information that describes the functions, implementation, use, maintenance, and distribution of RPMS packages.

##### 8.2.1.4 Sensitive Information

Sensitive information is information that requires protection due to the risk and magnitude of loss or harm that could result from inadvertent or deliberate disclosure, alteration, or destruction.

### **8.2.1.5 Adobe Acrobat Reader/Exchange**

Commercial software designed to bring electronic documents to a wide range of users. Cross-platform documents, which are created in Adobe Acrobat Portable Document Format (PDF), are called PDF documents. Acrobat Reader enables Windows, Windows NT, Macintosh, DOS, and UNIX users to review, navigate and print any PDF document.

## **8.2.2 Mandatory Documentation Components**

The four mandatory components of RPMS software documentation are

- 1) Installation Guide and Release Notes
- 2) Technical Manual
- 3) User Manual
- 4) Security Manual

## **8.2.3 Preparation**

All RPMS software documentation will be forwarded to the IHS Documentation group, which will prepare the document files for national release.

Preparation includes ensuring that the source Word file is 508 compliant, and converting the source file to a PDF file, using the Adobe Acrobat software. The PDF file will be distributed as part of the national release, and the file will reside as part of the archived distribution file.

## **8.2.4 Reference**

The RPMS *Standards and Conventions Developers' Handbook* is the resource of policies and procedures for development and certification of RPMS applications. This document includes a developer tools section. This guide describes programming conventions common to all national packages and is to be used as a reference source.

## **8.2.5 Installation Guide and Release Notes**

### **8.2.5.1 Purpose**

The Installation Guide and Release Notes should provide sufficient information about the package and how to install the package without additional assistance from OIT. The Installation Guide and Release Note must contain, at a minimum, the mandatory sections listed below.

## 8.2.6 Contents

### 8.2.6.1 Title Page (Mandatory)

Identify the package by name, version number, and release date.

### 8.2.6.2 Release Notes (Mandatory)

Describe the packages functionality and benefits to the user. Also, describe, if applicable, any modifications and enhancements to the national package software since prior release. This information is needed in advance of loading the software.

### 8.2.6.3 Required Resources (Mandatory)

Describe the resources required for the national package. Include Central Processing Unit (CPU) capacity, disk space, unique devices, and other pertinent resources. Provide a formula for sizing, if applicable.

### 8.2.6.4 Installation Issues (Mandatory)

Provide an instructional guide for installing the software. Describe issues that should be considered prior to initialization and how to prepare for initialization.

### 8.2.6.5 Installation Instructions (Mandatory)

Describe the installation process in logical steps and include a statement recommending where the software should be loaded (e.g., “initially load software into a test account and then finally into the production account”). Note any routines, globals, and such that may be removed from the system after completion of the installation.

For all KIDS installations, require user to run “Verify Checksums in Transport Global” after loading and before installing, to insure the transport file was not corrupted. Also include a list of routines included and their official checksums.

### 8.2.6.6 Sample Install (Mandatory)

Include screen copies of the entire installation process as an example of a successful install.

### 8.2.6.7 Installation Configuration (Mandatory)

Provide guidance and suggestions for system configuration and global placement. List any requirements necessary for successful installation of the package. List any reference material that may be required during the installation process. List items that the installer should produce from the system after installation of the national package.

## 8.2.7 Technical Manual

### 8.2.7.1 Purpose

Technical documentation should provide sufficient information about the software for programmers, Information System Coordinators, and OIT technical personnel to operate and maintain the program applications without additional assistance from the package developer(s). The Technical Manual must contain, at a minimum, the following sections flagged as “Mandatory.”

## 8.2.8 Contents

### 8.2.8.1 Title Page (Mandatory)

Include the name of the national software package, the version number, the preparation date, the name of the development center and the logo “IHS RPMS.”

### 8.2.8.2 Table of Contents (Mandatory)

Provide a table of contents with page references to major chapters and/or sections of the manual.

### 8.2.8.3 Preface (Optional)

Supply a brief statement identifying the document in terms of its purpose, scope, and targeted audience.

### 8.2.8.4 Introduction (Mandatory)

Include an overview that describes the package. The introduction should convey to the reader the major function(s) and purpose(s) of the package, and how the software accomplishes the objective(s).

### 8.2.8.5 Orientation (Optional)

Address package specific notations or directions (e.g., symbols used to indicate terminal dialogues or user responses).

### 8.2.8.6 Implementation and Maintenance (Mandatory)

Provide information to assist the Information System Coordinators (ISCs), OIT personnel, and Professional Specialty Groups (PSGs) in the implementation and maintenance of the national package. This section may include information regarding the entry of required site-specific data; a description of parameters configured to meet the needs of individual sites; sample configurations; and work sheets to assist in determining the parameters to be entered for the site.

#### **8.2.8.7 Package Security**

See Section 8.2.11, “Security Manual.”

#### **8.2.8.8 Routine Descriptions (Mandatory)**

Provide a list of routines with comprehensive descriptions of the function.

#### **8.2.8.9 File List (Mandatory)**

Include a list and brief description of files that come with the package. The description should indicate what data comes with the file and whether that data will overwrite existing data, if applicable (this will normally apply only to VA packages, since including data with a file is against IHS Standards).

#### **8.2.8.10 Cross-references (Mandatory)**

Provide a brief description of all cross-references exported with the package.

#### **8.2.8.11 File Diagram/Flowchart (Optional)**

For packages that include numerous files, provision of a chart representing the relationship among files is highly desirable.

#### **8.2.8.12 Callable Routines (Mandatory)**

List all entry points in the package that can be called by other applications. This list must include the actual entry points, a brief description of the function of these entries, a description of all required variables, and any restrictions on the use of the entry points.

#### **8.2.8.13 External Relations (Mandatory)**

Explain any special relations and agreements between the routines and/or files/fields in this package and the routines and/or files/fields in other packages. List any routines essential to the functions of this package. For example, could an outpatient facility function without programming related to inpatient activity and avoid system failure? Specify the version of VA FileMan, VA Kernel, and other packages required to run the package.

#### **8.2.8.14 Internal Relations (Mandatory)**

Identify, if applicable, any routines, files, or options within this package that cannot function independently of other programs. For example, which menus can stand alone? Does the functioning of a particular option assume that entry/exit logic of another option has already occurred? List such options with their programming SACC approval dates.



#### **8.2.8.15 How to Generate On-line Documentation (Mandatory)**

Provide the file numbers and/or file number ranges, and namespaces, along with any special templates. Inform the users where to find the Kernel documentation and how to print the data dictionaries and menu diagrams.

#### **8.2.8.16 SAC Requirements/Exemptions (Mandatory)**

Provide a listing of any items required by the SAC to appear in the Technical Manual. All exemptions granted by the SACC must be noted, specifying the date of the exemption and the actual exemption.

#### **8.2.8.17 Glossary (Mandatory)**

Provide a glossary of terms that relate to the specific national package.

#### **8.2.8.18 Index (Optional)**

Provide a package-specific index.

### **8.2.9 User Manual**

#### **8.2.9.1 Purpose**

The User Manual is a document designed to be helpful to the user. This document will provide sufficient information for users to operate the national software package competently. Variability in the content is accepted, but at minimum, should contain a reference to all of the package functions and menu options. The User Manual must contain, at a minimum, the following sections flagged as “Mandatory.”

#### **8.2.10 Contents**

##### **8.2.10.1 Title Page (Mandatory)**

Include the name of the software package, the version number, the preparation date, the name of the Development Center, and the logo “IHS RPMS.”

##### **8.2.10.2 Table of Contents (Mandatory)**

Provide a table of contents with page references to chapters and/or sections of the manual.

##### **8.2.10.3 Preface (Optional)**

Supply a brief statement identifying the document in terms of its purpose, scope, and targeted audience.

#### **8.2.10.4 Introduction (Mandatory)**

Provide an overview that sets forth a description of the software package. Note related RPMS and/or VA manuals and other reference materials for a modular manual and the purpose for the individual module. Distinguish the major topics and issues within the package. The introduction should convey to the reader the major function(s) and purpose(s) of the package, and how the software accomplishes the objective(s).

#### **8.2.10.5 Navigation (Optional)**

Address any package-specific notations or directions (e.g., symbols used to indicate terminal dialogues or user responses).

#### **8.2.10.6 Package Management (Mandatory)**

Address unique legal requirements pertaining to the package and necessary security measures to protect the integrity of the package and its data (e.g., a package may use an electronic signature code or data that may not be changed because it is supplied by another agency).

If this application generates security or event logs, describe how and when those logs are to be reviewed with suggestions for actions to be taken. If applicable, describe how audits incorporated into this application are to be reviewed and documented.

Package management should not be its own section; the User Manual should contain the information for package management in the sections where it pertains.

#### **8.2.10.7 Package Operation (Mandatory)**

Describe what the user needs to know in order to competently operate the package. Package operation should not be its own section; the entire user manual should describe how to use the software package. The information should include how the user can access on-line documentation.

#### **8.2.10.8 Rules of Behavior (Mandatory)**

Address the particular rules of behavior required by users of this application. Copy the current IHS RPMS Rules of Behavior (ROB); then add those specific to this application. Since users will not have access to the RPMS Rules of Behavior, they must be duplicated here. All developers will have access to the current ROB manual for reference. Include all rules regarding the Privacy Act, HIPAA, and related laws as they pertain to the data in this application. The location of this information may be either the last section before appendices or an appendix.

#### **8.2.10.9 Glossary (Mandatory)**

Provide a glossary of terms that relate to the specific package.

### **8.2.10.10 Index (Optional)**

Provide a package-specific index.

## **8.2.11 Security Manual**

### **8.2.11.1 Purpose**

A package Security Manual will be created detailing the specific security controls in place in this national software package. This manual will not be included in any Freedom of Information Act (FOIA) request releases. Distribution of this section is limited to the Information Systems Coordinator (ISC) and OIT personnel. Since certain keys and authorizations must be delegated for proper management of the system, information about these items may be found in the technical and user manuals.

### **8.2.12 Contents**

The following sections outline the format to be used. The developer fills in items in *italics*. Items in quotations (“ ”) are to be added to the manual as is (remove quotes). For more details, developers can reference the *RPMS System Security Plan* (SSP 06-02).

#### **8.2.12.1 Introduction**

“The Security categorization for this application is HIGH in accordance with FIPS 199. Security controls used in this application adhere to those outlined in the *System Security Plan for RPMS* (SSP 06-02), with specific details as follows in this document.”

#### **8.2.12.2 AC - Access Control**

*Describe how access to this application and its data is controlled.*

- 1) *Describe who should have which level of access and how that is enforced.*
- 2) *Include*
  - a) *menu options and which ones are locked*
  - b) *descriptions of who should be allocated security keys which unlock options*
  - c) *file security levels (FileMan access codes)*
  - d) *use of electronic signatures, if applicable*

3) *Detail security measures taken to protect any data transmitted from this application to other*

- a) *RPMS applications*
- b) *applications (COTS, GOTS)*
- c) *entities via transfer protocols*

### **8.2.12.3 AT - Awareness & Training**

*Describe tools available in application including manuals that address security issues to users of this application. Also reference training materials used.*

### **8.2.12.4 AU - Audit & Accountability**

*Describe any auditing capability in this application. Include*

- 1) *Description of auditable events*
- 2) *Content of audit records*
- 3) *Storage capacity & retention issues*
- 4) *How audit failures are handled*
- 5) *Protection of audit data*
- 6) *Audit monitoring, analysis and reporting functions*

### **8.2.12.5 CA - Certification, Accreditation & Security Assessment**

*Describe the methods used to*

- 1) *assess and certify that security controls detailed are implemented correctly*
- 2) *operate as intended*
- 3) *produce the desired outcome, both before accreditation and on an on-going basis*

*Describe how connections to other information systems will be monitored and controlled.*

*Detail who accredited the system and how any deficiencies noted during on-going assessments will trigger a plan of action with milestones to remedy the situation.*

### **8.2.12.6 CM - Configuration Management**

*Describe the methodology to be used to track changes to this application from proposal and justification through testing, accreditation and release. Detail what checks will be performed to determine that security features are functioning properly after the change has been installed and implemented.*

#### **8.2.12.7 CP - Contingency Planning**

*Give guidance to facilities running this application with suggestions on what to include in their Continuity of Operations Plan and Business Recovery Plan as it pertains to security of this application's data. This may include specific data checks to perform, planned paper backups, suggested manual operations, communication plans and checklists for bringing system back to normal operations.*

#### **8.2.12.8 IA - Identification & Authentication**

*Describe, if applicable, any additional controls used by this application to authenticate the identity of a user or device in addition to that used by the RPMS Kernel system.*

#### **8.2.12.9 IR - Incident Response**

*Give guidance to facilities running this application with suggestions on what to include in their plans to handle security incidents. This may include*

- 1. how to obtain audit data,*
- 2. how to trace the series of events and*
- 3. suggested corrective actions.*

#### **8.2.12.10 MA - System Maintenance**

*“Local controls are to be put in place according to IHS policy and the *IHS General User Security Handbook*.”*

#### **8.2.12.11 MP - Media Protection**

*“Local controls are to be put in place according to IHS policy and the *IHS General User Security Handbook*.”*

*If this application has specific security concerns for removable media, please describe them here.*

#### **8.2.12.12 PE - Physical & Environmental Protection**

*“Local controls are to be put in place according to IHS policy and the *IHS General User Security Handbook*.”*

#### **8.2.12.13 PL - Planning & Rules of Behavior**

*“This security manual is the System Security Plan for this application. The Rules of Behavior are to be found in this application's User Manual.”*

*(See Section 8.2.10.8 for details.)*

#### **8.2.12.14 PS - Personnel Security**

“In accordance to the local security plan, controls are in place prior to allowing access to this application.”

#### **8.2.12.15 RA - Risk Assessment**

*Describe the risk and magnitude of harm that could result from*

- 1) *unauthorized access*
- 2) *use*
- 3) *disclosure*
- 4) *disruption*
- 5) *modification*
- 6) *destruction of this applications data and software*

*Describe tools that can be used to monitor vulnerabilities in this application and how risk assessment changes due to changes in the information system or facility.*

#### **8.2.12.16 SA - System & Services Acquisition**

*Identify the Software Development Life Cycle used in this application and describe how security concerns were addressed in each phase.*

*Detail how the confidentiality, integrity, and availability of the information in this application were addressed in each phase.*

*Describe the developer’s configuration and change management process that insured security issues were addressed adequately.*

*Describe the development security testing performed, including tests that show that patients flagged through the Sensitive Tracking module are properly tracked and access restricted.*

*Include software usage restrictions including licensing issues.*

*If applicable to this application, describe security specifications used on acquisition contracts and third party provider contracts.*

#### **8.2.12.17 SC - System & Communications Protection**

*If this is a web or GUI application, describe how the user interface software is partitioned from the data management layer.*

*For data transmissions, describe how the integrity and confidentiality of the data is protected during transmission.*

*If applicable, describe boundary protection mechanisms in place for connections between this application and the Internet or external networks.*

*If this is a publicly available system or uses mobile code technologies, detail the security controls used to protect this application's information.*

#### **8.2.12.18 SI - System & Information Integrity**

*Describe how this application protects itself from unauthorized changes to its software and information.*

*Detail the way security flaws are remedied.*

*Describe the controls in place to insure the accuracy, completeness, and validity of information input through this application (no matter which application's database stores said information).*

*Include details on any archiving and purging capabilities in this application and how the data is protected in each case.*

#### **8.2.13 SAC Developers' Handbook**

The IHS Resource and Patient Management System (RPMS) *Standards and Conventions Developers' Handbook* has been adopted as a reference source for developers of RPMS applications.

#### **8.2.14 Patch Documentation Requirements**

##### **8.2.14.1 Notes**

All patches to certified RPMS software require a notes file outlining system requirements, contents of the distribution, modifications to the software, reason for the patch, list of routines changed and their new checksums, and other information. Developers should follow the format outlined in the RPMS *Standards and Conventions Developer's Handbook*, Section 1.4.6.6.2.

##### **8.2.14.2 Patch User Manual Addendum**

If a patch contains changes for how a user will interface with the package or contains new or altered functionality, a user manual addendum may be distributed with the patch. The addendum should contain enough information so the user can operate the package after the patch is installed. If the patch is cumulative of previous patches, so must the patch addendum of previous addenda.

## 8.3 Documentation Style Standards

All RPMS package documentation must follow and meet the style guidelines in the *OIT Documentation Style Guide*, which is available on the RPMS SAC page of the IHS RPMS internet web site,

<http://www.ihs.gov/Cio/RPMS/index.cfm?module=home&option=index>

All OIT documentation is designed to be viewed onscreen (per the Paperwork Reduction Act of 1996).

The following sections summarize some of the main points related to the documentation style standards.

### 8.3.1 Using the RPMS Templates

A set of RPMS templates has been created, which comply with Section 508 compliancy requirements.

- RPMS Install
- RPMS Technical
- RPMS User
- RPMS Addendum
- RPMS Security

Each of these templates includes a pre-defined Cover Page, TOC setup, headers and footers, page numbering, “boilerplate” information, and a set of valid paragraph and font styles.

#### 8.3.1.1 Page dimensions

All documents use standard Letter, 8.5-inch by 11-inch size paper.

#### 8.3.1.2 Margins

All documents contain 1-inch margins on all sides.

#### 8.3.1.3 Text orientation

The default text orientation is portrait. If necessary, landscape orientation may be used within its own section.



### 8.3.2 Using Styles

The RPMS templates include defined styles, which comply with Section 508 compliancy requirements.

**Note:** The use of styles is *required* to comply with Section 508.

Styles streamline the updating and formatting process of documentation. Use the defined paragraph and character styles to align and format text. When using styles, limit manual adjustments to paragraph line and page breaks. Avoid making new or styles or adjusting existing styles, as such changes compromise the file's 508 compliance status.

### 8.3.3 Cover Page Layout

Element	Description	RPMS Style
Application Name	Name of the RPMS module; for example, Accounts Receivable or Emergency Room System	Title Page
Namespace	Corresponding namespace of the application; for example BAR or AMER	Namespace
Manual Type	<ul style="list-style-type: none"> <li>• Installation Guide and Release Notes</li> <li>• Technical Manual</li> <li>• User Manual</li> <li>• Addendum</li> <li>• Security Manual</li> </ul>	Manual Type
Version	the Release Version number of the application, using the format Version #.#; for example Version 5.0 or Version 1.3  Also, Version #.# Patch #, for example Version 2.5 Patch 15	Version
Month Year	Release date, using the format <i>Month YYYY</i> ; for example, June 2008, February 2009	month_year

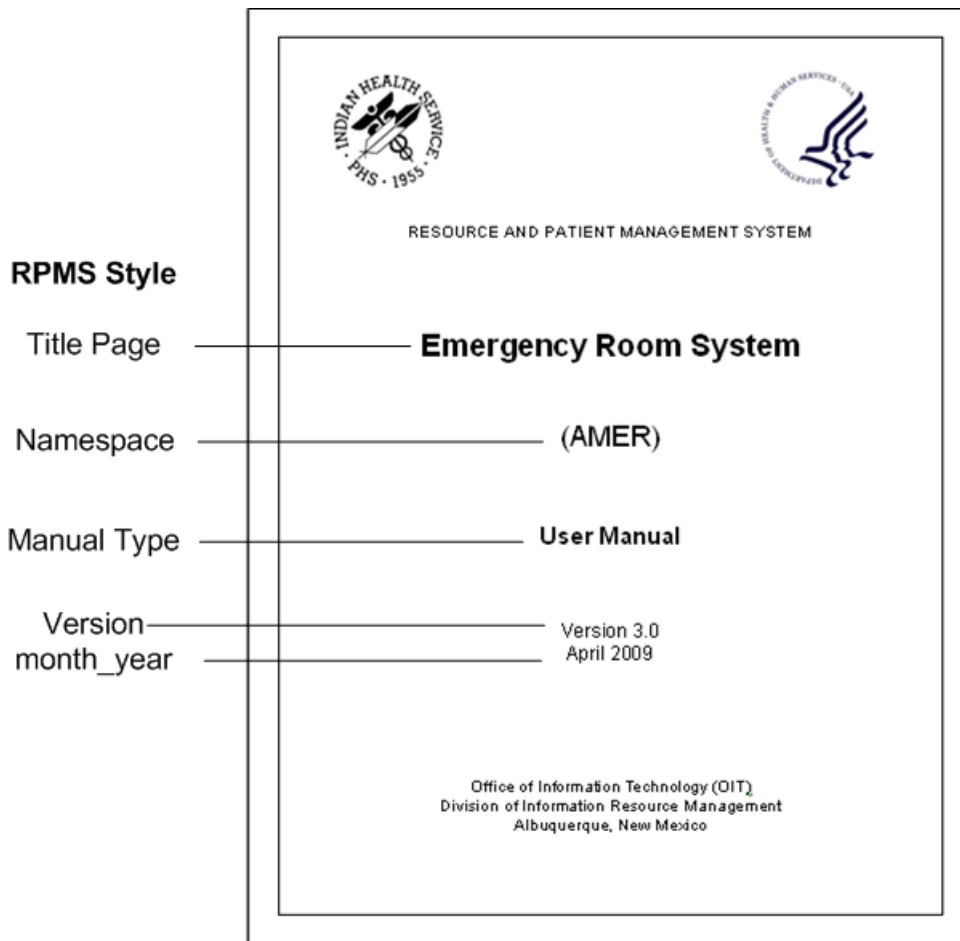


Figure 8-1: Example of an RPMS Cover page, highlighting corresponding RPMS styles

### 8.3.4 Headers and Footers

All documents require headers and footers. However, the Title page does not have a header or footer.

#### 8.3.4.1 Header

The document Header includes the

- Application Name
- Namespace of the application
- Version number

The left side of the header displays the Application Name and Namespace. These fields reference the Title Page and Namespace styles, and are populated automatically.

The right side of the header displays the Version number, and if applicable, the Patch number. This field references the Version style. The format is Version ## or Version ## Patch #.

The header is followed by a line to separate it from the document text; for example,



Emergency Room System (AMER) Version 3.0

---

Figure 8-2: Example of an RPMS document header

### 8.3.4.2 Footer

The document Footer includes

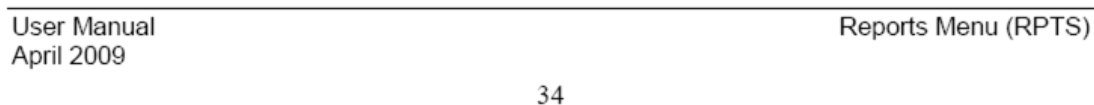
- Manual type
- Release date
- section title
- Page number

The left side of the footer displays the Manual type and release date. These fields reference the Manual Type and month\_year styles, and are populated automatically.

The right side of the footer displays the section title (e.g., Table of Contents, Introduction). This field references the style assignment of the particular section (Heading 1, Contents, Section, etc.) and is populated automatically.

The page number is centered, below the footer information. Those pages between the Title/Cover page and Section 1 use lowercase Roman numerals; all pages, beginning with the first page of Section 1, use Arabic numerals.

The footer is preceded by a line to separate it from the document text; for example,



---

User Manual Reports Menu (RPTS)  
April 2009

34

Figure 8-3: Example of an RPMS document footer

### 8.3.5 Page Numbering

Pages should be numbered in standard format, starting with page 1 at the Introduction. The only exceptions to this numbering system is the Front Matter, which uses lowercase Roman numerals, where

- Title Page is not numbered
- Table of Contents begins with Roman numeral, ii
- Other information following the TOC and before the first section, such as a Version History Table, continue with the next Roman numeral after the last page of the TOC.

The RPMS templates are configured to number the pages correctly.

### 8.3.6 Page Breaks

#### Section page breaks

The RPMS templates are configured with standard section breaks between sections (Cover page, Table of Contents) as much as possible. To add a section to a document,

1. Go to the end of the current section and press Enter.
2. At the location of the cursor, insert a Next page Section Break
3. Then enter the title of the next section and apply the Heading 1 style.

#### Hard page breaks

*DO NOT USE* hard page breaks to keep text together across a natural page line. Instead, check the **Keep lines together** and **Keep with next** paragraph properties, which allow the Word program to determine the best place to break the text.

#### Forcing even and odd start pages

*DO NOT USE* the Section Break (Odd Page), or Section Break (Even Page) options when inserting section breaks.

### 8.3.7 Font

#### Default font and type size

Use Times New Roman 12-point font for standard documentation text (and defined in RPMS template styles).

#### Bold type

Use **bold** type for emphasis, such as a menu option when it is first introduced, a Glossary term the first time it appears in the text, specific user input (see Section 8.3.19).

### 8.3.8 Italic type

Use *italic* type for emphasis, sparingly; for example,

Site parameters must be set *before* performing any DCM functions.

Italic type is used for document titles, for example,

*The Chicago Manual of Style*, 15th edition

*Accounts Receivable (BAR) User Manual*, v1.8

### 8.3.9 Underlined type

**Note:** Underlined type is restricted to hyperlinks.

Do not use underlining to emphasize particular sections of text or set headings apart from the text. Instead, use the template's paragraph and character styles.

### 8.3.10 Headings and Subheadings

Use the RPMS template styles for Headings and subheadings (Heading 1, Heading 2, etc.).

All headings between heading 1 and heading 4 are numbered in legal/outline style (1.0, 1.1, 1.1.1, 1.1.1.1, etc.) and are defined in the RPMS template Heading styles

### 8.3.11 Figure Numbering

Figures should be numbered in the 1-1 format, where the first number corresponds to the Section number and the second number corresponds to the figure number within that section.

For example, the first figure in Section 1.0 would be Figure 1-1, the third figure would be Figure 1-3; the first figure in Section 2.0 would be Figure 2-1. The figure number will appear in the figure caption. For more on figure captions, see Section 8.3.14.1.

### 8.3.12 Item/List Numbering

Items in lists should be numbered *only* if the items must be completed in a particular order or if the numbering is critical to understanding the list or items. In cases where there is no explicit reason to use a numbered list, use a bulleted list instead.

### 8.3.13 Appendix

#### 8.3.13.1 Appendix titles

Appendices should be treated independently and lettered accordingly. All appendices are labeled/ titled in the following format:

Section Number Appendix Letter: Title

Note that a colon (:) separates the Appendix letter and the title; for example,

11.0 Appendix A: List of Names

#### 8.3.13.2 Appendix location

Appendices should be located in order, after the final section and before the Glossary and/or Contact Information sections.

#### 8.3.13.3 Referring to multiple appendices

While many dictionaries consider both Appendices and Appendixes correct terms for referring to more than one appendix, OIT uses Appendices.

### 8.3.14 Figures

RPMS uses following types of figures:

- Computer output (roll and scroll), for example,

```
+++++
|          THIRD PARTY BILLING SYSTEM - VER 2.5          |
+                   Add/Edit Claim Menu                   +
|                   sitename                               |
+++++
User: LASTNAME, FIRSTNAME                                21 JAN 2009 10:35 AM

CG1P  Claim Generator, One Patient
EDCL  Edit Claim Data
LOOP  Claim Editor Loop
NEW   Add New Claim (Manual Entry)
RBCL  Rebuild Items from PCC

Select Add/Edit Claim Menu Option:
```

Figure 8-4: Example of a screen capture of computer output (roll and scroll)

Since “roll and scroll” screen captures are text-based, they can be formatted using one of the computer output styles.

- Screen captures with/without callouts

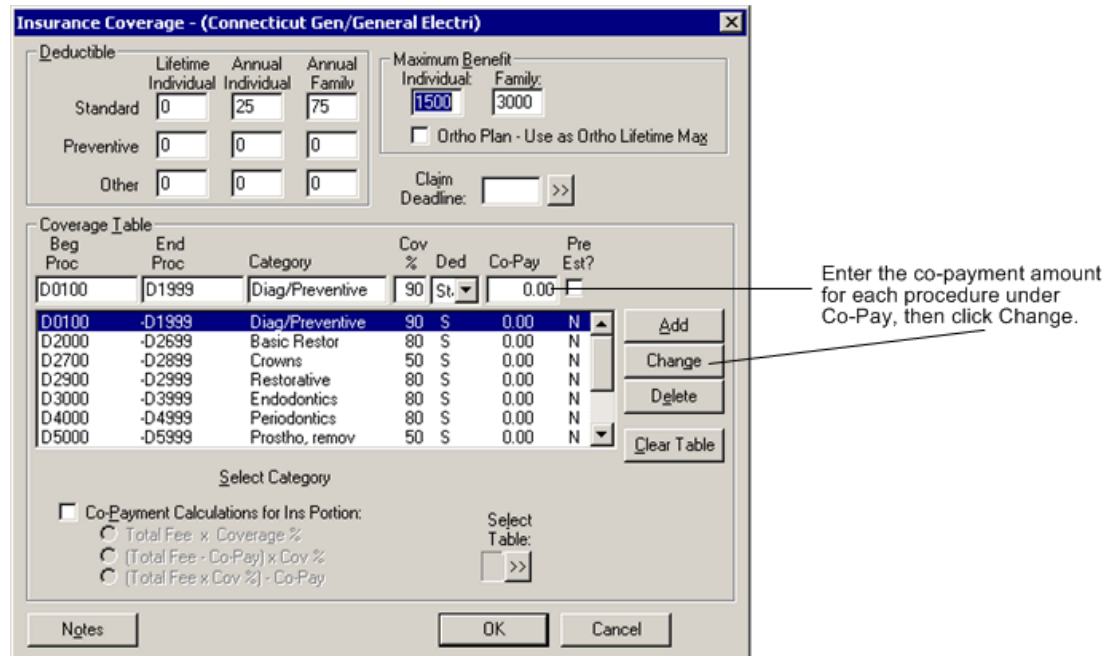


Figure 8-5: Example of a screen capture (dialog box) with callouts

**Note:** *DO NOT USE* Word graphic elements to create callouts, as they cause 508 compliance problems when converting to another format. Instead, use Microsoft Visio or other graphic design software to create the annotated dialog box and save as non-interlaced GIF files. Then insert the GIF picture in the Word document.

- Process flows and similar diagrams created with graphic design elements; for example,

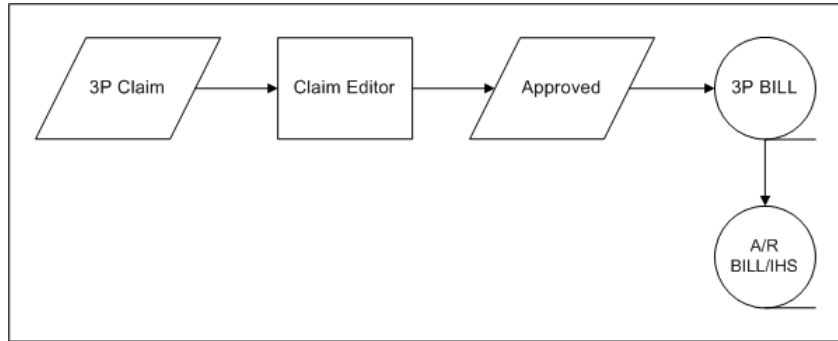


Figure 8-6: Example of a flow chart

**Note:** *DO NOT USE* Word graphic elements, as they cause 508 compliance problems when converting to another format. Instead, use Microsoft Visio or other graphic design software to create such diagrams and save as non-interlaced GIF files. Then insert the GIF picture in the Word document.

#### 8.3.14.1 Placement of figures

Figures should be placed after the first reference to them. Figures should not be placed against each other without text to separate them, unless they are representing consecutive pages of a screen or report and there is no text to place between them.

#### 8.3.14.2 Figure captions

Figure captions, located immediately after the figure, include the figure number followed by a colon (:) and a summary title/description of the figure. If applicable, reference step numbers in parentheses that are covered in the figure. For example:

Figure 3-30: Adding a new patient (steps 2-4a)

Use Word's Insert Caption feature (Insert > Reference > Caption) to create a figure caption.

#### 8.3.14.3 Figure numbering

Figure are numbered, using the #-# format, where the first number corresponds to the Section number and the second number corresponds to the consecutive figure number within that section.

Word's Insert Caption feature (Insert > Reference > Caption) generates figure numbers automatically.



## 8.3.15 Screen Captures

### 8.3.15.1 Screen capture authenticity

Screenshots should appear as close as possible to what the user really sees onscreen. Cropping for the sake of space is acceptable as long as the user can still orient him/herself on the screen.

- Do not make any unnecessary changes to spelling, grammar, etc. in text capture screenshots (roll and scroll).
- Do not use a graphic editor to cleanup or alter a screenshot unless you are altering sensitive information.

Make sure that you review your screenshots before the final product is packaged to make sure that any last minute changes to the program still match your documented screenshots. Callouts and sample dialog between a hypothetical user and the system should not be so subtle that the user expects them to appear on his or her screen.

### 8.3.15.2 Screen captures as figures

A screen capture should be treated as a figure if it

- Represents a complete RPMS procedure or a GUI dialog box or window.
- Is referred to often in the text.

When a screen capture represents only a specific portion of an RPMS procedure or GUI dialog box or window, treating it as a figure is at the writer's discretion.

### 8.3.15.3 Screen captures as alternate content only

Per Section 508 regulations, screenshots/images cannot contain information that is not contained elsewhere in the document. If you use a screenshot to illustrate which values to enter at a series of prompts, you must also include that information textually in that section. This does not apply to text-capture screenshots (roll and scroll), as the information is in textual form, which a reader program can interpret.

### 8.3.15.4 Using default settings

As much as possible, use the default settings for the desktop when you take a screenshot. For GUI screenshots, include the frame of the window. For roll-and-scroll screenshots, do not include the window frame, as many users are using dumb terminals and do not have an awareness of the RPMS system in a windowed environment.

### 8.3.15.5 Adding callouts

*DO NOT USE* Word graphic elements to add callouts or lines, as such elements create 508 compliancy issues. Instead bring the screen capture image into Microsoft Visio or other graphic design software and add the callouts. Then save as a non-interlaced GIF file.

When visually linking callout text with the section of the image it applies to, use straight horizontal and vertical lines when possible. Arrows pointing to the related screenshot are optional. For an example, see Figure 8-5.

### 8.3.15.6 Adding a border

If a border is needed, add one in Microsoft Visio or other graphic design software and save as a non-interlaced GIF file. Then insert the GIF picture in the Word document.

### 8.3.15.7 Adding alternate text

Alternate Text is a Section 508 requirement. All pictures inserted into a Word document from a file *must* have an alt text value. This alt text value must explain to the user what the picture is displaying.

To add alternate text,

1. Select the picture and right click the mouse to display the menu.
2. Click Format Picture to display the dialog box and click the Web tab.
3. Enter alternate text describing the picture; for example,

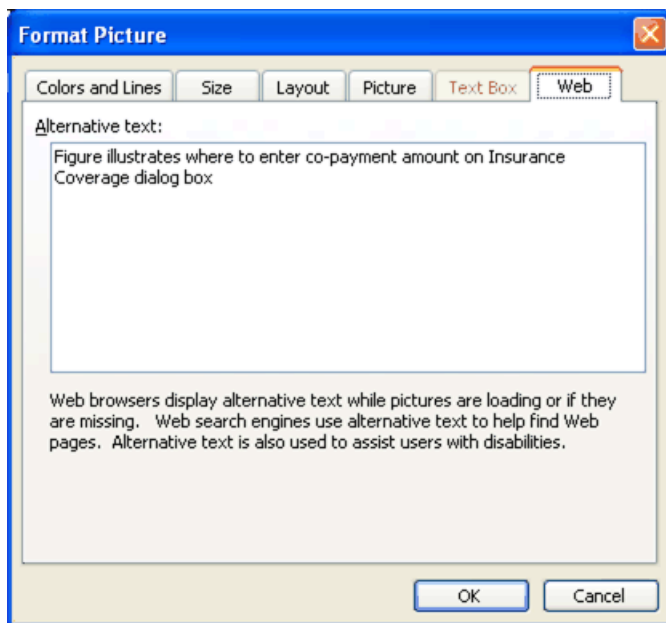


Figure 8-7: Example of alternate text, added to Figure 8-5 screen shot

### 8.3.16 Referring to Package Names

Use initial capitalization when referring to package names, for example, Laboratory, Patient Registration. When you refer to the package for the first time, spell out the full package name and follow it with its abbreviation or namespace in parentheses. After you have introduced a package in this fashion, you may refer to it by its abbreviation, namespace, or synonym throughout the document. Always use the word *package* when you are referring to a package, but do not capitalize *package* (unless the entire phrase is initial capped, such as in headings).

### 8.3.17 Introducing Procedures

To introduce a series of procedure items, use a complete sentence ending with a colon and follow the same grammatical pattern for all introductions in a document. Apply the Subhead style. For example,

**To print the document, follow these steps:**

### 8.3.18 Referring to System Prompts

When referring to RPMS and other system prompts in text, such as instruction or procedure steps,

- Enclose the prompt with double quotes ( “ ” )
- Do not include the ending colon.
- Follow the prompt name with the word “prompt.”

For example, the following RPMS prompt

```
Enter Patient Name:
```

would be referenced in text as

At the “Enter Patient Name” prompt, type the name of the patient and press Enter.

Additionally, consider the following when referring to RPMS prompts:

- If a prompt ends with a question mark, include the question mark within the quotation marks. For example, the following RPMS prompt

```
Continue? Y//
```

would be referenced in text as

At the “Continue?” prompt, press Enter to continue.

- If an RPMS prompt includes a default response, do not include the default response. For example, the following RPMS prompt

```
Select Beginning Date: 12 SEP 2002//
```

would be referenced in text as

At the “Select Beginning Date” prompt, type the beginning date and press Enter.

- If the prompt has any internal punctuation, include it in the prompt name reference. For example, the following RPMS prompt

```
Choose One, Return to Continue, or ^ to Cancel:
```

would be referenced in text as

At the “Choose One, Return to Continue, or ^ to Cancel” prompt, press Enter to display additional options.

### 8.3.19 User Input

Use **bold** font to indicate specific user input; for example,

At the “Main Menu” prompt, type **PTRG** and press Enter.

Use regular font when user input is descriptive; for example,

At the “Patient Name” prompt, type the name of the patient and press Enter.

At the “Letter” prompt, type the number corresponding to the letter you want to print and press Enter.

### 8.3.20 Computer Output

For “roll and scroll” computer output, apply a computer output style, which includes a border and shaded the background. For an example,

```
Select Debt Collection Menu Option: DCP <Enter>

NOTE:  You must be logged into the facility for which you wish to process
       Debt Collection.  You are logged into <Facility-Name>

Continue? Y// <Enter>

Starting Date:
Ending Date:

Select 3P Approval date range for this Debt Collection process...

Select Beginning Date: 010106 <Enter>  (JAN 01, 2006)
Select Ending Date: T <Enter>  (JAN 30, 2007)
Enter the Debt Collection Minimum Bill Balance Amount:  (20-5000): 20//
<Enter>

Start Date: 01/01/2006
End Date: 01/30/2007

$$ Limit: 20.00

Do you want to proceed? N// Y <Enter>
```

Figure 8-8: Example of computer output from the Accounts Receivable (BAR) Debt Collection option

Note that user input is in **bold** font and <Enter> represents pressing the Enter key.

For system-level or programming code examples, apply the computer character font style, which is Courier New, 10 point. For example,

```
I $D(XYZ),XYZ'="" S ^GBL("AC",XYZ)=""
```

### 8.3.21 Placement of Tables

Tables should be placed after the first reference to them. Tables should not be placed adjacent to each other without text to separate them.

### 8.3.22 Referring to Acronyms

On first appearance, spell out the acronym and include the acronym in parentheses ( ), and then define it, if necessary; for example,

Indian Health Service (IHS)

However, if the acronym is the more standard usage, include the full text in the parenthesis and use the acronym; for example, IBM (International Business Machines).

After the first usage, use only the acronym without the parenthetical explanation. All acronyms should appear in the glossary with the full phrase defined (minimum) and the definition, if it will be helpful to the users.

### 8.3.23 Capitalization

In general, you should capitalize

- All letters in acronyms
- Initial letter of the first word in a list
- Initial letter of the first word in a callout
- Initial letter of a key name
- Initial letter of a sentence. Avoid starting a sentence with a command name or application name, if it has a lowercase initial letter.

#### 8.3.23.1 Capitalization of key names

Match capitalization on the keyboard. Do not capitalize the word *key* when it follows the name of a key, for example, F1 key.

#### 8.3.23.2 Capitalization of field names

You can either match the onscreen capitalization of field names or just capitalize the first letter of each word, as long as you maintain a consistent approach to the entire document. If a lot of fields are all heavily capitalized, following the onscreen capitalization is highly discouraged, as too many capital letters make a document difficult to read.

#### 8.3.23.3 Capitalization of file names

Use initial caps for all file names, but do not capitalize the word *file* when it follows the file name. The word *file* must follow all references to file names to distinguish file names from option names.

#### 8.3.23.4 Capitalization of option names

Use initial caps for all option names, but do not capitalize the word *option* when it follows the option name. The word *option* must follow all references to option names to distinguish option names from file names.

#### 8.3.24 Punctuation as Command, Navigation Response

Many of the RPMS packages use punctuation marks as command and navigational responses. When a punctuation mark is used in this context, you must spell out its name and follow with its symbol within parentheses. For example,

At the “Name” prompt, type a question mark (?) to see a list of options.

At the “Device” prompt, type a caret (^).

#### 8.3.25 Keys

Keys that the user is supposed to press are referred to by an article before the key name and the word *key* after the key name; for example, “the F1 key.”

Use a hyphen to connect keystrokes that are performed simultaneously. If necessary, explain this notation in the Navigation section of the User Manual.

**Example:** Ctrl-D [press the Ctrl key and the D key simultaneously]

Use a comma and space to separate keystrokes that are pressed in sequence.

**Example:** Press Ctrl-Shift, N to display the New window. [Press the Ctrl key and the Shift key at the same time, then release and press the N key.]

#### 8.3.26 Enter/Return Key

When instructing the user to press the Enter or Return key, use the phrase, “Press the Enter key” or “Press Enter.” You may use either, but choose one and use it consistently throughout your documentation.

Notice that the phrase uses the word “press.” Capitalize the key name as it appears on the keyboard. For example,

At the system prompt, type your name and press the Enter key.

At the “Main Menu” prompt, type **PTRG** and press Enter.

Do not use any additional punctuation or formatting to make the Enter/Return key reference stand out.

Exception: When making notations in a recreated computer output example, the notation **<Enter>** is an acceptable way to indicate where the Enter key was pressed. Notice that the word Enter is bolded.

### 8.3.27 Caret (^)

Within RPMS packages, the caret has navigational properties and is referred to frequently in the manuals. You can refer to this punctuation mark as the caret or “up-hat,” but never as the up-arrow. When referring to the caret, always include its symbol in parentheses (^).

### 8.3.28 General Word Usage Guidelines

- Define new terms that are not listed in a regular dictionary the first time they appear in the text. You should include these terms in the glossary as well.
- Do not use slang or undefined jargon.
- Do not use terms that have several different meanings.
- Use your defined terms consistently throughout your documentation. Do not use synonyms.
- Ensure that your spelling is correct.
- Avoid general adjectives that can be misinterpreted. For example, “the user-friendly Windows desktop” could mean that either the desktop is user-friendly or that Windows is user-friendly.

#### 8.3.28.1 Allow

Avoid using the word “allow” when referring to software features. Allow implies permission. Either rewrite the sentence or use the word “enable.”

#### 8.3.28.2 Appear (verb)

Use the phrase “is displayed” instead of the word “appears.”

#### 8.3.28.3 Blank

When no information is present in a field, refer to the field as blank. Do not use the word “empty” to refer to a field.

#### 8.3.28.4 Choose/Select

Avoid using the word “choose.” The word “select” is usually more explicit.

#### 8.3.28.5 Enter (verb)

When you are instructing the user to type specific information into specific fields, use the word “type” instead. For example,

At the “Select Patient Name” prompt, type the patient’s full name.

Use “enter” when referring to higher-level or general function. For example,

Use the Patient Registration Menu option to enter patient data into the system.



**8.3.28.6 Field**

A field is the part of the interface where users can type information to add data to or search a database. It is also referred to as a prompt.

**8.3.28.7 Hit (verb)**

Use “press” instead of “hit” (or “strike”); for example,

Press the F1 key.

Press Enter

**8.3.28.8 Pick (verb)**

Use “select” for selecting things from menus; for example,

Select the ABC report option from the Reports menu.

**8.3.28.9 Press (verb)**

Use “press” for keyboard actions; for example,

Press Enter to continue.

Use “click” for GUI actions; for example,

Click the Apply button.

**8.3.28.10 Type**

Use “type” instead of “enter”; for example,

Type your name to start the program.

Do not use “type in” or any other variation of the word “type.”

## 9.0 Appendix F: IHS RPMS GUI Standards

### 9.1 Purpose

The purpose of these Graphical User Interface (GUI) standards is to promote visual and functional consistency in RPMS GUI software. This document defines user interface and programming standards to follow when programming in a GUI environment.

### 9.2 General Specifications and Guidelines

All IHS RPMS GUI software designed to execute in the Microsoft Windows operating environment will comply with the specifications and guidelines defined in *Windows User Experience*, and with the IHS-specific extensions embodied in this document.

*Windows User Experience* may be purchased in book form (Microsoft Press, 2004: ISBN 0735605661), or viewed online on the Microsoft Developer's Network web site at

<http://msdn.microsoft.com/en-us/library/aa511258.aspx>

### 9.3 IHS Programming Naming Conventions

All IHS applications submitted for verification must contain all source code associated with the application. This section defines how a developer should name various elements, modules, forms, variables, routines, classes, web services, etc. to allow reviewers to follow the logic of the application.

All IHS applications submitted will include:

- A detailed statement indicating the Development Tools (IDE) used to create the application. The statement will include the version, manufacturer, and development language.
- A detailed statement indicating the dependencies required to run the application. The statement will include the each dependency, its version, manufacturer, and description.
- A detailed statement indicating the Windows OS required to run the application. The statement will include all versions supported and minimum operational requirements.

Use consistent and standard comment formats.

Additional requirements include:

- Explicit dimensioning of data types. Strong Naming Encryption is required for .Net assemblies.
- Support for error handling techniques to prevent unhandled exception messages to occur.

The following notations are strongly recommended:

- Use of Hungarian prefix notation to designate variable type and scope. (e.g., "strName").
- Use of the Microsoft "camel-case" notation (e.g., "PatientName"). Use "Object-Verb" notation.

Use of commercially purchased controls requires SACC approval through the Request for Exemption process. The request must also contain the manufacturer's licensing and deployment agreements.

Applications should support user-defined windows and themes. Applications should not override user selection of fonts, colors, size, and resolution. Applications should adhere to the 508 Section of the Rehabilitation Act.

An About Form must be present in all GUI applications and will show the current version of the product along with the current version of all dependent products. Whenever a dependent product is upgraded and used with the primary product, both the dependent product version and the primary product version must be incremented.

All IHS GUI applications will have a robust on-line HELP context.

## 9.4 File Systems

IHS GUI applications

- Install themselves into an application subdirectory descendant from the system-defined "Program Files" directory. The first subdirectory descendant below the Program Files directory is "Indian Health Service." All applications certified by IHS will be in descendant subdirectories of "Indian Health Service."
- Make use of locations (and localized names) of standard system folders. Any application that registers a COM Object will register the object in the application subdirectory. Registering a COM object in any other Windows' directory requires exemption from the SACC.
- Make use of locations (and localized names) of temporary storage directories based on the environmental variables defined by the system (e.g., the temp folder may be C:\TMP).
- That create permanent files such as INI files, store such files in folders descendant from their application folder.

- That create files containing sensitive data must hash or encrypt such data, if the data is to remain on the users system. IHS GUI applications will remove any temporary stored sensitive data upon closing the application session. IHS GUI applications will delete any files containing sensitive data that may exist from an abnormal shutdown of the previous application session.
- That include files containing sensitive data must adhere to the Security requirements in Section 9.7.

## 9.5 Installation

IHS GUI applications will supply an installation package which supports the Windows Installer and which adheres to the *Windows User Experience* guidelines for installation and un-installation. The installation package will be namespaced according to the officially assigned application namespace. All temporary and permanent data storage files will be removed during un-installation.

## 9.6 Namespacing

### 9.6.1 COM Objects

IHS applications that register COM objects on the system must show the Product Name, Internal Name, and Version when the object's properties are displayed.

The Product Name contains the product's descriptive name.

The Internal **Name** contains the application specific namespace assigned by the DBA.

The Version must contain both major and minor numbers and must increment with each release. A modification to a RPMS Remote Procedure call or RPMS routines related to the object requires that a new object be created with the matching version number associated with the RPMS change.

The Technical Manual must identify all objects submitted to verification, including dependencies. Identity must list the product name, the internal name, and the version number.

### 9.6.2 .Net Objects

.NET Assemblies developed for IHS using the .NET framework will be created under the IndianHealthService namespace.

.NET created objects must contain property information as described to COM objects described in section 9.6.1.

### 9.6.3 RPMS Kids Files

Each new or patched object within an application will result in a unique Build File entry. The major part of the release version must coincide with the major release version of the client package distribution. (NN.NN - Major and minor version number of the namespace associated with the Windows application/object - This is the value that is placed in the Current Version field of the Package File.) Each time OIT Verification releases a distribution associated with the package namespace and the major part of the number changes (NN before the dot) all dependent client files of the namespace will be released with the same proper increment.

### 9.6.4 Client Naming Conventions

Client side software generally is identified by the extension of the file name. EXE identifies the application as an executable program. DLL identifies the application as a dynamic link library. OCX identifies the application as an active X control. BAT identifies the application as a batch run type program.

When the properties of the file are viewed, the version number must be clearly identified. Newer or patched releases of this file must be identified with a new name or an incremented version number. The major part of the release version must coincide with the major release version of the kids package distribution. (NN.NN - Major and minor version number of the namespace associated with the Windows application/object. This is the value that is placed in the Current Version field of the Package File.) Each time OIT Verification releases a distribution associated with the package namespace and the major part of the number changes (NN before the dot) all dependent client files of the namespace will be released with the proper increment.

## 9.7 Security

### 9.7.1 Data Storage on Client

Data such as default settings and previous used values can be stored on the client however no sensitive data or data associated with maintaining security can be stored on a client without SACC approval through the Request for Exemption process. IHS GUI applications that create files containing sensitive data must hash or encrypt such data if the data is to remain on the users system. IHS GUI applications will remove any temporary stored sensitive data upon closing the application session. IHS GUI applications must check and delete any files containing sensitive data that may exist from an abnormal shutdown of the previous application session.

If data is to be stored in a registry setting, then specific settings and directions to support trust relationships to the registry is required in the technical documentation.

If data is to be stored in a file, then specific directory settings, ownership details, and directions to support trust relationships to the file and directory is required in the technical documentation.

### 9.7.2 Broker Interface

Any RPMS database access must use a SAC approved broker for communication. Access by any other means requires SACC approval through the Request for Exemption process.

The GUI application must use the approved Kernel access controls when establishing a connection. The access/verify data will be hashed or encrypted.

GUI applications must use published Remote Procedure calls for security access, patient context and visit context. Access by any other means requires SACC approval through the Request for Exemption process.

The broker listener interface port used is subject to IHS DBA assignment.

## 10.0 Appendix G: Data Dictionary

### 10.1 Field Numbering and Data Placement Conventions

The following conventions for numbering fields and placing data in pieces are extracted from a mail message dated 25 Feb 88, and are considered to be the conventions to be followed when assigning field numbers for FileMan files.

1. There is a direct correlation between the field number and the node and piece, and for multiples, between the field number and the sub-file number.
2. Fields beginning with a "." are all .01-.n and are in the 0 node. Where possible, files only have a 0 node. This reduces the number of disc accesses required. A field number must be canonic; therefore, there is no .10 field. It goes from .09 to .11. That means piece 10 will always be NULL.
3. Where the entire entry cannot be put in one node, there are more nodes, generally grouped by logically related fields into field numbers within some range, say 1101-1116. These would be node 11 pieces 1-16, and in this case piece 10 is allowed because it is canonic.
4. Multiple fields are always four (4) digits. The first two digits are the next higher group. Using the previous example, 11 would be the next higher group. The second two digits are always 00. The subscript for that multiple is always the first two digits of the multi-valued field number; 11 in this case.

The sub-file number is always the parent file number with the first two digits of the multi-valued field number appended. If we were in file 9000001 in the previous example, the sub-file for field 1100 would be 9000001.11, and the subscript would be 11. Now, if we added a multiple to that sub-file, for example, field number 1500, its sub-file would be 9000001.1115, and its subscript would be 15. In the data global it would look like `^AUPNPAT(DA(1),11,DA,15,0)`.

The assigning of sub-file numbers is important, because if you let FileMan do it, FileMan will assign numbers that may fall within the number space of primary files, using our file number assigning logic.

5. There are special cases that do not follow the rules. On most of the pointed to files, we have added a field number 9901 MNEMONIC, which is used on a site-by-site basis. If you have a very high percentage of your lookups to two or three entries, you can add data to the MNEMONIC field, for example, 1, 2, and 3, and instead of responding CLAREMORE to a LOCATION lookup, you can respond 1. This field is in node 88 piece 1. It is 8801, so the MNEMONIC field would be the same number in all dictionaries, regardless of how many fields and field numbers a particular file had already.
6. Computed fields, wherever possible, immediately follow the field from which they are computed, and the computed field number is the same as the real field followed by a 9. If the field above was .12, the computed field would be .129. If you wanted more than one computed field off of .12, they would be .1291 and .1292.
7. There is another class of computed field. That is a computed field that points back to the VA PATIENT file. Those fields have a .2 following the field number, to indicate that it is not really a computed field but just a pointer back to the VA PATIENT file.