



OSEHRA Forum Initial Design Document

**Version 1.0
March 18, 2014**

*©2014 Open Source Electronic Health Record Alliance, Inc.
OSEHRA Forum Initial Design by Frederick D. S. Marshall
is licensed under a
Creative Commons Attribution-ShareAlike 4.0 International License.*

Revision History

Date	Version	Description	Author
03/06/2014	0.1	Initial Draft Submission for discussion	Marshall
03/07/2014	0.2	Reformatting Draft	Hewitt
3/13/2014	0.3	Finish Reformatting and address comments	Yaw
3/17/2014	0.4	Touch-up pass on latest comments	Hewitt
3/17/2014	0.5	Added missing WBS table, reformatted numbered lists to be consistent	Yaw
3/18/2014	1.0	Accepted Initial Design Document	

Table of Contents

1	Introduction	6
1.1	Background	6
1.1.1	History	6
1.1.2	OSEHRA's Life Cycle Work to Date	6
1.2	Purpose of Document	6
2	Phase One Project Goals	8
2.1	Differences From Veterans Affairs (VA) Life Cycle	8
2.2	Similarities To The Indian Health Service (IHS) Life Cycle.....	8
2.3	Primary & Secondary Development	8
2.4	Multiple Software Streams	8
3	OSEHRA Forum Functionality	9
3.1	Life Cycle Component One – FOIA VistA Patches (Secondary Development, Re-releases).....	10
3.1.1	OSEHRA Forum's Approach	10
3.1.2	Phase One Implementation.....	11
3.1.2.1	"In Review" Status.....	11
3.1.2.2	"Secondary Completion" Status.....	11
3.1.2.3	"Secondary Release" Status	11
3.1.2.4	Other Changes.....	12
3.1.3	Related WBS Tasks	12
3.2	Life Cycle Component Two - OSEHRA Vista Patches From VA (Secondary Development, Re-releases)	17
3.2.1	OSEHRA Forum's role	17
3.2.1.1	Review Incoming FOIA VistA Patches.....	17
3.2.1.2	Renumber Patches for OSEHRA Vista	17
3.2.1.3	Track Equivalent Patches	17
3.2.1.4	Provide Easy Identification of Originating Patch Stream	18
3.2.1.5	Add Concept of Multiple Patch Streams	18
3.2.2	Phase One Implementation.....	18
3.2.2.1	Review Incoming FOIA VistA Patches.....	18
3.2.2.2	Renumber Patches for OSEHRA Vista	18
3.2.2.3	Tracking Equivalent Patches.....	18
3.2.2.4	Provide Easy Identification of Which Stream Patch is From	18
3.2.2.5	Add Concept of Multiple Patch Streams	19
3.2.3	Related WBS Tasks	19
3.3	Life Cycle Component Three - OSEHRA Vista Patches From Community (Secondary Development, Extensions & Modifications)	24
3.3.1	OSEHRA Forum's Approach	24
3.3.2	Phase One Implementation.....	24
3.3.3	Related WBS Tasks	24
3.4	Life Cycle Component Four - OSEHRA Vista Replacements For FOIA VistA Patches (Secondary Development, Modifications)	24
3.4.1	Four Kinds of Relationships	25
3.4.2	OSEHRA Forum's Approach	26
3.4.3	Phase One Implementation.....	27
3.4.4	Related WBS Tasks	27

3.5	Life Cycle Component Five · New Submissions To OSEHRA From The Community (Primary Development)	27
3.5.1	OSEHRA Forum's Approach	27
3.5.2	Phase One Implementation	28
3.5.3	Related WBS Tasks	28
4	Unified Architecture	29
4.1	Integrating The Patch Module With The KIDS Module Of The Kernel Package.....	29
4.2	Integrating The Patch Module With Git, Gerrit, Jira, & OSEHRA Technical Journal.....	29
4.3	Integrating The OSEHRA Forum System With OSEHRA Community Systems	29
5	Current User Interface	31
6	Planned Unit Tests	31
7	Conclusion	33
	Appendix A: Forum Project Phase One Work Breakdown Structure	34
1	Introduction	34
2	Work Breakdown Structure	34
3	Conclusion	40
	Appendix B: Forum Project Phase One Milestones	41
1	Purpose	41
2	Background	41
3	Tasks	41
3.1	Task 1 · Consulting on OSEHRA VistA Product Definition.....	41
3.2	Task 2 · Development of OSEHRA Forum Phase One	41
3.3	Task 3 · System Administration of OSEHRA Forum Phase One.....	42
4	Points of Contact (POC).....	42
5	Deliverables.....	42
5.1	• Task one	42
5.2	• Task two	42
5.3	• Task three.....	42
6	Schedule	43
7	Milestones.....	43
7.1	• Milestone One · Initial Forum System	43
7.2	• Milestone Two · Enhanced Forum System.....	43
7.3	• Milestone Three · Initial Operating Capability (IOC)	43
	Appendix C: Forum Configuration Nomenclature	45
1	Purpose	45
2	Executive Summary	45
3	VistA Dialect · Software Life Cycle · Upgrade Stream	45

3.1	VistA Dialect.....	45
3.2	What Your VistA Dialect Tells Us.....	47
3.3	Software Life Cycle	48
3.4	Upgrade Stream	48
3.5	New Dialects	48
4	VistA Version • Snapshot or Service Pack	49
4.1	VistA Snapshot	49
4.2	VistA Service Pack	49
4.3	Naming Snapshots and Service Packs.....	50
5	VistA Applications • Packages, Versions, & Patches	50
5.1	Applications	51
5.2	Versions & Packages.....	51
5.3	Patches.....	51
5.4	Patch Names	51
5.5	Distribution Names	52
6	Mappings • Streams, Secondary Development, Manifests	52
6.1	Mapping Patches to Software Streams	52
6.2	Mapping Original & Derived Software Streams.....	52
6.3	Mapping Original & Derived Patches	53
6.4	Manifest Section 3 • Application Versions	53
6.5	Manifest Section 2 • Deltas	53
6.6	Manifest Section 1 • Highlights	54
7	Conclusion	54
8	Basic Patch Nomenclature • Names, Subjects, Files	55
9	The Variety of Patch E-mail Subject Formats	56
10	Patch Listings	56

1 Introduction

1.1 Background

As the custodial agent for VistA's open-source codebase and its corresponding life cycle, OSEHRA needs to be able to function not just as a codebase source for new adopters but also as a hub of open-source development for existing adopters.

“Forum” is the name of a special VistA system within VA that is not directly used to support patient care. Instead, it is used to support VistA development by acting as the coordination hub for all VistA development within VA. It runs special VistA modules that no other VistA system runs, such as the Patch Module, which is described in some detail in this document, especially in Section 3, OSEHRA Forum Functionality. This OSEHRA project aims to set up a second Forum system outside VA, to be run by OSEHRA and used by the OSEHRA community to coordinate VistA development worldwide.

1.1.1 History

Prior to this project, OSEHRA has focused on new VistA adopters by creating a FOIA VistA reference environment, a Github repository for tracking versions of software components, a Gerrit code review system for tracking testing and certification, the OSEHRA Technical Journal for community submissions, a Jira issue-tracking system, the osehra.org community website for discussing projects and sharing materials, and a workgroup system supported by conference calls for organizing the work.

1.1.2 OSEHRA's Life Cycle Work to Date

OSEHRA is now expanding its support for existing adopters by (a) setting up a Forum system for the OSEHRA community, (b) upgrading Forum to support the complex kinds of VistA development the OSEHRA community needs and (c) integrating these VistA-specific tools with the standard open-source tools so they support each other. These are the three fundamental objectives of the OSEHRA Forum Project.

1.2 Purpose of Document

This document describes the Phase One OSEHRA Forum design, including five life cycle components, a unified architecture, the current user interface, and the planned unit tests.

Appendix A: Forum Project Phase One Work Breakdown Structure (WBS) is the outline of primary and secondary design elements around which the project was designed. The main document is a reorganized narrative expansion of this appendix. The five sections of the main document that describe this phase's life cycle components each end with a subsection that identifies the WBS tasks corresponding to the narrative. These subsections are written from a cumulative perspective, identifying the tasks each component adds to those required to support earlier components. Certifiers may use these subsections to help cross-reference the narrative by WBS task.

Appendix B: Forum Project Phase One Milestones is designed to help us manage Phase One by defining the milestones. It is also designed to help certifiers by identifying requirement milestones and tracing them back to design features in the WBS and narrative.

Appendix C: Forum Configuration Nomenclature describes how dialects, software streams, snapshots, service packs, and patches are named and numbered and why, including the different ways we name them in the various formats and contexts in which they may appear.

2 Phase One Project Goals

Phase One of this project is focusing on achieving an initial operating capability (IOC) that may not be ideal, but will be sufficient for the community to be able to create and receive patches from OSEHRA. Later phases of the project will focus on upgrading that IOC to create a Full Operating Capability (FOC).

2.1 Differences From Veterans Affairs (VA) Life Cycle

For now, most of OSEHRA's patches will come from VA, with a small but growing percentage being developed by the OSEHRA community. Eventually, as OSEHRA's mission to promote the open-source Vista community advances, **there will come a time when more of OSEHRA's patches come from the community than from VA, but there will always be a mix.** OSEHRA's Forum will need to be able to handle this reality.

2.2 Similarities To The Indian Health Service (IHS) Life Cycle

No VA patch is ever sent directly from VA to an IHS site. IHS intercepts all of the VA patches, reviews them for conflicts with its own software, modifies those that have conflicts, inserts its own patches, and then releases the results to the IHS sites. This is also what OSEHRA needs to do. **OSEHRA will need to extend the Forum software to make it support ingesting, modifying, and re-releasing an existing software stream.** Rephrased, that is one of the principal goals of Phase One of the OSEHRA Forum Project.

2.3 Primary & Secondary Development

OSEHRA does some primary development, and IHS does a lot, but much of what both organizations do is modifying and extending existing VA code. VA's software hub is a Forum system that only supports primary development. **What the OSEHRA Forum Project's Phase One will do is upgrade the Forum software to be able to support both primary and secondary development.** Secondary development is discussed in more depth in section 3.

2.4 Multiple Software Streams

The details of the OSEHRA Forum IOC are described in the rest of the document, but one feature deserves further introduction first, because it affects everything that follows. VA's existing Forum system only supports a single outbound stream of patches, and IHS's planned Forum system would have done likewise, but **OSEHRA Forum needs to be able to support two outbound software streams, VA FOIA Vista and OSEHRA Vista, now and any number of streams later on.**

Further, if IHS wishes to take advantage of OSEHRA Forum in the future rather than creating its own IHS Forum, then OSEHRA Forum also needs to be able to support more than one inbound software stream, since it would also need to be able to consume IHS's stream of patches for distribution without mixing up the VA and IHS patches with each other.

What these multiple software streams really are and how they will work will come up repeatedly in the discussion that follows.

3 OSEHRA Forum Functionality

In a nutshell, the rest of this paper is about OSEHRA's perspective on VA Forum, a perspective that is being deeply enriched through the OSEHRA Forum project. The resulting OSEHRA Forum system will be a shared VistA-community resource that can make possible for the first time a VistA life cycle that integrates the vertical silos of the current VistA community into a truly unified VistA-development community.

We turn now to a discussion of the five components of the VistA life cycle that OSEHRA plans to support, what role OSEHRA Forum will play for each one, and how Phase One of this project will address that role. The five components are:

- 1) FOIA VistA Patches
- 2) OSEHRA VistA Patches from VA
- 3) OSEHRA VistA patches From Community
- 4) OSEHRA VistA Replacements For FOIA Vista Patches
- 5) New Submissions To OSEHRA From The Community

The first four components are forms of *secondary development*. Secondary development is when you are modifying or extending someone else's VistA package. It is important to be able to identify and understand this kind of development, because it involves working in software that someone else will change in the future, raising special version-control challenges to handle potential collisions between the two groups' work.

As described in [VISTA Mastery](#) (Marshall & Ice, MUMPS Books, 2014), VistA has a unique architecture designed to support secondary development; most of VistA's architecture is built around extensible frameworks, in which most of the software consists of tiny modular plugins that are called by shared software engines. All VistA developers agree on rules for how to name or number these extensions; as long as each developer is assigned a namespace and number space for the exclusive use of that developer, none of that developer's work will collide with any other developer's work, even if another developer's package is being extended. The first two components strictly involve the simplest form of extension—to rerelease someone else's code—and the third sometimes involves adding extensions to someone else's code or modifying just those extensions.

The alternative form of secondary development is called modification, and it occurs when we have no alternative but to work outside of our assigned namespace or numberspace, when we are modifying someone else's code. This guarantees a future collision, since sooner or later the main development team is going to update that code, overwriting our modification. When this occurs, if we do not want to retire our modification, we must modify the main development team's patch, to replace it with a version that adds our modification back to the code. The fourth component supports modifications, though in rare cases the third component will also be used to support them.

The fifth component is different; it supports *primary development* that originates not from VA Forum but from the OSEHRA community itself, when we are not modifying VA's code but creating or modifying our own. File Manager version 22.1 is an example of primary development, since in this release we took responsibility for an entire application rather than just

inserting secondary development into VA's application. Other examples would be the initial or subsequent versions of brand new VistA applications developed by the OSEHRA community. When doing primary development, the software life cycle is different, and much simpler. Rather than dancing around the primary developers' past and future work, we ourselves are the primary developers, so we just work within our own namespace and numberspace and leave the rest of the world to dance around our past and future work.

A subset of that fifth component, primary development by VA developers, is the only form of VistA development that VA Forum currently supports. Merely by setting up our own Forum system open to the entire OSEHRA community, we are expanding our support to include all primary development, by VA or by any other primary developer. It is the other four components of the life cycle that require modifying and extending the Patch Module to give it new capabilities, so we can support both forms of secondary development as well, 1) extension by developing within a developer or organization's namespace and numberspace, and 2) modification of existing VA VistA code. In this way we can support the rich, complex software life cycle OSEHRA VistA requires.

Given that discussion of OSEHRA's perspective on VA Forum, let's turn to the five components one by one and explore what they are and how OSEHRA Forum will support each of them.

3.1 Life Cycle Component One – FOIA VistA Patches (Secondary Development, Re-releases)

Under the terms of VA's original OSEHRA Request for Proposals (RFP), a core part of OSEHRA's mission is to ensure that VistA is kept always available for new adopters, by maintaining snapshots and version-control repositories of VA's Freedom of Information Act (FOIA) release of VistA. OSEHRA has accomplished the letter of this VA requirement, but meeting its spirit requires additional work VA did not originally think to request. Specifically, VA continually updates its codebase, usually in the form of patches, so to keep its snapshots and repositories up to date, OSEHRA must continually install these new VA patches.

Because of the complex architecture of medical software, OSEHRA members who adopt this software cannot update their VistA systems merely by installing new snapshots from OSEHRA on top of their production systems. Instead of upgrading their systems, that would merely erase all their data and discard all their local development, which would be catastrophic. Instead, much like OSEHRA, community members must continually install these new VA patches. Therefore, to meet the spirit of its duties, OSEHRA needs to make all VA patches readily and promptly available to its community members.

Experienced community members outside VA are used to obtaining their patches in the form of files, from VA's download site, which VA updates approximately monthly. VistA adopters copy the new files for the month to their systems, then manually load them one at a time into their VistA systems and install them. This is, however, the inferior method.

3.1.1 OSEHRA Forum's Approach

VA sites use a superior method. Instead of having to remember to check a download site and periodically grab and load all the updates, VA sites continually receive patches in the form of e-mail messages, which are sent from VA's Forum system the moment they are released and are delivered directly to the VistA environments in which they need to be installed.

This superior method is more automated and harder to forget to do, and it is one step shy of a much longed for improvement—to further automate patching to the extent possible, which would save many man-hours of programmer time across the VistA world. Meanwhile, even without further automated patching beyond what VA has implemented internally, just recreating the VA's auto-distribution system would save community members time and effort and improve their ability to stay current with patches. OSEHRA Forum Phase One will do that, helping OSEHRA play the same kind of role for OSEHRA community members that VA Forum plays for VA sites.

3.1.2 Phase One Implementation

During Phase One, the OSEHRA Forum Workgroup will modify Forum's Patch Module software to give it the ability to create patches by importing them, either from files (the FOIA method of patch distribution) or from e-mail messages (the internal VA method). The new import option(s) will need to be able to auto-configure the Patch Module as they are run, adding new inactive users to the system so we can document who developed, completed, and released the imported patches. They will need to add package definitions to the system as they are imported; giving some of the imported users the roles of inactive developer, completer, support, or verifier based on their contributions to the imported patches. Finally, they will set up a default system for who will actively play these roles on OSEHRA Forum. The Patch Module expects to notify these users of status changes in the patches they are responsible for, so they can take the appropriate next actions in processing the patches. This new import software will thus act as a way to initialize OSEHRA Forum's soon-to-be extensive database of FOIA VistA patches and to keep current with new VA VistA patches.

3.1.2.1 "In Review" Status

A new patch status will be added to the Patch Module—in review—which these loaded patches will automatically be set to. Patches require review because VA redacts some of its software before release to avoid releasing proprietary vendor information or confidential materials like VA's cryptographic algorithms. Redaction may be fine for VA to meet its legal obligations, but it results in the distribution of broken software to the rest of the world. OSEHRA needs to be able to review incoming patches so that it can flag patches containing redacted software, to warn community members that the patch is broken. Flagged patches will have a new Redaction Description field that the reviewer will have to fill in to explain what has been redacted, why, and what the consequences are.

3.1.2.2 "Secondary Completion" Status

Another new status of secondary completion will be used to signal the certification team that a FOIA VistA patch has been reviewed and is ready for certification.

3.1.2.3 "Secondary Release" Status

Finally, a third new status of secondary release will be used to release FOIA VistA patches. Redacted patches certainly should not have the same status as the original released VA patch. Though most VA patches contain no redactions, we still will need to make clear that they have been reviewed and certified as containing no redactions, so here, too a VA VistA patch's original released status should not be used. In other words, all FOIA VistA patches released by OSEHRA Forum will be given a status of secondary release.

3.1.2.4 Other Changes

In addition to these three new patch statuses and the new patch-import option(s), the project group will need to change the main patch-development options to support these new statuses, and will need to create new reports to make it easy to identify and manage patches while they are in these statuses. OSEHRA will also need to use Forum's existing features to maintain lists of subscribers who will receive FOIA VistA patches when they undergo secondary release. Also, necessarily, a VistA instance must be set up as OSEHRA's Forum system, configured as Forum, and maintained as a production system.

Finally, to fully support a FOIA VistA patch stream from OSEHRA, we need support for OSEHRA's software life cycle added to Forum. For Phase One, our focus is on the initial, minimal, necessary hooks that can be used to start the process of weaving Git and Gerrit into Forum:

- 1) The Patch Module needs to notify Gerrit when patches change in status.
- 2) The Patch Module needs web services that Gerrit can invoke to get or send a patch.
- 3) We need to invent new flat-file formats for many VistA software elements so they can be submitted to version-control systems.
- 4) The Patch Module needs to automatically submit a patch and its components to Gerrit when the patch achieves the right status.

These four steps will bring FOIA VistA patch releases more under the control of OSEHRA's software life cycle tools. Most of these tasks are on the project's agile backlog, to be done if time permits; the only required tasks involve the development and export of the initial flat-file format for VistA software elements. What we learn from developing and using this work will guide the further integration we add in future phases of the project.

3.1.3 Related WBS Tasks

	Mailman
4.00	Mailman Connectivity
4.01	Get VistA Infrastructure Configured & Operational
4.02	Configure VistA Environment as FORUM.OSEHRA.ORG
4.03	Configure DNS Externally to Recognize FORUM.OSEHRA.ORG
4.04	Configure GW.OSEHRA.ORG & Postfix to Send to Internet
4.05	Resolve Firewall Issues Blocking Outbound Mailman
4.06	Configure Mailman in XINETD to Receive E-Mail from Internet
4.07	Configure Q-PATCH.OSEHRA.ORG to Receive Patches from Developers
4.08	Resolve Disruptions Caused by Rackspace Port Checker
	Ingest, Review, Revise, & Release Patches

5.00	Import FOIA Patches
5.01	New Server Option A1AE LOAD RELEASED PATCH [Load Released Patch Into Patch Module] to Load VA Patches into Forum
5.02	New Routine to Load a Released VA Patch E-Mail into PM
5.03	Write Code to Process Message Header
5.04	Write Code to Process the Patch Description (Boilerplate—Parsed Fields Plus Word Processing)
5.05	Write the Code to Process the KIDS Distribution (Structured, so Should be Decomposed & Filed as Fields)
5.06	Create Bulletin A1AE LOAD RELEASED PATCH [New Patch Loaded into Patch Module]
5.07	Write a Subroutine to Email the Bulletin A1AE LOAD RELEASED PATCH to the Development Group
5.08	Test the Server Option A1AE LOAD RELEASED PATCH [Load Released Patch into Patch Module]
5.09	Integrate Cameron's Code to Load Released VA Patch from Host Files
5.10	Integrate Cameron's Code to Release a Patch as Host Files
5.11	Decompose Compound Builds in a Host-File Patch Into Separate PM Patches
5.12	Compose Patches into a Compound Host-File Patch
5.13	Write & Integrate Code to Create a Properly Named & Numbered Copy of an Ingested Patch for Each Patch Stream, Based on A1AESTRM · Send the Bulletin A1AE LOAD RELEASED PATCH for Each Copy
5.14	Modify the Before Checksum Handling to Support Multiple Patch Streams · This is Likely to Consist of Many Subtasks
5.15	Recalculate the Associated Patches to Refer to Derived Patches, Not Original Patches
5.16	Handle Cross-Package Associated Patches During System Initialization
5.17	Add Field "Update To Patch"
5.18	Initialize Patch Module with All Past Patches
6.00	New Statuses, Rules, and Fields for Secondary Development
6.01	Add New Statuses to PM
6.02	Add New Status "In Review" to PM
6.03	Add New Rules to Go with "In Review" Status
6.04	Add Rule that Users Cannot Change a Patch's Status to "In Review" Manually

6.05	Add Rule that A1AE LOAD RELEASED PATCH Sets the Status of the Loaded Patch to “In Review” as Its Last Step
6.06	Add Fields to Support Flagging Redacted Patches and Describing the Redaction; Add Report and Other-Option Support for Displaying these New Fields.
6.07	Add Rule that an “In Review” Patch may be Changed to Either “Secondary Development”, “Secondary Completion”, or “Not For Secondary Release”
6.08	Add New Status “Secondary Development”
6.09	Add Rule that a “Secondary Development” Patch may be Changed to Either “Secondary Completion” or “Not For Secondary Release”
6.10	Add New Status “Secondary Completion”
6.11	Add Rule that a “Secondary Completion” Patch may be Changed to Either “Secondary Release” or “Not For Secondary Release”
6.12	Add New Status “Secondary Release”
6.13	Add Rule that when Status Set to “Secondary Release” the Patch Email Message is Generated and Sent to All Subscribers
6.14	Add New Status “Not For Secondary Release”
6.15	Add New Field to Explain Why “Not For Secondary Release”
6.16	Add New Field to Explain What was Done as Part of “Secondary Development”
6.17	Add New Field Type to Distinguish Original Patches from Derived Ones
6.18	Add New Field Original Patch to Point Back to a Derived Patch’s Original
6.19	Make Type a Conditional Identifier
6.20	Make Original Patch a Conditional Identifier
6.21	Add Patch Stream, Type, & Original Patch to Patch Header
7.00	Modify Templates & Options to Support Secondary Management of a Patch
7.01	Input Template A1AE ADD/EDIT PATCHES
7.02	Sort Template A1AE FULL SUMARY SORT
7.03	Sort Template A1AE FULL SUMMARY BY DATE
7.04	Sort Template A1AE PATCH COMPL/COMM SORT
7.05	Sort Template A1AE SEQ REV SAVE
7.06	Sort Template A1AE SEQ REV SUMMARY SORT
7.07	Sort Template A1AE SEQ SAVE
7.08	Sort Template A1AE SEQ SUMMARY SORT

7.09	Sort Template A1AE STANDARD SORT
7.10	Sort Template A1AE SUMMARY SORT
7.11	Sort Template A1AEZPKE
7.12	Print Template A1AE FULL SUMMARY BY DATE
7.13	Print Template A1AE PATCH COMPL/COMMENT RPT
7.14	Print Template A1AE PATCH COMPLIANCE PRT
7.15	Print Template A1AE PATCH SUMMARY
7.16	Print Template A1AE STANDARD HEADER
7.17	Print Template A1AE STANDARD PRINT
7.18	Print Template A1AE VERIFIED PATCH SUMMARY
7.19	Print Template A1AEZPKE
7.20	Option A1AE PHADD [Add a Patch]
7.21	Option A1AE PHEDIT [Edit a Patch]
7.22	Option A1AE PHVER [Release a Patch (And/or Edit Internal Comments)]
7.23	Option A1AE PRT SUM DT ORG [Released Patch Summary Report by Date]
7.24	New Option A1AE PRTCOMDETD [Released Patches by Date Detailed Report]
7.25	New Option A1AE PRTCOMPHDTE [Completed Patch Summary Report by Patch Stream]
7.26	New Option A1AE PRTCOMSUMPS [Released Patch Summary Report by Patch Stream]
7.27	New Option A1AE PRTDETD ORG [Detailed Report of Released Patches by Patch Stream]
7.28	Option A1AE PRTPHA [All Released Patches for a Package, Detailed]
7.29	New Options for Reports on Patches with the New Statuses
7.30	New Menu for Options that Help Manage a Patch Stream
7.31	New Option to Show Patch Relationships—Dependencies & Derivations
7.32	Reorg Menus to Make Them Coherent
7.33	Other Package Elements
	Git, Gerrit, & Forum
8.00	When Patches Change to the Right Statuses, Export Patch to Gerrit
8.01	New Protocol for Gerrit Export

9.00	Make PM Update Gerrit at All the Right Times
9.01	Change PM Options to Trigger that Protocol When Patch is Edited
10.00	Improve Integration with Github · Initial Flat-File Formats for KIDS Components
10.01	Work Out Initial Formats for All Twenty-Four or so KIDS Components
10.02	Write Initial Exporter
10.03	Integrate into Export Protocol: Send Individual Components Along with Patch

3.2 Life Cycle Component Two - OSEHRA VistA Patches From VA (Secondary Development, Re-releases)

The problem with the FOIA VistA dialect and patch stream is that from the community's perspective it is read-only; it leaves no room for the OSEHRA community to collaborate and extend VistA, which is a key goal of OSEHRA's. To support this dimension of OSEHRA's mission, the Forum system needs to support a second dialect of VistA, OSEHRA VistA, which will have a read-write patch stream that welcomes community participation. OSEHRA VistA will release three kinds of patches:

- those substantively unchanged from their FOIA VistA version (which this section discusses),
- those contributed by the OSEHRA community (which section 3.3 discusses), and
- those that come from FOIA VistA but have to be changed, such as to repair redacted patches (which section 3.4 discusses).

Our approach to these three kinds of patches builds upon the excellent example set by IHS, but adds the automation IHS has been missing, to simplify the work involved.

3.2.1 OSEHRA Forum's role

Even when a FOIA VistA patch's description and software payload remain unchanged when it is released as part of OSEHRA VistA—which for the immediate future will be the case with most patches—there are still five changes required:

- 1) Review Incoming FOIA VistA Patches
- 2) Renumber Patches for OSEHRA VistA
- 3) Track Equivalent Patches
- 4) Provide Easy Identification of Which Stream Patch is From
- 5) Add Concept of Multiple Patch Streams

3.2.1.1 Review Incoming FOIA VistA Patches

The first change is that, as with life cycle component one, we need to review the incoming FOIA VistA patches to determine which of the three kinds of patch it is; that means we need associated options, database changes, and workflows, as described in the previous section.

3.2.1.2 Renumber Patches for OSEHRA VistA

The second and more significant change is that FOIA VistA's numbering of patches leaves no room for the insertion of community-developed patches. OSEHRA VistA patches need to be able to have different patch and sequence numbers; between FOIA VistA patches 9 and 10, we may need to insert three community-developed patches, so that FOIA VistA patch 10 would be called OSEHRA VistA patch 13.

3.2.1.3 Track Equivalent Patches

The previous change requires that we develop a way to keep track of equivalent patches, so we know that OSEHRA VistA patch 13 is FOIA VistA patch 10.

3.2.1.4 *Provide Easy Identification of Originating Patch Stream*

The fourth change, which follows from the first three, is that we need to be very sure which patches are FOIA VistA patches and which are OSEHRA VistA patches, so that, for example, FOIA VistA patch 10 is not inadvertently installed instead of OSEHRA VistA patch 10 (a completely different patch). Thus the OSEHRA Forum patch nomenclature must differentiate clearly among multiple sources for patches.

3.2.1.5 *Add Concept of Multiple Patch Streams*

The fifth change, implicit in all of the above, is the need to add the concept of different patch streams. We need to know that we can add community-developed patches to OSEHRA VistA without impacting FOIA VistA, while still applying applicable FOIA VistA patches to OSEHRA VistA.

OSEHRA Forum will supply database, business, and user-interface support for these five changes.

3.2.2 Phase One Implementation

Phase One will address each of the five changes outlined in the previous section in the following ways:

3.2.2.1 *Review Incoming FOIA VistA Patches*

We will review each incoming FOIA VistA patch to determine which of the three kinds of patch it is; that means we will create associated options, database changes, and workflows.

3.2.2.2 *Renumber Patches for OSEHRA VistA*

To support the second change, even when OSEHRA VistA patches are identical to the FOIA VistA patches they are derived from, we will still create a completely separate copy of the patch for naming, management, and distribution within the OSEHRA VistA patch stream. That is, the process of importing FOIA VistA patches will result in the creation of two patches, one for FOIA VistA under its original patch number, the other for OSEHRA VistA under a new OSEHRA-numberspaced patch number.

3.2.2.3 *Tracking Equivalent Patches*

To support the third change, we will point derived patches back to their originals, so we know, for example, that OSEHRA VistA patch 13 is derived from FOIA VistA patch 10. This will be captured both in the database and in revisions to the patch boilerplate, so both human readers and the software are perfectly clear about these relationships. This will also help us build the recalculated list of associated patches, since now we will know which OSEHRA VistA patches are derived from which FOIA VistA patches.

3.2.2.4 *Provide Easy Identification of Which Stream Patch is From*

To support the fourth change, we will change patch naming conventions to add both explicit and implicit designation of the dialect and patch stream. The dialect will be named in patch subjects, headers, and file names, and in the database itself. In addition, because patch numbers are the current way the VistA community distinguishes patches, patch numbers will be numberspaced by dialect, to reduce the chances of their being mixed up with each other. For a complete

description of the new numberspacing conventions, see Appendix C: Forum Configuration Nomenclature.

Also to support the fourth change, we must recalculate a patch's associated patches; this is the list of earlier patches that modified one or more of the exact same software components. These are the patches whose installation we do not dare skip if we intend to install this patch, because this patch partly overwrites their work. The patch numbers shown when we import a patch will be the old FOIA VistA patch numbers; we need to replace that list with the equivalent OSEHRA VistA patch numbers. This will require initializing the OSEHRA Forum Patch Module with all past VistA patches (at least for the current versions of the applications), so that all the earlier patches are available to be selected as associated patches.

3.2.2.5 Add Concept of Multiple Patch Streams

To support the fifth change, Phase One will introduce a new file to the Patch Module called DHCP Patch Stream (11007.1), which will record the VistA dialects supported by Forum and describe their relationships sufficiently to create their associated patch streams. For now, of all the dialects listed, only OSEHRA VistA will be set to a type of primary, which means it is the one the community can contribute development to.

3.2.3 Related WBS Tasks

	Version, Patch, & Sequence Numbers
1.00	Longer Patch Numbers
1.01	Update KIDS Data Dictionaries (DDs) to Support Longer Patch Numbers
1.02	Update KIDS Routines to Support Longer Patch Numbers
1.03	Update Patch Module (PM) DDs to Support Longer Patch Numbers
1.04	Update PM Routines to Support Longer Patch Numbers
1.05	Update Non-routine Components of KIDS & PM to Support Longer Patch Numbers · For Example, Input Templates
2.00	Longer Sequence Numbers
2.01	Update KIDS DDs to Support Longer Sequence Numbers
2.02	Update KIDS Routines to Support Longer Sequence Numbers
2.03	Update PM DDs to Support Longer Sequence Numbers
2.04	Update PM Routines to Support Longer Sequence Numbers
2.05	Update Non-routine Components of KIDS & PM to Support Longer Sequence Numbers · For Example, Input Templates
3.00	Tie Patch Numbers to Numberspacing & to Patch Stream

3.01	New DHCP Patch Stream File
3.02	New Name Field in DHCP Patch Stream
3.03	New Numberspace Field in DHCP Patch Stream
3.04	Input Transform on Numberspace to Restrict Use of 1 to Forum.VA.Gov
3.05	New Type Field (Primary Or Secondary) to Characterize Patch Streams · Only One Can Be Primary
3.06	New A1AE PATCH STREAM Compound Cross-Reference
3.07	Load A1AE PATCH STREAM into Local Array A1AESTRM
3.08	Set Top Node of A1AESTRM to the Primary Patch Stream
3.09	Make Patch Module Honor A1AESTRM
3.10	New Patch Stream Field in DHCP Patch File to Point to the DHCP Patch Stream
3.11	Make Patch Stream Field a Conditional Identifier (If Site Has More Than One)
	Patch Streams for OSEHRA VistA & FOIA VistA
	Mailman
4.00	Mailman Connectivity
4.01	Get VistA Infrastructure Configured & Operational
4.02	Configure VistA Environment as FORUM.OSEHRA.ORG
4.03	Configure DNS Externally to Recognize FORUM.OSEHRA.ORG
4.04	Configure GW.OSEHRA.ORG & Postfix to Send to Internet
4.05	Resolve Firewall Issues Blocking Outbound Mailman
4.06	Configure Mailman in XINETD to Receive E-Mail from Internet
4.07	Configure Q-PATCH.OSEHRA.ORG to Receive Patches from Developers
4.08	Resolve Disruptions Caused by Rackspace Port Checker
	Ingest, Review, Revise, & Release Patches
5.00	Import FOIA Patches
5.01	New Server Option A1AE LOAD RELEASED PATCH [Load Released Patch Into Patch Module] to Load VA Patches into Forum
5.02	New Routine to Load a Released VA Patch E-Mail into PM
5.03	Write Code to Process Message Header

5.04	Write Code to Process the Patch Description (Boilerplate—Parsed Fields Plus Word Processing)
5.05	Write the Code to Process the KIDS Distribution (Structured, so Should be Decomposed & Filed as Fields)
5.06	Create Bulletin A1AE LOAD RELEASED PATCH [New Patch Loaded into Patch Module]
5.07	Write a Subroutine to Email the Bulletin A1AE LOAD RELEASED PATCH to the Development Group
5.08	Test the Server Option A1AE LOAD RELEASED PATCH [Load Released Patch into Patch Module]
5.09	Integrate Cameron's Code to Load Released VA Patch from Host Files
5.10	Integrate Cameron's Code to Release a Patch as Host Files
5.11	Decompose Compound Builds in a Host-File Patch Into Separate PM Patches
5.12	Compose Patches into a Compound Host-File Patch
5.13	Write & Integrate Code to Create a Properly Named & Numbered Copy of an Ingested Patch for Each Patch Stream, Based on A1AESTRM · Send the Bulletin A1AE LOAD RELEASED PATCH for Each Copy
5.14	Modify the Before Checksum Handling to Support Multiple Patch Streams · This is Likely to Consist of Many Subtasks
5.15	Recalculate the Associated Patches to Refer to Derived Patches, Not Original Patches
5.16	Handle Cross-Package Associated Patches During System Initialization
5.17	Add Field "Update To Patch"
5.18	Initialize Patch Module with All Past Patches
6.00	New Statuses, Rules, and Fields for Secondary Development
6.01	Add New Statuses to PM
6.02	Add New Status "In Review" to PM
6.03	Add New Rules to Go with "In Review" Status
6.04	Add Rule that Users Cannot Change a Patch's Status to "In Review" Manually
6.05	Add Rule that A1AE LOAD RELEASED PATCH Sets the Status of the Loaded Patch to "In Review" as Its Last Step
6.06	Add Fields to Support Flagging Redacted Patches and Describing the Redaction; Add Report and Other-Option Support for Displaying these New Fields.
6.07	Add Rule that an "In Review" Patch may be Changed to Either "Secondary

	Development”, “Secondary Completion”, or “Not For Secondary Release”
6.08	Add New Status “Secondary Development”
6.09	Add Rule that a “Secondary Development” Patch may be Changed to Either “Secondary Completion” or “Not For Secondary Release”
6.10	Add New Status “Secondary Completion”
6.11	Add Rule that a “Secondary Completion” Patch may be Changed to Either “Secondary Release” or “Not For Secondary Release”
6.12	Add New Status “Secondary Release”
6.13	Add Rule that when Status Set to “Secondary Release” the Patch Email Message is Generated and Sent to All Subscribers
6.14	Add New Status “Not For Secondary Release”
6.15	Add New Field to Explain Why “Not For Secondary Release”
6.16	Add New Field to Explain What was Done as Part of “Secondary Development”
6.17	Add New Field Type to Distinguish Original Patches from Derived Ones
6.18	Add New Field Original Patch to Point Back to a Derived Patch’s Original
6.19	Make Type a Conditional Identifier
6.20	Make Original Patch a Conditional Identifier
6.21	Add Patch Stream, Type, & Original Patch to Patch Header
7.00	Modify Templates & Options to Support Secondary Management of a Patch
7.01	Input Template A1AE ADD/EDIT PATCHES
7.02	Sort Template A1AE FULL SUMARY SORT
7.03	Sort Template A1AE FULL SUMMARY BY DATE
7.04	Sort Template A1AE PATCH COMPL/COMM SORT
7.05	Sort Template A1AE SEQ REV SAVE
7.06	Sort Template A1AE SEQ REV SUMMARY SORT
7.07	Sort Template A1AE SEQ SAVE
7.08	Sort Template A1AE SEQ SUMMARY SORT
7.09	Sort Template A1AE STANDARD SORT
7.10	Sort Template A1AE SUMMARY SORT
7.11	Sort Template A1AEZPKE
7.12	Print Template A1AE FULL SUMMARY BY DATE

7.13	Print Template A1AE PATCH COMPL/COMMENT RPT
7.14	Print Template A1AE PATCH COMPLIANCE PRT
7.15	Print Template A1AE PATCH SUMMARY
7.16	Print Template A1AE STANDARD HEADER
7.17	Print Template A1AE STANDARD PRINT
7.18	Print Template A1AE VERIFIED PATCH SUMMARY
7.19	Print Template A1AEZPKE
7.20	Option A1AE PHADD [Add a Patch]
7.21	Option A1AE PHEDIT [Edit a Patch]
7.22	Option A1AE PHVER [Release a Patch (And/or Edit Internal Comments)]
7.23	Option A1AE PRT SUM DT ORG [Released Patch Summary Report by Date]
7.24	New Option A1AE PRTCOMDETD [Released Patches by Date Detailed Report]
7.25	New Option A1AE PRTCOMPHDTE [Completed Patch Summary Report by Patch Stream]
7.26	New Option A1AE PRTCOMSUMPS [Released Patch Summary Report by Patch Stream]
7.27	New Option A1AE PRTDETD ORG [Detailed Report of Released Patches by Patch Stream]
7.28	Option A1AE PRTPHA [All Released Patches for a Package, Detailed]
7.29	New Options for Reports on Patches with the New Statuses
7.30	New Menu for Options that Help Manage a Patch Stream
7.31	New Option to Show Patch Relationships—Dependencies & Derivations
7.32	Reorg Menus to Make Them Coherent
7.33	Other Package Elements

3.3 Life Cycle Component Three - OSEHRA Vista Patches From Community (Secondary Development, Extensions & Modifications)

The next simplest component is inserting community patches into the OSEHRA Vista patch stream. This situation is closer to what VA national developers do—that is, it is closer to what VA Forum was already designed to support—than either of the prior two life cycle components. The only really new things needed to support it are things the previous two components already handle—new numbering of patches, understanding the concept of different patch streams, and so on.

3.3.1 OSEHRA Forum's Approach

OSEHRA Forum makes this possible. This component above all is what the community currently has no way to do. We predict that this project's support for this specific component of patches will lead to an accelerating growth of community participation in the development of OSEHRA Vista, which in turn will gradually transform VA's relationship with OSEHRA for the better.

Most of the patches we're discussing in this paper will have passed through the original three stages of VA development—development, completion, verification/release—and then have added to them at least a review and a secondary release. Component-three patches are different. They are very much like VA Vista patches; they have a similar life cycle, and pass through the same three statuses. The other patches discussed in this document will pass through at least five and maybe as many as seven statuses, but component-three patches will only have three statuses—development, completion, and verification/release—just like VA Vista patches.

This is because the OSEHRA Vista patch stream is different in kind from the VA Vista or FOIA Vista patch streams. They contain homogenous patches with similar life cycles. OSEHRA Vista is a more complex patch stream that contains heterogeneous patches—some that are original to the community, many that are just rereleases of VA patches, and some that are replacements for VA patches (as discussed in section 3.4 below, as part of component four). Above all, this capacity for heterogeneity is what is fundamentally new about the OSEHRA Vista patch stream. It is what VA Forum has never supported; it is what this project is designed to create; and it is why this project was necessary for OSEHRA and the community. It is what will make this project draw together the existing Vista community to collaborate within OSEHRA, because they will finally have the capacity to create shared software streams from fundamentally different kinds of sources.

3.3.2 Phase One Implementation

Interestingly, this involves no new functionality, beyond the upgrades already covered by the previous two components, when added to the original capabilities of VA Forum.

3.3.3 Related WBS Tasks

Nothing new, just refinement of prior tasks.

3.4 Life Cycle Component Four - OSEHRA Vista Replacements For FOIA Vista Patches (Secondary Development, Modifications)

The fourth component is the trickiest of the five, but like component three it mostly leverages the work done to support the prior components.

The need for OSEHRA VistA replacements is partly a result of those redacted FOIA VistA patches. We would like to replace the redacted code in the equivalent OSEHRA VistA patches so the OSEHRA version of the patches returns the functionality the redactions took out. That means the OSEHRA community needs to be able to do secondary development upon the OSEHRA VistA version of the patch until they have healed the damage done by the redaction; the resulting modified patch is then released, under its own OSEHRA VistA numbers.

This also partly comes up when the OSEHRA community wants to do development on a package that cannot be completely isolated from the development VA is doing. For example, if we consider something in the VA code to be intrinsically broken, or too VA-centric, or not flexible enough, then we need to be able to change that code that VA has developed, and which they are likely to have patched before and may patch again. This form of secondary development—modification—has a complex and well understood software life cycle, but until now it has had no support from Forum or the Patch Module and has been entirely a manual process.

Modification is a vital element of the VistA software life cycle that provides (a) an escape valve to take pressure off an application's primary developers and (b) an arena in which innovation is given precedence over standardization. Given the lack of automation support in the past for modification, it has led to a certain amount of chaos between different VistA systems, given the accumulating differences between them, but suppressing it would be a strategic mistake. Doing so would create an increasing logjam in the pace of VistA development and a tendency toward one-size-fits-none solutions, with dire consequences for patient safety and general usability. Indeed, all non-VA VistA developers understand that VA VistA as is cannot be used outside VA; it is too VA-centric.

Secondary development—including modification—is necessary to make VistA widely useful, and automation support is necessary to prevent modification from wreaking havoc on code-convergence and other code-management efforts.

3.4.1 Four Kinds of Relationships

Before we go on to the remaining forms of support this takes, let us step aside to summarize the four kinds of dependencies patches may have upon one another, two of which are supported below.

First, patches are applied to a specific version of a specific application, and all patches to that version of that application are arranged into a default installation sequence, identified by the patch sequence number.

Second, however, this is only a default sequence. When considering a patch for possible installation, you can often install it out of the default sequence order, so long as you at least first install the other earlier patches that changed one or more of the same software elements this patch changes. Perhaps surprisingly, there are times when this is advisable, to be discussed in some other, future document. To support this out-of-sequence patch installation, every patch identifies its *associated patches*, which is a list of those absolute prerequisites.

Third, in the case of secondary development, a derived patch always has some original patch from which it is derived. In the ordinary course of patching, this is interesting to know, but in the case of developing a modification—a replacement patch—it is essential.

Fourth, the other essential thing to know whenever developing a replacement patch, is if there was a previous version of this replacement patch. This happens when a modification is too

extensive to fit within the confines of a modification to a single patch. In the case of such big modifications, instead of modifying the patch, we develop a local patch to follow right after it, so the two are installed together, one after the other. In such cases, it is not unusual to see the primary and secondary developers releasing patches in a kind of oscillating dance, where repeated primary patches to the affected code are each followed by a secondary patch that reapplies the modification, or in some cases even redesigns the modification to take advantage of new capabilities introduced by the primary developers. In such oscillations, each new revision of the secondary patch is not only derived from the primary patch (the third kind of relationship), it is also a new version of the previous secondary patch (the fourth kind of relationship).

When doing secondary development that involves modifying primary code, the developer always has to take into account the first three relationships and sometimes has to take into account the fourth relationship. This is what makes maintaining ongoing modifications to VISTA code such a labor-intensive process, and why it is so ripe for change, for improved automation.

With these four relationships in mind, let us turn to the ways component four supports modification.

3.4.2 OSEHRA Forum's Approach

OSEHRA Forum will supply this component of the VistA life cycle that is missing from VA Forum. Automation support for modification, using an upgraded Patch Module, available to the VistA community through the OSEHRA Forum system, will for the first time in VistA's history let us fully exploit the innovative power of modification without paying the chaotic price incurred by haphazard manual processes.

The first form this support takes is by supporting the ability of a patch reviewer (see component one) to open a released patch back up for further development, completion, and verification/release. This will be done by extending the in review status discussed in component one to support setting a patch's status to under a new secondary development status, to allow it to be modified, and thence to secondary completion, to allow it to be verified and certified, and thence to secondary release. This alternate redevelopment pathway will let secondary developers perform their modifications with the Patch Module's full awareness and support.

The second form of support is the introduction of another new status—not for secondary release—so that VA patches not desired at all by the OSEHRA community (such as patches that erase functionality VA does not want any longer, but that the rest of the community wants to keep) can be suppressed. This is a subtle but important form of modification that IHS and other secondary developers rely on; it is infrequent but necessary.

The third form of support was implemented in component two—the ability to point a derived OSEHRA VistA patch back to its original FOIA VistA version; that is, it was for the third kind of patch relationship. In component two, that was just used to compensate for the renumbering of patches in the OSEHRA VistA patch stream, but here in this component it is used to tie a modified OSEHRA VistA patch back to its unmodified original FOIA VistA version. At present, developers try to keep track of these kinds of relationships manually; VA's Patch Module does not support this at all, but OSEHRA's Patch Module will.

The fourth form of support is for the fourth kind of relationship, described above, which comes up when supporting oscillations of primary and secondary patches.

3.4.3 Phase One Implementation

Phase one will introduce the necessary new patch statuses and associated business logic to allow patches to move through the life cycle stages of secondary development needed to support modification, to create replacement patches or oscillating follow-up patches as needed. It will also add new field Update to Patch to file DHCP Patches (11005) to let us represent the fourth relationship. Considerably more support for replacement patches is desirable, but it will be deferred until phase two, so we can restrict the scope of Phase One enough to make it testable and finishable in the time allowed. This will be sufficient to get OSEHRA Forum to an initial operating capability for supporting replacement patches.

3.4.4 Related WBS Tasks

5.17	Add Field “Update To Patch”
6.07	Add Rule that an “In Review” Patch may be Changed to Either “Secondary Development”, “Secondary Completion”, or “Not For Secondary Release”
6.08	Add New Status “Secondary Development”
6.09	Add Rule that a “Secondary Development” Patch may be Changed to Either “Secondary Completion” or “Not For Secondary Release”
6.10	Add New Status “Secondary Completion”
6.11	Add Rule that a “Secondary Completion” Patch may be Changed to Either “Secondary Release” or “Not For Secondary Release”
6.12	Add New Status “Secondary Release”
6.13	Add Rule that when Status Set to “Secondary Release” the Patch Email Message is Generated and Sent to All Subscribers
6.14	Add New Status “Not For Secondary Release”
6.15	Add New Field to Explain Why “Not For Secondary Release”
6.16	Add New Field to Explain What was Done as Part of “Secondary Development”

3.5 Life Cycle Component Five · New Submissions To OSEHRA From The Community (Primary Development)

The final component is the easiest of the five—to support the OSEHRA community’s ability to do primary development, that is, to develop brand-new software that is neither an extension of nor a modification of existing VA VistA code. VA Forum’s native capabilities support everything needed to do this, except for properly numberspacing OSEHRA VistA contributions, which component two above already provides.

3.5.1 OSEHRA Forum’s Approach

More than any specific change to the Patch Module, it is OSEHRA Forum’s presence and operation outside VA that makes this component of the OSEHRA VistA life cycle possible.

3.5.2 Phase One Implementation

This fifth component involves no new functionality, beyond the upgrades already covered by the previous four components, above all when added to the original capabilities of VA Forum.

3.5.3 Related WBS Tasks

None needed.

4 Unified Architecture

Creating a unified architecture for Forum and the Patch Module would involve three kinds of integration that are not scheduled for Phase One, as described below.

4.1 Integrating The Patch Module With The KIDS Module Of The Kernel Package

Eventually, the long-term plan for the Patch Module must be to give up its independent status as a separate package, to fold it into the KIDS module of the Kernel package where it belongs, and to remove some of the rough edges and seams by which the two have until now been welded together. For scoping reasons, none of this is planned for any of the early phases of this project, because it involves renamespacing and renumberspacing every component of the package to fit it within KIDS's namespace and numberspace. We have done this successfully before with other packages, but it should not be undertaken until we are at the peak of our understanding of the existing package, to ensure success.

For these early phases, the initial design for the unified architecture is to keep the Patch Module exactly as it is—a unique application in the Washington VA Field Office's A1A namespace that runs only on a single Forum system—and to continue to have it interact across package boundaries with the Kernel and Mail Manager packages so that together the three applications create a unified software life cycle for VistA. All three applications will run on OSEHRA Forum, and Kernel and Mailman will run in each VistA development and production environment to support the development, distribution, and installation stages of the life cycle.

4.2 Integrating The Patch Module With Git, Gerrit, Jira, & OSEHRA Technical Journal

Likewise, the long-term plan is for the Patch Module to become increasingly aware of and integrated with OSEHRA's other software-life cycle tools, such as Git, Gerrit, and Jira, to gradually create a unified experience for those involved in the OSEHRA VistA software life cycle. For now this is hampered in part by a lack of silent interfaces between the Patch Module and these other tools, but it is also blocked by the Patch Module's dialog-mode user interface, which cannot be smoothly blended with these other tools' web interfaces. The solution is to develop a web interface for the Patch Module, but this too must wait for future phases, and for the same reason—because replacing a VistA application's user interface is invasive work that requires complete mastery of the application before starting. These early phases will develop the needed mastery, so that later phases can tackle this problem and make a unified software-life cycle experience conceivable.

Here, too, then, for these early phases, the initial design for the unified architecture is to keep the Patch Module as a separate application that will be extended with an increasing number of interface hooks between it and the other OSEHRA life cycle tools.

4.3 Integrating The OSEHRA Forum System With OSEHRA Community Systems

Finally, the Forum system itself is not yet fully aware of its subscribers, of the OSEHRA community systems that will receive patches from it. It will maintain subscriber lists, to which it will send its OSEHRA VistA patches as they are released, but it will not yet receive back

information about which systems are up to date with installing their patches. This is something VA Forum tracks, using a small application called DHCP Tracking, which acts in concert with KIDS, Mailman, and the Patch Module to accomplish the necessary messaging. We will want to get this application from VA, configure it, get it operational, and patch KIDS in OSEHRA VistA environments to send its confirmation e-mails to it instead of to VA Forum—or rather, to send the confirmation e-mails to different destinations depending on the patch stream. The result of all this work will be a more integrated, more unified architecture across all VistA systems in the OSEHRA community.

For Phase One, this degree of cross-system integration is not necessary, so we defer it to keep Phase One's scope under control. However, it should be put in place before we begin signing up OSEHRA Forum users at the June VistA Community Meeting, so it should be part of phase two's design.

All told, these three kinds of integration are not present in Phase One's IOC architecture, because they are not needed to demonstrate the system's potential. However, these kinds of integration will be needed later to achieve full operating capability, and to create a truly unified architecture on all three levels. For Phase One, we focus on the art of the possible.

5 Current User Interface

As described in 4.2 above, although Gerrit, Git, Jira, and the OSEHRA Technical Journal have modern web user interfaces, VA Forum uses a traditional terminal-based user interface, as will OSEHRA Forum's initial operating capability. Upgrading Forum to a web UI is nontrivial work, too difficult to achieve in the early phases of the project, because the Patch Module's database and business logic are closely intertwined with its UI, to support a rich user dialog.

The early phases of this project, including Phase One, are beginning that disentangling process, especially by opportunistically replacing hard-wired UI output built into the database to instead use more modern UI tools capable of redirecting output to arrays for silent operation, in preparation for sending those arrays to GUIs or web UIs (WUIs?). Even so, for these early phases the Patch Module will remain solidly terminal-based.

6 Planned Unit Tests

Adding unit tests to a mature and robust application that lacks them is a time-consuming process best done iteratively, opportunistically, and following a triage process that prioritizes bracketing planned changes with before-and-after unit tests to demonstrate that the changes have not damaged the application. We plan to follow this approach, with unit tests built to bracket each major change that is part of the project design.

At such a primitive stage of testing, pragmatism also dictates that we seek breadth over depth initially, to replace nothing with something as widely as feasible within the project constraints. Therefore, we also intend to improve the depth and granularity of unit tests iteratively, with early unit tests just ensuring the code is in the ballpark and later phases refining those same unit tests to distinguish singles from doubles, so to speak.

To balance the competing needs of modifying and extending an existing application under tight time constraints while still rolling out unit tests to eventually cover the whole application in fine-grained tests, our plan for the phases of this project is to alternate focus from one phase to the next. Phase one will focus more on development than testing, inserting unit tests opportunistically as described above. Phase two will reverse those priorities, with a lighter development plan and a heavier load for adding and refining unit tests. It is our intent to continue alternating focus from phase to phase for the duration of the project, to ensure that each of these two paramount priorities—development and testing—gets its due attention.

The other principle we are applying to this project is to focus unit testing on functionality, not database elements. For the latter, we are opportunistically tuning up the Patch Module's database integrity, adding true database keys (especially natural keys), identifiers, security, and other database elements to make File Manager protect database quality, reducing the immediate need for database-focused unit tests. This buys us more bandwidth to apply to unit tests that cover the code, where most of our changes are happening, and therefore where most of the problems are likely to occur. Later, once initial operating capabilities and the unit tests that cover them have been developed and refined, we would like to revisit database integrity by developing database-integrity checkers for each of the Patch Module's files.

Based on the above, the planned unit tests correspond exactly to the non-database tasks described in this document and listed in the Phase One Work Breakdown Structure. Here is a list of the unit tests we have developed so far:

- 1) Make the OSEHRA VistA patch stream
- 2) Make a package in file Package (9.4)
- 3) Make users in file New Person (200)
- 4) Add a package to Patch Module file DHCP Patch/Problem Package (11007)
- 5) Setup a package in the Patch Module
- 6) Setup a new version of a package
- 7) Delete all e-mail messages in Q-PATCH basket (used to receive patches)
- 8) Mail a KIDS build to XXX@Q-PATCH.OSEHRA.ORG (simulating a developer)
- 9) Get Postmaster basket for Q-PATCH in variable QUE
- 10) Obtain next patch number
- 11) Set up a patch a la 1+3^A1AEPH1
- 12) Add routine set in file Message a la 1+5^A1AEPH1
- 13) Get messages matching A1AEPD in Q-PATCH queue in variable A1AERD
- 14) Create a patch (status: under development)
- 15) Complete the patch (status: completed/unreleased)
- 16) Verify the patch (status: released)
- 17) Create a second patch - complete this one (leave completed)
- 18) Create a third patch - don't complete or verify (leave under development)
- 19) Test Report 5^A1AEPH2 (option Summary Report for a Package)
- 20) Test Report 1^A1AEPH2 (option Completed/unverified Patch Report)
- 21) Try to exceed 999 (this always passes now, may phase out)
- 22) Import KIDS test files into Mailman messages
- 23) Import KIDS test files directly into the Patch Module

We will certainly be developing more unit tests related to the various new statuses described in earlier sections, and more unit tests to ensure that existing Patch Module reports (beyond just the two in tests 19 & 20 are still generating results. Beyond that, we will continue to seek opportunities to bracket other discrete changes that are part of Phase One.

7 Conclusion

Explaining the initial design that Phase One of this project is implementing is nontrivial, because understanding the changes requires understanding the Forum system, which in turn requires understanding the Patch Module, which in turn requires understanding some vital but fairly complex elements of VistA's software life cycle.

Our plan for this paper is like our plan for Forum itself—to develop it iteratively.

Appendix A: Forum Project Phase One Work Breakdown Structure

1 Introduction

This appendix describes how the statement of work (OSEHRA Forum Phase One Agreement between Open Source Electronic Health Record Agent, Inc. and VISTA Expertise Network, Attachment A, Statement of Work [SOW]) breaks down into subtasks. Specifically, Task 2 - Development of OSEHRA Forum Phase 1, Deliverable 2 - Working Phase 1 Forum System, described in the SOW as follows:

This system will be capable of demonstrating the features described in Section 2.2 above. The deliverable system must be certified to a minimum of OSEHRA Level 3 (Level 4 preferred), which will mandate the delivery of accompanying documentation & unit tests.

And here's description from Section 2.2 Task 2 - Development of OSEHRA Forum Phase 1:

OSEHRA Forum will provide multiple product support functions for OSEHRA Vista, primarily in the area of patch distribution & release management. Phase One will be a usable but limited system that will:

- 1. Implement approved version, patch, & sequence number conventions*
- 2. Provide two patch streams, one for OSEHRA Vista & the other for FOIA Vista*
 - 2.1. Implement Mailman connectivity*
 - 2.2. Provide a mechanism to ingest, review, revise, & release patches to the appropriate patch stream*
- 3. Manage interaction between Git, Gerrit, & OSEHRA Forum*

The document that follows is drawn from the notes the VISTA Expertise Network built during the proposal stage of this project. It also lists optional tasks in an agile backlog.

2 Work Breakdown Structure

Refno	WBS Item
	Version, Patch, & Sequence Numbers
1.00	Longer Patch Numbers
1.01	Update KIDS Data Dictionaries (DDs) to Support Longer Patch Numbers
1.02	Update KIDS Routines to Support Longer Patch Numbers
1.03	Update Patch Module (PM) DDs to Support Longer Patch Numbers
1.04	Update PM Routines to Support Longer Patch Numbers
1.05	Update Non-routine Components of KIDS & PM to Support Longer Patch Numbers · For Example, Input Templates

2.00	Longer Sequence Numbers
2.01	Update KIDS DDs to Support Longer Sequence Numbers
2.02	Update KIDS Routines to Support Longer Sequence Numbers
2.03	Update PM DDs to Support Longer Sequence Numbers
2.04	Update PM Routines to Support Longer Sequence Numbers
2.05	Update Non-routine Components of KIDS & PM to Support Longer Sequence Numbers · For Example, Input Templates
3.00	Tie Patch Numbers to Numberspacing & to Patch Stream
3.01	New DHCP Patch Stream File
3.02	New Name Field in DHCP Patch Stream
3.03	New Numberspace Field in DHCP Patch Stream
3.04	Input Transform on Numberspace to Restrict Use of 1 to Forum.VA.Gov
3.05	New Type Field (Primary Or Secondary) to Characterize Patch Streams · Only One Can Be Primary
3.06	New A1AE PATCH STREAM Compound Cross-Reference
3.07	Load A1AE PATCH STREAM into Local Array A1AESTRM
3.08	Set Top Node of A1AESTRM to the Primary Patch Stream
3.09	Make Patch Module Honor A1AESTRM
3.10	New Patch Stream Field in DHCP Patch File to Point to the DHCP Patch Stream
3.11	Make Patch Stream Field a Conditional Identifier (If Site Has More Than One)
	Patch Streams for OSEHRA VistA & FOIA VistA
	Mailman
4.00	Mailman Connectivity
4.01	Get VistA Infrastructure Configured & Operational
4.02	Configure VistA Environment as FORUM.OSEHRA.ORG
4.03	Configure DNS Externally to Recognize FORUM.OSEHRA.ORG
4.04	Configure GW.OSEHRA.ORG & Postfix to Send to Internet
4.05	Resolve Firewall Issues Blocking Outbound Mailman

4.06	Configure Mailman in XINETD to Receive E-Mail from Internet
4.07	Configure Q-PATCH.OSEHRA.ORG to Receive Patches from Developers
4.08	Resolve Disruptions Caused by Rackspace Port Checker
	Ingest, Review, Revise, & Release Patches
5.00	Import FOIA Patches
5.01	New Server Option A1AE LOAD RELEASED PATCH [Load Released Patch Into Patch Module] to Load VA Patches into Forum
5.02	New Routine to Load a Released VA Patch E-Mail into PM
5.03	Write Code to Process Message Header
5.04	Write Code to Process the Patch Description (Boilerplate—Parsed Fields Plus Word Processing)
5.05	Write the Code to Process the KIDS Distribution (Structured, so Should be Decomposed & Filed as Fields)
5.06	Create Bulletin A1AE LOAD RELEASED PATCH [New Patch Loaded into Patch Module]
5.07	Write a Subroutine to Email the Bulletin A1AE LOAD RELEASED PATCH to the Development Group
5.08	Test the Server Option A1AE LOAD RELEASED PATCH [Load Released Patch into Patch Module]
5.09	Integrate Cameron's Code to Load Released VA Patch from Host Files
5.10	Integrate Cameron's Code to Release a Patch as Host Files
5.11	Decompose Compound Builds in a Host-File Patch Into Separate PM Patches
5.12	Compose Patches into a Compound Host-File Patch
5.13	Write & Integrate Code to Create a Properly Named & Numbered Copy of an Ingested Patch for Each Patch Stream, Based on A1AESTRM · Send the Bulletin A1AE LOAD RELEASED PATCH for Each Copy
5.14	Modify the Before Checksum Handling to Support Multiple Patch Streams · This is Likely to Consist of Many Subtasks
5.15	Recalculate the Associated Patches to Refer to Derived Patches, Not Original Patches
5.16	Handle Cross-Package Associated Patches During System Initialization
5.17	Add Field "Update To Patch"
5.18	Initialize Patch Module with All Past Patches

6.00	New Statuses, Rules, and Fields for Secondary Development
6.01	Add New Statuses to PM
6.02	Add New Status “In Review” to PM
6.03	Add New Rules to Go with “In Review” Status
6.04	Add Rule that Users Cannot Change a Patch’s Status to “In Review” Manually
6.05	Add Rule that A1AE LOAD RELEASED PATCH Sets the Status of the Loaded Patch to “In Review” as Its Last Step
6.06	Add Fields to Support Flagging Redacted Patches and Describing the Redaction; Add Report and Other-Option Support for Displaying these New Fields.
6.07	Add Rule that an “In Review” Patch may be Changed to Either “Secondary Development”, “Secondary Completion”, or “Not For Secondary Release”
6.08	Add New Status “Secondary Development”
6.09	Add Rule that a “Secondary Development” Patch may be Changed to Either “Secondary Completion” or “Not For Secondary Release”
6.10	Add New Status “Secondary Completion”
6.11	Add Rule that a “Secondary Completion” Patch may be Changed to Either “Secondary Release” or “Not For Secondary Release”
6.12	Add New Status “Secondary Release”
6.13	Add Rule that when Status Set to “Secondary Release” the Patch Email Message is Generated and Sent to All Subscribers
6.14	Add New Status “Not For Secondary Release”
6.15	Add New Field to Explain Why “Not For Secondary Release”
6.16	Add New Field to Explain What was Done as Part of “Secondary Development”
6.17	Add New Field Type to Distinguish Original Patches from Derived Ones
6.18	Add New Field Original Patch to Point Back to a Derived Patch’s Original
6.19	Make Type a Conditional Identifier
6.20	Make Original Patch a Conditional Identifier
6.21	Add Patch Stream, Type, & Original Patch to Patch Header
7.00	Modify Templates & Options to Support Secondary Management of a Patch
7.01	Input Template A1AE ADD/EDIT PATCHES
7.02	Sort Template A1AE FULL SUMMARY SORT
7.03	Sort Template A1AE FULL SUMMARY BY DATE

7.04	Sort Template A1AE PATCH COMPL/COMM SORT
7.05	Sort Template A1AE SEQ REV SAVE
7.06	Sort Template A1AE SEQ REV SUMMARY SORT
7.07	Sort Template A1AE SEQ SAVE
7.08	Sort Template A1AE SEQ SUMMARY SORT
7.09	Sort Template A1AE STANDARD SORT
7.10	Sort Template A1AE SUMMARY SORT
7.11	Sort Template A1AEZPKE
7.12	Print Template A1AE FULL SUMMARY BY DATE
7.13	Print Template A1AE PATCH COMPL/COMMENT RPT
7.14	Print Template A1AE PATCH COMPLIANCE PRT
7.15	Print Template A1AE PATCH SUMMARY
7.16	Print Template A1AE STANDARD HEADER
7.17	Print Template A1AE STANDARD PRINT
7.18	Print Template A1AE VERIFIED PATCH SUMMARY
7.19	Print Template A1AEZPKE
7.20	Option A1AE PHADD [Add a Patch]
7.21	Option A1AE PHEDIT [Edit a Patch]
7.22	Option A1AE PHVER [Release a Patch (And/or Edit Internal Comments)]
7.23	Option A1AE PRT SUM DT ORG [Released Patch Summary Report by Date]
7.24	New Option A1AE PRTCOMDETD [Released Patches by Date Detailed Report]
7.25	New Option A1AE PRTCOMPHDTE [Completed Patch Summary Report by Patch Stream]
7.26	New Option A1AE PRTCOMSUMPS [Released Patch Summary Report by Patch Stream]
7.27	New Option A1AE PRTDETD ORG [Detailed Report of Released Patches by Patch Stream]
7.28	Option A1AE PRTPHA [All Released Patches for a Package, Detailed]
7.29	New Options for Reports on Patches with the New Statuses
7.30	New Menu for Options that Help Manage a Patch Stream
7.31	New Option to Show Patch Relationships—Dependencies & Derivations
7.32	Reorg Menus to Make Them Coherent

7.33	Other Package Elements
	Git, Gerrit, & Forum
8.00	When Patches Change to the Right Statuses, Export Patch to Gerrit
8.01	New Protocol for Gerrit Export
9.00	Make PM Update Gerrit at All the Right Times
9.01	Change PM Options to Trigger that Protocol When Patch is Edited
10.00	Improve Integration with Github · Initial Flat-File Formats for KIDS Components
10.01	Work Out Initial Formats for All Twenty-Four or so KIDS Components
10.02	Write Initial Exporter
10.03	Integrate into Export Protocol: Send Individual Components Along with Patch
	Agile Backlog · More Git, Gerrit, & Forum
11.00	Make PM Externally Responsive to Gerrit
11.01	New Remotely Triggered Procedure Get Patch
11.02	New Remotely Triggered Procedure Send Patch
12.00	Create the Actual Interface for Gerrit to Use
12.01	New Web Service to Invoke the Get Patch Remote Procedure
12.02	New Web Service to Invoke the Send Patch Remote Procedure
13.00	Improve Integration with Github · Initial Flat-File Formats for KIDS Components
13.01	Re-express Initial Formats as a Universal Format for Any Component
13.02	Write Universal Importer
13.03	Write Universal Exporter
13.04	Integrate into Export Protocol: Send Individual Components Along with Patch

3 Conclusion

The goal of Phase One is an initial operating capability (IOC); it will include manual steps and unpolished software, but it will be correct, unit tested, and operational. This restraint defining Phase One is wise, because the changed data-architecture of the IOC has ripple effects throughout the Patch Module's options, as shown above.

Some WBS items will expand into further subtasks. For example, calculation of before-checksums for patches must expand because the original Patch Module was never designed to manage multiple patch streams from the same VistA environment. We should update this Phase 1 WBS at each week's sprint planning meeting based on what we learn from the previous sprint's work.

Appendix B: Forum Project Phase One Milestones

1 Purpose

This appendix, derived from the project contract's statement of work, identifies the contract requirements and corresponding project milestones. It also identifies the technical items derived from the WBS and narrative, and inserts them into the appropriate milestones.

2 Background

The Department of Veterans Affairs uses a system called Forum for patch distribution and related services. OSEHRA intends to implement an improved version of Forum to provide services to the open-source community in support of OSEHRA VistA, a community-driven distribution of the VistA EHR. This project will be undertaken in three phases to allow Phase One work to begin while the specifications for later phases are defined. This approach will also allow later phases to incorporate lessons learned from the community use of Phase One functionality.

3 Tasks

This statement of work contains three primary tasks as follows:

3.1 Task 1 · Consulting on OSEHRA VistA Product Definition

This task includes time from Rick Marshall in support of product-definition efforts on the OSEHRA VistA distribution. Work will be in support of Mike Henderson, and will include, but not be limited to identification of package functionality, dependencies, functional relationships, and commonality among major VistA distributions. These efforts will be used to improve the quality and specificity of the product description, and potentially populate a DBA Database instance for OSEHRA VistA that will serve as a configuration management baseline.

3.2 Task 2 · Development of OSEHRA Forum Phase One

OSEHRA Forum will provide multiple product-support functions for OSEHRA VistA, primarily in the area of patch distribution and release management. Phase One will be a usable but limited system that will:

- 1 · implement approved version, patch, and sequence number conventions;
- 2 · provide two patch streams, one for OSEHRA VistA, the other for FOIA VistA, including:
 - 2.1 · implementing Mailman connectivity and
 - 2.2 · providing a mechanism to ingest, review, revise, and release patches to the appropriate patch stream; and
- 3 · manage interaction between Git, Gerrit, and OSEHRA Forum.

3.3 Task 3 · System Administration of OSEHRA Forum Phase One

VEN will provide system administration on OSEHRA-provided virtual machine(s), including configuration, updates, GT.M backup, GT.M journaling, and a secondary system with failover capabilities.

4 Points of Contact (POC)

The OSEHRA POC for all performance under this statement of work will be Mike Henderson, director of open-source product management.

VEN's point of contact for task one will be Rick Marshall. VEN's POC/Project Manager for task two will be Sam Habel. VEN's POC for task three will be Larry Landis.

For tasks two and three, VEN will update OSEHRA via a weekly status call, and summarize that call in a status e-mail. The e-mail will include any action items for either party that are required to facilitate timely project completion. Any planned downtime or upgrades, as well as any system incidents, will be reported on the call and summarized in the e-mail.

5 Deliverables

The following deliverables will be provided under this statement of work:

5.1 · Task one

o deliverable one · Technical consulting assistance as requested

5.2 · Task two

o deliverable one · Documentation · Configuration Nomenclature Document. This document will describe in detail the scheme and nomenclature to be used in assigning version, patch, and sequence numbers to OSEHRA VistA. This document will be developed in collaboration with OSEHRA · Initial Design Document. This document will describe the Phase One Forum design, including user interface and planned unit tests.

o deliverable two · Working Phase-one Forum system · This system will be capable of demonstrating the features described above in Section 2: Tasks. The deliverable system must be certified to a minimum of OSEHRA Level 3 (Level 4 preferred), which will mandate the delivery of accompanying documentation and unit tests.

5.3 · Task three

o deliverable one · Documentation · Additional System Installation/Administration Document, which will provide details of hosting and system administration. This document will be delivered directly to the OSEHRA POC, and not made publicly accessible.

o deliverable two · Phase-one Forum system administration · This system will be administered to the standard described in Section 2.3 above. The quality of system administration must essentially be production-ready, as described in the accompanying documentation.

6 Schedule

Task one will continue coincident with task two. The initial schedule for task two will be from 1 January through 31 March 2013. Task three will continue coincident with tasks one and two.

7 Milestones

The components of the three milestones are as follows:

7.1 • Milestone One • Initial Forum System

task one deliverable one · consulting on OSEHRA VistA product definition

task two deliverable two · an initial Forum system · WBS item (d) Mailman Connectivity · Patch Module setup, configuration · initial unit tests · life cycle component one: FOIA VistA patches (secondary development, rereleases) · life cycle component five: new submissions to OSEHRA from the community (primary development)

task three deliverable two · system support for an initial Forum system · initial Forum system administration

7.2 • Milestone Two • Enhanced Forum System

task one deliverable one · consulting on OSEHRA VistA product definition

task two deliverable one · documentation · Configuration Nomenclature Document (which will describe in detail the scheme and nomenclature to be used in assigning version, patch, and sequence numbers to OSEHRA VistA) · Initial Design Document (which will describe the Phase One Forum design, including user interface and planned unit tests)

task two deliverable two · an enhanced Forum system · WBS items (a) Longer Patch Numbers, (b) Longer Sequence Numbers, (c) Tie Patch Numbers to Numberspacing & to Patch Stream, and (e) Import FOIA Patches · enhanced unit tests · life cycle component two: OSEHRA VistA patches from VA (secondary development, rereleases) · life cycle component three · OSEHRA VistA patches from community (secondary development, extensions & modifications)

task three deliverable one · documentation: Additional System Installation/Administration Document (which will provide details of hosting and system administration)

task three deliverable two · enhanced Forum system administration

7.3 • Milestone Three • Initial Operating Capability (IOC)

task one deliverable one · consulting on OSEHRA VistA product definition

task two deliverable two · a working IOC Forum system · WBS items (f) New Statuses, Rules, and Fields for Secondary Development, (g) Modify Templates & Options to Support Secondary Management of a Patch, (h) When Patches Change to the Right Statuses Export Patch to Gerrit, (i) Make PM Update Gerrit at All the Right Times, and three parts of (l) Improve Integration with Github & Initial Flat-file Formats for KIDS Components · WBS Agile Backlog items if

time allows · IOC unit tests · life cycle component four: OSEHRA VistA replacements for FOIA
VistA patches (secondary development, modifications)

task three deliverable two · IOC Forum system administration

Appendix C: Forum Configuration Nomenclature

1 Purpose

This appendix surveys the scheme and nomenclature to be used in assigning version, patch, and sequence numbers to OSEHRA VistA.

2 Executive Summary

How do you answer the question “What version of VistA are you running?” In three ways:

- 1) You name your VistA dialect, which identifies the gold codebase your version of VistA is based on, and which also identifies the stream of upgrades you follow in evolving your VistA system.
- 2) If your VistA system is brand new, you name the VistA snapshot it was created from; otherwise, you name the most recent service pack you applied.
- 3) Optionally, for more detail you can describe the version and patch level of one or more of the VistA applications you are running; the easiest way to do this is to consult your snapshot’s or most recent service pack’s manifest.

3 VistA Dialect · Software Life Cycle · Upgrade Stream

VistA has a complex lineage going back thirty-seven years across many organizations (see The Lineage of VistA chart on the next page).

3.1 VistA Dialect

When a VistA adopter produces a unique, stable version of VistA, upgrades it consistently over time, and continually releases those upgrades to the rest of the VistA community in a complete stream of updates, we call what they have created a VistA dialect. When someone asks “What version of VistA are you running,” often all they want to know is which dialect you are using.

THE LINEAGE OF VISTA

FRIDAY, 24 JANUARY 2014

Defunct dialects
Troubled dialects
Proprietary dialects
Incomplete dialects
Open & Healthy Dialects



© 2014, Carol Monahan &
Frederick D. S. Marshall

VISTA Expertise Network's *The Lineage of VISTA* is licensed under the *Creative Commons Attribution-Noncommercial-Share Alike 3.0 United States License* (<http://creativecommons.org/licenses/by-nc-sa/3.0/us/>).

Some VistA dialects have been replaced, some are defunct, some are so unsupported as to be problematic to use, and still others are too incomplete to use, but six or so are production ready.

OSEHRA is creating a seventh—OSEHRA VistA. Unlike the other dialects, OSEHRA VistA is not intended to replace the other dialects but to be a converged common core code base from which the other dialects can be derived, to improve the VistA community's productivity by increasing software sharing and decreasing reinvention of the wheel.

3.2 What Your VistA Dialect Tells Us

We care about the dialect as an answer to the question “What version of VistA are you running?” because your dialect tells us four crucial things.

First, it tells us which constellation of VistA applications you are running. Although there is a large overlap among dialects, the differences in functionality are easiest to summarize in terms of which applications you have or do not have, and this is determined mostly by your choice of dialect—and of course the actual services your hospital provides and needs automation for. If we know your VistA dialect, we know the basic pool of applications you are drawing from.

Second, it tells us about your primary source of innovation, because different dialects are primarily developed by different organizations. When you adopt a VistA dialect, you become partly dependent on its development organization, at worst wholly dependent, at best interdependent. Much like life itself, VistA is far, far too complex to go it alone; dependency and interdependency are necessary parts of adopting and developing an EHR. Your fate is therefore determined at least in part by your relationships, especially by who you choose to be your primary development organization, that is, by who primarily develops your chosen VistA dialect.

Third, it tells us about your rate of innovation, the pace at which your applications improve and new applications are added. Different primary development organizations follow different software life cycles, which results in different rates of productivity. More specifically, it also tells us about the rates of innovation for different applications, since different organizations have different priorities; an organization with an otherwise lethargic development pace might be unusually good with one or more applications you care about.

Fourth, it tells us about your ability to exchange data and software with other VistA adopters, a crucial consideration.

In terms of data, the world is leaving behind the era of medical feudalism, in which hospitals owned patient and treatment data, and moving into a new era of medical-data interchange, in which patients own their medical and treatment information. Despite all the past, present, and future standards for medical-data exchange, it remains true that systems with more compatible data structures have an easier time representing each other's data. If we know your VistA dialect, we know its pedigree, so we know how compatible its data structures are with our own, and thus how effectively we can exchange data.

In terms of software, since we cannot go it alone with VistA, we have to divide up the development work. Different organizations focus on different VistA applications. The more easily we can exchange software, the more easily we can benefit from each other's innovations and accelerate our software's evolution. If we know your dialect, then we know your software architecture and its compatibility with our own, so we know how easily we can exchange software, and thus how much we can help each other out.

3.3 Software Life Cycle

These four things your VistA dialect tells us are important, but they become more important when viewed through the lenses that VistA experts use. The pace of change in medical science and practice is staggering, and it destroys the validity of the ways most of us go about shopping for EHRs. In such a complex field, the hardest principle for non-experts to grasp is the extent to which the software's current capabilities do not matter as much as its future capabilities. If it is wrong today, it can become right tomorrow; if it is right today, it can become wrong tomorrow. Whether it is right or wrong tomorrow is determined not by the software itself but by the pace, direction, and quality of innovation; these are the pre-eminent concerns a VistA expert has when evaluating VistA dialects.

Fortunately, this other system of evaluation is also captured by knowing which VistA dialect you have. The primary developers of the VistA dialects are different organizations, each of which follows its own software-life cycle practices, which in turn results in higher or lower productivity, more or less collaborative relationships with their adopters, and above all a greater or lesser degree of responsiveness to the shifting understanding of their medical experts. That latter quality determines how evidence-based a development organization's software life cycle is, and thus how reality-based its resulting software will be. We do not need to know in advance where specifically IHS, for example, will innovate in their VistA dialect, so long as we know that innovation will be driven above all by their medical users.

3.4 Upgrade Stream

VistA innovation comes to us in the form of a stream of upgrades, as will be described in the rest of this paper. What matters foremost about your upgrade stream is that (a) most of your future innovations will come from it, (b) your upgrade stream is specific to your VistA dialect, (c) your upgrade stream is created by your primary development organization, and (d) your upgrade stream is creating using the software life cycle selected by your primary development organization. Thus, knowing your VistA dialect tells us a great deal about your future.

3.5 New Dialects

When a new VistA dialect is created (rightly, a rare event), its adopters cannot describe their version of VistA just by naming it, because no one will have heard of it or know what it means. It must be explained in terms of existing dialects. The dialect used as the original code base should be named, its main differences from that code base should be described, any borrowings from other dialects should be mentioned to help establish its relationship to them, the primary development organization should be named, that organization's software life cycle should be described, the source of the upgrade stream should be named so we know where to go for updates, and any coherent plans for the future of this dialect should be described.

For example, OSEHRA VistA was created to solve a problem that no existing VistA dialect can solve but that they all need solved—reunification of the VistA dialects, to be a repository for the community's code-convergence efforts. It is derived from FOIA VistA, mixed with contributions from outside VA aimed at improving the core codebase and converging our dialects as closely as practically feasible. It borrows from every healthy dialect, the primary development organization is the OSEHRA community collaborating within OSEHRA, and the sources of the upgrade stream are the OSEHRA VistA Github repository for snapshots and OSEHRA Forum for patches (see next section). Plans for the future are driven by the OSEHRA community within its work groups, which meet and discuss plans openly; everyone is invited to participate.

Now that OSEHRA VistA exists, its adopters can use it to help answer the question “What version of VistA are you running?” We can say “OSEHRA VistA” or more typically in days to come we can say we are running XXX VistA, “derived from OSEHRA VistA.” Likewise, an adopter might say they are running “IHS RPMS” or “VA VistA,” and these are equally powerful summations of a great deal of information about the version of VistA they are running.

4 VistA Version • Snapshot or Service Pack

Knowing the dialect tells us a lot, and it is often all the questioner wants to know, but sometimes it is not enough. Dialects change over time. Unless we know where in time you are within your dialect, we still do not know which version of VistA you are running. When we need to know more than the dialect, usually we want to know the snapshot or service pack. Here’s why.

VistA is among the most complex works of software ever created; it has to be, because medicine is among the most complex of human endeavors. It is too complex to install the way most software is installed, and it is too complex to upgrade the way most software is upgraded.

4.1 VistA Snapshot

VistA is installed by cloning a living VistA system. The same methods we use to keep production VistA systems up to date, we also use to keep each VistA dialect’s reference system up to date. Periodically, we make a copy of that system, what we call a snapshot of it, and that copy is published. In the case of the U.S. Department of Veterans Affairs (VA) and Indian Health Service (IHS), which are federal agencies, the creation and publication of these VistA snapshots is a function of their duties under the Freedom of Information Act (FOIA). Other organizations produce snapshots to support their commercial services or charitable missions.

Each organization produces snapshots on its own rhythm, some monthly, some quarterly, some inconsistently. Time is the right way to identify these snapshots, because VistA is upgraded over time; knowing when a snapshot was taken gives us a pretty good idea which upgrades it includes and which it does not. After knowing the dialect, this is the closest thing there is to knowing what version of VistA you plan to install to set up a new VistA system.

4.2 VistA Service Pack

Unfortunately, VistA cannot be upgraded the same way it is installed. One of the big differences between VistA and most consumer software is that the software and data are not separate in VistA; they are and must be mingled very closely together, intertwined, to allow the complex medical data to drive the execution of the software. Most consumer software is upgraded mainly by replacing the software files with new and improved software files, but to do that to VistA would not only replace the software but also all of the data, erasing all of the patient and other medical information you have so carefully recorded. So once a VistA system has been created, new VistA snapshots are close to useless to that system thereafter; we never use new snapshots to upgrade an existing VistA system.

VistA instead uses a more biological system of upgrades, almost viral, in which small packages of new software and data are delivered to your VistA system using special VistA software, which then installs the update in a way that allows it to properly weave itself into your system. For historical reasons, we call these installation packages “patches,” but they are used as much for delivering enhancements as they are for delivering repairs. VistA’s system of patching is

intricate, precise, and effective, but it is too complex to let us give a short answer to the question “What version of VistA are you running?” For this reason, outside VA most primary development organizations periodically bundle up all their recent patches into service packs, to simplify their release and installation.

Typically, a service pack will include all the patches since the previous service pack that have passed the organization’s certification processes. Whenever the service pack covers a period during which the patches and/or versions of a major multi-application development initiative were released, it is important that the service pack include all of the patches and releases that together deliver that upgrade. We never want to split a major release of interdependent patches across two service packs, because we must ensure that a VistA system is architecturally coherent after installing each service pack.

Unlike snapshots, these service packs can be used to upgrade an existing VistA system, and they are, so identifying the most recent service pack you have installed is another very good way to explain which version of VistA you are running.

4.3 Naming Snapshots and Service Packs

Snapshots and service packs are almost always identified by the time they were taken, such as the FOIA VistA September 2009 snapshot. OSEHRA will create snapshots and service packs four times a year, once per quarter, and they will be named accordingly: OSEHRA VistA 2014Q1, OSEHRA VistA 2014Q2, OSEHRA VistA Service Pack 2014Q1, OSEHRA VistA Service Pack 2014Q2, and so on.

Every so often, a primary developer releases emergency patches that should not wait until the next quarterly service pack to be distributed. In such cases, OSEHRA will release emergency service packs whose names are derived from the most recently released quarterly service pack, such as OSEHRA VistA Service Pack 2014Q1A.

So for new or proposed VistA systems, the best answer to “What version of VistA are you running?” is to name the VistA dialect and snapshot; for established VistA systems, it is to name the VistA dialect and most recent service pack.

5 VistA Applications • Packages, Versions, & Patches

These ways of describing your VistA system’s version are clear and concise, and they will do for most casual situations—certainly they are the best for casual conversation or executive summaries—but they are not precise, and they were not always available. Until recent years, we used a system that is far more complex—too complex to be a polite or casual answer to the question “What version of VistA are you running?”—but far more precise. Even today, VistA software engineers and architects can and should begin with the answers described above, but they soon need to drill down to the more detailed answers we have used for thirty-seven years.

Those answers involve describing the version not of your VistA system as a whole but instead of each of the VistA applications you are running on your system. We describe those versions in three ways: (a) by identifying the applications, (b) by listing the current version of each application, and (c) by naming the most recently installed patch to each application.

5.1 Applications

Each VistA application automates part or all of one hospital or clinic service. Just as different clinics and hospitals offer different suites of services, so their VistA systems will run different applications. Listing the applications running within a VistA system tells us a lot about it, and about the adopter it belongs to.

5.2 Versions & Packages

Major upgrades of applications are called versions, and the distributions of application versions are called packages. When an application is new, that first package is called version 1. Thereafter, new versions may be numbered by integers or decimals. A detailed accounting of the version of VistA you are running therefore requires identifying not just the list of applications but the current version of each one running on your system.

With applications like Laboratory that have not had a new version in nineteen years, knowing the version may not tell us much, because all modern VistA systems are running the same version. But with applications like File Manager, which just saw the release of new version 22.2 last year, knowing the version tells us a great deal about the differences between VistA systems.

Some applications have client software that is versioned independently of the main application's version. For example, VistA's flagship application, the Clinical Patient Record System (CPRS), designed for doctors and other healthcare professionals, has not had a new version in over a decade; it remains stuck at version 3, while its client is approaching version 30. Therefore, we also need to know the version of the client, if there is one, not just the main application.

5.3 Patches

As described above, most upgrades to VistA applications are smaller than complete new versions; most take the form of "patches," small, frequent, incremental upgrades to the application. Patches are often developed in parallel, but they are released sequentially, resulting in a stream of upgrades, the patch stream.

Since patches are almost always installed in order, knowing the most recently installed patch for each application gives us a very precise understanding of the version of VistA running on your system. Therefore, the most precise way to answer the question "What version of VistA are you running?" is to list all the VistA applications you are running, the version number of each, and the latest installed patch for each application.

5.4 Patch Names

Each version of each application has its own patch stream. That is why each patch is identified by (1) the application, (2) the version, and (3) a numeric ID. To this we add a fourth piece of information, (4) the sequence number, which tells us in which order to install the patches within its patch stream. Patches from VA VistA and FOIA VistA follow this pattern, with names like OR*3.0*283 SEQ #272 or DI*22.0*169 SEQ #149.

OSEHRA VistA patches are different, as are patches from any other VistA dialect if they are released from OSEHRA's new Forum system. These patches add a fifth piece of information to patch names: (5) the name of the VistA dialect. To help avoid confusion with VA patches, we also numberspace the patch's numeric ID. Each VistA dialect has its own range of numbers used to assign this value; OSEHRA VistA numeric IDs start with 10,000. So, for example, the first

patch to be released for OSEHRA VistA's File Manager version 22.2 will be called "OSEHRA VistA DI*22.2*10001 SEQ #1."

5.5 Distribution Names

Patches are distributed by e-mail message or by a pair of host files. Message subjects and file names are each derived from the name of the patch they distribute.

Until now, a patch like LR*5.2*413 SEQ #326, if distributed as an e-mail message would have a subject like "Released LR*5.2*413 SEQ #326," if distributed as host files would have names like "LR-5P2_SEQ-326_PAT-413.TXT" and "LR-5P2_SEQ-326_PAT-413.KID." The occasional emergency patch is different only in the e-mail subject line, which would read, for example, "EMERGENCY Released LR*5.2*403 SEQ #319." Likewise test patches, such as "TEST LR*5.2*438 TEST v1."

Distributions from OSEHRA's new Forum system will be just a little different. For example, patch "OSEHRA VistA DI*22.2*10004 SEQ #3," if distributed as an e-mail message would have a subject of "[OSEHRA VistA] Released DI*22.2*10004 SEQ #3," if distributed as host files would have names of "DI-22P2_OSEHRA-VistA_SEQ-3_PAT-10004.TXT" and "DI-22P2_OSEHRA-VistA_SEQ-3_PAT-10004.KID."

6 Mappings · Streams, Secondary Development, Manifests

When we move beyond dialect, snapshot, and service pack, when we need more information, part of what we need is a mapping between these things and patches, and among the patches themselves. It can be a complicated dance, so a little mapping can help simplify things for us by connecting the dots.

6.1 Mapping Patches to Software Streams

Every patch needs to be mapped to its software stream, to avoid any possible confusion about which patches a VistA adopter should or should not install on their system. This is done (a) in the patch name and distribution subject or file name, as described above, (b) in each patch's descriptive text, where the VistA dialect it applies to is named, and (c) in the database itself, to support the distribution and installation process.

6.2 Mapping Original & Derived Software Streams

Some software streams, like VA VistA, contain exclusively primary development, meaning the source of every patch in the stream is the primary developers of each application. This is called an original software stream.

Others, like IHS RPMS, contain a mix of primary development and secondary development, in which the source of many of the patches is some original software stream, but the source of others consists of additional development. This is called a derived software stream. In the case of IHS RPMS, the original stream on which it is based is VA VistA. When the additional development consists of new applications, it is called primary development; when it consists of modifications of or extensions to the patches in the original software stream, it is called secondary development.

The resulting IHS RPMS software stream is more complex than the VA VistA stream upon which it is based, because it contains primary development from VA developers, new primary

development from IHS developers, and secondary development from IHS developers, all carefully interleaved to create a new stream of upgrades. This interleaving usually requires renaming and resequencing the patches—even the unchanged ones—to make room for the new patches inserted between them.

For this to work, the relationships between original and derived software streams must be mapped out. Mainly this is done in the database itself, to support distribution and installation.

6.3 Mapping Original & Derived Patches

Just as the streams need to be mapped, so do the individual patches themselves, so we are clear on the relationship between them. Patches derived from an original stream by renaming and resequencing but otherwise not editing them need to be mapped back to their original patches. Patches derived from an original stream by doing secondary development upon them likewise need to be mapped back, though their more distant relationship needs to be distinguished from mere renaming and resequencing. New primary-development patches that need to be inserted between patches that were adjacent in the original stream need to identify their dependencies upon the original patches, to support proper sequencing.

All three forms of mapping between patches in the original software stream and patches in the derived stream is done (a) in the patch description's header text and (b) in the database itself. This ensures that both human beings and the software itself can trace the provenance of all patches in the derived software stream.

6.4 Manifest Section 3 • Application Versions

To map the high-level description of a VistA version in terms of dialects, snapshots, and service packs to the detailed description in terms of applications, versions, and patches, we need to ensure that each snapshot and service pack includes a version manifest. Each manifest will include three sections, which we will describe in reverse order.

The third section describes the state of the applications. It is a list of all of the VistA applications included in the snapshot or service pack, the version of each application (and of its client, if any and if different), and the latest patch installed for each application. It gives the detailed answer to the question “What version of VistA are you running?” as described above. This list can be automatically generated by the Forum system when a snapshot or service pack is created.

6.5 Manifest Section 2 • Deltas

The second section describes the changes (deltas) in this snapshot or service pack compared to the previous one. It is a list of the new applications, new versions, and new patches that together created this version of VistA from the previous version. For all patches in a derived software stream, it also lists the corresponding original patch, to assist technical experts in mapping this snapshot or service pack back to the version of the original software stream upon which it is based. For a derived software stream, it also lists any original patches excluded from release in this software stream, as sometimes happens. It is indispensable to VistA adopters preparing to apply a service pack or comparing two adjacent snapshots, because it details the differences between them. This list too can be automatically generated by the Forum system when a snapshot or service pack is created.

6.6 Manifest Section 1 • Highlights

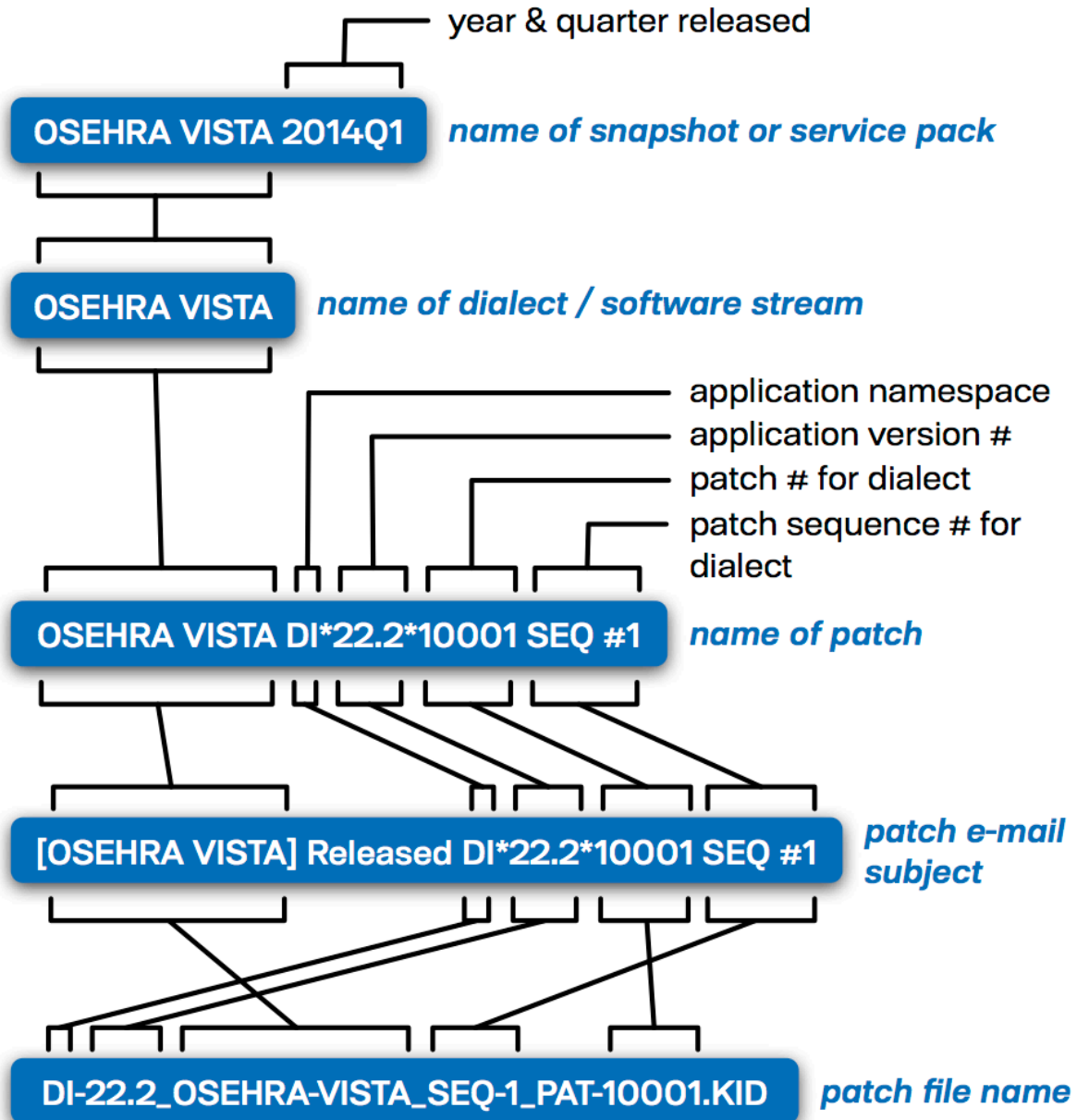
The first section is an executive summary that picks out just the most essential or interesting versions and deltas included in this snapshot or service pack. It should always include the version numbers of indicator packages, such as the CPRS client, which drives a lot of other VistA development, or File Manager, which enables it. It answers the related question “Why am I most likely to care about this version of VistA?” Most people who read a snapshot or service pack’s manifest will only read this first section, but VistA software engineers and superusers will need to study all three sections in the performance of their jobs. This short list must be generated manually by experts who understand the significance of the other two sections.

7 Conclusion

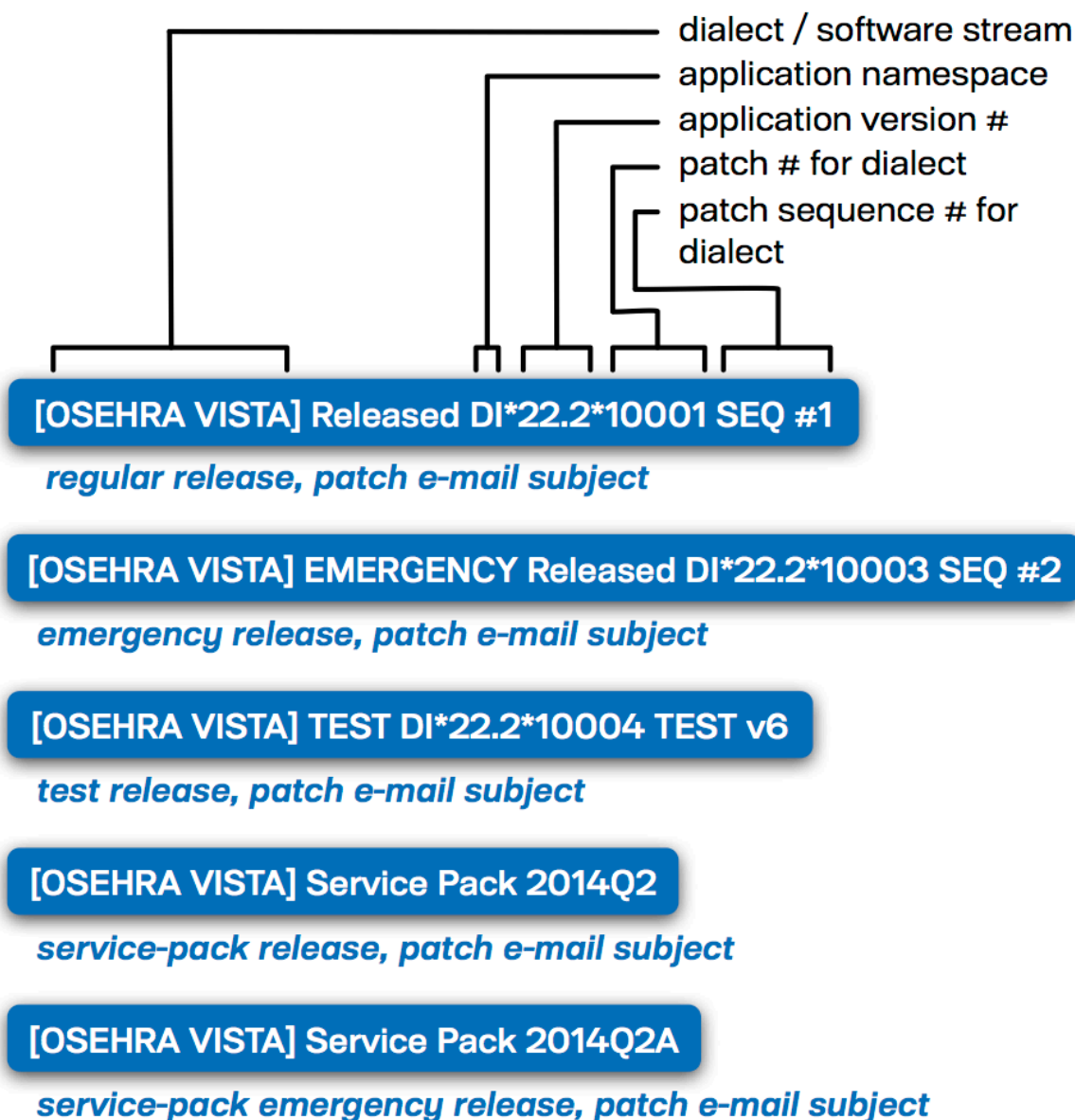
The scheme and nomenclature to be used in assigning version, patch, and sequence numbers to OSEHRA VistA, described in this paper, is an extension of the systems currently in use in VA, in IHS, and outside the U.S. federal government. The main differences are that this system (a) makes it easier to summarize your VistA system’s version, (b) allows you to map that summary to a precise reckoning of the details of your VistA version, and (c) involves new automated support being created as part of OSEHRA’s Forum open-source working group.

We encourage the OSEHRA community to master and adopt this new, extended nomenclature to make it easier for us all to talk about our VistA systems with each other.

8 Basic Patch Nomenclature · Names, Subjects, Files



9 The Variety of Patch E-mail Subject Formats



10 Patch Listings

record #	name	stream	subject	status	developer
80290	PRCA*4.5*10261	[OV]	FIX <NOLINE> ERRORS	IN TH	
	derived from [FV]PRCA*4.5*261				

Every patch stream has an abbreviation, such as OV for OSEHRA VistA, or FV for FOIA VistA, used in condensed displays like patch listings.