

VistA Services Assembler (VSA) Phase 2



System Design Document (SDD)

Software Version 1.0.0

September 2014



**Department of Veterans Affairs (VA)
Office of Information and Technology (OI&T)
Product Development (PD)**




Revision History


Table 1. Document Revision History



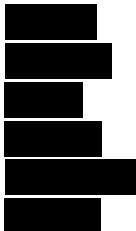
Date	Document Version	Software Version	Description	Author
09/11/2014	1.0	1.0.0.	<p>Tech Edits:</p> <ul style="list-style-type: none"> Baselined document as final for Increment 1. All edits and comments accepted or removed. <p>(NOTE: This document is considered to be a “living” document as VSA development and document reviews continue, so changes may be added in the future.)</p> <ul style="list-style-type: none"> Removed watermark in preparation for electronic signatures. Created accompanying PDF version of this document for electronic signatures. 	[REDACTED]
09/10/2014	0.94	1.0.0	<p>Tech Edits:</p> <ul style="list-style-type: none"> Spelled out first occurrence of acronyms in each chapter. Corrected or removed broken links. Updated Figure 14, Figure 15, Figure 16, Figure 25, Figure 26, and Figure 27. Updated the following tables/sections to make sure the content is a statement of proposed functionality rather than a requirement: Table 5, Sections 2.5.2, 2.5.3, 2.5.4: Table 13, and 2.5.5. Updated Table 32. Updated Table 34. Updated S. Oster signature block in Section 10. 	[REDACTED]
09/05/2014	0.93	1.0.0	<p>Tech Edits:</p> <ul style="list-style-type: none"> Added a list of components 	[REDACTED]

Date	Document Version	Software Version	Description	Author
			<p>distributed with VSA to Section 2.1 (moved from Section 2.5.7).</p> <ul style="list-style-type: none"> Updated text in Section 3.1.1. Also, added Figure 7. Updated Sections 3.1.3.1 and 3.1.3.2. Added Figure 10, Figure 11, and Figure 12. In Section 3.2.1, replaced Figure 13 with reference jump back to Figure 6. Added spreadsheet/link for Terremark development/test servers in Section 3.3.4.1. Updated JAX-RS and JAX-WS in Table 7. Added reference to Enterprise Systems Engineering (ESE) SharePoint site for SEDR VSA-Phase 2 (P2) Excel spreadsheet containing VSA environment information in Section 3.3.4.1 and 6.2.1.1.3. Replaced “Michael P. Clarke” name with “Wilma J. Back” as new SDE representative in Section 10. Updated Figure 14, Figure 15, Figure 16, Figure 25, Figure 26, and Figure 27; based on new footers added to VSA Wizard screens. These will be updated in a subsequent SDD version. Updated Section 4. 	
08/18/2014	0.92	1.0.0	<p>Tech Edits:</p> <ul style="list-style-type: none"> Updated VistA software references in Section 2.4.1, 3.3, 4.2.1, 6.2, and 6.2.1.1.3. Updated Figure 4 and Figure 5. 	

Revision History

Date	Document Version	Software Version	Description	Author
			<ul style="list-style-type: none"> Need to update Figure 14, Figure 15, Figure 16, Figure 25, Figure 26, and Figure 27; based on new footers added to VSA Wizard screens. These will be updated in a subsequent SDD version. Changed references from "Oracle® WebLogic Server 12c (Release 12.1.2)" to "Oracle® WebLogic Server 12c (Release 12.x)." 	
08/13/2014	0.91	1.0.0	Tech Edits: Updated signature block for the Integration Architect, Terry Luedtke, in Table 9 and Section 10 .	
08/07/2014	0.9	1.0.0	Tech Edits: <ul style="list-style-type: none"> Updated Figure 4 and added Figure 5, based on feedback from J. Garcia. Updated acronyms in Table 7. Added document reference to Table 14. Updated Section 4.3: Added Table 28. Updated Section 6.2.1.1.4: Added reference to Table 28. Updated Regional Data Processing Center (RDPC) locations in Table 22. 	
08/05/2014	0.8	1.0.0	Tech Edits: <ul style="list-style-type: none"> Added Section 2.5.2.1 based on feedback from J. Garcia. Updated Sections 2.5.2, 2.5.2.2, and 2.5.2.3 based on feedback from J. Garcia and RSD updates. Deleted the "Capacity," "Degradation Mode," and "Resource Use" sections from Section 2.5.2, as per J. 	

Date	Document Version	Software Version	Description	Author
			<p>Garcia.</p> <ul style="list-style-type: none"> Deleted the “Accuracy” section in Section 2.5.3, as per J. Garcia. Updated Section 2.5.3.2, as per J. Garcia. Updated Section 2.5.4 and Table 13, as per J. Garcia. Updated Section 2.5.5 and Table 14 as per J. Garcia. Updated Sections 2.5.6 and 3.3.1 based on feedback from K. Cox and S. Oster regarding VSA and system criticality. Updated Section 2.5.7, as per J. Garcia. Updated Section 3.1.3, as per J. Garcia. Added introductory text from Section 2.5.2. Updated Table 21, as per J. Garcia. Updated Section 3.1.3.2, as per J. Garcia and S. Oster. Changed all references from “WebLogic Server 11g Release 1 (10.3.6)” to “WebLogic Server 12c (Release 12.1.2).” 	
08/01/2014	0.7	1.0.0	<p>Tech Edits:</p> <ul style="list-style-type: none"> Changed “demonstrate” to “automate” in Sections 1.3, 2.4.1, and 4.4. Changed Sections 2.5.6 and o description of VSA Phase 2 to be “mission critical” based on feedback from S. Oster. Added introductory text to Sections 3.2.4 and 8. Reconfigured and modified content in Section 8. 	



Date	Document Version	Software Version	Description	Author
07/29/2014	0.6	1.0.0	Tech Edits: <ul style="list-style-type: none"> Added Section 2.5.3.3. Added reference note to Sections 2.5.6, 3.1.3.2, and 3.3.1. Added eMI SharePoint site reference link wherever ESB/eMI were mentioned. 	
07/24/2014	0.5	1.0.0	Tech Updates: <ul style="list-style-type: none"> Changed all references from VSA Software Version 2.0 to 1.0 throughout. Made updates or added comments to the following sections based on a review by K. Cox: <ul style="list-style-type: none"> Section 2.5.2.1: Removed first two bullets. Section 2.5.2.3: Added introductory text. Section 3.1.3: Differentiated location information tables between the development phase and implementation phase. See K. Cox's and T. Blom comments. Section 3.3.4: Consolidated multiple notes following figures into one top-level note and corrected text with regard to use of the ESB. Sections 4.3 and 6.2.1.1.4: Added same ESB Note from Section 3.3.4. 	
07/16/2014	0.4	1.0.0	Tech Updates: <ul style="list-style-type: none"> Updated the Wizard, REST, Leverages Existing Code, and Federation Support descriptions in Table 4. <p>Also, moved "Reference</p>	

Date	Document Version	Software Version	Description	Author
			<p>Implementation” entry from Table 5 (Scope Exclusion) to Table 4 (Scope Inclusion).</p> <ul style="list-style-type: none"> • Changed all WebLogic supported version references from “WebLogic J2EE Server 10.3.6” to “WebLogic Server 11g Release 1 (10.3.6).” • Changed references from “Windows 7” to “Windows Server 2008” throughout the document when referring to the VSA Wizard. • Changed all Linux references from “Linux 6.4” to “Red Hat Enterprise Linux release 6.5.” • Updated Consumer Applications entry in Table 6. • Added Representational State Transfer (REST) acronym and service definitions to Table 7. • Updated VSA features in Section 2.1. • Updated Figure 3 and Table 8 caption descriptions to indicate they pertain to the design-time process (using the Wizard to create a service). • Updated Section 2.2 to indicate VSA has Design-time and Run-time components. • Updated entries in Table 8. • Updated Section 2.3. • Replaced Maureen Coyle with Dave Peters and added Kathleen L Frisbee and [REDACTED] Davis in Table 9 based on feedback from S. Oster. • Updated Table 10. • Updated Table 11. • Updated the following 	

Date	Document Version	Software Version	Description	Author
			<p>sections based on feedback from E Laurel:</p> <ul style="list-style-type: none"> ○ Section 2.5.1. ○ Section 2.5.2. ○ Section 2.5.3. ○ Section 2.5.4. ○ Section 2.5.5. • Updated Section 2.5.7 based on feedback from J. Garcia. • Updated Table 15. • Updated Table 17. • Added description of Steps 6-10 in Table 18. • Updated Figure 8: Removed reference to WSDL (#2) and reprinted #22. • Updated Table 19. • Updated Table 21 and Table 23. • Changed Java references from “Java 2 Platform Enterprise Edition (J2EE)” to “Java Platform Enterprise Edition (Java EE).” • Updated Section 3.1.3. • Added Section 3.2.2 and Figure 13 as per G. Smullen. • Deleted “Developer Workstation” software list from Sections 3.3 and 6.2.1.1.3. • Updated Table 27. • Updated Section 4.2.1. • Updated Section 4.4.2. • Updated Section 4.5: Added reference note to prior section listing VSA-related software. • Updated Section 6.2. • Updated Section 6.2.2.3.16. • Updated Section 6.2.2.3.27. • Updated Section 6.4 and all 	

Date	Document Version	Software Version	Description	Author
			<p>sub-sections.</p> <ul style="list-style-type: none"> Updated Sections 7.1 and 7.2.1. Updated Section 8. Updated the APILIST section in the XSARPC routine in Table 31 based on feedback from R. Yorty. Updated Section 10 to add architect, EDE, and VSA demo use case services for external project representatives to the signature page. 	
07/11/2014	0.3	1.0.0	<p>Tech Updates:</p> <ul style="list-style-type: none"> Updated section 2.5.7. Updated section 3.1.3. Consult with A Chan. Updated section 3.2.2: Added VistA VA ^XTMP to Table 23. Updated section 3.3.3; Table 27: Updated WebLogic version. Updated section 3.3.4.1. Updated all comment Notes related to ESB. Updated section 4.2: Added a Note below Figure 22. Updated section 4.4.2: Indicated VSA internal web service is <i>private</i>. Updated section 4.5: Added a Note related to Log Reporting tools. Updated section 5.1, Table 29: Added entries for ^XTMP files. Updated Section 5.3: Listed Kernel parameters and RPCs related to VSA. Updated Section 6.2.1.1.3: Updated software versions. Updated Section 6.2.2.3.1: Updated XSAUTL (Table 33) and added new XSAUTL1 	<div></div> <div></div>

Revision History

Date	Document Version	Software Version	Description	Author
			<p>(Table 34).</p> <ul style="list-style-type: none"> Updated Section 6.2.2.3.6: Added XTMP global to Table 39. Updated Section 6.2.2.3.11: Added the following new RPCs: <ul style="list-style-type: none"> XSA GET LOGGING STATUS XSA SET LOGGING XSA GET LOG ENTRY XSA LOG LIST RETRIEVAL Updated Section 6.4. Updated Section 6.4.1. Updated Section 6.4.2. Updated Section 9.1. Updated Section 9.2. Updated/Added Figure 15, Figure 16, Figure 26, and Figure 27. 	
06/27/2014	0.2	1.0.0	<p>Tech Updates:</p> <ul style="list-style-type: none"> Updated Process IDs 11 – 14 in Table 8 based on feedback from M. A. Added RESTful service files to Section 4.2.2. Added RESTful service files to Table 30. 	
06/26/20014	0.1	1.0.0	Initial VSA 1.0.0 Phase 2 document: Based on original Prototype; Proof-of-Concept document.	

Artifact Rationale

The System Design Document (SDD) is a dual-use document that provides the conceptual design as well as the as-built design. This document will be updated as the product is built, to reflect the as-built product. Per the Project Management Accountability System (PMAS) Guide, the SDD with conceptual design is required prior to the Milestone 1 Review. The as-built for each delivery *must* be incorporated prior to the Milestone 2 Review.

Table 2. Template Revision History

Date	Version	Description	Author
March 2014	2.5	Section 508 repairs to new version approved by Architecture Engineering Review Board (AERB) Chair approved	Process Management
August 2013	2.3	Replaced the Service Architecture sub-section with new sub-sections for consumed and provided services. Also applied miscellaneous feedback from VA team.	Architecture, Strategy, and Design (ASD)
June 2013	1.2	Address inconsistencies in Section 3, Conceptual Design	Process Management
March 2013	1.1	Formatted to current ProPath documentation standards and edited to conform with latest Alternative Text (Section 508) guidelines	Process Management
January 2013	1.0	Initial Version	PMAS Business Office

Contents

Revision History	ii
Tables and Figures	xix
1 Introduction	1
1.1 Purpose of the SDD	1
1.2 Identification	1
1.2.1 Standards and Guidelines	1
1.3 Scope	3
1.4 Constraining Policies, Directives and Procedures	6
1.5 User Characteristics	6
1.6 Relationship to Other Documents and Plans	7
1.6.1 Remote Procedure Calls (RPCs)	8
1.6.2 Application Programming Interfaces (APIs)	9
1.6.3 Integration Agreements (IAs)	9
1.7 Definition, Acronyms, and Abbreviations	10
1.8 References	13
2 Background	14
2.1 Overview of the System	14
2.2 Overview of the Business Process	16
2.3 Business Benefits	21
2.3.1 Business Customer—Key Stakeholders	22
2.3.2 VSA Project Team Members	23
2.3.3 VSA Software Components	26
2.4 Assumptions and Constraints	28
2.4.1 Design Assumptions	28
2.4.2 Design Constraints	28
2.4.3 Design Trade-offs	29
2.5 Overview of the Significant Requirements	30
2.5.1 Overview of Significant Functional Requirements	30
2.5.2 Overview of Functional Workload/Performance Requirements	30
2.5.2.1 Multiple Concurrent Clients	33
2.5.2.2 Response Time	33

2.5.2.3	Throughput	33
2.5.2.4	Scalability	33
2.5.2.5	Reliability	34
2.5.3	Overview of Operational Requirements.....	34
2.5.3.1	Scalability	34
2.5.3.2	Failure.....	34
2.5.3.3	Disaster Recovery (DR)	35
2.5.4	Overview of the Technical Requirements.....	36
2.5.5	Overview of the Security or Privacy Requirements.....	37
2.5.6	Overview of System Criticality and High Availability Requirements.....	38
2.5.7	Single Sign-on Requirement	38
2.5.8	Requirement for Use of Enterprise Portals	39
2.5.9	Special Device Requirements	39
2.6	Legacy System Retirement.....	39
3	Conceptual Design.....	40
3.1	Conceptual Application Design.....	40
3.1.1	Application Context	40
3.1.2	High-Level Application Design	45
3.1.3	Application Locations	54
3.1.3.1	VSA Development and Integration Testing Phase	54
3.1.3.2	VSA Implementation (Production Release) Phase	58
3.2	Conceptual Data Design.....	63
3.2.1	Project Conceptual Data Model	63
3.2.2	System Topology and Message Workflow	63
3.2.3	Database Information.....	65
3.2.4	User Interface Data Mapping	66
3.2.4.1	VSA Wizard Web Application Screen Interface.....	67
3.2.4.1.1	VSA Wizard—Main Page.....	67
3.2.4.1.2	VSA Wizard—Service Descriptor Form	69
3.2.4.1.2.1	RESTful Web Services	69
3.2.4.1.2.2	SOAP Web Services	70
3.2.4.1.2.3	Service Descriptor Form fields	71

	3.2.4.2	Application Report Interface	76
	3.2.4.3	Unmapped Data Element.....	76
3.3		Conceptual Infrastructure Design.....	77
	3.3.1	System Criticality and High Availability	77
	3.3.2	Special Technology	78
	3.3.3	Technology Locations	78
	3.3.4	Conceptual Infrastructure Diagram.....	80
	3.3.4.1	Location of Environments and External Interfaces	80
	3.3.4.2	Conceptual Production String Diagram.....	84
4		System Architecture	85
4.1		Hardware Architecture	85
4.2		Software Architecture.....	85
	4.2.1	Methodology, Tools, and Techniques	86
	4.2.2	VSA Wizard.....	87
	4.2.3	VSA-generated Web Service	87
	4.2.4	VSA Listener.....	87
	4.2.4.1	VMRCS	87
	4.2.4.2	VMRCA	87
	4.2.5	Code Generator	88
	4.2.6	VistaServiceInvoker.....	88
4.3		Network Architecture	89
4.4		Service Oriented Architecture / Enterprise Shared Services	91
	4.4.1	Services Provided.....	91
	4.4.2	Service Required/Consumed.....	92
4.5		Enterprise Architecture.....	92
5		Data Design	93
5.1		DBMS Files	93
5.2		Non-DBMS Files.....	94
5.3		Data View	95
6		Detailed Design	96
6.1		Hardware Detailed Design	96
6.2		Software Detailed Design.....	96
	6.2.1	Conceptual Design	97

6.2.1.1	Product Perspective	97
6.2.1.1.1	User Interfaces	98
6.2.1.1.2	Hardware Interfaces	98
6.2.1.1.3	Software Interfaces	98
6.2.1.1.4	Communications Interfaces	99
6.2.1.1.5	Memory Constraints.....	100
6.2.1.1.6	Special Operations	100
6.2.1.2	Product Features	100
6.2.1.3	User Characteristics.....	100
6.2.1.4	Dependencies and Constraints.....	100
6.2.2	Specific Requirements	101
6.2.2.1	Database Repository	101
6.2.2.2	System Features.....	101
6.2.2.3	Design Element Tables.....	102
6.2.2.3.1	Routines (Entry Points).....	102
6.2.2.3.2	Templates.....	111
6.2.2.3.3	Bulletins	112
6.2.2.3.4	Data Entries Affected by the Design.....	112
6.2.2.3.5	Unique Records.....	113
6.2.2.3.6	File or Global Size Changes	113
6.2.2.3.7	Mail Groups	113
6.2.2.3.8	Security Keys.....	114
6.2.2.3.9	Options.....	115
6.2.2.3.10	Protocols	116
6.2.2.3.11	Remote Procedure Call (RPC)	117
6.2.2.3.12	Constants Defined in Interface.....	119
6.2.2.3.13	Variables Defined in Interface	119
6.2.2.3.14	Types Defined in Interface.....	119
6.2.2.3.15	GUI	120
6.2.2.3.16	GUI Classes	120
6.2.2.3.17	Current Form.....	120

	6.2.2.3.18	Modified Form	120
	6.2.2.3.19	Components on Form.....	120
	6.2.2.3.20	Events.....	121
	6.2.2.3.21	Methods	121
	6.2.2.3.22	Special References	121
	6.2.2.3.23	Class Events	122
	6.2.2.3.24	Class Methods.....	122
	6.2.2.3.25	Class Properties.....	122
	6.2.2.3.26	Uses Clause.....	122
	6.2.2.3.27	Forms	123
	6.2.2.3.28	Functions.....	123
	6.2.2.3.29	Dialog.....	125
	6.2.2.3.30	Help Frame.....	125
	6.2.2.3.31	HL7 Application Parameter	126
	6.2.2.3.32	HL7 Logical Link	126
	6.2.2.3.33	COTS Interface	127
6.3		Network Detailed Design	127
6.4		Service Oriented Architecture / Enterprise Shared Services Detailed Design	128
6.4.1		Service Description for VsaCommonService	128
6.4.2		Service Design for Provided Services.....	129
	6.4.2.1	Introduction	129
	6.4.2.1.1	Purpose and Scope of Service.....	129
	6.4.2.1.2	Links to Other Documents	129
	6.4.2.2	Service Details.....	130
	6.4.2.2.1	Service Identification.....	130
	6.4.2.2.2	Service Versions	130
	6.4.2.2.3	Summary of Design and Platform Details	131
	6.4.2.2.3.1	SOA Patterns Implemented.....	131
	6.4.2.2.3.2	COTS Platform vendor names and versions for hosting platform.....	131
	6.4.2.3	Dependencies.....	131
	6.4.2.4	Service Design Details	131

6.4.2.4.1	Interface Technical Specs	131
6.4.2.4.1.1	Service Invocation Type	131
6.4.2.4.1.2	Service Interface Type	131
6.4.2.4.1.3	Service Name	131
6.4.2.4.1.4	Interface	131
6.4.2.4.1.5	End Points	131
6.4.2.4.1.6	Operations or Methods	132
6.4.2.4.1.7	Message Schemas	132
6.4.2.4.2	Information Model	132
6.4.2.4.2.1	Class Diagram and Description of Entities Involved	132
6.4.2.4.2.2	Mappings from ELDM to Standards Based Schemas	132
6.4.2.4.3	Behavior Model (a.k.a. Use Case Realization)	132
6.4.2.4.3.1	Use Cases (Use Case Model)	132
6.4.2.4.3.2	Interaction Diagrams	132
6.4.2.5	Gap Analysis	133
6.4.2.5.1	Variances from Enterprise Target Architecture	133
6.4.2.5.2	Variances from SLDs	133
6.4.2.5.3	Variances from Standards and Policies	133
6.4.2.5.4	Justification for Exceptions and Mitigation	133
7	External Interface Design	134
7.1	Interface Architecture	134
7.2	Interface Detailed Design	135
7.2.1	VistaServiceInvoker Application	135
7.2.2	VistA M Routine Calling Service	135
7.2.3	VistA M Routine Calling Adapter	135
8	Human-Machine Interface	136
8.1	Overview	136
8.2	Interface Design Rules	136
8.2.1	Inputs	137
8.2.2	Outputs	138

8.3	Navigation Hierarchy	139
8.3.1	VSA Wizard—Main Page.....	139
8.3.2	VSA Wizard—Service Descriptor Form	140
9	Security and Privacy	142
9.1	Security.....	142
9.2	Privacy.....	142
10	Attachment A—Approval Signatures.....	143
11	Appendix A—Additional Information	145
11.1	RTM	145
11.2	Package and Installation.....	145
11.3	Design Metrics	145
11.4	Acronym List and Glossary.....	145
11.5	Required Technical Documents	145
11.5.1	VSA Documentation	145
11.5.2	VistA Documentation	146
11.5.3	Standards and Guidelines.....	146

Tables and Figures

Tables

Table 1. Document Revision History.....	ii
Table 2. Template Revision History	xi
Table 3. VSA Project Scope—Goals and Objectives	3
Table 4: VSA Scope Inclusions	4
Table 5: VSA Scope Exclusion— <i>Future</i> phases/iterations	5
Table 6: VSA Application Users	6
Table 7: Acronym List and Glossary	10
Table 8: VSA business process that pertains to the VSA design-time process (using the Wizard to create a service).....	18
Table 9. VSA Business Customer—Key Stakeholders.....	22
Table 10. VSA Project Team Members (team members are listed by role; within a role, names are listed in alphabetical order)	23
Table 11. VSA Software Components	26
Table 12. Operational Requirements	35
Table 13. Technical Requirements.....	36
Table 14. Security Requirements	37
Table 15: VSA Application Context Description—Object.....	41
Table 16. VSA Application Context Description—Interfaces External to OI&T	42
Table 17. VSA Application Context Description—Interfaces Internal to OI&T	42
Table 18. VSA Application Context Description—Externally Shared Data Stores	44
Table 19: VSA High Level Application Design—Objects	47
Table 20. VSA High Level Application Design—Internal Data Stores	53
Table 21: VSA Application Locations—Development and Integration Phases	54
Table 22: VSA Application Locations—Integration and Implementation (Release) Phase	60
Table 23: VSA Database Inventory	65
Table 24: VSA Wizard—Main Page (Screen) Description	68
Table 25: VSA Wizard—Service Descriptor Form Description.....	71
Table 26: VSA Special Technology Requirements	78

Table 27: VSA Technology Location Details.....	78
Table 28: VSA Communication Requirements.....	90
Table 29: VSA DBMS files	93
Table 30: VSA Non-DBMS files	94
Table 31: VSA Routines—XSARPC.....	102
Table 32: VSA Routines—XSAVMRCA.....	106
Table 33: VSA Routines—XSAUTL.....	108
Table 34: VSA Routines—XSAUTL1.....	110
Table 35: VSA Templates.....	111
Table 36: VSA Bulletins	112
Table 37: VSA Data Entries Affected by the Design	112
Table 38: VSA Unique Record ID	113
Table 39: VSA File or Global Size Changes	113
Table 40: VSA Mail Groups	113
Table 41: VSA Security Keys.....	114
Table 42: VSA Options.....	115
Table 43: VSA Protocols	116
Table 44: VSA RPCs—XSA GET RPC INFO.....	117
Table 45: VSA RPCs—XSA RPC LIST	117
Table 46: VSA RPCs—XSA GET LOGGING STATUS	118
Table 47: VSA RPCs—XSA SET LOGGING	118
Table 48: VSA RPCs—XSA GET LOG ENTRY	118
Table 49: VSA RPCs—XSA LOG LIST RETRIEVAL.....	119
Table 50: VSA Constants Defined in Interface	119
Table 51: VSA Variables Defined in Interface	119
Table 52: VSA Types Defined in Interface	119
Table 53: VSA Wizard GUI	120
Table 54: VSA Wizard GUI Classes.....	120
Table 55: VSA Components on Form.....	121
Table 56: VSA Events	121
Table 57: VSA Methods	121

Table 58. VSA Special References	121
Table 59: VSA Class Events	122
Table 60: VSA Class Methods.....	122
Table 61: VSA Class Properties.....	122
Table 62: VSA Forms	123
Table 63: VSA Functions.....	123
Table 64: VSA Dialog	125
Table 65: VSA Help Frame	125
Table 66: VSA HL7 Application Parameter	126
Table 67: VSA HL7 Logical Link	126
Table 68: VSA COTS Interface	127
Table 69. VSA Provided Services—Service Identification	130
Table 70. VSA Provided Services—Service Versions	130
Table 71. VSA Provided Services—Operations or Methods.....	132
Table 72. VSA Provided Services—Gap Analysis	133

Figures

Figure 1. VSA Web Service—VistA service process overview	7
Figure 2 VSA Web Service—VistA M code "wrappers" (layers)	8
Figure 3. VSA Process Overview—VSA design-time process cross-functional diagram (swim lanes) using the VSA Wizard to create a service.....	17
Figure 4. VSA External Dependencies for the Federating Platform—Centralized (“As Is”).....	31
Figure 5. VSA External Dependencies for the Federating Platform—Distributed (“To Be”).....	32
Figure 6: VSA Design-time Context Diagram.....	40
Figure 7. VSA Run-time Context Diagram—VSA Federating Platform Run-time Dependencies (in WebLogic)	41
Figure 8: VSA High-Level Application Design.....	45
Figure 9. VSA Wizard—Component Flow.....	46
Figure 10. VSA Wizard Service Creation—Terremark environment (sample).....	57
Figure 11. VSA test to same development environment (one VistA system).....	58

Figure 12. VSA Run-time Context Diagram—VSA Federating Platform Run-time Dependencies (WebLogic): Production	61
Figure 13. VSA System Topology and Message Workflow	63
Figure 14: VSA Wizard—Main Page (Screen).....	67
Figure 15. VSA Wizard—Service Descriptor Form for RESTful Services.....	69
Figure 16. VSA Wizard—Service Descriptor Form for SOAP Services.....	70
Figure 17. VSA Conceptual Networks and Environments (1 of 4)	80
Figure 18: VSA Conceptual Networks and Environments (2 of 4)	81
Figure 19. VSA Conceptual Networks and Environments (3 of 4)	82
Figure 20. VSA Conceptual Networks and Environments (4 of 4)	83
Figure 21. VSA Conceptual Single Production String Diagram.....	84
Figure 22. VSA—Logical Component Model	85
Figure 23. VSA—Network Communications Architecture.....	89
Figure 24. VSA—Network Communications Architecture.....	99
Figure 25. VSA Wizard—Main Page (Screen).....	139
Figure 26. VSA Wizard—Service Descriptor Form for RESTful Services.....	140
Figure 27. VSA Wizard—Service Descriptor Form for SOAP Services.....	141

1 Introduction

1.1 Purpose of the SDD

The purpose of this System Design Document (SDD) is to describe how the VistA Services Assembler (VSA) is to be constructed. The SDD translates the requirement specifications into a document from which the developers can create the actual system. It identifies the following:

- Top-level system architecture
- Hardware
- Software
- Communication
- Interface components



NOTE: The SDD is a *conceptual* document that is continuously fine-tuned as software development solutions are identified and implemented. Thus, this document is considered to be a “living” document that is subject to change. Any content changes will be noted in the “[Revision History](#)” section in [Table 1](#).

1.2 Identification

This SDD applies to the VSA 1.0.0 Phase 2 software release. This software is being developed for national release.

1.2.1 Standards and Guidelines

The VSA Phase 2 project team, software, and test servers adhere to the following directives, policies, procedures, standards, and guidelines:

- [Program Management Accountability System \(PMAS\)](#) and [ProPath](#) guidelines and processes—To monitor project progress.
- [Technical Reference Model \(TRM\)](#)—For approved tools and standards.
- MUMPS (M) American National Standards Institute (ANSI) X11.1-1995 standard: <http://71.174.62.16/Demo/AnnoStd> (Annotated).
- VA Standards & Conventions Committee (SACC) Codes Standards and Conventions: [communities/app_dev/sac/default.aspx](#)
- Java Platform, Enterprise Edition (Java EE) Architecture (Eclipse documentation): <http://help.eclipse.org/indigo/index.jsp?topic=%2Forg.eclipse.jst.j2ee.doc.user%2Ftopics%2Fejarch.html>.
- [World Wide Web Consortium \(W3C\) Simple Object Access Protocol \(SOAP\) Standard](#).

- [World Wide Web Consortium \(W3C\) Extensible Markup Language \(XML\) Standard.](#)
- [Federal Guidelines—IAM Service: VA Handbook 6500-Information Security Program.](#)
- VA Directive 6102—For the Enterprise Knowledge Management Solution and to serve the purpose of guaranteeing all VA Internet and Intranet standards for systems operations are considered.
- Section 508 of the Rehabilitation Act (29 U.S.C. § 794d): [VA Section 508 Home Page.](#)
- Enterprise Shared Services/Service Oriented Architecture (ESS/SOA): [REDACTED]
- National Institute of Standards and Technology (NIST) and Federal Desktop Core Configuration (FDCC) Guidelines—For servers regarding build-outs and security settings. Servers use standard protocols and ports for communication and network settings:
 - NIST SP 800-44 Checklist to ensure specifications are listed in compliance for securing public Web servers; *Guidelines on Securing Public Web Servers* document: <http://csrc.nist.gov/publications/nistpubs/800-44-ver2/SP800-44v2.pdf>.
 - *NIST Electronic Authentication Guideline* document: <http://csrc.nist.gov/publications/nistpubs/800-63-1/SP-800-63-1.pdf>.
- [Federal Identity, Credential, and Access Management \(FICAM\) Roadmap and Implementation Guidance Roadmap and Implementation Guidance](#) document.
- Electronic and Information Technology Accessibility Standards (36 CFR 1194).
- Federal Information Security Management Act (FISMA) of 2002.
- VAAR 852.273-75 Security Requirements for Unclassified Information Technology Resources (interim Oct 2008).
- FIPS Pub 201, Personal Identity Verification for Federal Employees and Contractors, February 25, 2005.

1.3 Scope

VSA Phase 2 creates a set of software utilities for national release. VSA provides the basic (uncomplicated) functionality necessary to automate the creation of VistA SOA services and infrastructure components necessary to support and operate those services. It further develops this set of utilities to include the VistA Services Assembler Wizard and assorted components.

The VSA Phase 2 project also identifies and produces multiple, sample VistA SOA business services as a "reference implementation" to confirm the use of VSA services. The specific business services to be produced will be identified based on a consideration of organizational initiatives that have a need for VistA SOA services in the near *future*. This approach ensures that services produced as a part of the "reference implementation" are both critically tested and of immediate value to the organization.

Table 3. VSA Project Scope—Goals and Objectives


Goal/Objective	Desired Outcome	Measurement	Impact
Provide utilities for automating the creation of VistA SOA services and supporting their use.	Infrastructure utilities that support automated VistA SOA service creation and execution.	VistA SOA services are consumable by external systems/applications.  NOTE: In the <i>future</i> , web services may be managed by the Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI), which could serve as a web services registry and repository.	VistA can be easily integrated with external systems/applications through an SOA compliant architecture.
Enable the exposure of existing VistA data and methods as SOA compatible web services.	Use of VSA utilities to create business services that provide business value to VA mission and objectives.	Current VA development and system integration initiatives are consumers of VistA SOA Services implemented by VSA.	Advanced state of SOA architecture implementation in the VA and associated software functionality and sustainability benefits.
Abstract VistA system developers and integrators from multiple technology orientation.	Consuming systems / applications can be integrated without extensive knowledge of VistA technology.	VA software development staff members are able to rapidly create and deploy VistA SOA Services with minimal re-training.	Improved use of existing staff skills, leveraging of previously implemented VistA application functionality.
Improve VistA development "time to market" and overall system sustainability.	Automated, standardized production of VistA SOA Service components.	Development time for VistA related services is notably reduced supporting efficient software delivery.	Provides cost effective methodology for the integration of VistA with external systems.

Table 4: VSA Scope Inclusions

Includes
<p>Multiple Platform Support—VSA Wizard runs, creates web services, and tests the services on the following supported platforms:</p> <ul style="list-style-type: none"> • Windows Server 2008 • Linux
<p>VSA Wizard—Web application that provides a web page interface that accepts user input of an RPC definition and generates a Service Descriptor file in Extensible Mark-up Language (XML) format, which is used to automatically generate a web service operation that invokes that RPC.</p> <p>This application was written using the Spring MVC 3.2.4 (Model-View-Controller [MVC] framework).</p>
<p>Wizard Form Validation—VSA Wizard user interface checks to make sure the user input is valid.</p>
<p>RPC Parameter Passing—RPC Invocation Application Programming Interface (API) allows passing single or multiple parameters for those RPCs that expect multiple parameters.</p>
<p>Multiple Operations—VSA supports single or multiple operations in one web service. In other words, VSA supports multiple web service methods in one web service Web Archive (WAR) file.</p>
<p>Return Types—VSA returns the following types:</p> <ul style="list-style-type: none"> • String • JavaScript Object Notation (JSON)-formatted object • List • Map
<p>REST: Representational State Transfer (alternative to SOAP)—Looking beyond SOAP, VSA supports RESTful web services in which VistA data and functionality are exposed as resources and accessed via Uniform Resource Identifiers (URIs), using a stateless communication HyperText Transfer Protocol (HTTP) protocol¹.</p>
<p>Leverages Existing Code—VSA leverages VistALink code (as a model) and design patterns. This results in:</p> <ul style="list-style-type: none"> • Richer functionality. • Improved fault-tolerance. • Reduction of code and complexity.

¹ Wikipedia website (<http://en.wikipedia.org/wiki/RESTful>) and other sites state that HTTP 1.1 was designed as being RESTful. HTTP 1.1 was designed as being RESTful. By transitivity, a service (web API) that uses as many, and properly, features of HTTP is RESTful. Since a SOAP service uses only the HTTP POST method, and improperly, it is *not* RESTful.

Includes
<p>Federation Support:</p> <ul style="list-style-type: none"> • Federate queries and responses across VistA systems. • Leverage Master Patient Index (MPI)/ Master Veteran Index (MVI) services to discover all instances of patient data. • Implement multi-threaded data retrieval and data aggregation to improve performance. • Provide the ability to define the creation of services which reference a single VistA system or federate the routing of edits, queries and aggregation of responses across one, many or all VistA systems. • Provide VistA SOA service generation utilities capable of creating services that can respond to multithreaded queries across multiple VistA systems where indicated.
Continuous Integration—Scripting and automation of building, deployment, unit and integration testing of all VSA code to achieve immediate and reproducible test results.
Enhanced Security—Implementation of Hypertext Transfer Protocol Secure (HTTPS) with Secure Socket Layer (SSL) and other security mechanisms as needed to protect Personal Health Identifiers (PHI), Personally Identifiable Information (PII) and other sensitive VA data in motion and at rest.
Improved Error Handling—Currently, VistA M Routine Calling Adapter (VMRCA) listener has improved Error handling. It is still evolving and expected to continue improvement. Other VSA error handling will be improved to catch more exceptions and provide more meaningful messages. These error codes will be documented in the VSA end-user documents (e.g., Developer's Guide or Systems Management Guide).
Reference Implementation—Develop one or two web services that exercise and showcase VSA.

Table 5: VSA Scope Exclusion—*Future* phases/iterations

Excludes
True Multi-User Support—Currently some VSA Wizard functionality does <i>not</i> support concurrent users.
Additional Constructs—Expose additional constructs from VistA (e.g., Application Programming Interfaces [APIs], VA FileMan, etc.) as Intranet web services.
Vendor specific components—VSA service descriptors include information related to vendor specific components (e.g., payload transformation utilities, etc.) used in service descriptions where indicated.
Provide the ability to generate services that are SOA compliant and can be fully integrated with organizational SOA infrastructure (e.g., Enterprise Service Bus [ESB]/Enterprise Messaging Infrastructure [eMI], repository, etc.).
Composite Web Services—Currently, VSA allows one Remote Procedure Call (RPC) call per operation. Need to explore all of the use cases where multiple RPC calls in one operation are desirable.
Established Deployment Process—Define/document/provide the software distribution process and utilities necessary for the deployment of VistA SOA Services.
Auditing—In addition to "service descriptors," the VSA Wizard maintains an audit record of VistA SOA service generation actions that have occurred (including the actions taken, developer, date/time, etc.).

1.4 Constraining Policies, Directives and Procedures

VSA design and development strives to leverage Free Open Source Software (FOSS) as much as possible while adhering to all the VA standards and conventions. In certain, specific cases where VA has chosen to use proprietary and commercially licensed products over the FOSS alternatives, VSA followed the VA standards and conventions. The VSA Phase 2 implementation uses the following commercial products:

- InterSystems® Caché 2011.1.2.701.0.12942
- Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5



NOTE: *Future* phases of VSA development will explore and include FOSS alternatives to the commercial products above while always maintaining the priority to use the VA-chosen software.

As development is expanded to *future* project phases/iterations, VSA will comply with all the necessary legal disclaimers described in the VA 6102 handbook and all Section 508 disclaimers relevant to the VSA solution.

There are no other constraining directives, policies, procedures, standards, and guidelines, other than those defined with the standard VistA Massachusetts General Hospital Utility Multi-Programming System (MUMPS); abbreviated as M) platform configuration and framework.



REF: For a list of specific directives, policies, procedures, standards, and guidelines, see Section [1.2.1](#).

1.5 User Characteristics

VSA is designed for users (e.g., Developers for Producing or Consuming applications) to easily create web services from VistA RPCs.

[Table 6](#) identifies the user roles of the individuals who will use VSA.

Table 6: VSA Application Users

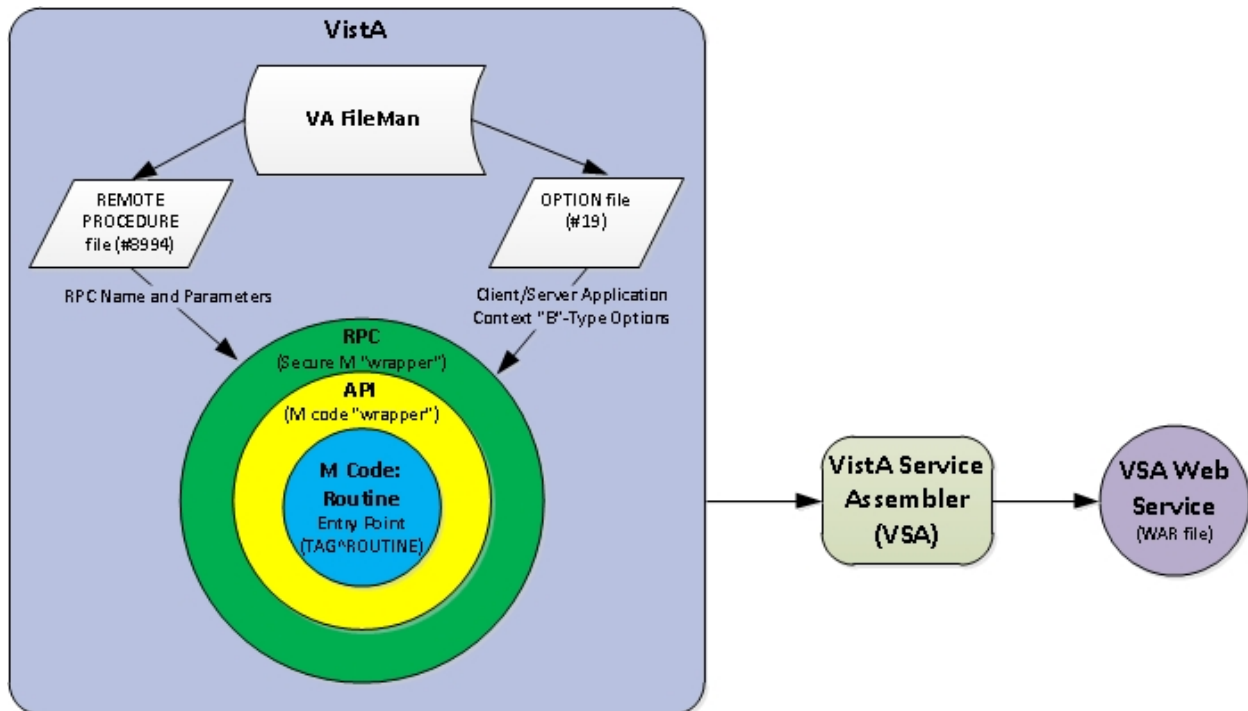
Application Component	Location	User
Producing Applications	All sites.	Producer/Provider (e.g., developer) that creates new VSA web services from VistA RPCs.
Consuming Applications	All sites.	Consumer (e.g., application developers) that uses VSA-generated web services.

VSA allows non-VistA M developers to more easily interact with VistA and access VistA data through VSA web services. Users do *not* need to be proficient in M or building Java web services, since the VSA Wizard allows developers to easily build VSA web services from existing VistA RPCs.

1.6 Relationship to Other Documents and Plans

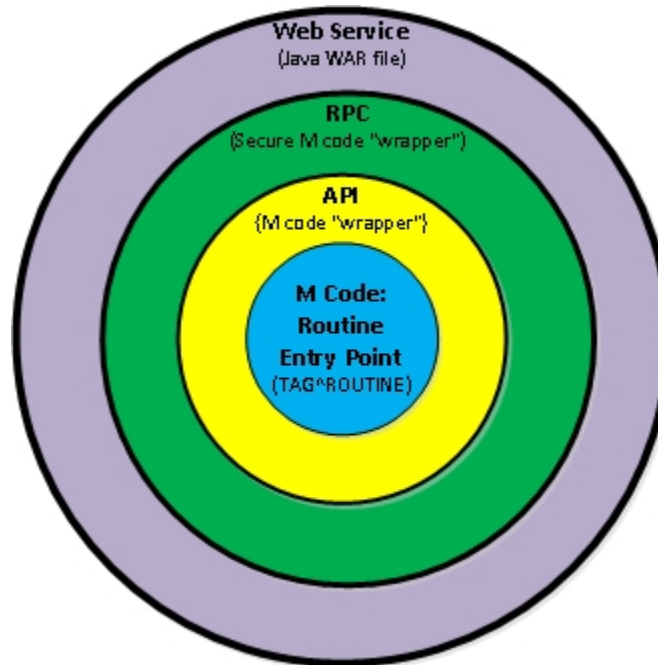
The VSA Phase 2 software provides the capability to create *new* web services from *existing* VistA Remote Procedure Calls (RPCs). Thus, VSA works across and has a relationship to *all* VistA applications and associated RPCs.

Figure 1. VSA Web Service—VistA service process overview



VSA-created web services are Java WAR files that are essentially wrappers around VistA Remote Procedure Calls (RPCs).

Figure 2 VSA Web Service—VistA M code "wrappers" (layers)



1.6.1 Remote Procedure Calls (RPCs)

RPCs are essentially security wrappers around VistA [Application Programming Interfaces \(APIs\)](#), which allow client/server applications to securely call an API. RPCs are written in M, and for security, require that client/server applications create the following:

- Entry in the REMOTE PROCEDURE file [#8994]. This registers the RPC to a client/server application. This file contains all available and authorized VistA RPCs.
- Client/Server application context "**B**"-type option (i.e., "**B**" = Broker options) in the OPTION file (#19). These "**B**"-type option are assigned to users like any other assigned option in VistA. It can be put on the user's primary menu tree or as a secondary option/menu as part of his/her suite of permitted options. The application *must* specify the option and that option *must* be in a user's menu tree to run the application RPC.

The server receives a message from the client-side of the application and parses out the name of the remote procedure call and its parameters. The server-side of the application looks up the remote procedure call in the REMOTE PROCEDURE file (#8994), verifies that the RPC is allowed to run in the context of the application (i.e., "**B**"-type option), and executes the RPC using the passed-in parameters. At this point, the server side of the application processes the request and returns the result of the operation. The result of the call contains either several values or a single value. If the operation is a query, then the result is a set of records that satisfy that query. If the operation is to simply file the data on the server or it

is unnecessary to return any information, then, typically, notification of the success of the operation will be returned to the client.



NOTE: VSA itself does *not* create new or modified RPCs. It only creates web services from existing VistA RPCs.

1.6.2 Application Programming Interfaces (APIs)

APIs are essentially wrappers around VistA M routines that are accessible at a specific callable entry point (i.e., TAG^ROUTINE). APIs are program calls provided for use by application developers. APIs are M code that can take optional parameters to do some work and then return either a single value or an array back to the client calling application. APIs allow developers to carry out standard computing activities eliminating the need to duplicate utilities in their own software. APIs also further DBA goals of system integration by channeling activities through defined callable entry points.

VistA APIs are units of programming code provided by a custodial development domain (e.g., Infrastructure domain, such as RPC Broker or Kernel) to permit developers outside the custodial domain (e.g., clinical domain, such as Computerized Patient Record System [CPRS], Pharmacy or Laboratory) to accomplish a specified purpose. APIs are also known as callable entry points or callable (sub) routines. APIs in VistA can be defined as follows:

- Extrinsic functions
- Extrinsic special variables
- Label references to a routine (i.e., TAG^ROUTINE).



NOTE: VSA itself does *not* create new or modified APIs. It only creates web services from existing VistA RPCs.

1.6.3 Integration Agreements (IAs)

The Database Administrator (DBA) maintains a list of Integration Control Registrations (ICRs)/Integration Agreements (IAs) or mutual agreements between software developers allowing the use of internal entry points or other software-specific features that are *not* available to the general programming public. VistA APIs fall into the following three categories:

- Supported API—Callable routines that are supported for general use by all VistA applications.
- Controlled Subscription API—Callable routines that require an Integration Agreement with the custodial software.
- Private API—Callable routines where only a single VistA application is granted permission to use an attribute/function of another VistA software application. These IAs are granted for special cases, transitional problems between versions, and release coordination.



NOTE: VSA itself does *not* create new or modified ICRs/IAs. It only creates web services from existing VistA RPCs. Integration Control Registrations should be established between the application creating the service and the custodial VistA application providing the remote procedure call (RPC) or API.

1.7 Definition, Acronyms, and Abbreviations

Table 7: Acronym List and Glossary

Term	Meaning
AITC	Austin Information Technology Center
API	Application Programming Interface
BRD	Business Requirements Document
CDS	Clinical Data Services
CIO	Chief Information Office
COTS	Commercial-Off-The-Shelf
CPRS	Computerized Patient Record System
CSP	Caché Server Pages
CVS	Conformance Validation Statement
DoD	Department of Defense
EDE	Enterprise Development Environment. The EDE Data Center is managed by: <ul style="list-style-type: none"> • VA—Enterprise Operations (EO) • Contractor—MITRE Corporation Possible server location: Austin Information Technology Center (AITC).
eMI	Enterprise Messaging Infrastructure
ESB	Enterprise Service Bus
ESS	Enterprise Shared Services
Federated Web Service	A published WAR file on the Federating Platform.
Federating	Controlled cross-platform access.
Federating Platform	The Federating Platform serves as the registry and repository for storing Service Descriptor and WAR files (web services). It federates routing of queries from provider and consumer "service" requests to and from Veterans Health Information Systems and Technology Architecture (VistA). It also provides the security interface that controls single sign-on (SSO) access, so a user's authentication token is trusted across multiple VA Information Technology (IT) systems or organizations.
FO	Field Operations
GFE	Government Furnished Equipment

Term	Meaning
GUI	Graphical User Interface
HL7	Health Level Seven
HTTPS	Hypertext Transfer Protocol Secure
IAM	Identity and Access Management
IHS	Indian Health Service
IT	Information Technology
IV&V	Independent Verification and Validation
JAX-RS	Java API (Java Standard Library) for XML REST Web Services
JAX-WS	Java API I (Java Standard Library) for XML SOAP Web Services
JSON	JavaScript Object Notation
M (MUMPS)	Massachusetts General Hospital Utility Multiprogramming System (now known as “M”)
MDWS	Medical Domain Web Services
MPI	Master Patient Index
MVI	Master Veteran Index
MVC	Model-View-Controller
NCA	National Cemetery Administration
OI&T	Office of Information and Technology
Operation	An operation corresponds to a VistA remote procedure that is to be invoked, and it has response type and input parameter specifications.
Payloads	Transmitted data. For example, VSA-related payloads that get translated between M routine compatible syntax and various external formats (e.g., XML and JSON).
PD	Product Development
PITC	Philadelphia Information Technology Center
PMP	Project Management Plan
R&D	Research and Development
REST	Representational State Transfer (RESTful)
REST Service	A REST web service “defines a set of architectural principles by which you can design web services that focus on a system's resources, including how resource states are addressed and transferred over HTTP by a wide range of clients written in different languages. When using REST, you can

Term	Meaning
	<p>characterize web services as 'RESTful.'</p> <p>REST is an alternative to other distributed-computing specifications, such as SOAP and Web Services Description Language (WSDL)-based Web services, because it is a considerably simpler style to use."²</p> <p>With REST, "each unique URL is a representation of some object. You can get the contents of that object using an HTTP GET, to delete it, you then might use a POST, PUT, or DELETE to modify the object (in practice most of the services use a POST for this)."³</p>
RPC	Remote Procedure Call
RSD	Requirements Specification Document
SDD	Systems Design Document
SDLC	System Development Life Cycle
SEDR	Systems Engineering and Design Review
SME	Subject Matter Expert
SOA	Service Oriented Architecture
SOAP	Simple Object Access Protocol
SOAP Service	<p>A SOAP web service is "a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards."⁴</p>
SoapUI	Simple Object Access Protocol User Interface
SSL	Secure Socket Layer
SSO	Single Sign-on
SSOe	Single Sign-on External
SSOi	Single Sign-on Internal
TCP/IP	Transmission Control Protocol/Internet Protocol
TLS	Transport Layer Security
TRM	Technical Reference Model
URI	Uniform Resource Identifier

² DeveloperWorks website: <http://www.ibm.com/developerworks/library/ws-restful/index.html> ; "RESTful Web services: The basics" topic.

³ Pete Freitag website: <http://www.petefreitag.com/item/431.cfm>; "REST vs SOAP Web Services" topic.

⁴ Wikipedia: "Web Service" definition: http://en.wikipedia.org/wiki/Web_service#cite_note-1; "Web Services Glossary". W3C. February 11, 2004. Retrieved 2011-04-22.

Term	Meaning
URL	Uniform Resource Locator
VA	Department of Veterans Affairs
VAMC	Veterans Affairs Medical Center
VBA	Veterans Benefits Administration
VHA	Veterans Health Administration
VIA	VistA Integration Adaptor
VistA	Veterans Health Information Systems and Technology Architecture
VM	Virtual Machine
VMRCA	VistA M Routine Calling Adapter (listener). VSA server component.
VMRCS	VistA M Routine Calling Service (listener). VSA server component.
VSA	VistA Services Assembler
WAR	Web ARchive. When a WAR file is published on the Federating Platform, it is a service.
WMB	WebSphere Message Broker
WSDL	Web Services Description Language
WSRR	WebSphere Registry and Repository
XML	Extensible Mark-up Language. The VSA Service Descriptor files are written in XML format.

1.8 References

For a complete list of documents referenced in this SDD, see the “[Required Technical Documents](#)” section.

2 Background

Department of Veterans Affairs (VA) computing origins consist of legacy systems that functioned in a largely autonomous manner. Veterans Health Information Systems and Technology Architecture (VistA) began with a collection of decentralized systems capable of serving the needs of a single medical center.

It has become necessary to identify a new design approach for providing Service Oriented Architecture (SOA)-compliant integration between VistA applications and external systems in the course of Information Technology (IT) application modernization, which includes the following:

- Architectural revisions for achieving SOA implementation.
- Integration of external systems.
- Diverse technologies.
- Commercial-Off-The-Shelf (COTS) products.

Implementing SOA design in the VA has presented the challenge of bridging the technical gap between the mainframe procedural language environment of VistA and disparate (often object-oriented) technologies of COTS products and external systems. In addition to the need for identifying a technical solution, past efforts to implement SOA in the VA have illuminated an organizational gap as well, wherein software development staff involved with VistA generally lacked the technical skills necessary to integrate VistA directly with other systems.

2.1 *Overview of the System*

The VistA Services Assembler (VSA) Phase 2 effort provides a software solution to the business need, including:

- Automation of VistA Service Oriented Architecture (SOA) service generation.
- Infrastructure utilities necessary for the execution of VistA SOA services.
- Variety of "reference implementation" service examples that prove the concept.

VA objectives prioritize the implementation of SOA as an Enterprise design. The VSA design was identified as the preferred architectural design for SOA-based systems going forward. The intent is to eventually replace interim connectivity design approaches (e.g., Medical Domain Web Services [MDWS], VistA Integration Adapter [VIA], Clinical Data Services [CDS], etc.). VSA satisfies the consumer needs currently supplied by those other design approaches. It significantly enhances:

- VistA security.
- System performance.
- Sustainability.

VSA provides both a technical and organizational solution:

- **Technical Solution:**
 - Provides legacy system integration compatibility.
 - Provides SOA implementation compliance and infrastructure integration.
 - Enhances compliance with VA Enterprise Architecture and software development standards relative to SOA implementation in legacy systems.
 - Provides technical abstraction across disparate environments.
 - Provides standardization of technical characteristics and maintainability.
 - Expands (significantly) VistA extensibility and system integration opportunities by facilitating the implementation of SOA architecture.
 - Leverages existing "service assets:"
 - Provides tools and utilities that facilitate the creation and execution of auto-generated VistA SOA services (i.e., VSA Wizard).
 - Provides the ability to automate the generation of "service" needs across major initiatives.
- **Organizational Solution:**
 - Bridges and minimizes staff skill set and technical orientation boundaries.
 - Provides a standardized technical solution:
 - Addresses the organizational challenge of approaching disparate technologies relevant to legacy systems and SOA.
 - Increases VA SOA software sustainability and VistA integration extensibility.
 - Improves compatibility with "open source" participation objectives.
 - Increases software development efficiency through the automation of VistA SOA service creation.
 - Reduces investment.
 - Improves "time to market."

The VSA Phase 2 software includes the following features and functionality:

- Platform Support—VSA Wizard runs, creates web services, deploys web services, and provides a link to test the services on the following supported platforms:
 - Oracle WebLogic Server 11g on Microsoft Windows Server 2008.
 - Oracle WebLogic Server 11g on Red Hat Enterprise Linux 6.5.
- Wizard Form Validation—VSA Wizard user interface checks to make sure the user input is valid.
- RPC Parameter Passing—RPC Invocation Application Programming Interface (API) allows passing single or multiple parameters for those RPCs that expect multiple parameters.
- Multiple Operations—VSA supports single or multiple operations in one web service. In other words, VSA supports multiple web service methods in one web service Web Archive (WAR) file.

- Return Types—VSA returns the following types:
 - String
 - JavaScript Object Notation (JSON)-formatted object
 - List
 - Map
- Leverages Existing Code—VSA leverages VistaLink code and design patterns. This results in:
 - Richer functionality.
 - Improved fault-tolerance.
 - Reduction of code and complexity.

VSA plans to distribute the following two products:

- VSA Wizard (design-time)
- Federating Platform (run-time)

The following components will be distributed with the VSA Wizard and Federating Platform:

- vsaJersey (web archive [war])
- vsaXml (java archive [jar])
- vsaVistaClient (jar)
- vsaMviWebServiceClient (jar)
- vsaVistaServiceInvoker (jar)
- vsaCommonServiceWSClient (jar)
- vsaCommonServiceLib (jar)
- vsaCommonService (war)—dynamically generated
- VSAServiceGenerator (jar)
- vsaWizard (war)
- vsaWizardApp (enterprise archive [ear])

2.2 Overview of the Business Process

VSA has the following process phases:

- Design-time—Creation and auto-generation ([Figure 3](#))
- Run-time—Execution

Figure 3. VSA Process Overview—VSA design-time process cross-functional diagram (swim lanes) using the VSA Wizard to create a service

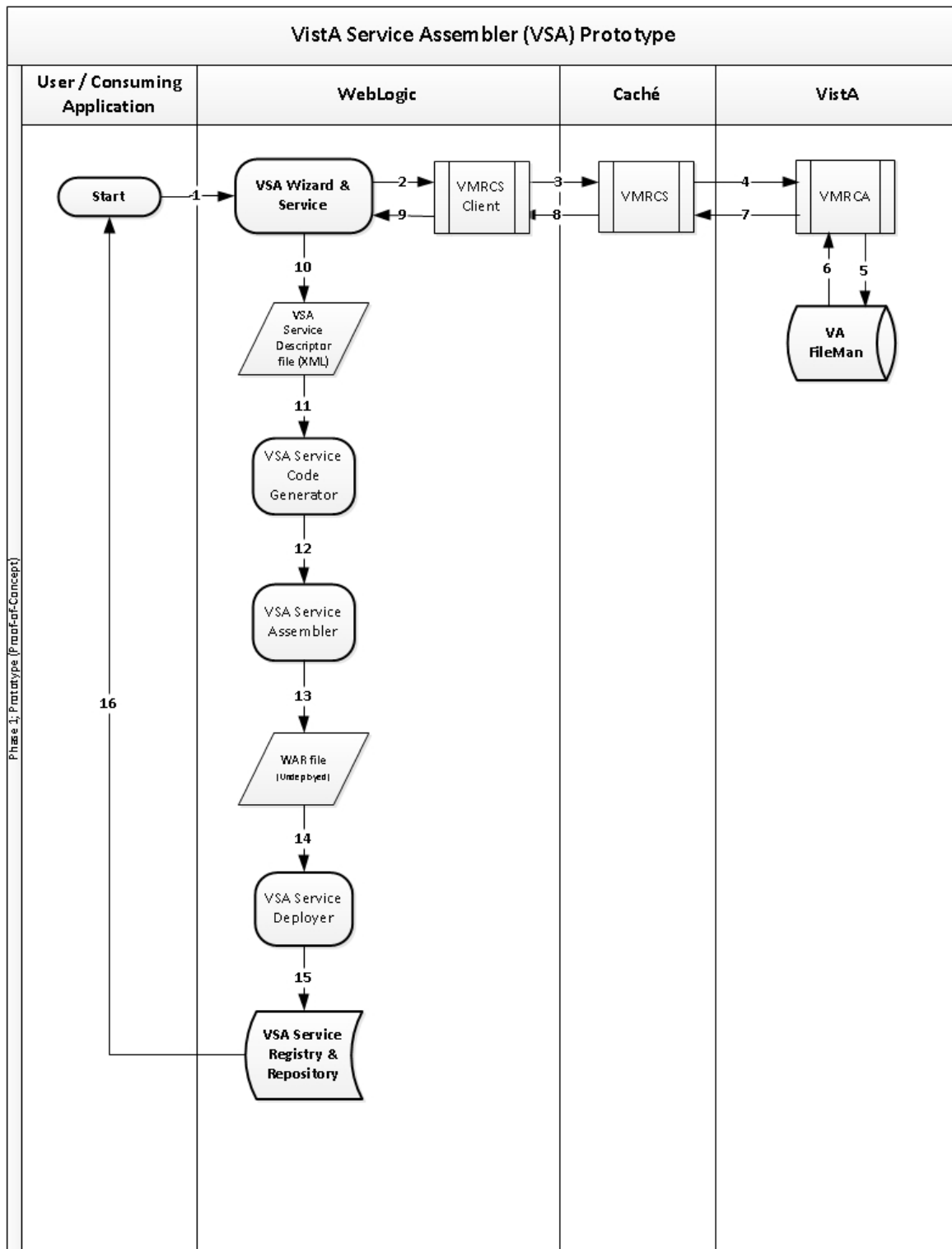



Table 8: VSA business process that pertains to the VSA design-time process (using the Wizard to create a service)

Business Process ID	Business Process Name	Type	Owner	Description
1	Start VSA Wizard (WebLogic).	Modernized	Developers or System Integrators	<p>Developers or System Integrators of Consuming Applications use the VSA Wizard as a design-time tool to create a web service. A consuming application would then use the generated web service at run-time.</p> <p>The developer or system integrator opens a web browser and starts the VSA Wizard, which is the web interface to the VSA web service on WebLogic. The developer or system integrator uses the VSA Wizard Service Descriptor form to:</p> <ul style="list-style-type: none"> • Create a new VSA web service. <p>Or</p> <ul style="list-style-type: none"> • Edit an existing VSA web service.
2	Connect/Transmit RPC data to VMRCS Listener (WebLogic).	Modernized	VSA Wizard (WebLogic Web Service)	<p>VSA Wizard web service on WebLogic:</p> <ul style="list-style-type: none"> • Connects to the VistA M Routine Calling Service (VMRCS) client service listener on WebLogic. • Passes VistA RPC information.
3	Connect/Transmit data to VMRCS Listener (Caché).	Modernized	VMRCS Listener (WebLogic Client Service)	<p>VMRCS listener on WebLogic:</p> <ul style="list-style-type: none"> • Connects to the VMRCS listener on Caché. • Passes VistA RPC information.
4	Connect/Transmit RPC data to VMRCA Listener (VistA).	Modernized	VMRCS Listener (Caché SOAP Web Service)	<p>VMRCS listener on Caché:</p> <ul style="list-style-type: none"> • Invokes the VMRCA listener on VistA. • Passes VistA RPC information. <p> NOTE: VMRCS listener is a Caché SOAP web service that is used only by VSA itself. This is distinct from the SOAP and RESTful services that VSA helps generate.</p>

Business Process ID	Business Process Name	Type	Owner	Description
5	Invoke the RPC (VistA).	Modernized	VMRCA Listener (VistA M Code)	VMRCA listener on VistA: <ul style="list-style-type: none"> Sets up the user environment. Confirms the user is authorized to invoke the RPC, based on the information in the NEW PERSON file (#200), the REMOTE PROCEDURE file (#8994), and the OPTION file (#19). Invokes the RPC.
6	Return Validated RPC data to VMRCA Listener (VistA).	Modernized	VSA Remote Procedure (VistA)	VSA Remote Procedure returns information about Remote Procedure Calls (RPCs): <ul style="list-style-type: none"> Retrieves RPC data from the REMOTE PROCEDURE file (#8994). Returns the data to VMRCA listener.
7	Return RPC data to VMRCS Listener (Caché).	Modernized	VMRCA Listener (VistA M Code)	VMRCA listener on VistA returns the RPC data to the VMRCS listener.
8	Return RPC data to VMRCS Listener (WebLogic).	Modernized	VMRCS Listener (Caché SOAP Web Service)	VMRCS listener on Caché returns VistA RPC information.
9	Return data to VSA Wizard (WebLogic).	Modernized	VMRCS Listener (WebLogic Client Service)	VMRCS listener on WebLogic returns VistA RPC information.
10	Create Service Descriptor file. (WebLogic)	Modernized	VSA Wizard (WebLogic Web Service)	VSA Wizard web service on WebLogic: <ul style="list-style-type: none"> Validates and retrieves user input from the VSA Service Descriptor form. Creates the Service Descriptor file (XML format).

Business Process ID	Business Process Name	Type	Owner	Description
11 & 12	Create Java Source Code (WebLogic)	Modernized	VSA Service Code Generator	<p>VSA Service Code Generator:</p> <ul style="list-style-type: none"> • Extracts data from the Service Descriptor file in XML format (11). • Generates the (transient) Java and XML source code files (12) for SOAP or RESTful web service • Creates SOAP service files: <ul style="list-style-type: none"> ○ Creates the Java API I (Java Standard Library) for XML SOAP Web Services (JAX-WS) annotated Java class (the .java source file) that defines the SOAP-based service and its operations. ○ Creates WEB-INF/web.xml. ○ Creates /applicationContext.xml (for Spring injection). ○ Creates pom.xml. ○ Creates RESTful service files. ○ Creates Java API (Java Standard Library) for XML REST Web Services (JAX-RS) annotated Java and XML files that define the RESTful service and its operations. ○ Creates WEB-INF/web.xml. ○ Creates WEB-INF/weblogic.xml. ○ Creates pom.xml.
13 & 14	Create WAR file (WebLogic)	Modernized	VSA Service Assembler	<p>VSA Service Assembler:</p> <ul style="list-style-type: none"> • Inputs the Java and XML source code files (13). • Compiles the Java class and bundles everything into a WAR file (undeployed VSA web service), including the required JAR files (both third-party and VSA JARs) under WEB-INF/lib (14). • Deletes the (transient) Java source code files.

Business Process ID	Business Process Name	Type	Owner	Description
15 & 16	Deploy WAR file (WebLogic)	Modernized	VSA Service Deployer	VSA Service Deployer: <ul style="list-style-type: none"> Inputs the undeployed VSA web service WAR file (15). Deploys the VSA-generated web service to the WebLogic Server.

2.3 Business Benefits

In recent years, an increasing priority has been placed on the modernization of VA computing environments. This includes the integration of VA computing systems with:

- COTS products that extend business functionality
- External systems, such as Department of Defense (DoD), Indian Health Service (IHS), etc.

This modernization would be done across the VA business domains:

- Healthcare—Veterans Health Administration (VHA)
- Benefits—Veterans Benefits Administration (VBA)
- Memorial and Burial Affairs—National Cemetery Administration (NCA)

Previous approaches for VistA SOA implementation have included:

- Replacement (e.g., HealtheVet).
- Encasement—Variety of solutions that attempt to compensate for the technical/design differences of VistA by providing connectivity adaptors.

These approaches have been less than cost effective and in the case of the latter set of designs have been problematic in terms of the following:

- System performance
- Software redundancy
- Security
- Overall sustainability

There is a recognized need for a VistA approach that provides a simplified (yet architecturally correct) technical solution, as well as addressing the organizational challenge of performing application development across multiple technical environments (e.g., traditional M computing and modern object-oriented web services design). Efforts to resolve this challenge have been exacerbated by an increasing number of high priority efforts involving the integration of disparate systems suited to the service-oriented architecture approach.

Both Health Level Seven (HL7) messaging and web services are regarded as methods for creating SOA-compliant services. And while VistA has implemented and used HL7 messaging for several decades, the ability to interact directly from VistA in the form of web services has been elusive.

There is also a need to provide a solution that alleviates the need for VistA M developers to understand Java to create web services, as well as to provide an automated VistA SOA service assembly methodology that minimizes to the extent possible, further development burden on VistA applications to expose VistA logic as Remote Procedure Calls (RPCs).

In addition to bridging the technology gap between traditional M computing and the object-oriented/web services-oriented design, the solution needs to facilitate the location of patient data across the 128 decentralized VistA systems and aggregate query responses returned to requesting systems. Similarly, there is a need to combine fine grained VistA methods into more coarsely grained or composite services that reduce network "chattiness" between VistA SOA Services and consuming applications.

VSA helps solve these overarching business needs. As the software continues to improve with *future* phases/iterations, all business needs can be met.

2.3.1 Business Customer—Key Stakeholders

Table 9. VSA Business Customer—Key Stakeholders

Name	Role	Role Description and Assigned Task
[REDACTED]	Executive Sponsor Acting Deputy Chief Information Officer (DCIO) Product Development ADCIO of Development Management (DM) ADCIO for Product Support (PS)	Serves as Office of Information and Technology (OI&T) project advocate, assists with obtaining funding for the project.
[REDACTED]	Business Sponsor Supervisory Management & Program Analyst. VSA Integrated Project Team (IPT) Member	Serves as the project advocate who drives the business strategy, priorities, and assesses risk and business objectives.
[REDACTED]	Technical Analysis Competency Manager VSA Integrated Project Team (IPT) Member	Oversees program and VSA project team.
[REDACTED]	Integration Architect Office of Information and Technology (OI&T) Architecture, Strategy and Design (ASD)	VSA Integration Architect consultant.
[REDACTED]	VSA Integrated Project Team (IPT) Chair	Oversees VSA IPT and project team.

Name	Role	Role Description and Assigned Task
	VSA Project Manager	
	CBO; Director Veteran Point of Service (VPS) Project	Reference Implementation—As an exercise to showcase VSA and demonstrate the VSA tool. VSA is creating test web services of specific Use Cases for the Veteran Point of Service (VPS) Project.
	Co-Director Connected Health VHA Office of Informatics & Analytics Representative	Reference Implementation—As an exercise to showcase VSA and demonstrate the VSA tool. VSA is creating test web services of specific Use Cases for the VistA Evolution/Mobile Development; Patient Viewer Phase 2 Project.



REF: For a detailed list of other stakeholders, see the VSA Integrated Project Team (IPT) Charter.

2.3.2 VSA Project Team Members

Table 10. VSA Project Team Members (team members are listed by role; within a role, names are listed in alphabetical order)

Name	Role	Role Description and Assigned Task
	Integrated Project Team (IPT) Chair Project Manager	Oversees VSA project team and IPT.
	Project Planner	VSA project planning tasks.
	Risk Manager Facilitator	Risk analysis tasks. Assists with VSA project oversight.
	Functional Analyst Recorder	All functional analysis tasks (e.g., requirements).
	Functional Analyst	To Be Determined (TBD): All functional analysis tasks.
	Technical Analyst	TBD: Technical Analyst group.
	Technical Analyst	TBD: Technical Analyst group.
	Technical Analyst	All analysis tasks (e.g., requirements).
	Developer: Java/Other	Develops, Enhances, and Maintains: <ul style="list-style-type: none"> Service Code Generator. Service Assembler using Apache® Maven 3.1.


Name	Role	Role Description and Assigned Task
[REDACTED]	Developer Consultant Developer: Java/Other	Develops, Enhances, and Maintains as consulted.
[REDACTED]	Development Technical Lead Developer: Java/Other	Develops, Enhances, and Maintains: <ul style="list-style-type: none"> Federating Components. VistaServiceInvoker. SSL Testing. VSA Wizard.
[REDACTED]	Developer: Java/Other	TBD
[REDACTED] K [REDACTED]	Developer: Java/Other	Developed, Enhanced, and Maintained: <ul style="list-style-type: none"> VMRCS Listener. Vista M Routine Calling Adapter (VMRCA) listener.
[REDACTED]	Development Technical Lead Developer: Java/Other	Develops, Enhances, & Maintains: <ul style="list-style-type: none"> VSA Wizard. Vista M Routine Calling Service (VMRCS) Listener. Federating Components.
[REDACTED]	Developer: Vista/M	Develops, Enhances, and Maintains: <ul style="list-style-type: none"> VMRCA Listener. RPC Security. RPC Service for Wizard
[REDACTED]	Developer: Vista/M	Develops, Enhances, and Maintains: <ul style="list-style-type: none"> VMRCA Listener RPC Security. RPC Service for Wizard
[REDACTED]	System Administrator Developer: Java/Other	Performs/Maintains: <ul style="list-style-type: none"> Integration Tests. SoapUI Tests. Master Patient Index (MPI) Mock Service. Continuous Integration. VSA Test Environments Setup: <ul style="list-style-type: none"> Temporarily working with the Repositories project team (e.g., Health Data Repository [HDR]) and their environment to borrow their Virtual Machine (VM) servers for VSA. Working with Enterprise Development Environment (EDE) to set up sanctioned and managed VM servers for VSA:




Name	Role	Role Description and Assigned Task
		<ul style="list-style-type: none"> ▪ Linux: Two (2) VM. ▪ Windows: Eight (8) VM ("Dev-in-a-box;" primarily for contractors without Government Furnished Equipment (GFE) laptops. <p>The EDE Data Center is managed by:</p> <ul style="list-style-type: none"> ▪ VA—Enterprise Operations (EO) ▪ Contractor—MITRE Corporation <p>Possible server location: Austin Information Technology Center (AIRC).</p>
	Software Quality Assurance (SQA)/Test Analyst	Quality assurance and testing VSA.
	SQA/Test Analyst	Quality assurance and testing VSA.
	Integration Control Registration (ICR) Processing Lead	Downstream partner with VSA.
	Technical Writer	<p>Reviewing, Editing, and Maintaining:</p> <ul style="list-style-type: none"> • PMAS/ProPath-related project documentation (e.g., Charters, BRD, RSD, and SDD). • End-User documentation: <ul style="list-style-type: none"> ○ <i>VSA Installation Guide</i> ○ <i>VSA Developer's Guide</i> ○ <i>VSA Systems Management Guide</i> ○ <i>VSA Wizard User Guide</i> ○ VSA Wizard Online Help

2.3.3 VSA Software Components

[Table 11](#) lists the VSA software components:

Table 11. VSA Software Components

Component	Description
VSA Wizard	<p>This web application provides a web interface that:</p> <ul style="list-style-type: none"> • Accepts user input of information required to create a SOAP or RESTful service based on one or more VistA RPCs. • Saves that information in a Service Descriptor file in XML format. • Invokes the service generator to create the web service that invokes those RPCs. <p>This application is written using the Spring MVC 3.2.4 (Model-View-Controller [MVC] framework).</p>
RPC Service for Wizard	<p>An internal VSA web service that the VSA Wizard uses to interrogate VistA for available RPCs. It retrieves all of the details for a given RPC to pre-populate most of the VSA Wizard fields.</p> <p>The RPC Service for Wizard web service allows users to display a list of RPCs by pressing the Search button. The system populates the "Results" field with the list of RPCs returned from VistA.</p>
VistaServiceInvoker	<p>This Java class does the following:</p> <ul style="list-style-type: none"> • Takes inputs from the generated business service. • Packages the inputs into a proper instruction object. • Calls the VistA M Routine Calling Service (VMRCS) listener. <p> NOTE: This class is specific to VSA services.</p>
VistA M Routine Calling Service (VMRCS) Listener	<p>VMRCS listener is an internal Caché web service written and deployed in a VistA instance:</p> <ul style="list-style-type: none"> • Serves as an HTTP Listener using Caché objects. • Employs a two-way SSL connectivity and is responsible for delegating requests to run M routines to the VistA M Routine Calling Adaptor (VMRCA) listener. • Facilitates the use of M routines as the basis for VistA SOA service application business logic. • Transforms payloads between M routine compatible syntax and various external formats (e.g., XML and JavaScript Object Notation [JSON]), and type conversions to JSON, etc.
VMRCS Client Listener	<p>The VMRCS client listener is used by VSA-generated services to invoke the VMRCS Web Service listener on the Caché system.</p>
VistA M Routine Calling Adapter (VMRCA) Listener	<p>A set of M routines bundled as a VSA package (VSA namespace = "XSA"), which:</p> <ul style="list-style-type: none"> • Integrates the VMRCS listener component with the traditional VistA M computing environment. • Checks a VMRCS instruction for correctly formatted and required legacy

Component	Description
	<p>process invocation information.</p> <ul style="list-style-type: none"> Builds the legacy M API signature to execute the legacy process and returns the results to the VMRCs listener. Performs Kernel authentication. Provides for the invocation of M routines.
Service Code Generator	<p>This internal VSA module extracts data from the *Service Descriptor XML file and generates the necessary Java classes and artifacts that are later compiled and combined with other libraries into a deployable web service WAR by the Service Assembler.</p> <p> NOTE: The Service Descriptor file in XML format is the key configuration file that is created via the VSA Wizard.</p>
Service Assembler	<p>This internal VSA component takes all source Java classes that define a service and combines them with other necessary dependencies to produce a deployable WAR file.</p>
Service Deployer	<p>This internal VSA component deploys the WAR file generated by the Service Assembler to the VSA Federating Platform.</p> <p> NOTE: This component currently uses Apache® Maven 3.1; however, it could change to the WebLogic Deployment API in the <i>future</i>.</p>
Federating Classes	<p>Federating classes sit between the generated business service and the VistaServiceInvoker. For example, for MPI/MVI the Federating class knows how to:</p> <ul style="list-style-type: none"> Talk to MPI/MVI and parse the results. <ul style="list-style-type: none"> Interpret the MPI/Master Veteran Index (MVI) results. Address the necessary VistA systems. Multi-task and run the requests in <u>parallel</u>. Assemble the results (for now just a simple merge). <p> NOTE: Working with MPI/MVI to get ID list of sites visited by a patient for "Federation" needs.</p>

2.4 Assumptions and Constraints

2.4.1 Design Assumptions

VSA provides the basic (uncomplicated) functionality necessary to automate the creation of VistA SOA services and the infrastructure components necessary to support and operate those services.

This document is written with the assumption that the reader is familiar with the following, but it does *not* require in-depth knowledge of each in order to use the VSA Wizard:

- VistA computing environment:
 - Kernel 8.0—VistA M Server software
 - Kernel Toolkit 7.3—VistA M Server software
 - VA FileMan 22.0 data structures and terminology—VistA M Server software
 - Remote Procedure Call (RPC) Broker 1.1—VistA Client/Server software
 - VistALink 1.6—VistA Server and Middle Tier Server software
- Microsoft® Windows Server 2008 environment
- Red Hat Enterprise Linux release 6.5 environment
- Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5 environment
- M programming language (e.g., Remote Procedure Calls [RPCs] and Application Programming Interfaces [APIs])
- Java programming language

2.4.2 Design Constraints

Hardware resources (e.g., development/test servers and databases) are *not* currently dedicated to VSA, so special permissions and accommodations have been made with the following organizations:

- Repositories Project Team (e.g., Health Data Repository [HDR])—temporarily supplied a development environment and access to their Virtual Machine (VM) servers for VSA.
- Enterprise Development Environment (EDE)—Set up sanctioned and managed development environment with access to VM servers, which includes the following platforms:
 - Linux: 2 VM.
 - Windows: 8 VM ("Dev-in-a-box;" primarily for contractors without Government Furnished Equipment (GFE) laptops.

The EDE Data Center is managed by:

- VA—Enterprise Operations (EO)
- Contractor—MITRE Corporation

Possible server location: Austin Information Technology Center (AITC)



NOTE: EDE charges for their setup and maintenance services, so VSA needs established funding to proceed with this route.

2.4.3 Design Trade-offs

VA objectives prioritize the implementation of SOA as an Enterprise design. The VSA design was identified as the preferred architectural design for SOA-based systems going forward. The intent is to eventually replace interim connectivity design approaches (e.g., Medical Domain Web Services [MDWS], VistA Integration Adapter [VIA], Clinical Data Services [CDS], etc.). In the long-term, VSA satisfies the consumer needs currently supplied by those other design approaches. It significantly enhances:

- VistA security.
- System performance.
- Sustainability.

The VSA Phase 2 software is a solution for developing web services for VistA RPCs. The VSA Phase 2 software design addresses the following factors:

- Performance
- Reliability and robustness
- Usability

Future VSA software iterations will enhance functionality and improve the following factors:

- Flexibility
- Interoperability
- Section 508 Conformance

Any design trade-offs, additional features, and enhanced functionality will be addressed in *future* phases/iterations of VSA.



REF: For a list of some functionality and design modifications to be included in *future* phases/iterations, see [Table 5](#).

2.5 Overview of the Significant Requirements



REF: For a more comprehensive list of business/functional and system requirements, see the *VSA Business Requirements Document (BRD)* and *VSA Requirements Specifications Document (RSD)*.

2.5.1 Overview of Significant Functional Requirements

Initial features of the VSA Wizard application tool will allow developers or system integrators at design-time to enter and manage Service Descriptor information, and retrieve the necessary details of an RPC from VistA used to auto-generate a web service. Execution of VSA-generated services by consuming and providing applications is dependent on the features provided by the following infrastructure components:

- VistA SOA Federating Services Platform
- VMRCs Listener
- VMRCs Listener



REF: For details of the functional requirements for the VSA Wizard design-time and run-time processes, see the “Functional Specifications” and “System Features” sections in the *VSA Requirements Specifications Document (RSD)*.

2.5.2 Overview of Functional Workload/Performance Requirements

VSA functional workload and performance requirements is dependent on the final topology chosen for deployment. The two topologies currently under consideration:

- Centralized—This is the “As Is”, interim topology.
- Distributed—This is the “To Be”, *future* topology; as intended by the original architecture documents.

In both cases, VSA performance is dependent on the location and responsiveness of its external dependencies.

[Figure 4](#) shows the external dependencies for the VSA Federating Platform. In the centralized topology, both the ESB and the VSA Federating Platform will be located in a central location and will have direct and equal access to the VistA systems.

In the distributed topology, both the ESB and VSA Federating Platforms will be located at each VA Regional Data Processing Center (RDPC)/Defense Enterprise Computing Center (DECC). A given instance of the VSA Federating Platform will have direct access to the VistA instances deployed at an RDPC/DECC, but the VSA Federating Platform will route requests to a corresponding VSA Federating Platform at the corresponding RDPC/DECC.

As represented in [Figure 4](#), VSA *has* the ability to:

- Define (expose) services on the ESB.
- Establish Transport Layer Security (TLS) between the ESB and the VSA Federating Platforms.
- Configure a fully functioning SOA architecture.

Figure 4. VSA External Dependencies for the Federating Platform—Centralized (“As Is”)

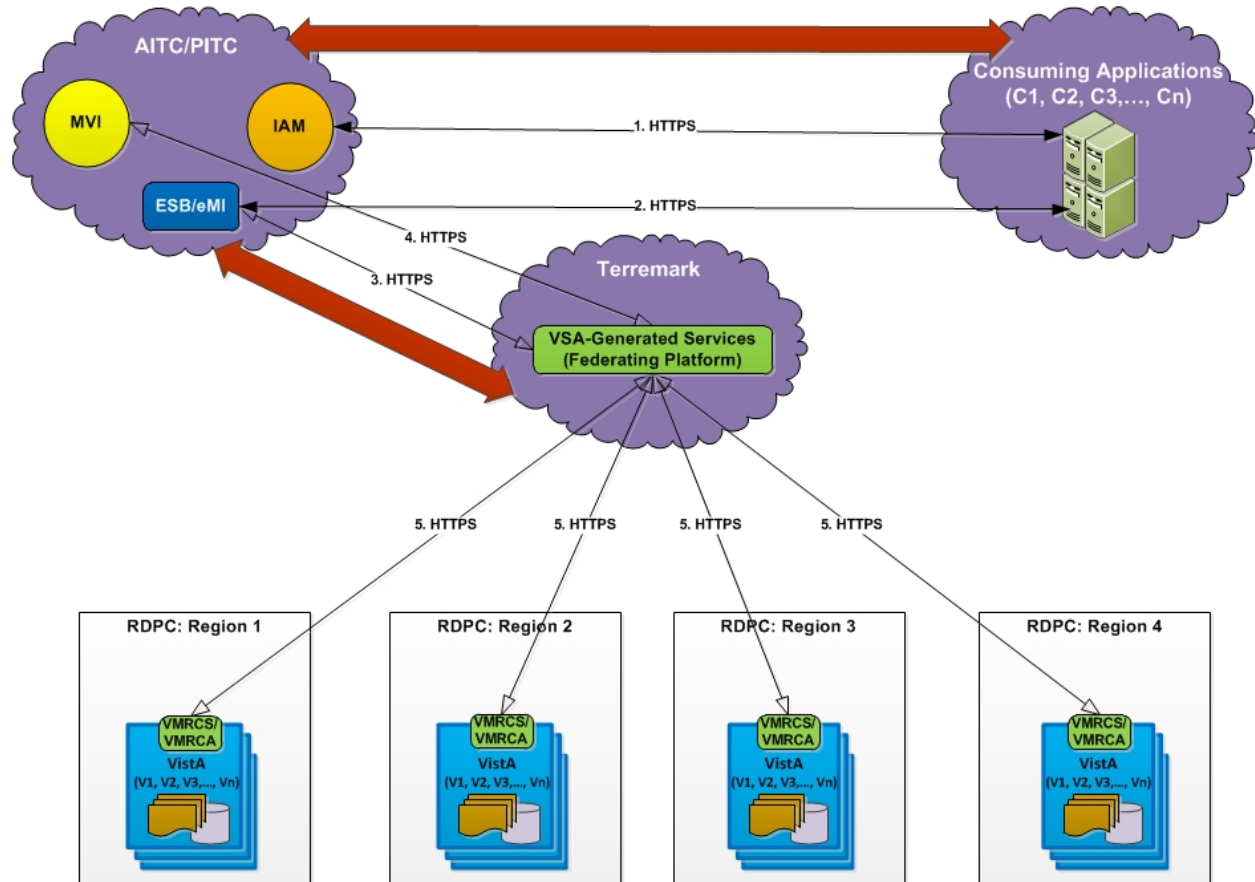
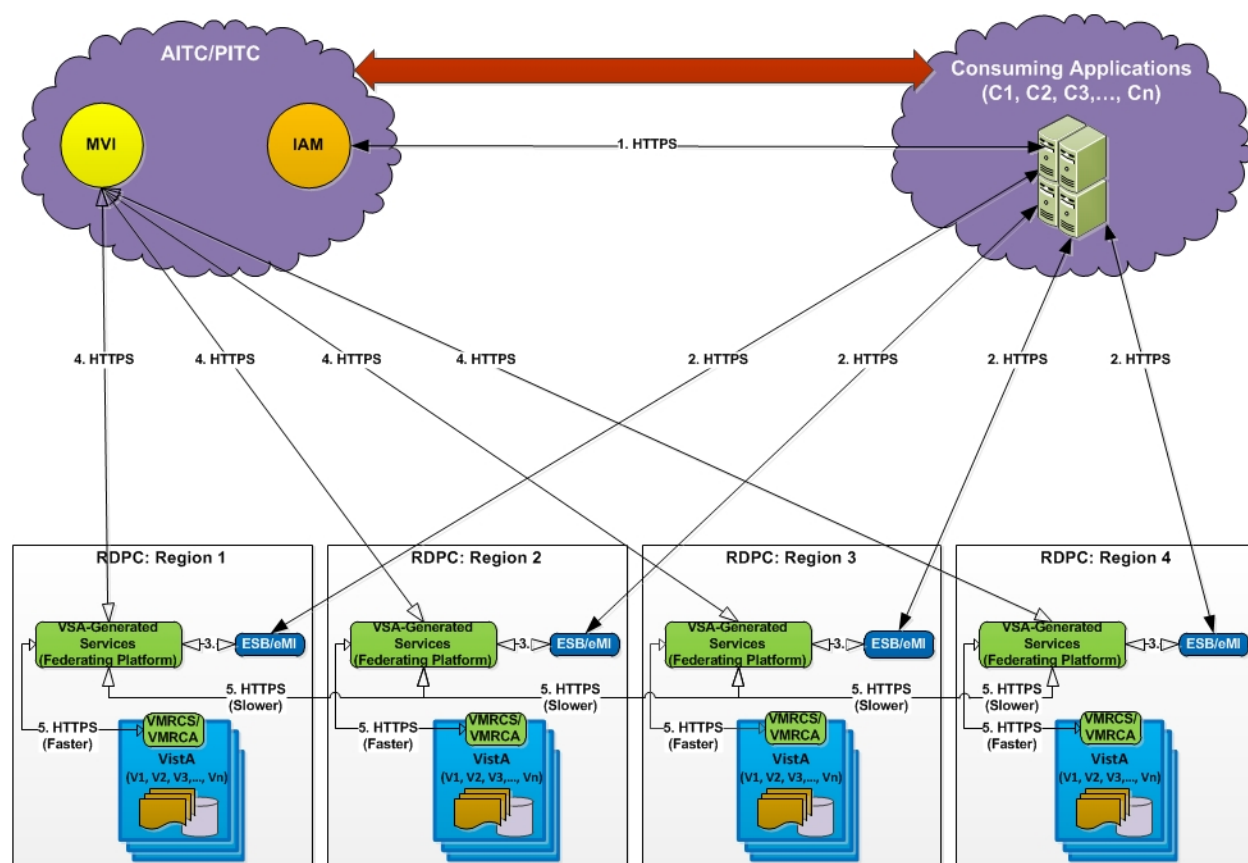


Figure 5. VSA External Dependencies for the Federating Platform—Distributed (“To Be”)

In order to meet the requirements, defined in the *VSA Requirements Specification Document (RSD)*, the VSA Federating platform will use industry standard application servers capable of handling:

- Multiple concurrent connections.
- Messages of various sizes.

VSA will execute performance, capacity, and independent testing of the VSA product, and as part of SQA analysis and testing. Recurring discussions with the Systems Engineering and Design Review (SEDR) workgroup are in progress to articulate requirements for performance and capacity transactions, including the specifications to stand up the environments at Terremark (*Non-VA*) as the “Initial” implementation of VSA. VSA will use HP Performance Lab testing tools (including LoadRunner) and services provided by the Enterprise Testing Services (ETS) group. Performance testing will be conducted at the Bay Pines Test Lab.

VSA is currently working with the Health Care Delivery Products (HCDP)/Service Integration Office (SIO) to gain access to a *Non-Production* Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI) environment. VSA will test the framework to include 150 VistA systems and consumers of data to/from VistA including (but *not* limited to) Kiosks (placed around all VHA health care facilities (about 1500), Enrollment (ESE – at AITC), and other applications that currently use MDWS, etc.



REF: For details of the VSA performance requirements, see the “Performance Specifications” section in the *VSA Requirements Specifications Document (RSD)*.

2.5.2.1 Multiple Concurrent Clients

- VSA can handle requests from multiple concurrent clients using two HTTP protocols:
 - SOAP
 - REST
- VSA can handle request directly from the consumer or through the ESB.
- VSA can handle many requests through each separate client connection or through a single connection from the ESB.

2.5.2.2 Response Time

The response time measured for VSA is dependent on the response time of the existing VistA RPC mechanism, since the VSA Listener (VMRCS) invokes Remote Procedure Calls (RPCs).



NOTE: In the *future*, VSA will have the capability of invoking other VistA functionality (e.g., routine APIs).

Therefore, the combined response time measured for VSA will include the response time for the following:

- ESB
- VSA Federating Platform.
- VSA Listeners (i.e., VMRCS and VMRCA).
- VistA functionality (RPC).

The VSA system has an average response time of 1 second; it will *not* to exceed 3 seconds in more than 0.1% of cases.

2.5.2.3 Throughput

VSA can handle a message throughput rate equivalent or better than the throughput rate documented for the ESB, which is 1500 messages per second for message sizes of 1KB to 1MB.

2.5.2.4 Scalability

The VSA implementation does *not* go from “zero to 60” on day one. Loads increase incrementally as client applications use VSA, requiring the need for capacity to grow over time. This allows usage to be throttled, so it does *not* overtake the scale up/out capacity.

- VSA allows for a scalable implementation as specified in the Program Requirements.
- VSA allows for *future* functional requirements to be integrated into the system without major changes to the hardware and software package.
- VSA will be fully integrated with existing systems within VistA. VistA instances can be located locally at the facility or in remote data centers as either an integrated or a single instance per site database. Additionally, VistA instances may reside on servers running various operating systems including but *not* limited to Windows, VMS, and Linux running InterSystems Caché.

2.5.2.5 Reliability

- VSA ensures the highest degree of reliability and accuracy of data collection based on existing VA defined business rules.
- All measurable reliability requirements for VSA have *not* been determined, nor have all specific reliability requirements been cited by the stakeholder. However, VSA provides a very high degree of reliability and accuracy.



REF: For details of the VSA functional workload and performance requirements, see the “Performance Specifications” section in the *VSA Requirements Specifications Document (RSD)*.

2.5.3 Overview of Operational Requirements

2.5.3.1 Scalability

- VSA allows for *future* functional requirements to be integrated into the system without major changes to the hardware and software package.
- VSA will be fully integrated with existing systems within VistA. VistA instances can be located locally at the facility or in remote data centers as either an integrated or a single instance per site database. Additionally, VistA instances can reside on servers running various operating systems including but *not* limited to Windows, VMS, and Linux running InterSystems Caché.

2.5.3.2 Failure

- VSA incorporates a strategy for retries and allows Service Providers and Service Consumers to handle incomplete sets of responses in a Federating request response.
- VSA includes in the aggregated responses an indication of a failure to response from any of the federated VistA systems.

The following VSA operational requirements are taken from the *VSA Business Requirements Document (BRD)*:

Table 12. Operational Requirements

Operational Environment Requirements
VSA provides sub-second performance for the execution of system connections and logic within the boundaries of the VSA infrastructure utilities.
Maintenance, including maintenance of externally developed software incorporated into the VSA utilities use system (contingency/failover, etc.) redundancy to ensure that scheduled maintenance does <i>not</i> cause system down time.
Information about response time degradation resulting from unscheduled system outages and other events that degrade system functionality and/or performance will be disseminated to the user community within 30 minutes of the occurrence. The notification will include the information described in the current Automated Notification Reporting (ANR) template maintained by the VA Service Desk. The business impact will also be noted.
A real-time monitoring solution will be provided for on-demand evaluation of system performance during normal operation or when technical issues/problems occur that may require a remediation.
System implementation includes Continuity of Operations Plan (COOP)/Disaster Recovery (DR) and 24/7 support consistent with organizationally established expectations relative to system availability.
A <i>Continuity of Operations Plan</i> will be created and provided to the technical and user community to follow if/when lapses in system availability occur despite the implementation of COOP/DR and 24/7 support arrangements.

2.5.3.3 Disaster Recovery (DR)

VA Enterprise application backup and restore procedures are to be followed for VSA, including:

- Application servers
- Service descriptors data
- Web service data

Disaster Recovery (DR) specifications will be determined after the VSA Increment 1 is brought into the Terremark environment as the near-term path for production implementation. The Contingency/Disaster Recovery (DR) environment will be hosted in the Terremark Nation Access Point (NAP) of the Americas facility in Miami, FL. The VSA project inherits the DR procedures of the VA hosting environment supporting the application.

It is anticipated that the *future* topology/implementation of VSA will be OI&T controlled, at the Austin Information Technology Center (AITC) and/or regional data centers. This system will inherit the data center's general DR specifications and plans.

Analysis will be necessary to develop complete DR requirements, to include assessment of the data stores on each server supporting VSA. There will be one-to-many and many-to-many relationships between data

stores. Disaster Recovery will require an analysis of the data stores used and built along the VSA path. Analysis to include the “what-if’s” for the combinations of failure that can occur at the steps along the path and how to recover so data store integrity is restored at all pieces of VSA.

VSA architecture does *not* have a database typical of regular applications. The current implementation saves generated xml files (service descriptors) within a deployable war file. Recovery of those war (and the xml) files can be determined, and in an alternate implementation those xml or war files could be saved in a centralized registry or database.

VSA accounts for synchronizing related data on Caché and WebLogic sides (e.g., VSA Logging). VSA Phase 2 explores such point-to-point relationships to determine the specifications.



REF: For details of other VSA operational requirements, see the “Disaster Recovery Specification” and “Reliability Specifications” sections in the *VSA Requirements Specifications Document (RSD)*.

2.5.4 Overview of the Technical Requirements

The VSA implementation uses software products in accordance with the VistA architecture and the VA Technical Reference Mapping (TRM) standards. If a chosen software component *cannot* meet any of the significant requirements, VSA will evaluate alternate software products to meet the requirements.



REF: For the complete list of standards used, see the TRM website: [\[REDACTED\]TRMHomePage.asp](#)

[Table 13](#), to the extent that they are known, provides a list of the pivotal (i.e., that force design decisions) technical requirements that drive the conceptual design.

Table 13. Technical Requirements

ID	Requirement
TBD	VSA uses WebLogic Server 12c (Release 12.x), which is an approved Java EE application server that hosts both SOAP and REST web services. It is expected that WebLogic meets all the performance requirements stated in this section.
TBD	VSA uses only standards-based SOAP and REST APIs to allow VSA-generated services to be hosted in other Java EE application servers, and uses standards-based Java APIs for both SOAP and REST. The respective standards used are: <ul style="list-style-type: none"> • JAX-WS • JAX-RS

2.5.5 Overview of the Security or Privacy Requirements

VSA will ensure layers of security through the use of system access privileges and authenticated login processes.



REF: For details of other VSA security and privacy requirements, see the “Security Specifications” section in the *VSA Requirements Specifications Document (RSD)*.

In the absence of a comprehensive VistA security model, the design of which is currently under consideration, an ad-hoc approach has been taken with regard to security. The VSA team collaborated with the Common Services Security Engineer to identify and create solutions to security-related risks specific to the functionality identified. It should be noted that the approaches taken as a result of this evaluation to mitigate these risks might *not* achieve optimal results given the ad-hoc nature of the approach. Requirements taken into account are:

- VSA complies with all privacy and security features as mandated by VA, Federal, State, and Local regulations.
- VSA complies with federal guidelines on Personal Health Identifiers (PHI) (Health Insurance Portability and Accountability Act [HIPAA]): [REDACTED] [hipaa/](#).
- VSA Privacy - System adheres to all VA requirements for Release of Information (ROI).
- VSA uses FIPS 140-2 encryption.
- VSA complies with NIST 800-53 Security Controls.

Table 14. Security Requirements

ID	Requirement
TBD	<p>VSA secures the communication between the following:</p> <ul style="list-style-type: none"> • ESB and VSA Federating Platform. • VSA Federating Platform and the VSA Listener (VMRCS). <p>VSA application server is configured to host its services using HTTPS, which uses SSL (TLS) and is the standard transport security protocol for encryption and authentication of both client system and server system.</p>
TBD	<p>VSA requires that the corresponding VSA Federating Platform authenticates using its digital certificate signed by a VA Central Authority to prevent an unauthorized system from invoking the VSA Listener (VMRCS) <i>without</i> the use of VSA Federating Platform.</p>
TBD	<p>VSA enforces user authentication as implemented and required by the Identity and Access Management (IAM).</p> <p>The IAM solution is <i>not</i> available yet; therefore, VSA implements the approved interim solutions as defined in the Identity User Design Patterns.</p> <p> REF: For more information, see the <i>High Level User Identity Design Pattern And Methodology for VistA Integration</i> document (High Level User Identity Design Pattern - VistA_20140416.docx; Author Keith Cox).</p>

ID	Requirement
TBD	The VSA Wizard web application enforces user security if the VSA Wizard is deployed in a centralized and shared environment. However, if the Wizard is deployed in a developer's own environment, the user security may <i>not</i> need to be enforced to allow for rapid design of VSA services.
TBD	<p>The VSA Wizard enforces VistA user security to its corresponding development VistA instance. Again, if the VSA Wizard is deployed in a centralized and shared environment, the VSA Wizard allows the user to specify its VistA development environment and provides the necessary credentials to connect securely with VistA. The current implementation of the VSA Wizard allows for a pair of the following values:</p> <ul style="list-style-type: none"> • DUZ • Station/Division

2.5.6 Overview of System Criticality and High Availability Requirements

The VSA Phase 2 is considered a mission critical system and requires high availability. It follows the same standard procedures used for all critical VistA systems. For example:

- System backups
- VistA disaster recovery



REF: For details of disaster recovery and other VSA operational requirements, see the [“Overview of Operational Requirements”](#) section.

2.5.7 Single Sign-on Requirement



NOTE: It is expected that the Identity and Access Management (IAM) solution will provide single sign-on mechanisms for VSA. Details will be provided in the *future*.

VSA Phase 2 does *not* mandate a single sign-on solution (i.e., Kerberos, and Active Directory). There are no VSA project requirements at this time that limit or inhibit use of the mandated single sign-on solution. Regarding other signon requirements, VSA needs to receive the user key data that allows identification of the user on the VistA instance where the legacy service is exposed.

The VSA Wizard is a tool to be used by developers at design-time, so there is no immediate need for single sign-on. When Identity and Access Management (IAM) and Kernel provide the Single Sign-on Internal (SSOi) solution, VSA can incorporate it into the VSA Wizard.



NOTE: Using Active Directory for signing into the VSA Wizard could be desirable, eliminating the need to remember yet another set of credentials.

For the Federating Platform at run-time, VSA will use the "interim" User Identity Design Pattern; then use the Single Sign-on External (SSOe) or Single Sign-on Internal (SSOi) solution being developed by IAM and Kernel.



REF: For more information on SSOi/SSOe at the VA Enterprise-level, see the Architecture, Strategy, and Design (ASD) *SOA Authentication* document on the eMI (a.k.a. SOA Suite) SharePoint site: [REDACTED]

2.5.8 Requirement for Use of Enterprise Portals

The VSA user interface uses approved Enterprise portals.

2.5.9 Special Device Requirements

VSA does *not* require the use of any special devices to be part of this system.

2.6 Legacy System Retirement

Not Applicable (N/A). VSA is *not* replacing any legacy Veterans Health Information Systems and Technology Architecture (VistA) systems.

3 Conceptual Design

This section of the SDD provides details about the following topics:

[Conceptual Application Design](#)

[Conceptual Data Design](#)

[Conceptual Infrastructure Design](#)

3.1 Conceptual Application Design

This section provides the conceptual design of the application that is being produced by this project.

3.1.1 Application Context

VSA as an application exists in two contexts:

- Design-time—[Figure 6](#) represents the design-time context and illustrates how VSA exposes VistA functionality as standard web services using a service wizard that generates and assembles the web services.
- Run-time—[Figure 7](#) represents the run-time context and illustrates how the assembled web services are deployed and consumed by future service consumers. It includes the VSA Wizard object (service) and includes each additional system, external service, or data store with which VSA interfaces, interacts, or shares with other systems.

Figure 6: VSA Design-time Context Diagram

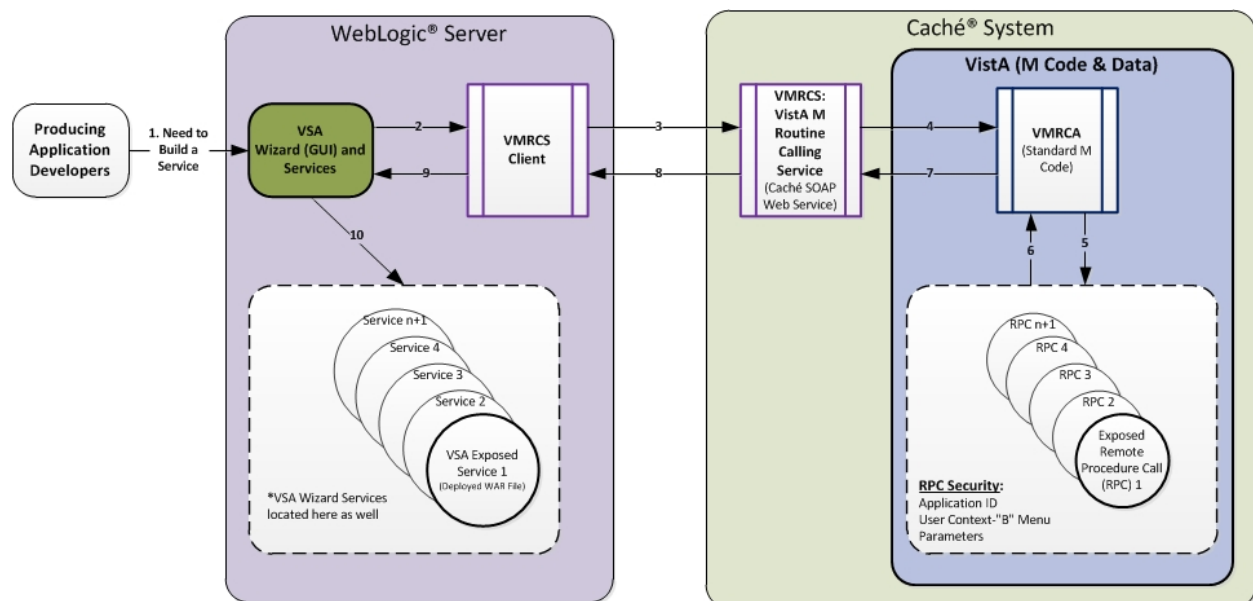


Figure 7. VSA Run-time Context Diagram—VSA Federating Platform Run-time Dependencies (in WebLogic)

Federating Platform (Current):

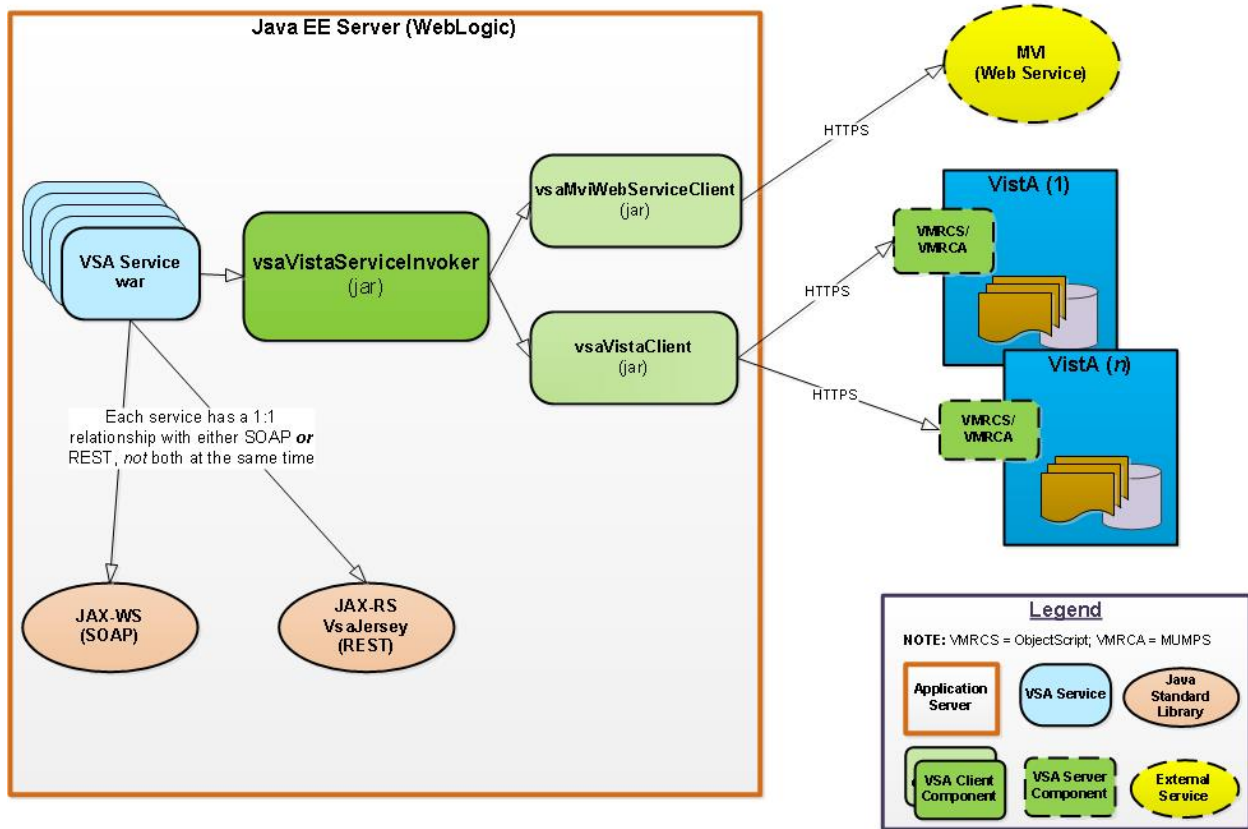


Table 15 through Table 18 describes the information in the application context diagram (Figure 6) in four sections:

Table 15: VSA Application Context Description—Object

ID	Name	Description	Interface Name	Interface System
1	VSA Wizard and Web Service	Web application that provides a web page interface that accepts user input of an RPC definition and generates a *Service Descriptor file in XML format to automatically generate a web service operation that invokes that RPC. Includes underlying services and processes that create the java WAR file and deploys it to WebLogic.	Graphical User Interface (GUI)	<ul style="list-style-type: none"> Vista M Routine Calling Service (VMRCS) Listener (WebLogic) Oracle® WebLogic Server

Table 16. VSA Application Context Description—Interfaces External to OI&T

ID	Name	Related Object	Input Messages	Output Messages	External Party
N/A	N/A	N/A	N/A	N/A	N/A

Table 17. VSA Application Context Description—Interfaces Internal to OI&T

ID	Name	Related Object	Input Messages	Output Messages	External Party
2	VMRCS Listener (WebLogic)	VSA Wizard	<ul style="list-style-type: none"> • RPC Name • RPC Context • Authentication Method • Authentication Value, RPC/API Parameters (Type and Value) • RPC Timeout (Optional, Default=10) • RPC Version 	<ul style="list-style-type: none"> • RPC Name • RPC Context • Authentication Method • Authentication Value, RPC/API Parameters (Type and Value) • RPC Timeout (Optional, Default=10) RPC Version	Enterprise Development Environment (EDE) Data Center WebLogic Server.
3	VMRCS Listener (Caché)	VMRCS Listener (WebLogic)	<ul style="list-style-type: none"> • RPC Name • RPC Context • Authentication Method • Authentication Value, RPC/API Parameters (Type and Value) • RPC Timeout (Optional, Default=10) • RPC Version 	<ul style="list-style-type: none"> • RPC Name • RPC Context • Authentication Method • Authentication Value, RPC/API Parameters (Type and Value) • RPC Timeout (Optional, Default=10) • RPC Version 	OI&T/Caché Server


ID	Name	Related Object	Input Messages	Output Messages	External Party
4	VistA M Routine Calling Adaptor (VMRCA) Listener	VMRCS Listener (Caché)	<ul style="list-style-type: none"> • RPC Name • RPC Context • Authentication Method • Authentication Value, RPC/API Parameters (Type and Value) • RPC Timeout (Optional, Default=10) • RPC Version <p> *NOTE: The VMRCA listener references the XSA DEVELOPER LOGGING and XSA SITE LOGGING parameters, which are part of legacy VistA. These parameters allow a site or a developer to turn on VMRCA process logging.</p>	<ul style="list-style-type: none"> • XSARES := local or global variable reference where RPC results is stored • XSAERROR := local or global variable reference where error data would be stored • When logging is turned on, the VMRCA listener logs an entry in the ^XTMP file. 	OI&T/VistA M Server

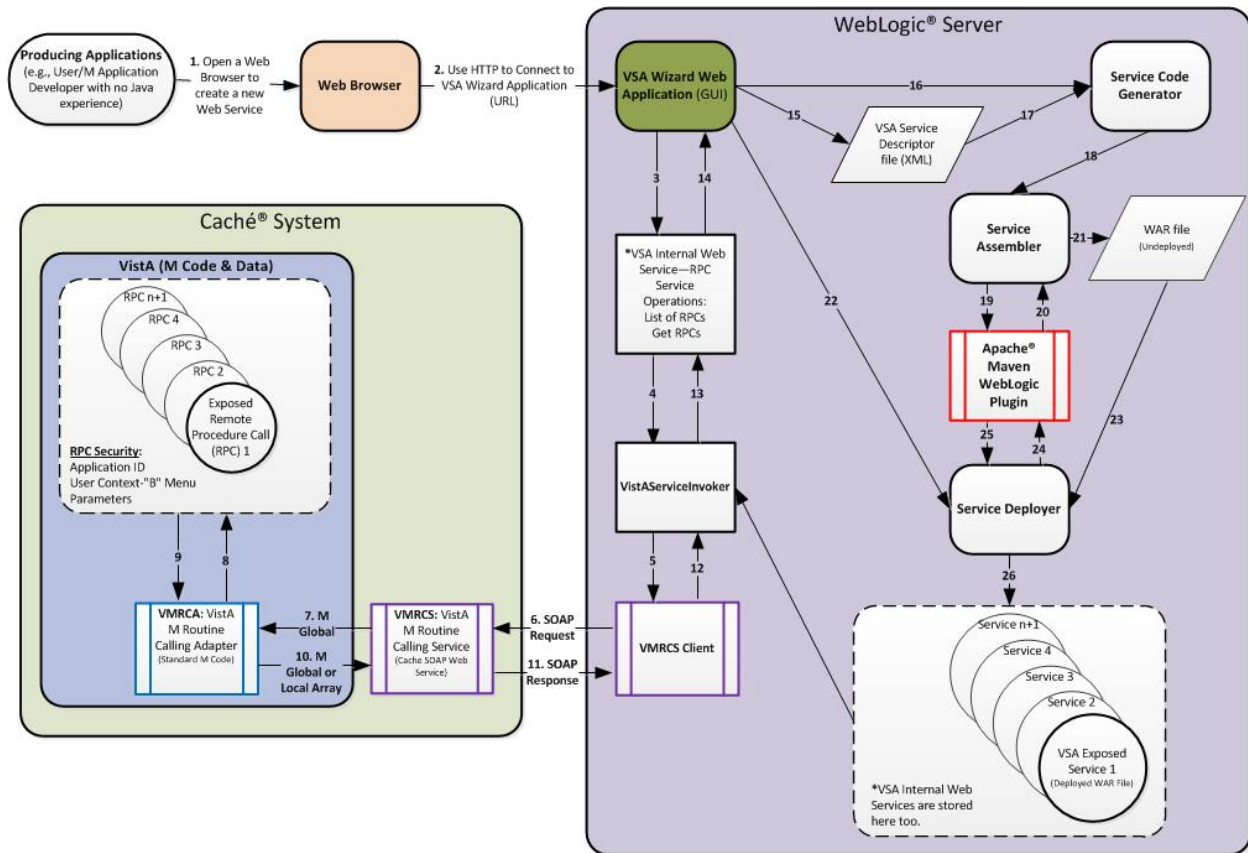
Table 18. VSA Application Context Description—Externally Shared Data Stores

ID	Name	Data Stored	Owner	Access
5	VA FileMan	Stores VistA files and data. VSA retrieves RPC information. VSA provides a set of utilities that provide for the automated generation and execution of SOA-compliant services, using VistA logic. These services support the full range of VistA M logic functionality including the retrieval, creation, editing, and deletion of VistA data—such as is allowed by established VistA application logic as referenced.	OI&T (site-specific)	Read / Write / Delete
6	VA FileMan	This process does <i>not</i> store data but passes/returns RPC information.	OI&T (site-specific)	Read
7	VMRCA Listener	This process does <i>not</i> store data but passes/returns RPC information.	OI&T (site-specific)	Read
8	VMRCS Listener (Caché)	This process does <i>not</i> store data but passes/returns RPC information.	OI&T (site-specific)	Read
9	VMRCS Listener (Client)	This process does <i>not</i> store data but passes/returns RPC information.	OI&T (site-specific)	Read
10	Oracle® WebLogic Server	Stores deployed VSA web services.	EDE Data Center WebLogic server. The EDE Data Center is managed by: <ul style="list-style-type: none"> • VA—Enterprise Operations (EO) • Contractor—MITRE Corporation Possible server location: Austin Information Technology Center (AIRC).	Read / Write (Deploy)

3.1.2 High-Level Application Design

[Figure 8](#) illustrates the VSA high-level application design in the form of a dataflow diagram. It identifies the major components of VSA and the relationships of the major application components to each other and to the surrounding applications. The major components of the VSA application are at the subsystem or top-level service area. Lower-level services are defined and documented in the Logical Application Design.

Figure 8: VSA High-Level Application Design



[Figure 9](#) shows the process flow and structure of the VSA Wizard's functional components that are available to service developer users.

Figure 9. VSA Wizard—Component Flow

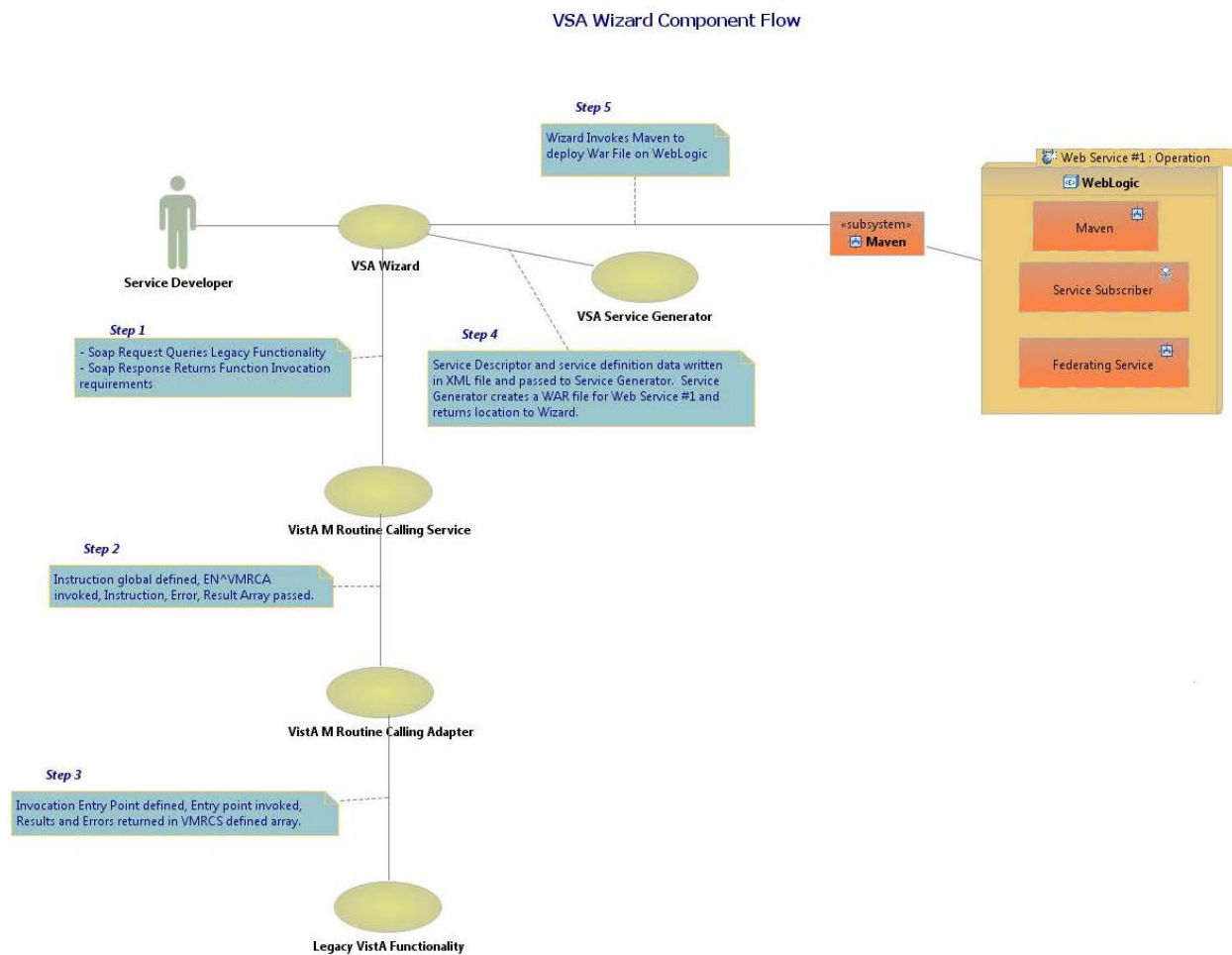




Table 19: VSA High Level Application Design—Objects

ID	Name	Description	Service or Legacy Code	External Interface Name	External Interface ID	Internal Interface Name	Internal Interface ID	SDP Sections 1&2
2	VSA Wizard	Web application that provides a web page Graphical User Interface (GUI) that accepts user input of an RPC definition and generates a Service Descriptor file in XML format to automatically generate a web service operation that invokes that RPC. Also, it has the ability to deploy the web service to the Oracle® WebLogic Server, and for SOAP services provides a link to the WebLogic SOAP Client Tester application.	Service	Web Browser GUI Oracle® WebLogic Server	N/A	<ul style="list-style-type: none"> VSA Internal Web Service Service Code Generator 	N/A	Being Developed
3	VSA Internal Web Service	Service that provides RPC service operations internal to the VSA Wizard: <ul style="list-style-type: none"> List of RPCs Get RPC data 	Service	N/A	N/A	<ul style="list-style-type: none"> VSA Wizard Service VistaServiceInvoker 	N/A	Being Developed

ID	Name	Description	Service or Legacy Code	External Interface Name	External Interface ID	Internal Interface Name	Internal Interface ID	SDP Sections 1&2
4	VistaServiceInvoker	<p>This Java class does the following:</p> <ul style="list-style-type: none"> • Takes inputs from the generated business service. • Packages the inputs into a proper instruction object. • Calls the Vista M Routine Calling Service (VMRCS) Listener. 	Service	N/A	N/A	<ul style="list-style-type: none"> • VSA Internal Web Service • VMRCS Client Listener 	N/A	Being Developed
5	VMRCS Client Listener	The VMRCS client listener resides on WebLogic and is needed to communicate with the actual VMRCS Web Service listener in Caché.	Service	N/A	N/A	<ul style="list-style-type: none"> • VistaServiceInvoker • VMRCS Listener (Caché SOAP Web Service) 	N/A	Being Developed

ID	Name	Description	Service or Legacy Code	External Interface Name	External Interface ID	Internal Interface Name	Internal Interface ID	SDP Sections 1&2
6	VMRCS Caché Listener	<p>VMRCS listener is an internal Caché web service written and deployed in a VistA instance that facilitates the use of M routines as the basis for VistA SOA service application business logic.</p> <p>VMRCS listener is a <i>private</i> service that can only be consumed by VSA-generated services.</p> <p>It employs a two-way SSL connectivity and is responsible for delegating requests to run M routines to the VistA M Routine Calling Adaptor (VMRCA) listener.</p> <p>Transformation of payloads between M routine compatible syntax and various external formats (e.g., XML and JavaScript Object Notation [JSON]), and type conversions to JSON, etc.</p>	Service	N/A	N/A	<ul style="list-style-type: none"> • VMRCS Client Listener • VMRCA Listener 	N/A	Being Developed

ID	Name	Description	Service or Legacy Code	External Interface Name	External Interface ID	Internal Interface Name	Internal Interface ID	SDP Sections 1&2
7	VMRCA Listener	<p>A set of M routines bundled as a VSA package (VSA namespace = "XSA"), which:</p> <ul style="list-style-type: none"> Integrates the VMRCS listener component with the traditional VistA M computing environment. Provides for the invocation of M routines. <p>VMRCA's listener functionality ensures:</p> <ul style="list-style-type: none"> Kernel authentication. Execution of M routines. 	Service	N/A	N/A	<ul style="list-style-type: none"> VMRCS Listener (Caché SOAP Web Service) VistA Remote Procedures, M, and APIs 	N/A	Being Developed

ID	Name	Description	Service or Legacy Code	External Interface Name	External Interface ID	Internal Interface Name	Internal Interface ID	SDP Sections 1&2
15	VSA Service Code Generator	<p>This internal VSA application parses Java source code from the Service Descriptor file in XML format and generates the necessary Java classes and artifacts that are later compiled and combined with other libraries into a deployable web service WAR by the Service Assembler.</p> <p> NOTE: The Service Descriptor file in XML format is the key configuration file that is created via the VSA Wizard.</p>	Service	N/A	N/A	<ul style="list-style-type: none"> VSA Wizard Service VSA Service Assembler 	N/A	Being Developed
18	VSA Service Assembler	<p>This internal VSA component takes all source Java classes that define a service and combines them with other necessary dependencies to produce a deployable WAR file.</p> <p> NOTE: This component uses Apache® Maven 3.1.</p>	Service	N/A	N/A	VSA Service Code Generator	N/A	Being Developed




ID	Name	Description	Service or Legacy Code	External Interface Name	External Interface ID	Internal Interface Name	Internal Interface ID	SDP Sections 1&2
22	VSA Service Deployer	<p>This internal VSA component deploys the WAR file generated by the Service Assembler to the VSA Federating Platform.</p> <p> NOTE: This component currently uses Apache® Maven 3.1; however, it could change to the WebLogic Deployment API in the future.</p>	Service	N/A	N/A	<ul style="list-style-type: none"> VSA Service Assembler Oracle® WebLogic Server 12c (Release 12.x) 	N/A	Being Developed
19 & 24	Apache® Maven 3.1	<p>WebLogic Plugin that provides support for WebLogic deployment capabilities as well as artifact (EAR, WAR, Resource Adapter Archive [RAR], etc.) compilation within the Maven 3.1 environment.</p> <p> REF: For more information on Apache® Maven 3.1, see: http://maven.apache.org/</p>	Service	N/A	N/A	<ul style="list-style-type: none"> VSA Service Assembler VSA Service Deployer 	N/A	Being Developed

Table 20. VSA High Level Application Design—Internal Data Stores


ID	Name	Data Stored	Steward	Access
8	VistA VA FileMan	VistA RPCs. VSA provides a set of utilities that provide for the automated generation and execution of SOA-compliant services, using VistA logic. These services support the full range of VistA M logic functionality including the retrieval, creation, editing, and deletion of VistA data—such as is allowed by established VistA application logic as referenced.	Office of Information and Technology (OI&T)	Read / Write / Delete
26	<p>Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5.</p> <p> REF: For more information on Oracle® WebLogic, see: http://www.oracle.com/tech/network/middleware/weblogic/overview/index.html</p>	<ul style="list-style-type: none"> • VSA Service Descriptor files (in XML format). • Deployed WAR files (services). 	Enterprise Development Environment (EDE) Data Center	Read / Write (Deploy)

3.1.3 Application Locations

3.1.3.1 VSA Development and Integration Testing Phase



[Table 21](#) specifies the planned locations at which the VSA application components are to be hosted for the development and integrated testing phases:

Table 21: VSA Application Locations—Development and Integration Phases

Application Components	Description	Location at Which Component is Run	Type
<ul style="list-style-type: none"> VSA Wizard Federating Platform: Deployed WAR files (services) 	<p>Development, testing, and demonstration environment for the VSA Project team only.</p> <p> NOTE: This environment may also host a partner's generated service for demonstration purposes only. The VSA Project team does <i>not</i> allow those partners access to the VSA Wizard and Federating Platform on these systems.</p> <p>It includes Java Platform Enterprise Edition (Java EE) application server for building and deploying applications and web services (e.g., VSA):</p>	<ul style="list-style-type: none"> Enterprise Development Environment (EDE) Data Center⁵, located at Austin Information Technology Center (AITC) in Austin, TX. <p>EDE is a private cloud that is built on EDE's cloud infrastructure at AITC.</p> <p>These servers are managed by:</p> <ul style="list-style-type: none"> VA—Enterprise Operations (EO) Contractor—MITRE Corporation <ul style="list-style-type: none"> Test servers at Oakland Office of Information Field Office (OIFO), located in Oakland, CA. 	<ul style="list-style-type: none"> Presentation Logic Business Logic Data Logic Interface Code

⁵ For more information on the EDE, see the EDE SharePoint site:

[\[REDACTED\]pm/EDE/default.aspx](#)

Application Components	Description	Location at Which Component is Run	Type
	<p>Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5.</p> <p> NOTE: This development environment does <i>not</i> interact with the Enterprise Service Bus (ESB), Master Veteran Index (MVI), or Identity and Access Management (IAM).</p>		
<ul style="list-style-type: none"> • VSA Wizard • Federating Platform: Deployed WAR files (services) 	<p>Integration testing environment.</p> <p>It includes Java Platform Enterprise Edition (Java EE) application server for building and deploying applications and web services (e.g., VSA): Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5.</p> <p> NOTE: This integration environment will interact with</p>	<ul style="list-style-type: none"> • Servers at the Verizon® Terremark⁶ Nation Access Point (NAP) of the National Capital Region (NCR), located in Culpeper, VA. <p>This is a Cloud Infrastructure that will reside on virtual hardware.</p> <ul style="list-style-type: none"> • Bay Pines Test Lab (BPTL): Part of the normal VA product release process. BPTL can host Vista test systems and will soon also host a VSA WebLogic server. 	<ul style="list-style-type: none"> • Presentation Logic • Business Logic • Data Logic • Interface Code

⁶ For more information on Verizon® Terremark, see the Verizon® Terremark official website: <http://www.terremark.com/>

Application Components	Description	Location at Which Component is Run	Type
	<p>the Enterprise Service Bus (ESB), Master Veteran Index (MVI), Identity and Access Management (IAM) and other VistA systems.</p>	<p>As a combination test, Terremark could host VSA Federating platform and BPTL could host the VistA test systems.</p> <ul style="list-style-type: none"> • Servers at the Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI) Development Test Evaluation Environments (DTE) located at Austin Information Technology Center (AIRC) in the EO Enterprise Development Environment (EDE) Cloud for development through Pre/Prod. 	

[Figure 10](#) and illustrates how the VSA Wizard and Federating Platform can be hosted and tested in Terremark, Oakland, BPTL, EDE, etc.

Figure 10. VSA Wizard Service Creation—Terremark environment (sample)

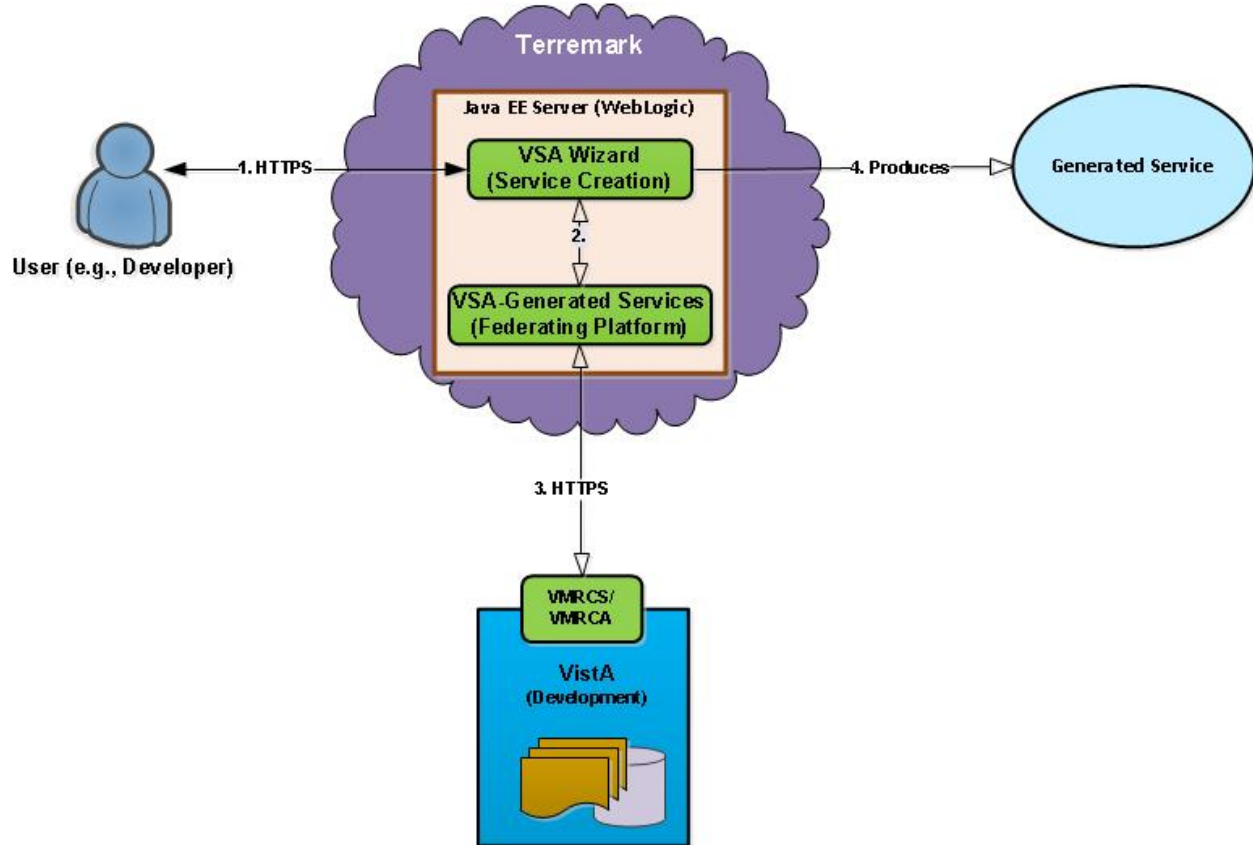
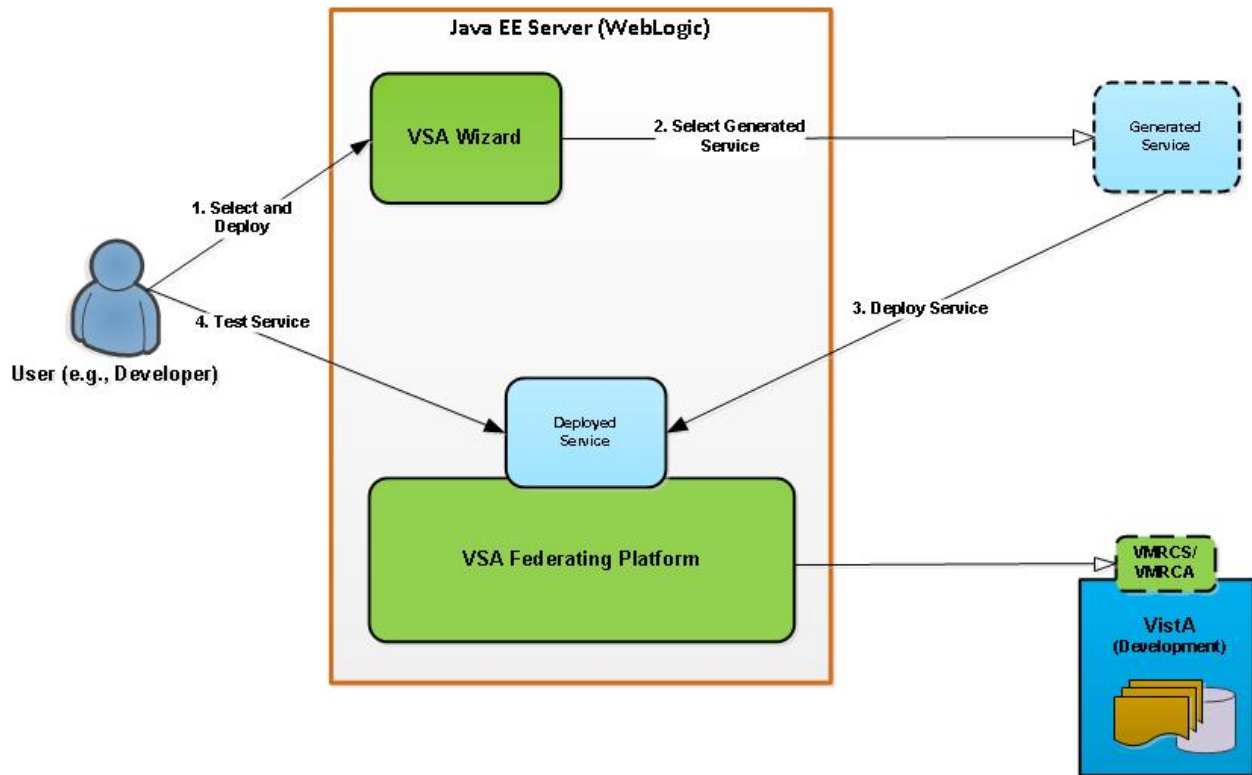


Figure 11. VSA test to same development environment (one VistA system)

3.1.3.2 VSA Implementation (Production Release) Phase

VSA functional workload and performance requirements is dependent on the final topology chosen for deployment. The two topologies currently under consideration:

- Centralized—This is the “As Is,” interim topology.
- Distributed—This is the “To Be,” *future* topology; as intended by the original architecture documents.

In both cases, VSA performance is dependent on the location and responsiveness of its external dependencies.

[Figure 4](#) shows the external dependencies for the VSA Federating Platform. In the centralized topology, both the ESB and the VSA Federating Platform will be located in a central location and will have direct and equal access to the VistA systems.

In the distributed topology, both the ESB and VSA Federating Platforms will be located at each VistA Regional Data Processing Center (RDPC)/Defense Enterprise Computing Center (DECC). A given instance of the VSA Federating Platform will have direct access to the VistA instances deployed at an RDPC/DECC, but the VSA Federating Platform will route requests to a corresponding VSA Federating Platform at the corresponding RDPC/DECC.

The hardware architecture for the VistA Services Assembler Phase 2 (VSA-P2) system will be a multi-phased rollout with an initial Cloud Infrastructure that will reside on virtual hardware in a dedicated enclave within the VA Dedicated Core. All resources are virtualized within the Terremark enclave using VMware technologies:

- VSA-P2 Production environments—Hardware hosted at Terremark Nation Access Point (NAP) of the National Capital Region (NCR) in Culpeper, VA. This is a dedicated infrastructure solely for the VA.
- VSA-P2 Contingency/Disaster Recovery (DR) environment—Hardware hosted at Terremark NAP of the Americas facility in Miami, FL.



REF: For details of disaster recovery and other VSA operational requirements, see the [“Overview of Operational Requirements”](#) section.

Future direction “To Be” will be for Field Operations (FO) to incorporate the VSA-P2 in the 6 Regional Data Centers (i.e., 4 regional centers and 2 regional backup data centers) using a total of 12 servers with a primary and backup server at each location. Ideally, this bed down will be collocated with each of the Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI) servers to optimize performance of the VSA-P2 utility by minimizing network latency between the utility and the Service Oriented Architecture (SOA) of the ESB/eMI and the associated applications using the utility for calls to the various VistA instances across the Enterprise.



REF: For more information on the ESB/eMI, see the eMI (a.k.a. SOA Suite) SharePoint




NOTE: Enterprise Systems Engineering (ESE) internal review process known as the System Engineering Design Review (SEDR) is being initiated to perform its oversight regarding the assurance of the following throughout the VA Enterprise infrastructure:

- Standards
- Performance
- Reliability
- Scalability
- Supportability

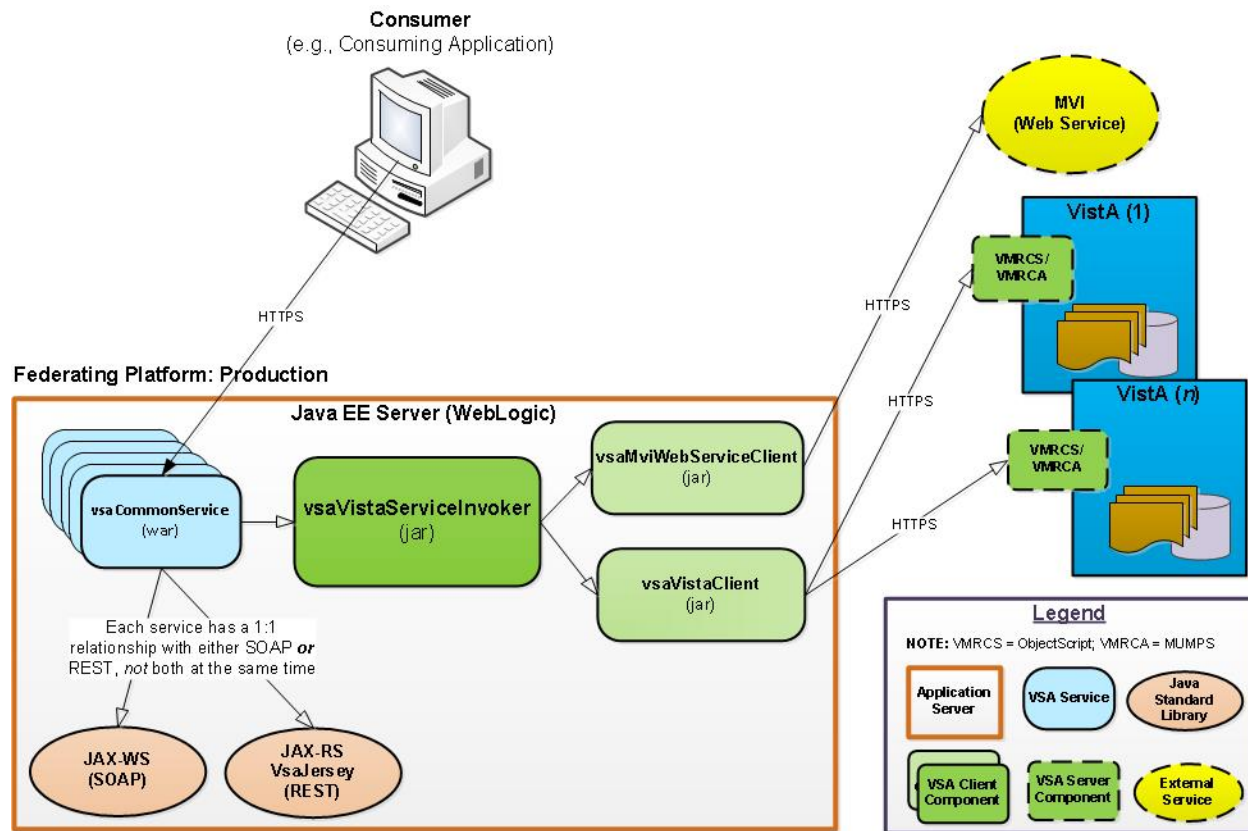
[Table 22](#) specifies the planned locations at which the VSA application components are to be hosted for the implementation phase (i.e., national deployment and release):

Table 22: VSA Application Locations—Integration and Implementation (Release) Phase

Application Components	Description	Location at Which Component is Run	Type
Federating Platform: Deployed WAR files (services)  NOTE: The Production Environment includes the VSA Federating Platform but <i>not</i> the VSA Wizard.	Production environment hosts the VSA Wizard (TBD) and Federating Platform: Deployed WAR files (services). Environment includes Java Platform Enterprise Edition (Java EE) application server for registering web services (e.g., VSA): Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5.	<ul style="list-style-type: none"> Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI) VA Information Technology Centers located at: <ul style="list-style-type: none"> Austin Information Technology Center (AITC). Philadelphia Information Technology Center (PITC). *(Future) VA Regional Data Processing Center (RDPC)/Defense Enterprise Computing Center (DECC) Servers (i.e., 4 regional centers and 2 regional backup/mirror site data centers): <ul style="list-style-type: none"> RDPC/DECC Locations: <ul style="list-style-type: none"> Region 1: Sacramento, CA Region 2: St. Louis, MO Region 3: Atlanta, GA Region 4: Brooklyn, NY RDPC Backup Locations: <ul style="list-style-type: none"> Region 1: Denver, CO for Sacramento Region 4: Philadelphia, PA for 	<ul style="list-style-type: none"> Presentation Logic Business Logic Data Logic Interface Code

Application Components	Description	Location at Which Component is Run	Type
		Brooklyn	
	Environment hosts the Contingency/ Disaster Recovery (DR)	<ul style="list-style-type: none"> Servers at the Verizon® Terremark NAP of the Americas facility, located in Miami, FL. 	<ul style="list-style-type: none"> Presentation Logic Business Logic Data Logic Interface Code

Figure 12. VSA Run-time Context Diagram—VSA Federating Platform Run-time Dependencies (WebLogic): Production



***NOTE:** The original VSA architecture provided by Travis Hilton showed the VSA platform deployed at each of 6 VistA Regional Data Centers (i.e., 4 regional centers and 2 regional backup data centers), instead of the Enterprise data center (AITC) or Philadelphia Information Technology Center (PITC). However, the AITC/PITC are still possible locations in case the Regional Processing Data Centers (RDPCs) are *not* yet ready (i.e., *not* all VA medical centers have moved to the regional data centers). 2020 has been mentioned as the estimated date to achieve a 100% move to the Regional Data Centers.



REF: For details of disaster recovery and other VSA operational requirements, see the [“Overview of Operational Requirements”](#) section.

3.2 Conceptual Data Design

3.2.1 Project Conceptual Data Model

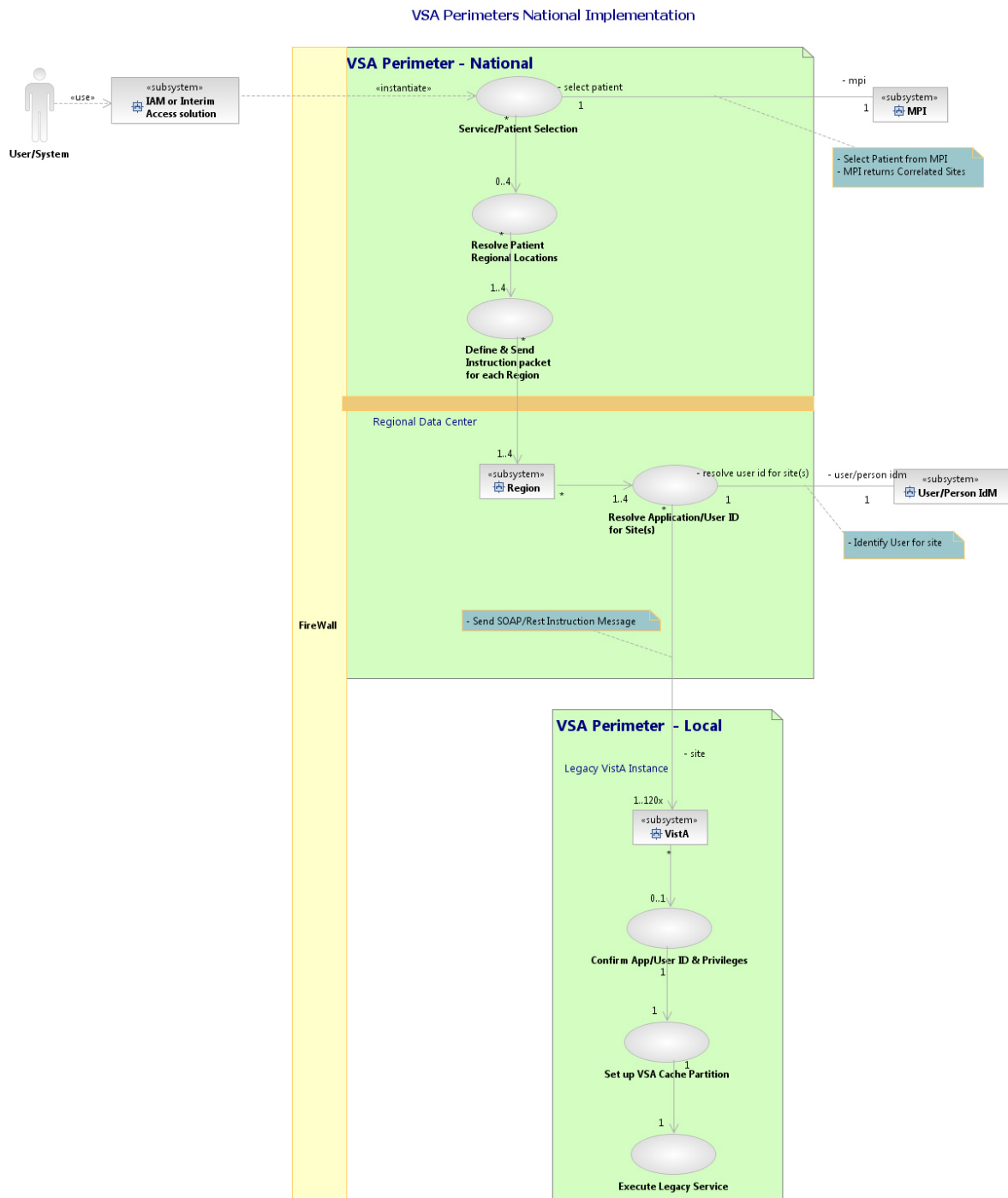
VSA is *not* a traditional application; there is no VSA database for application data. VSA provides a tool that allows users to generate web services, which are stored in WebLogic or the ESB/eMI.

[Figure 6](#) is the Conceptual Data Model (CDM) for VSA. It shows the services being created (and potentially new RPCs being created in VistA). The CDM is a high-level representation of the data entities and their relationships. It is a first step toward developing the more detailed logical data model (LDM) that will be provided during the Logical Data Design.

3.2.2 System Topology and Message Workflow

[Figure 13](#) illustrates VSA system topology and message workflow.

Figure 13. VSA System Topology and Message Workflow



3.2.3 Database Information

VSA interfaces with the following databases:

Table 23: VSA Database Inventory

Database Name	Description	Type	Steward
VistA VA FileMan	<p>VSA passes specified RPC names and parameters (input) and retrieves RPC data from VA FileMan (output).</p> <p>VSA provides a set of utilities that provide for the automated generation and execution of SOA-compliant services, using VistA logic. These services support the full range of VistA M logic functionality including the retrieval, creation, editing, and deletion of VistA data—such as is allowed by established VistA application logic as referenced.</p>	<p>M-based database.</p> <ul style="list-style-type: none"> • Presentation Logic • Business Logic • Data Logic • Interface Code <p>It allows M data to be:</p> <ul style="list-style-type: none"> • Added • Modified • Deleted 	Office of Information and Technology (OI&T) VistA
VistA VA ^XTMP	<p>VMRCS and VMRCA logging.</p> <p>VSA provides a set of services that:</p> <ul style="list-style-type: none"> • Turn logging on and off. • Retrieve lists of log entries and data from a specific log entry. 	M-based database.	Office of Information and Technology (OI&T) VistA.

Database Name	Description	Type	Steward
Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5	Stores VSA Service Descriptor files (in XML format) and deployed WAR files (services). The WebLogic server acts as a VSA web service registry and repository.	Java Platform Enterprise Edition (Java EE) platform: <ul style="list-style-type: none"> • Presentation Logic • Business Logic • Data Logic • Interface Code 	EDE

3.2.4 User Interface Data Mapping

This section describes and defines the information that is available for users of the VSA Wizard. Users can enter data and retrieve information for creating new VSA web services.

VSA provides a Graphical User Interface (GUI) of the VSA Wizard utility that is:

- Web-based
- Modular
- Customizable

The GUI consists of a structured *point-and-click* interface. There are also free-text fields on a Service Descriptor form for the user to enter information. VSA complies with VA's Web development standards, including Section 508 of the Rehabilitation Act. It consists of a Service Descriptor form that enables a user to enter and save information that can be used to create a VistA SOA service that invokes a remote procedure identified by a developer or system integrator.

The VSA Wizard web application user interface is a browser-based client that graphically integrates required system functions. It provides the ability for the service developer to search for and display information about a remote procedure on the development VistA system. Certain forms or fields are editable depending on the information already entered into the VSA Wizard form.

3.2.4.1 VSA Wizard Web Application Screen Interface

3.2.4.1.1 VSA Wizard—Main Page

[Figure 14](#) represents the VSA Wizard main page (screen). The VSA Wizard allows users to locate existing Service Descriptor files (in XML format) or create new Service Descriptor files for new web services.

Figure 14: VSA Wizard—Main Page (Screen)

VSA Wizard

Service Descriptor XML File

Locate an existing service descriptor file or create a new one

Navigate to the file path that contains the service descriptor file or type a path in the Path field and press Enter. Once the file is located, click the name of the file to begin working with it.

Press the Create button to create a new service descriptor file.

Path:

Current Location: localhost \ C: \ Oracle \ config \ domains_12.1.3 \ base_domain

- autodeploy
- bin
- config
- console-ext
- generatedWebService
- init-info
- lib
- nodemanager
- pending
- security
- servers
- tmp
- ☒ gov.va.med.test.TestService-1.0.vsa.sdf.xml

VistA Services Assembler (VSA)
Version: 1.0.0
Date of build: 20140904-1054

[Table 24](#) describes the fields/gadgets on the VSA Wizard main page ([Figure 14](#)).

Table 24: VSA Wizard—Main Page (Screen) Description

GUI Data Elements (Fields or Gadgets)	Table (Database Table that field connects to)	Field (Field in Table that the GUI field connects to)	Comments
Path Field	N/A	N/A	<p>This field allows the user to enter a file directory path. The user can navigate through the file system of the VSA Federating Platform (e.g., WebLogic) to locate the folder containing the Service Descriptor file (in XML format). The system:</p> <ul style="list-style-type: none"> • Displays the path of the file being selected. • Allows the user to select and edit the desired Service Descriptor XML file.
Create Button	N/A	N/A	This button allows the user to open a Service Descriptor form to create a new Service Descriptor file (in XML format), which will eventually become a VSA web service.
Current Location Field	N/A	N/A	This field displays the user's current location in the directory path (tree).
Directory List Area	N/A	N/A	The system displays a directory list for those directories that store VSA Service Descriptor files (in XML format).

3.2.4.1.2 VSA Wizard—Service Descriptor Form

3.2.4.1.2.1 RESTful Web Services

[Figure 15](#) represents the VSA Wizard's Service Descriptor form for Representational State Transfer (REST; RESTful) services. This form is used to enter the information needed to generate a new web service using REST:

Figure 15. VSA Wizard—Service Descriptor Form for RESTful Services

VSA Wizard [Back to Service Descriptor selection page](#)

Service Information

Enter information about this web service

The Service Name is the name of the service as exposed to client applications. The Version is the current version of this web service. The Service Namespace is a URI that corresponds to this web service. The Java Package identifies the package to use for generated Java classes.

Service Type: ☐ SOAP ☒ REST

Service Name:

Version:

Java Package:

REST-Specific Information:

URL Path:

Life Cycle:

Produces:

Consumes:

Vista Remote Procedure Selection

Look up an RPC and autogenerate an operation

Look up an RPC on the Vista system and autogenerate a web service operation that corresponds to the selected RPC.

Enter the first few characters of the RPC and press Search. A list of RPCs that start with the input value appears. Select an item in the list to display information about that RPC. Press Autogenerate to automatically generate an operation that corresponds to that RPC.

RPC Search

RPC Name:

Operation Information

Create web service operations

Edit the autogenerated operations or manually create and edit web service operations.

Operations:

RPC Name	Operation Name	Resp Type	HTTP Method	URL Path	Consumes	Produces		
+		string					+	x

Actions

Process the web service

Save the service descriptor XML file and create, deploy, and test the web service.

Step 1: Save the form to a service descriptor XML file.

Step 2: Create the web service as a WAR file.

Step 3: Deploy the web service to WebLogic.

Vista Services Assembler (VSA)
Version: 1.0.0
Date of build: 20140904-1054

3.2.4.1.2.2 SOAP Web Services

[Figure 16](#) represents the VSA Wizard's Service Descriptor form for Simple Object Access Protocol (SOAP) services. This form is used to enter the information needed to generate a new web service using SOAP:

Figure 16. VSA Wizard—Service Descriptor Form for SOAP Services

VSA Wizard
Back to Service Descriptor selection page

Service Information

Enter information about this web service

The Service Name is the name of the service as exposed to client applications. The Version is the current version of this web service. The Service Namespace is a URI that corresponds to this web service. The Java Package identifies the package to use for generated Java classes.

Service Type: ☒ SOAP ☐ REST

Service Name:

Version:

Java Package:

SOAP-Specific Information:

Service Namespace:

Vista Remote Procedure Selection

Look up an RPC and autogenerate an operation

Look up an RPC on the Vista system and autogenerate a web service operation that corresponds to the selected RPC.

Enter the first few characters of the RPC and press Search. A list of RPCs that start with the input value appears. Select an item in the list to display information about that RPC. Press Autogenerate to automatically generate an operation that corresponds to that RPC.

RPC Search

Operation Information

Create web service operations

Edit the autogenerated operations or manually create and edit web service operations.

Operations:

RPC Name	Operation Name	Resp Type	
<input type="text"/>	<input type="text"/>	string	<input type="button" value="+"/> <input type="button" value="x"/>

Actions

Process the web service

Save the service descriptor XML file and create, deploy, and test the web service.

Step 1: Save the form to a service descriptor XML file.

Step 2: Create the web service as a WAR file.



Step 3: Deploy the web service to WebLogic.



Vista Services Assembler (VSA)
Version: 1.0.0
Date of build: 20140904-1054

3.2.4.1.2.3 Service Descriptor Form fields

Table 25 describes the VSA Wizard Service Descriptor form fields for both RESTful (Figure 15) and SOAP (Figure 16) web services:

Table 25: VSA Wizard—Service Descriptor Form Description

GUI Data Elements (Fields or Gadgets)	Table (Database Table that field connects to)	Field (Field in Table that the GUI field connects to)	Comments
Back to Service Descriptor selection page (button)	N/A	N/A	Click this button to return to the VSA Wizard main page (Figure 14).
Service Information			
Service Type (radio button)	N/A	N/A	<p>Enter the service type for this web service. Select from one of the following options:</p> <ul style="list-style-type: none"> • SOAP • REST <p> NOTE: Some unique fields are displayed to the user depending on the service type selected. These unique service-type fields are indicated throughout this table by either of the following:</p> <ul style="list-style-type: none"> • SOAP only • RESTful only
Service Name	N/A	N/A	Enter a unique identifier for the new service name.
Version	N/A	N/A	<p>Enter the service version number.</p> <p> REF: To enter a properly formatted version, follow the Enterprise Shared Services (ESS) Guidance for Versioning Services Document on the "ESS SOA Design Guidelines" website: ea.oit.va.gov/ess-design-guidance-documents/</p>
Java Package	N/A	N/A	Enter a unique identifier (path and name) for the new service Java package.

GUI Data Elements (Fields or Gadgets)	Table (Database Table that field connects to)	Field (Field in Table that the GUI field connects to)	Comments
Service Namespace (SOAP only)	N/A	N/A	<p>Enter the namespace to which the new service belongs.</p> <p> REF: To enter a properly formatted namespace, follow the Namespace Guidance Document on the "ESS SOA Design Guidelines" website: </p>
URL Path (drop-down list) (RESTful only)	N/A	N/A	Enter a Uniform Resource Identifier (URI) path of the RESTful resource class, relative to the application URI. This corresponds to the @Path annotation of the class.
Life Cycle (drop-down list) (RESTful only)	N/A	N/A	Enter the life-cycle of the RESTful resource class. The default value is per-request, in which case a new instance of a root resource class is created every time the request URI path matches the root resource. Select singleton, which corresponds to the @Singleton class-level annotation, to indicate that only instance of the class should be created per web application.
Produces (drop-down list) (RESTful only)	N/A	N/A	<p>Enter the default Multimedia Internet Mail Extensions (MIME) media types a resource can produce and send back to the client. Select zero or more of the following:</p> <ul style="list-style-type: none"> • text/plain • text/html • application/json • application/xml • application/x-www-form-urlencoded <p>Corresponds to the @Produces annotation of the class.</p>

GUI Data Elements (Fields or Gadgets)	Table (Database Table that field connects to)	Field (Field in Table that the GUI field connects to)	Comments
Consumes (RESTful only)	N/A	N/A	Enter the default MIME media types sent by the client a resource can consume. Select zero or more of the following: <ul style="list-style-type: none"> • text/plain • text/html • application/json • application/xml • application/x-www-form-urlencoded Corresponds to the @Consumes annotation of the class.
Vista Remote Procedure Selection			
RPC Name	N/A	N/A	Enter an RPC name (minimum 1 character) in this field to get a list of Vista RPCs from which to choose. A drop-down list of RPCs is returned from Vista.
Search (button)	N/A	N/A	Click this button after entering the RPC search input data in the RPC Name field. The system returns the results as a list of RPCs that match the search input data.
Results (drop-down list)	N/A	N/A	Appears if the Search button returned results. Select an RPC to display the details about that RPC.
Auto-Generate Operation	N/A	N/A	Appears if the Search button returned results. Click this button after selecting an RPC to auto-generate an operation with default values based on the RPC definition.
Operation Information			
Add an Operation (button)	N/A	N/A	Click this button to add a new operation.
RPC Name	N/A	N/A	Enter the RPC name to be associated with the new operation.
Operation Name	N/A	N/A	Enter a unique identifier for the new operation name.
Response Type (drop-down list)	N/A	N/A	Enter the new operation response type. Select from one of the following options: <ul style="list-style-type: none"> • string • json • list • map

GUI Data Elements (Fields or Gadgets)	Table (Database Table that field connects to)	Field (Field in Table that the GUI field connects to)	Comments
HTTP Method (RESTful only)	N/A	N/A	Enter the HTTP method this operation processes. Select from one of the following options: <ul style="list-style-type: none"> • GET • POST • PUT • DELETE • HEAD
URL Path (RESTful only)	N/A	N/A	Enter the relative URI path of the RESTful operation. This corresponds to the @Path annotation of the generated Java method.
Consumes (RESTful only)	N/A	N/A	Enter the MIME media types sent by the client the operation can consume. Select zero or more of the following: <ul style="list-style-type: none"> • text/plain • text/html • application/json • application/xml • application/x-www-form-urlencoded This corresponds to the @Consumes annotation of the generated Java method.
Produces (RESTful only)	N/A	N/A	Enter the MIME media types a resource can produce and send back to the client. Select zero or more of the following: <ul style="list-style-type: none"> • text/plain • text/html • application/json • application/xml • application/x-www-form-urlencoded This corresponds to the @Produces annotation of the generated Java method.
Delete (button)	N/A	N/A	Click this button to delete an operation.
Insert (button)	N/A	N/A	Click this button to insert an operation before the current operation.
Show/Hide Input Parameters (toggle button)	N/A	N/A	Click this button to display (Show) input parameters or <i>not</i> display (Hide) the input parameters for an operation.

GUI Data Elements (Fields or Gadgets)	Table (Database Table that field connects to)	Field (Field in Table that the GUI field connects to)	Comments
Input Parameters			
Name	N/A	N/A	Enter the operation input parameter name.
Type (drop-down list)	N/A	N/A	Enter the operation input parameter type. Select from one of the following options: <ul style="list-style-type: none"> • string • ref • list • map • ICN to DFN (Integration Control Number [ICN] to Data File Number [DFN]) • Correlation List
Param Type (RESTful only)	N/A	N/A	Select how the value of this parameter is sent to the resource method. Select zero or one of the following options: <ul style="list-style-type: none"> • PathParam • QueryParam • MatrixParam • HeaderParam • CookieParam • Context This corresponds to the annotation of the parameter in the resource method signature.
Param Name (RESTful only)	N/A	N/A	Enter the name of the parameter. This corresponds to the argument to be used in any of the following the annotations of the input parameter of the generated Java method: <ul style="list-style-type: none"> • @PathParam • @QueryParam • @MatrixParam • @HeaderParam • @CookieParam
Default Value (RESTful only)	N/A	N/A	Enter the default value for this input parameter if one is <i>not</i> passed. This corresponds to the @DefaultValue annotation of the input parameter of the generated Java method.
Delete (button)	N/A	N/A	Click this button to delete an input parameter.

GUI Data Elements (Fields or Gadgets)	Table (Database Table that field connects to)	Field (Field in Table that the GUI field connects to)	Comments
Insert (button)	N/A	N/A	Click this button to insert an input parameter <i>before</i> the current parameter.
Add a Parameter (button)	N/A	N/A	Click this button to add a parameter to the <i>end</i> of the list.
Actions			
Save Service Descriptor (button)	N/A	N/A	Click this button after entering all operation and input parameter information. It generates and saves the Service Descriptor file (XML format).
Display contents (link)	N/A	N/A	Click on this link to display the contents of the Service Descriptor file (XML format).
Create Web Service (button)	N/A	N/A	Click this button to generate a web service (WAR file; undeployed) from the Service Descriptor file (XML format).
Display war file location (link)	N/A	N/A	Click on this link to display the directory path (location) for the newly created WAR file.
Deploy Web Service (button)	N/A	N/A	Click this button to deploy the WAR file to the WebLogic server as a new VSA service.
Test the web service (link)	N/A	N/A	Click on this link to test the new VSA web service (i.e., newly deployed WAR file).



NOTE: All of the values entered into the data fields on the Service Descriptor form are saved in the Service Descriptor file in XML format. This file is the key configuration file that is created via the VSA Wizard.

3.2.4.2 Application Report Interface

Not Applicable (N/A). The VSA Phase 2 software does *not* create any reports.

3.2.4.3 Unmapped Data Element

Not Applicable (N/A). The VSA Phase 2 software does *not* have any unmapped data elements.

3.3 Conceptual Infrastructure Design

The conceptual infrastructure design is a high-level overview of the infrastructure that is used to support VSA.

The VSA Phase 2 conceptual infrastructure design will become more detailed at later stages as more information is collected regarding the system, and the infrastructure requirements (i.e., capacity requirements) are better known.

The following software environments and applications are required for VSA development and web service deployment using the VSA Wizard:

- Web Browser (e.g., Microsoft® Internet Explorer)
- Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5—Development server and web service registry and repository.
- Red Hat Enterprise Linux release 6.5 environment
- Secure Shell (SSH) connectivity
- Apache® Maven 3.1
- Apache® Velocity templating library Version 1.7
- SoapUI 4.5.2
- Jersey RESTful Web Services Framework 1.17
- Spring MVC 3.2.4 (Model-View-Controller [MVC] framework)
- Caché:
 - InterSystems® Caché Studio 2011.1.2.701.0.12942
 - InterSystems® Caché Server Pages turned on
- VistA M Environment (fully patched)—Standard infrastructure required for all VistA systems at all sites. This includes but is *not* limited to the following applications:
 - Kernel 8.0
 - Kernel Toolkit 7.3
 - VA FileMan 22.0
 - Remote Procedure Call (RPC) Broker 1.1
 - VistALink 1.6

3.3.1 System Criticality and High Availability

The VSA Phase 2 is considered a mission critical system and requires high availability. It follows the same standard procedures used for all critical VistA systems. For example:

- System backups
- VistA disaster recovery



REF: For details of disaster recovery and other VSA operational requirements, see the [“Overview of Operational Requirements”](#) section.

3.3.2 Special Technology

None. VSA uses standard VA hardware and software. It does *not* require any special technology.

Table 26: VSA Special Technology Requirements

Special Technology	Description	National Location	TRM Status
N/A	N/A	N/A	N/A

3.3.3 Technology Locations

[Table 27](#) lists the various technology components that VSA uses.

Table 27: VSA Technology Location Details

Technology Component Phase 2	Location	Usage
Workstations	All VA Sites	Developers and consuming applications use client workstations to develop software and access the VSA Wizard for Phase 2 located on the Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5.
Interface Processors	N/A	N/A
WebLogic Application Server	Enterprise Development Environment (EDE) Data Center, which is managed by: <ul style="list-style-type: none"> VA—Enterprise Operations (EO) Contractor—MITRE Corporation Possible server location: Austin Information Technology Center (AIRC).	EDE—Set up sanctioned and managed development environment with access to VM servers, which includes the following platforms: <ul style="list-style-type: none"> Linux: 2 VM. Windows: 8 VM ("Dev-in-a-box;" primarily for contractors without Government Furnished Equipment (GFE) laptops.

Technology Component Phase 2	Location	Usage
Legacy Databases	VistA Regional Data Centers	The VistA VA FileMan database is used to store all VistA RPCs. VSA provides a set of utilities that provide for the automated generation and execution of SOA-compliant services, using VistA logic. These services support the full range of VistA M logic functionality including the retrieval, creation, editing, and deletion of VistA data—such as is allowed by established VistA application logic as referenced.
Certification	N/A	N/A
Education	SharePoint Site:	Review VSA demonstration recordings for an overview on the use of the VSA Wizard.
Test	WebLogic Server located at the EDE Data Center	Used to develop the VSA Phase 2 software.
Development	WebLogic Server located at the EDE Data Center	Used to develop the VSA Phase 2 software.

3.3.4 Conceptual Infrastructure Diagram



CAUTION: The diagrams depicted in this section are based on early architectural diagrams provided by Travis Hilton. The diagrams for VSA Phase 2 are subject to change as development continues.



NOTE: The Enterprise Service Bus (ESB) is included in some of the images that follow. During initial integration testing, VSA Phase 2 will continue to register and interact direct with VSA on the Oracle® WebLogic Server 12c (Release 12.x) federating platform. For deployment, VSA Phase 2 will register services with the ESB/eMI federating platform.

REF: For more information on the ESB/eMI, see the eMI (a.k.a. SOA Suite) SharePoint site: [\[Redacted\]](#)

3.3.4.1 Location of Environments and External Interfaces

[Figure 17](#) shows the VSA supported environments. This diagram includes the following:

- Local networks to which VSA is attached (Development, Test, and National release).
- Locations at which VSA is installed.
- External connections.

Figure 17. VSA Conceptual Networks and Environments (1 of 4)

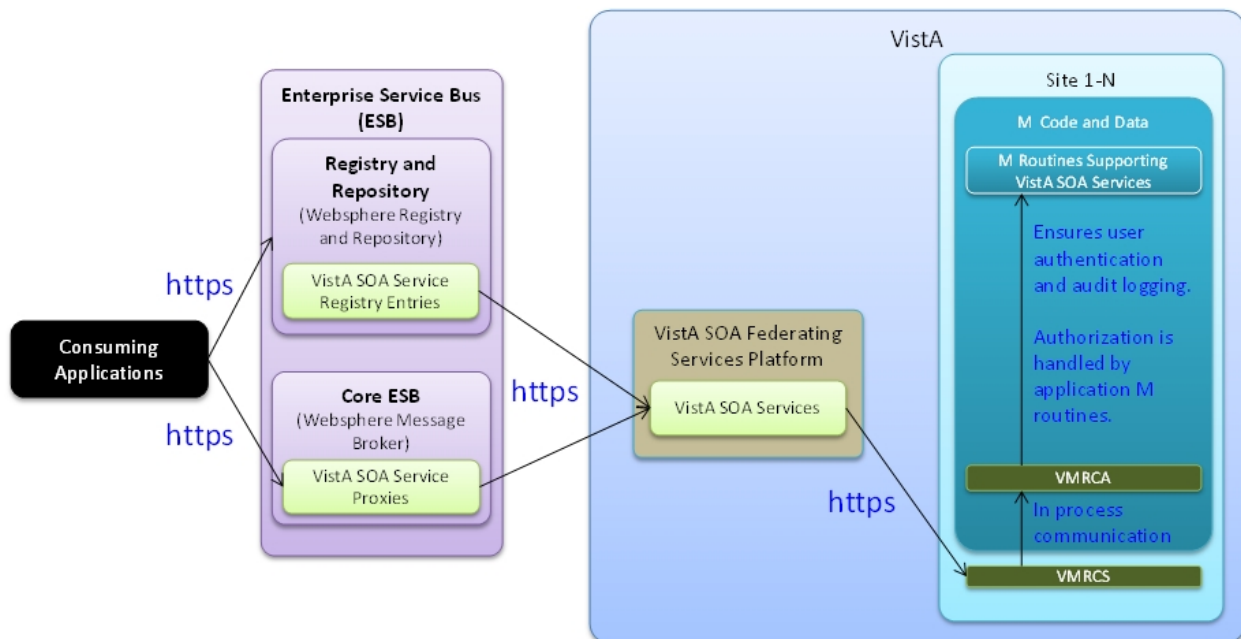


Figure 18: VSA Conceptual Networks and Environments (2 of 4)

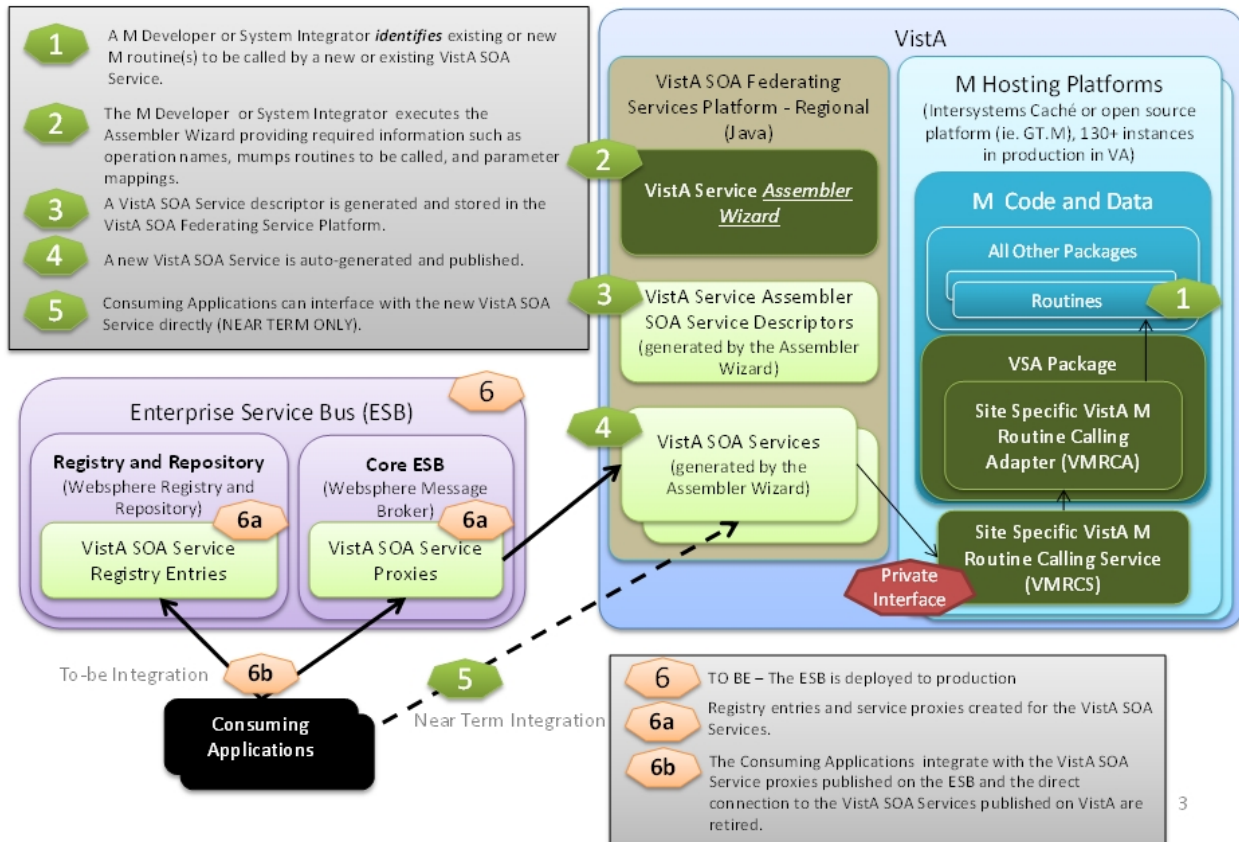
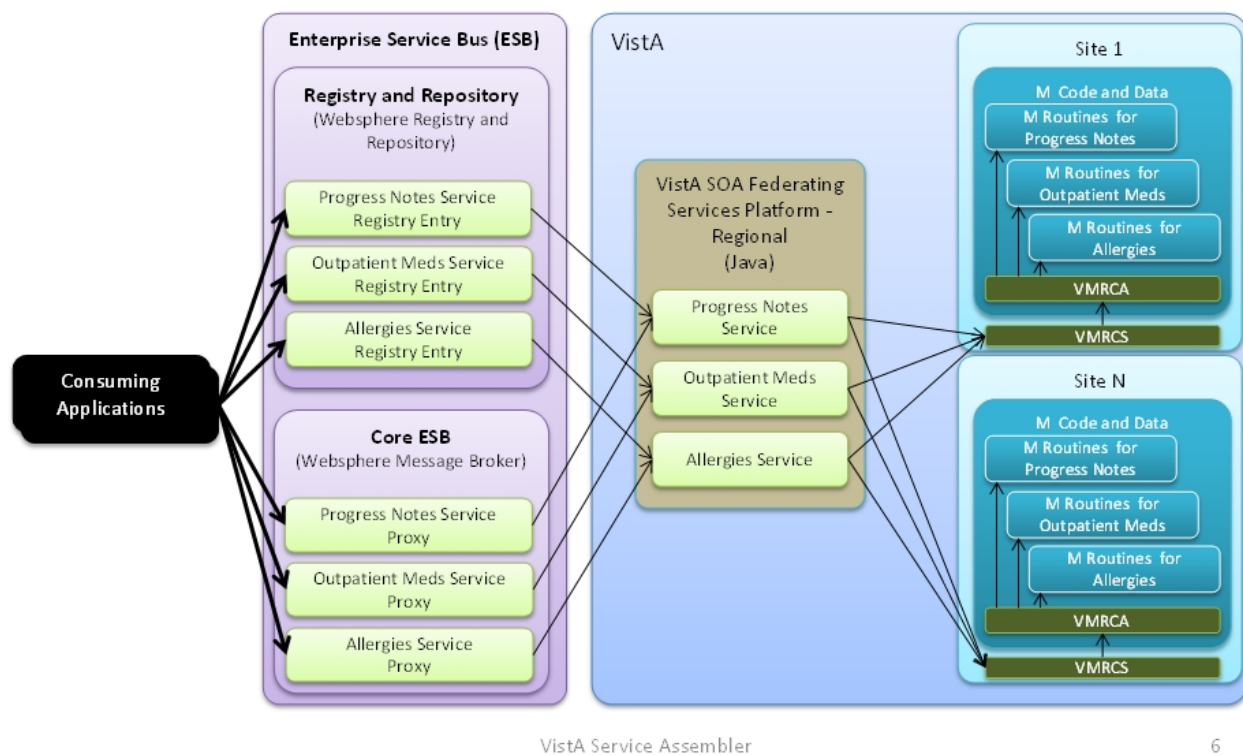


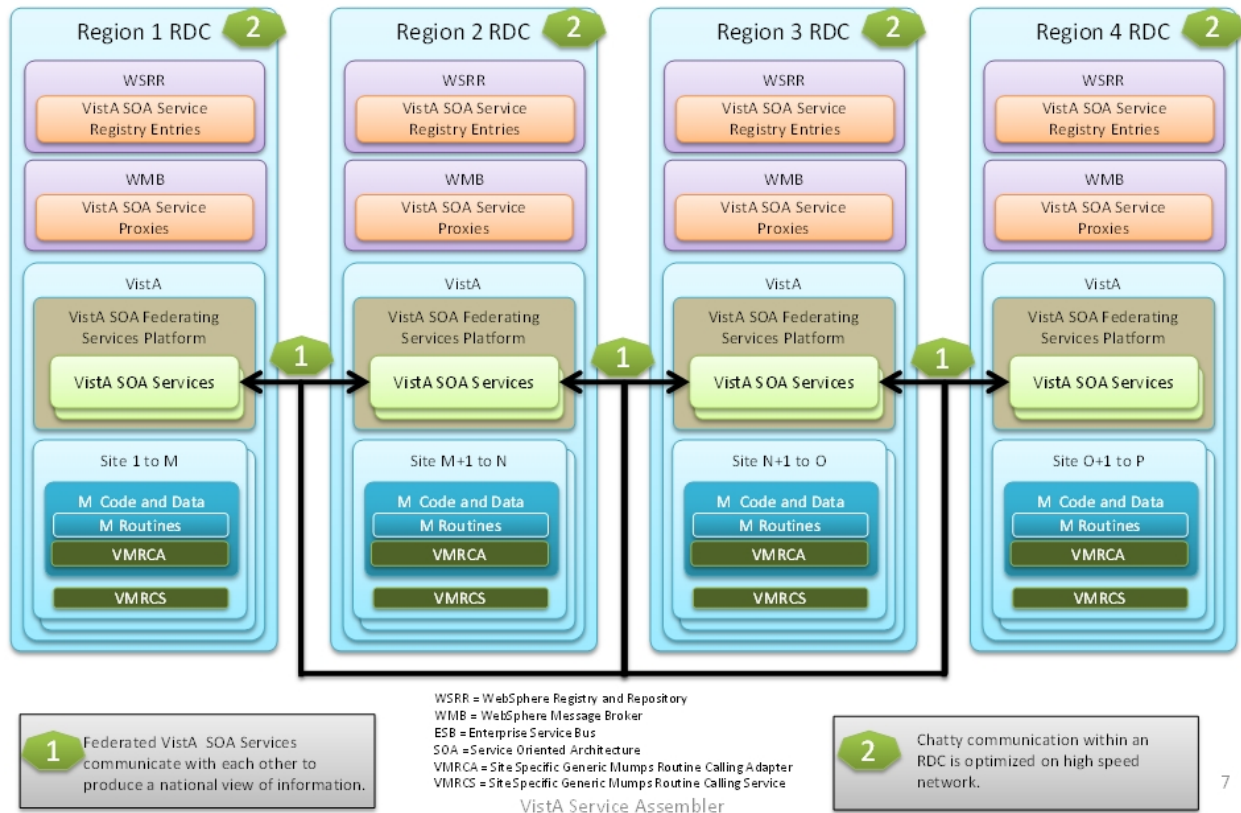
Figure 19. VSA Conceptual Networks and Environments (3 of 4)



6

Figure 20. VSA Conceptual Networks and Environments (4 of 4)

VSA National Production Deployment Configuration

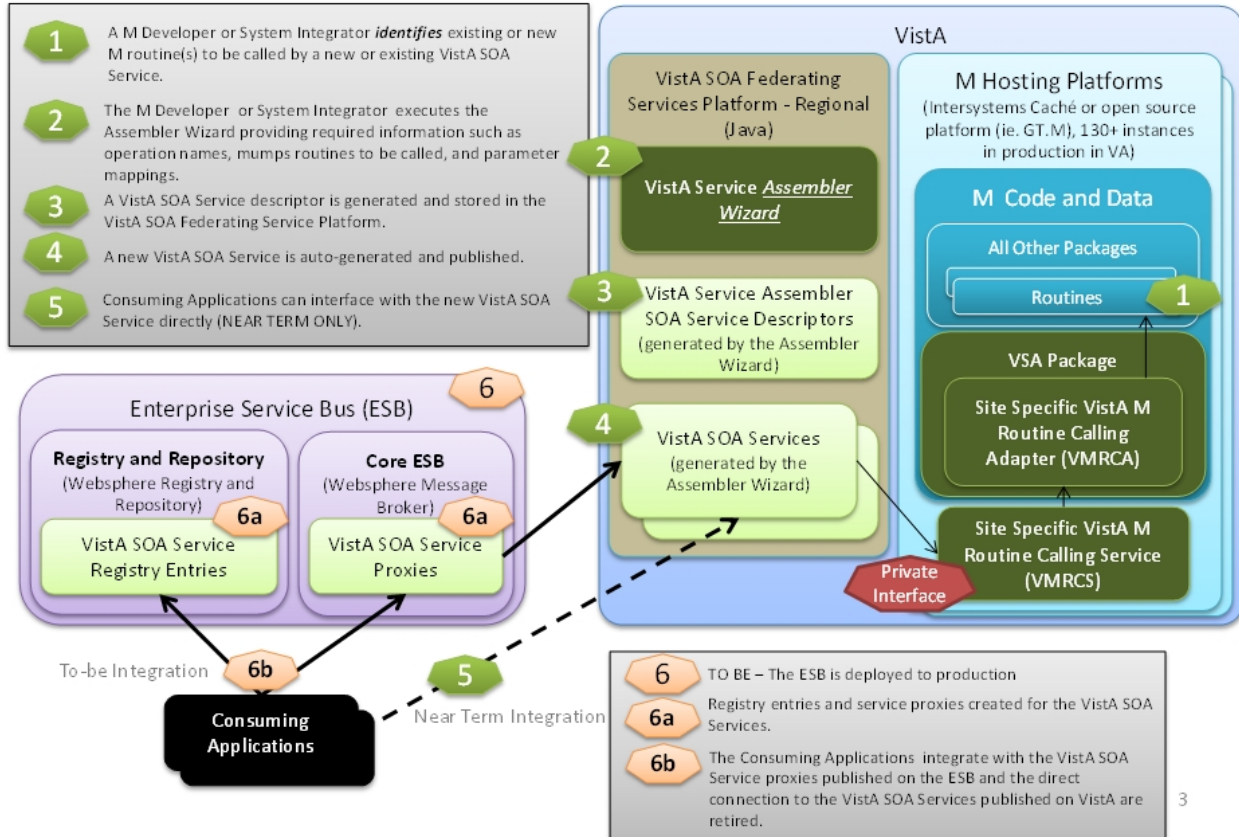


REF: For more detailed information regarding the VSA environments, see the Systems Engineering and Design Review (SEDR) VSA-P2 *Accountable Customer Experience (ACE)-Map Questions-VistA Services Assembler Phase 2_MMDDYY* Excel spreadsheet located on the Enterprise Systems Engineering (ESE) SharePoint site: [\[Redacted URL\]](#)

3.3.4.2 Conceptual Production String Diagram

[Figure 21](#) shows the configuration of a single production string to the extent that it is known.

Figure 21. VSA Conceptual Single Production String Diagram



4 System Architecture

This section describes the VSA architecture. It includes discussion about the general architectural decisions that have been approved.

VSA as an application exists in two contexts:

- Design-time (see [Figure 10](#)).
- Run-time (see [Figure 4](#) and [Figure 5](#)).

4.1 Hardware Architecture

VSA requires the following hardware:

- Client Workstations for development.
- Oracle® WebLogic Server 12c (Release 12.x).
- InterSystems® Caché and VistA M Environment Server.

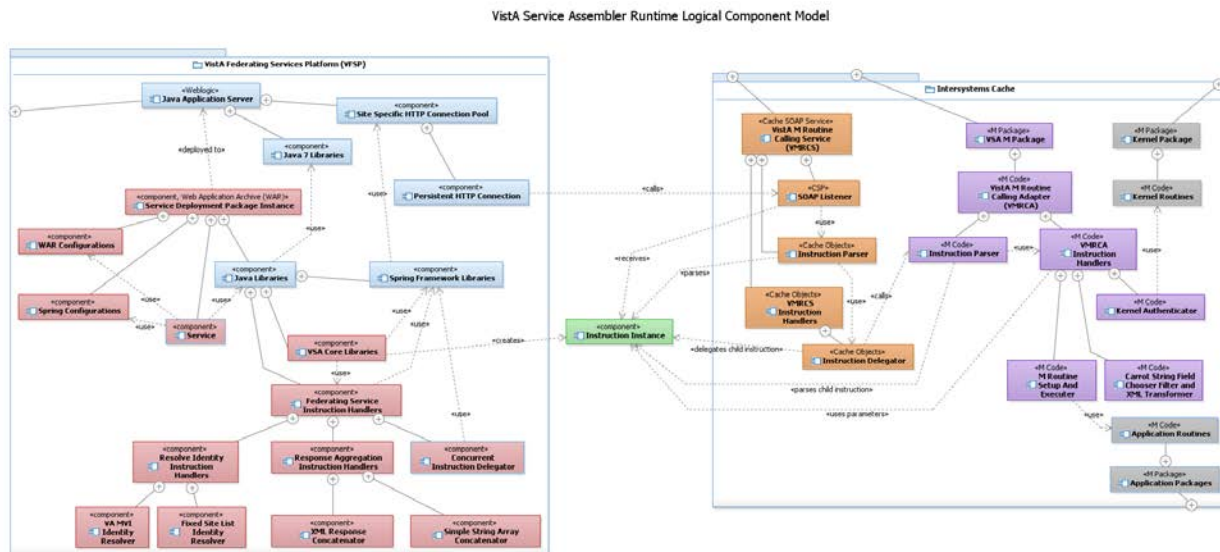


REF: For system diagrams, see the figures in [Section 3](#).

4.2 Software Architecture

The following system software architecture is required for VSA:

Figure 22. VSA—Logical Component Model





NOTE: The InterSystems Caché side of the diagram in [Figure 22](#) is in flux, but *not* determinately different from the current diagram. At some point this diagram might be updated.



REF: For system diagrams, see the figures in [Section 3](#).

4.2.1 Methodology, Tools, and Techniques

The VSA project uses IBM® Rational Jazz suite to manage change requests made during the development phase. The specific Jazz tools include:

- Rational Quality Manager (RQM)
- Rational Requirements Composer (RRC)
- Rational Team Concert-(RTC)
a.k.a. Change and Configuration Management (CCM)

The following software environments and applications are required for VSA development and web service deployment using the VSA Wizard:

- Web Browser (e.g., Microsoft® Internet Explorer)
- Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5—Development server and web service registry and repository.
- Red Hat Enterprise Linux release 6.5 environment
- Secure Shell (SSH) connectivity
- Apache® Maven 3.1
- Apache® Velocity templating library Version 1.7
- SoapUI 4.5.2
- Jersey RESTful Web Services Framework 1.17
- Spring MVC 3.2.4 (Model-View-Controller [MVC] framework)
- Caché:
 - InterSystems® Caché Studio 2011.1.2.701.0.12942
 - InterSystems® Caché Server Pages turned on
- VistA M Environment (fully patched)—Standard infrastructure required for all VistA systems at all sites. This includes but is *not* limited to the following applications:
 - Kernel 8.0
 - Kernel Toolkit 7.3
 - VA FileMan 22.0
 - Remote Procedure Call (RPC) Broker 1.1
 - VistALink 1.6

4.2.2 VSA Wizard

The VSA is a web application written in Java 6 and Java EE 5, and bundled in an Enterprise Archive (EAR). The VSA Wizard uses the following 3rd party libraries:

- Spring Framework 3.2.4 for dependency injection.
- Spring MVC 3.2.4 (Model-View-Controller [MVC] framework) for separation of data from presentation.
- JAXB 2.0 bundled in JDK 1.6 for converting XML data to and from Java objects.
- Jersey RESTful Web Services Framework 1.17 for an implementation of Java APIs (Java Standard Library) for XML REST Web Services (JAX-RS).

Additionally, the Wizard relies directly on the following VSA components:

- VSA Code Generator—A software module that reads a Service Descriptor XML document and generates a SOAP or RESTful web service in Java EE 5.
- VSA XML—Module that represents the Service Descriptor XML file as Java objects.
- VSA-generated Web Service—Internal service used by the Wizard to obtain information about Remote Procedures on the development VistA system.

4.2.3 VSA-generated Web Service

VSA does *not* contain or provide any web services for other applications. However, VSA can be used to create web services for other applications. A VSA-generated web service is a standard SOAP web archive (WAR) file that should be deployed on a Java EE 5 application server, such as Oracle® WebLogic is supported.

4.2.4 VSA Listener

4.2.4.1 VMRCS

The Vista M Routine Calling Service (VMRCS) listener is a VSA server component that is developed using Caché Objects/Caché Server Pages (CSP). It is an internal web serviced used by VSA to access the VistA system. Currently, the VMRCS listener runs on InterSystems® Caché version 2011.1.2.701.0.12942.

4.2.4.2 VMRCA

The Vista M Routine Calling Adapter listener is a VSA server component that is a set of M routines that is invoked by the VMRCS listener when a web service call is made. The VMRCA listener is written in M and executes the legacy VistA functions exposed by the web service.

4.2.5 Code Generator

The VSA Code Generator is invoked by the VSA Wizard. It is written in Java 1.6. It and does the following:

- Receives a Service Descriptor Java object from the VSA Wizard.
- Extracts the XML data.
- Generates the necessary Java classes that subsequently get compiled and built into a SOAP or REST web service.

When the Code Generator's task is done, it does the following:

- Packages the generated code into a deployable .war file.
- Deletes the generated code.
- Passes back to the VSA Wizard a location where the generated .war file resides on the WebLogic file system.

4.2.6 VistaServiceInvoker

The VistaServiceInvoker is written in Java 6. It is responsible for delegating all SOAP web service calls from VSA-generated web services to the VMRCS listener.

This Java component does the following at runtime:

- Takes inputs from the invoked business service.
- Converts the inputs into a proper instruction object.
- Calls the VistA M Routine Calling Service (VMRCS) listener web service.

4.3 Network Architecture

Authorized VA users access the VSA Wizard through a Web front-end via a Web browser. The browser connects from within the VA's Intranet and communicates with the Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5 via Hypertext Transfer Protocol Secure (HTTPS) standard Port 443 over Transport Control Protocol/Internet Protocol (TCP/IP).

The following system network communications interfaces are required for VSA.

- Hypertext Transfer Protocol Secure (HTTPS)
- Transport Control Protocol/Internet Protocol (TCP/IP)

Figure 23. VSA—Network Communications Architecture



NOTE: During initial integration testing, VSA Phase 2 will continue to register and interact direct with VSA on the Oracle® WebLogic Server 12c (Release 12.x) federating platform. For deployment, VSA Phase 2 will register services with the Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI) federating platform.

REF: For more information on the ESB/eMI, see the eMI (a.k.a. SOA Suite) SharePoint site: [REDACTED]

[Table 28](#) lists VSA communication requirements:

Table 28. VSA Communication Requirements

Requirement Identifier	Description
CI4.1.1	The VSA Wizard <i>shall</i> provide communication to one development VistA instance and use standard and secure communication protocols (Hypertext Transfer Protocol Secure [HTTPS]).
CI4.1.2	The VSA Federating Services Platform <i>shall</i> provide the necessary communication interfaces to interact with the necessary collaborating systems: <ul style="list-style-type: none"> • Client applications (or ESB) to the VSA Federating Services Platform. • VSA Federating Services Platform to MVI Web Services. • VSA Federating Services Platform to VistA.
CI4.1.3	The VSA solution <i>shall</i> allow provider applications to create their own SOAP and RESTful web service interfaces to its clients.
CI4.1.4	VSA <i>shall</i> use a single internal Caché SOAP web service (VMRCS) on the VistA system to: <ul style="list-style-type: none"> • Receive requests from the Federated Web service. • Transform the payload to an M-compatible format. • Pass the payload to VMRCA.
CI4.1.5	The VMRCA component <i>shall</i> call and invoke the corresponding RPC in VistA, passes the results back to VMRCS, which transforms the response and returns it in compatible format back to the Federated web service.
CI4.1.6	Web service requests from client applications to the Federated web service of the Federating Platform and from the federated web service to VMRCS <i>shall</i> be done through HTTPS using secure channels, such as Secure Socket Layer (SSL) encryption.

4.4 Service Oriented Architecture / Enterprise Shared Services

VSA Phase 2 creates a set of software utilities for national release. VSA provides the basic (uncomplicated) functionality necessary to automate the creation of VistA Service Oriented Architecture (SOA) services and infrastructure components necessary to support and operate those services. It further develops this set of utilities to include the VistA Services Assembler Wizard and assorted components.

The VSA Phase 2 project also identifies and produces multiple, sample VistA SOA business services as a "reference implementation" to confirm the use of VSA services. The specific business services to be produced will be identified based on a consideration of organizational initiatives that have a need for VistA SOA services in the near *future*. This approach ensures that services produced as a part of the "reference implementation" are both critically tested and of immediate value to the organization.



REF: For a diagram depicting the major components of the larger system, interconnections, and external interfaces, see [Figure 3](#).

4.4.1 Services Provided

VSA provides the *mechanism* to create services from Remote Procedure Calls (RPCs). It does *not* distribute any services; other than those services consumed internally by VSA itself.

As a test demonstration of the VSA tool, however, the VSA development team is collaborating with other project teams to create specific services based on identified use cases. For example:

- Veteran Point of Service (VPS).
- VistA Evolution/Mobile Development. The Patient Viewer Phase 2 will be writing orders. They will be the first application to test the VSA solution that will resolve or come up with a solution for the session management problem.



ATTENTION: These demonstration services are *not* owned or distributed by VSA; they are owned and maintained by the appropriate project teams.

All Web services will eventually be stored, maintained, and available for use at an Enterprise-level, such as through an Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI).



REF: For more information on the ESB/eMI, see the eMI (a.k.a. SOA Suite) SharePoint site: [REDACTED]

4.4.2 Service Required/Consumed

VSA created its own internal (*private*) VSA web service for use by the VSA Wizard to look up and display information about existing Remote Procedure Calls (RPCs) stored on the development VistA system. This *private* web service is distributed and deployed as part of the VSA Wizard application itself.



REF: For a diagram depicting the Enterprise Shared Services between the system and subsystem modules, see [Figure 8](#).

4.5 Enterprise Architecture

VSA uses standard VA hardware and software. It adheres to the Standards Profile (SP) and only uses software that is approved by the VA Technical Reference Model (TRM) with the following notation:



NOTE: VSA is currently evaluating Log Reporting tools, which are *not* yet on the TRM. Thus, these tools have *not yet* been selected for release as part of VSA.

VSA does *not* require any special technology.



REF: For a list of VSA-related software, see Section [4.2.1](#).

The standards, strategies, and guidelines establish the fundamental technologies enabling the VA to meet many of its business and information system goals. By using these standards, the VA can:

- Promote interoperability, portability, and adaptability within systems.
- Promote quality assurance.
- Place the VA in a position to use current technology.
- Provide a framework for IT application and infrastructure development.



REF: The current TRM/SP VA Enterprise Architecture (EA) is located at: [REDACTED]

5 Data Design

This section outlines the design of the database management system (DBMS) and non-DBMS files associated with VistA Services Assembler (VSA).

5.1 DBMS Files

VSA Phase 2 does *not* create any new DBMS files. VSA uses the existing VA FileMan database and underlying global and file structure.

Table 29. VSA DBMS files

File	Input, Output, or Both	Permanent or Temporary	Module
Veterans Health Information Systems and Technology Architecture (VistA): <ul style="list-style-type: none">REMOTE PROCEDURE file (#8994)OPTION file (#19)	Input	Permanent	VA FileMan
VistA: <ul style="list-style-type: none">^XTMP("XSA-LOG-IX;; xsatid value;;xsanow value;;,\$J value;;vsa source")^XTMP("XSA-LOG;;DATE;;xsanow value;;xsatid value,\$J value")	Both	Temporary	Standards and Conventions (SAC)-compliant <i>non</i> -VA FileMan file

5.2 Non-DBMS Files

The following *non*-DBMS files on the Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5 are associated with VSA:

Table 30. VSA Non-DBMS files

File	Input, Output, or Both	Permanent or Temporary	Module
Service Descriptor files (in Extensible Mark-up Language [XML] format)	Output	Permanent	VSA Wizard Service
Java API I (Java Standard Library) for XML SOAP Web Services (JAX-WS) annotated Java class (the .java source file) that defines the Simple Object Access Protocol (SOAP) or Representational State Transfer (REST; RESTful) service and its operations.	Output	Temporary	Service Code Generator
Web Artifact (WAR) File Artifacts for SOAP service: <ul style="list-style-type: none"> • Class file that defines service operations • WEB-INF/lib: Both third-party and VSA Java Archive (JAR) files • WEB-INF/web.xml • applicationContext.xml (for Spring injection) • pom.xml WAR File Artifacts for RESTful service: <ul style="list-style-type: none"> • Class file that defines service operations • WEB-INF/lib: Both third-party and VSA JARs • WEB-INF/web.xml • WEB-INF/weblogic.xml • pom.xml 	Output	Temporary	Service Code Generator
WAR file	Output	Permanent	Service Code Assembler

5.3 Data View

VSA includes the following persistent data objects in the “VSA” namespace:

- VistA Kernel Parameters:
 - XSA LOGGING PURGE DAYS
 - XSA SITE LOGGING
 - XSA DEVELOPER LOGGING
- VistA Remote Procedure Calls (RPCs):
 - XSA GET RPC INFO
 - XSA RPC LIST
 - XSA GET LOGGING STATUS
 - XSA SET LOGGING
 - XSA GET LOG ENTRY
 - XSA LOG LIST RETRIEVAL

6 Detailed Design

This section describes the proposed VSA design in detail. It includes the necessary information for the development team to integrate the hardware components and write the software code, so that the hardware and software components provide a functional VSA product.



NOTE: Every design item should map back to the Requirements Specification Document. These should be captured in the Requirement Traceability Matrix (RTM).

6.1 *Hardware Detailed Design*

VSA requires the following hardware:

- Client Workstations for development.
- Oracle® WebLogic Server 12c (Release 12.x).
- InterSystems® Caché and VistA M Environment Server.

6.2 *Software Detailed Design*

The VSA project uses IBM® Rational Jazz suite to manage change requests made during the development phase. The specific Jazz tools include:

- Rational Quality Manager (RQM)
- Rational Requirements Composer (RRC)
- Rational Team Concert-(RTC)
a.k.a. Change and Configuration Management (CCM)

Java side development and testing is done with:

- Java Development Kit (JDK) 1.6.0_29
- RTC Client Eclipse 3.7.2 with the MyEclipse 10.7.1 plugin
- Apache® Maven 3.1
- Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5
- Apache® Velocity templating library Version 1.7
- Spring MVC 3.2.4
- Jersey RESTful Web Services Framework 1.17

Caché components that make up the VMRCS listener are developed and tested with:

- InterSystems® Caché Studio 2011.1.2.701.0.12942
- SoapUI 4.5.2

VistA (M side) components are developed with:

- InterSystems® Caché 2011.1.2.701.0.12942
- VistA M Infrastructure software (fully patched):
 - Kernel 8.0
 - Kernel Toolkit 7.3
 - VA FileMan 22.0
 - Remote Procedure Call (RPC) Broker 1.1
 - VistALink 1.6

VSA Wizard is developed with:

- Spring Framework 3.2.4

6.2.1 Conceptual Design

For the conceptual design, see Section [3](#).

6.2.1.1 Product Perspective

The VistA Services Assembler (VSA) Phase 2 effort provides a solution to the VA OI&T business need, including:

- Automation of VistA Service Oriented Architecture (SOA) service generation.
- Infrastructure utilities necessary for the execution of VistA SOA services.
- Variety of "reference implementation" service examples that prove the concept.

VA objectives prioritize the implementation of SOA as an Enterprise design. The VSA design was identified as the preferred architectural design for SOA-based systems going forward. The intent is to eventually replace interim connectivity design approaches (e.g., Medical Domain Web Services [MDWS], VistA Integration Adapter [VIA], Clinical Data Services [CDS], etc.). VSA satisfies the consumer needs currently supplied by those other design approaches. It significantly enhances:

- VistA security.
- System performance.
- Sustainability.

6.2.1.1.1 User Interfaces

The VSA Wizard web application on the WebLogic server is the user interface. The VSA Wizard has a *point-and-click* interface with free text fields from its Service Descriptor form. It can be accessed via any VA Intranet browser.



REF: For more information on the user interface, see Sections [3.2.4](#) and [8](#).

6.2.1.1.2 Hardware Interfaces

This is *not* applicable (N/A) for VSA.

6.2.1.1.3 Software Interfaces

VSA requires the following software interfaces:

- Web Browser (e.g., Microsoft® Internet Explorer)
- Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5—Development server and web service registry and repository:
 - VSA Wizard and associated services
 - Apache® Maven 3.1
 - SSL connectivity
 - VMRCS Client Listener (VSA component)
- InterSystems Caché 2011.1.2.701.0.12942:
 - Caché Server Pages turned on
 - VMRCS Listener (VSA component)
 - VMRCA Listener (VSA component)
 - VistA M Environment—Standard infrastructure required for all VistA systems at all sites. This includes but is *not* limited to the following applications:
 - Kernel 8.0
 - Kernel Toolkit 7.3
 - VA FileMan 22.0
 - Remote Procedure Call (RPC) Broker 1.1
 - VistALink 1.6
- VA Developer Workstations:
 - Microsoft® Windows 7 environment
 - Spring MVC 3.2.4 (Model-View-Controller [MVC] framework)



REF: For a list of VSA-related software, see Section [1.7](#).



REF: For more detailed information regarding the VSA environments, see the Systems Engineering and Design Review (SEDR) VSA-P2 *ACE-Map Questions-VistA Services Assembler Phase 2_MMDDYY* Excel spreadsheet located on the Enterprise Systems Engineering (ESE) SharePoint site: [\[redacted\]](#)

6.2.1.1.4 Communications Interfaces

Authorized VA users access the VSA Wizard through a web front-end via a web browser. The browser connects from within the VA's Intranet and communicates with the Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5 via Hypertext Transfer Protocol Secure (HTTPS) standard Port 443 over Transport Control Protocol/Internet Protocol (TCP/IP).

The following system communications interfaces are required for VSA.

- Hypertext Transfer Protocol Secure (HTTPS)
- Transport Control Protocol/Internet Protocol (TCP/IP)

Figure 24. VSA—Network Communications Architecture



NOTE: During initial integration testing, VSA Phase 2 will continue to register and interact direct with VSA on the Oracle® WebLogic Server 12c (Release 12.x) federating platform. For deployment, VSA Phase 2 will register services with the Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI) federating platform.

REF: For more information on the ESB/eMI, see the eMI (a.k.a. SOA Suite) SharePoint site: [\[redacted\]](#)



REF: For a list of communication requirements, see [Table 28](#).

6.2.1.1.5 Memory Constraints

The VSA Phase 2 software does *not* have any memory constraints or limits on partition size.

6.2.1.1.6 Special Operations

The VSA Phase 2 software does *not* require any special operations by the user, external devices, and archiving operations.

6.2.1.2 Product Features

The VSA Phase 2 software includes the following features and functionality:

- Platform Support—VSA Wizard runs, creates web services, and tests the services on the following supported platforms:
 - Windows Server 2008
 - Linux
- Wizard Form Validation—VSA Wizard user interface checks to make sure the user input is valid.
- RPC Parameter Passing—RPC Invocation API allows passing single or multiple parameters for those RPCs that expect multiple parameters.
- Multiple Operations—VSA supports single or multiple operations in one web service. In other words, VSA supports multiple web service methods in one web service WAR file.
- Return Types—VSA returns the following types:
 - String
 - JSON-formatted object
 - List
 - Map
- Leverages Existing Code—VSA leverages VistaLink code and design patterns. This results in:
 - Richer functionality.
 - Improved fault-tolerance.
 - Reduction of code and complexity.

6.2.1.3 User Characteristics

For user characteristics see Section [1.5](#) and the *VSA Requirements Specification Document (RSD)*.

6.2.1.4 Dependencies and Constraints

For a list of dependencies and constraints, see Sections [1.4](#) and [2.4.2](#).

6.2.2 Specific Requirements

6.2.2.1 Database Repository

For a list of VSA database repositories, see [Table 20](#).

6.2.2.2 System Features

VSA provides both a technical and organizational solution:

- **Technical Solution:**
 - Provides legacy system integration compatibility.
 - Provides SOA implementation compliance and infrastructure integration.
 - Enhances compliance with VA Enterprise Architecture and software development standards relative to SOA implementation in legacy systems.
 - Provides technical abstraction across disparate environments.
 - Provides standardization of technical characteristics and maintainability.
 - Expands (significantly) VistA extensibility and system integration opportunities by facilitating the implementation of SOA architecture.
 - Leverages existing "service assets:"
 - Provides tools and utilities that facilitate the creation and execution of auto-generated VistA SOA services (i.e., VSA Wizard).
 - Provides the ability to automate the generation of "service" needs across major initiatives.
- **Organizational Solution:**
 - Bridges and minimizes staff skill set and technical orientation boundaries.
 - Provides a standardized technical solution:
 - Addresses the organizational challenge of approaching disparate technologies relevant to legacy systems and SOA.
 - Increases VA SOA software sustainability and VistA integration extensibility.
 - Improves compatibility with "open source" participation objectives.
 - Increases software development efficiency through the automation of VistA SOA service creation.
 - Reduces investment.
 - Improves "time to market."

6.2.2.3 Design Element Tables

6.2.2.3.1 Routines (Entry Points)

The VSA Phase 2 software adds the following VistA routines (entry points).

Table 31. VSA Routines—XSARPC

Routines	Activities
Routine Name	XSARPC
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
RTM	N/A
Related Options	RPCs: <ul style="list-style-type: none"> • XSA RPC LIST • XSA GET RPC INFO

Related Routines	Routines "Called By"	Routines "Called"
	N/A	N/A

Routines	Activities
Data Dictionary (DD) References	REMOTE PROCEDURE file (#8994)
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output Reference <input checked="" type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local
Input Attribute Name and Definition	Name: APILIST(RETURN,START) Definition: Return list of RPCs in List array return type. Name: GETRPC(RETURN,XSARY) Definition: Return data dictionary data from single RPC in Global Array return type.
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic
N/A (new routine)

Modified Logic (New routine or changes are in bold)

```

XSARPC * * 123 LINES, (total 4938, comments 1899) BYTES Page 1
      RSUM: old 4646086, new 18072314
      UCI: VSA,ROU Site: VEHU MASTER JAN 8,2014@09:46

XSARPC ;ALB/MRY - VSA Service Assembler ;10/16/2013 08:30
      ;;5.2;Vista Services Assembler;***###*;Sep 4, 2013;Build 10
      ;
      Q
GETRPC(RESULTS,XSARY) --
      ;This RPC is called to return an RPC's #8994 file entry values.
      ; RPC: XSA GET RPC INFO
      ;INPUT: XSARY - literal value (rpc name)
      ; XSACAMEL - if passed in as a variable:
      ; 0 = DataDictionary label name
      ; 1 = Camelcase lettering (default)
      ;
      ;OUTPUT: RESULTS - Array of RPC information
      ;
      ;Example:
      ; ^TMP("XSA-RESULT", $J, "NAME")="XOBV TEST MULT ARRAY PARAMS"
      ; ^TMP("XSA-RESULT", $J, "TAG")="MARRAYS"
      ; ^TMP("XSA-RESULT", $J, "ROUTINE")="XOBVLT"
      ; ^TMP("XSA-RESULT", $J, "RETURN VALUE TYPE")="ARRAY"
      ; ^TMP("XSA-RESULT", $J, "AVAILABILITY")="RESTRICTED"
      ; ...
      ; ^TMP("XSA-RESULT", $J, "DESCRIPTION")=<all on one node, with lines
      ; delimited by some character like $C(10).>
      ; ...
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 1, "INPUT PARAMETER")="ARRAY"
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 1, "PARAMETER TYPE")="LIST"
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 1, "MAXIMUM DATA LENGTH")=.
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 1, "REQUIRED")="YES"
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 1, "SEQUENCE NUMBER")=1
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 1, "DESCRIPTION")="An array of
      ; information"
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 2, "INPUT PARAMETER")="ARRAY2"
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 2, "PARAMETER TYPE")="LIST"
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 2, "MAXIMUM DATA LENGTH")=.
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 2, "REQUIRED")="YES"
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 2, "SEQUENCE NUMBER")=2
      ; ^TMP("XSA-RESULT", $J, "INPUT PARAMETER", 2, "DESCRIPTION")="An array of
      ; information"
      ; ...
      ;
      N CNT,I,II,TLFLDS,FN,FNL,FLD0,FLD,FLDV,FIELD,RPCIEN,RPCIENS,MULTFLDS,X
      N WP,WPCNT,WPSTR,ERRMSG,ARRAY,ARRAY1,RESULT
      K ^TMP("XSA-RESULT", $J), ^TMP("DILIST", $J)
      S RESULT=" ^TMP("XSA-RESULT", $J) "
      S XSARY=$G(XSARY) ; rpc name
      S RPCIEN=+$FIND1^DIC(8994,"", "X", XSARY)
      I '$D(XSACAMEL) S XSACAMEL=1 ;default - write field name using camelcase
      naming convention
      I 'RPCIEN D G ERRMSG
      . S ERRMSG="RPC: "_XSARY_" not found."
      TOPLFDS ;
      S TLFLDS=".01;.02;.03;.04;.05;.06;.07;.08;.09;.11",RPCIENS=RPCIEN_", "
      D GETS^DIQ(8994,RPCIENS,TLFLDS,"NR","ARRAY")
      S ARRAY1="ARRAY(8994,"_SC(34)_RPCIENS_SC(34)_")"

```

Modified Logic (New routine or changes are in bold)

```

S FN="" F S FN=$O(@ARRAY1@(FN)) Q:FN="" D
. S:X$ACAMEL FNL=$$LOWER(FN) S:'X$ACAMEL FNL=FN
. S @RESULT@ (FNL)=@ARRAY1@(FN)
;-word processing fields #1,#3
F FLD=1,3 D
. K WP,WPSTR S X=$$GET1^DIQ(8994,RPCIENS,FLD,"Z","WP")
. Q:'$D(WP)
. F II=1:1 Q:'$D(WP(II,0)) D
.. I $D(WPSTR) S WPSTR=WPSTR_$C(10)_WP(II,0)
.. E S WPSTR=WP(II,0)
. K FIELD D FIELD^DID(8994,FLD,"","LABEL","FIELD")
. S:X$ACAMEL FNL=$$LOWER(FIELD("LABEL")) S:'X$ACAMEL FNL=FIELD("LABEL")
. S @RESULT@ (FNL)=WPSTR
;
MULTFDS ;(#2) Input Parameter
K ^TMP("DILIST",$J)
S RPCIENS="","_RPCIEN_", "
S MULTFLDS=".01;.02;.03;.04;.05"
D LIST^DIC(8994.02,RPCIENS,"","P","","","")
I '$D(^TMP("DILIST",$J)) Q
S CNT=$P(^TMP("DILIST",$J,0),"^") Q:'CNT
F I=1:1:CNT D
. K ARRAY
. S RPCIENS=I_,"_RPCIEN_", "
. D GETS^DIQ(8994.02,RPCIENS,MULTFLDS,"NR","ARRAY")
. S ARRAY1="ARRAY(8994.02,"_$C(34)_RPCIENS_$C(34)_")"
. S FN="" F S FN=$O(@ARRAY1@(FN)) Q:FN="" D
.. S:X$ACAMEL FNL=$$LOWER(FN) S:'X$ACAMEL FNL=FN
.. S @RESULT@ ($S(X$ACAMEL:"inputParameter",1:"INPUT PARAMETER"),I,FNL)=@
ARRAY1@(FN)
;-word processing field #1 of Input Parameter field #2
. S RPCIENS=I_,"_RPCIEN_", "
. K WP,WPSTR S X=$$GET1^DIQ(8994.02,RPCIENS,1,"Z","WP")
. Q:'$D(WP)
. F II=1:1 Q:'$D(WP(II,0)) D
.. I $D(WPSTR) S WPSTR=WPSTR_$C(10)_WP(II,0)
.. E S WPSTR=WP(II,0)
. K FIELD D FIELD^DID(8994.02,1,"","LABEL","FIELD")
. S:X$ACAMEL FNL=$$LOWER(FIELD("LABEL")) S:'X$ACAMEL FNL=FIELD("LABEL")
. S @RESULT@ ($S(X$ACAMEL:"inputParameter",1:"INPUT PARAMETER"),I,FNL)=WP
STR
S RESULTS=$NA(@RESULT)
K ^TMP("DILIST",$J) ;clean up
K X$ACAMEL
Q
;
LOWER(FNAME) --
;Change name to camelcase lettering
N I,F,FSTR
S FNAME=$G(FNAME)
I FNAME="" Q ""
F I=1:1 S F=$P(FNAME," ",I) Q:F="" D
. I $D(FSTR) D
.. S FSTR=FSTR_$E(F,1)_STR($E(F,2,$L(F)),"ABCDEFGHIJKLMNOPQRSTUVWXYZ","a
bcdefghijklmnopqrstuvwxyz")
. E S FSTR=$TR(F,"ABCDEFGHIJKLMNOPQRSTUVWXYZ","abcdefghijklmnopqrstuvwxyz")
Q FSTR
;
APILIST(RESULT,START) --

```

Modified Logic (New routine or changes are in bold)

```

;
;retrieve a list and return it in RESULT
N %
K ^TMP("DILIST",$J),RESULT
D FIND^DIC(8994,"",.01,"P",START,"","B")
I +$G(^TMP("DILIST",$J,0))=0 D Q
. S RESULT(1)="-1^No RPCs found."
;I +$G(^TMP("DILIST",$J,0))>25 D Q
;. S RESULT("ERROR MESSAGE")="Too many RPCs returned (>25), please narrow your search."
S %=0 F S %=$O(^TMP("DILIST",$J,%)) Q:%="" D
. S RESULT(%)=$P(^(%,0),"^",3)
K ^TMP("DILIST",$J) ;clean up
Q
ERRMSG ;
S @RESULT@("errorMsg")=ERRMSG
S RESULTS=$NA(@RESULT)
Q

```

Table 32: VSA Routines—XSAVMRCA

Routines	Activities
Routine Name	XSAVMRCA
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
RTM	N/A
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	Method invokeVMRCA^VMRCS.cls	XSAUTL \$\$NOW^XLFDT \$\$ACTIVE^XUSER DIV4^XUSER \$\$NNT^XUAF4 \$\$TOCHK^XOBVLIB \$\$STRPSUFF^XOBSCAV \$\$STA^XUAF4 \$\$KSP^XUPARAM \$\$IEN^XUAF4 \$\$ACTIVE^XUAF4 \$\$CRCONXT^XOBSCAV \$\$ENCRYP^XUSRB \$\$CHKCTXT^XOBSCAV \$\$SETTO^XOBVLIB T0^%ZOSV LOGRSRC^%ZOSV \$\$GETTO^XOBVLIB SET1^XUS SET2^XUS \$\$EZBLD^DIALOG \$\$LOW^XLFSTR Kernel and VA FileMan supported calls

Routines	Activities
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	Need following (<i>not</i> yet defined): Reference Parameter File entries.

Routines	Activities
Data Passing	<input checked="" type="checkbox"/> Input <input checked="" type="checkbox"/> Output Reference <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local
Input Attribute Name and Definition	Name: VMRCS Instruction Array Definition: per VMRCS
Output Attribute Name and Definition	Name: VMRCS Result Array & Error Array. Definition: Per VMRCS.

Current Logic
N/A (new routine)

Modified Logic (New routine or changes are in bold)
<ul style="list-style-type: none"> • Receives Instruction Array, Result Array, Error Array names. • Checks for Instruction errors. • Checks for security breaches (user access to RPC & RPC Context). • Checks VMRCA logging parameters. • Defines API signature and invoke RPC. • Returns results and/or errors in respective VMRCS array. • Cleans up and returns execution to VMRCS.

Table 33. VSA Routines—XSAUTL

Routines	Activities
Routine Name	XSAUTL
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
RTM	N/A
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	XSAVMRCA (various points) RPC: XSA SET LOGGING RPC: XSA GET LOGGING STATUS	Kernel supported APIs

Routines	Activities
Data Dictionary (DD) References	PARAMETER DEFINITION file
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input checked="" type="checkbox"/> Input <input checked="" type="checkbox"/> Output Reference <input type="checkbox"/> Both <input checked="" type="checkbox"/> Global Reference <input type="checkbox"/> Local
Input Attribute Name and Definition	Name: Logging Text, Logging Indicators, Date/time, & current ^XTMP counter variable. Definition: Self descriptive variables and documented in code comments.
Output Attribute Name and Definition	Name: Logging Level Indicator Variable and ^XTMP node (for Log item). Definition: Self descriptive in code comments.

Current Logic
N/A (new routine)

Modified Logic (New routine or changes are in bold)
<p>Functions defined in APIs:</p> <p>LOGCHK: Check for Logging status (level and on/off)</p> <p>LOGHDR: Create header node for Log file</p> <p>LOGTRACE: Create code trace entry</p> <p>LOGDATA: Create code trace entry</p> <p>LOGTRAP: Record a Log entry when logging creates a Kernel Error trap entry</p> <p>PROCTRCE: Trace conditional quits from PROCESS^XSAVMRCA</p> <p>LOGSET: Set VMRCA System Logging</p> <p>DEVSET: Set VMRCA Developer System Logging</p> <p>DERR: Set DEVERR variable</p> <p>LOGSTAT: Return a list of enabled logging</p> <p>LOGTYPE: Check for Logging & return types and names of enabled logging</p> <p>LOGLVEXT: Return External descriptions for enabled Log Level</p> <p>DATSUB1: Return the 1st subscript for ^XTMP global</p> <p>IDXSUB1: Return the 1st subscript for ^XTMP index global</p>

Table 34. VSA Routines—XSAUTL1

Routines	Activities
Routine Name	XSAUTL1
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
RTM	N/A
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	xsa.Vmrcs.cls (various points) RPC: XSA LOG LIST RETRIEVAL RPC: XSA GET LOG ENTRY	Kernel supported APIs

Routines	Activities
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input checked="" type="checkbox"/> Input <input checked="" type="checkbox"/> Output Reference <input type="checkbox"/> Both <input checked="" type="checkbox"/> Global Reference <input type="checkbox"/> Local
Input Attribute Name and Definition	Name: Logging Text, Logging Indicators, Date/time, and current ^XTMP counter variable. Definition: Self-descriptive variables and documented in code comments.
Output Attribute Name and Definition	Name: Logging Level Indicator Variable and ^XTMP node (for Log item). Definition: Self-descriptive in code comments.

Current Logic
N/A (new routine)

Modified Logic (New routine or changes are in bold)

Functions defined in APIs:

VMRCSLOG: Initialize and update VMRCS logging data

INITLOG: Initialize VMRCS Logging variable

XTMPCLN: Cleanup all Logging entries in ^XTMP [Developer Support API]

LOGLIST: Return array of ^XTMP log entry indexes

SETLGTMP: Set Log List ^TMP global

TIDQT: QUIT Check for TID loop in SETLGTMP

DATEINDX: Loop XSA Log Date Index [^XTMP("XSA-LOG-IX","DATE")] for efficient date range retrieval

GETND1: Return XSA Node 1 value prior to start node

LPPRCSRC: Loop process and source nodes and set ^TMP("XSA LOG ENTRY LIST") global

SETLG1: Set Log List ^TMP global

DATEIDX1: Loop XSA Log Date Index [^XTMP("XSA-LOG-IX","DATE")] for efficient date range retrieval

LOGSEL: Return an array with Log data from a single entry in ^XTMP log

QTCHK: Check for loop quit condition

GETSUB: Return subscript

PARAMLOG: Loop logged Input parameter data and set in result array

NODE2PRC: Logging entry node 4 processing check

DATALOG: Loop logged Input/Output data and arrays and set in result array

INSTLOG: Loop logged Instructions and set in result array

TRACELOG: Loop Traced Code logged entries and set in array

SUBLST: Return a list of array subscripts from ^XTMP logging global

6.2.2.3.2 Templates

The VSA Phase 2 software does *not* add, modify, or delete any VistA templates.

Table 35: VSA Templates

Templates	Description
Template Name	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change
RSD	N/A
Template Type	<input type="checkbox"/> Sort <input type="checkbox"/> Input <input type="checkbox"/> Print <input type="checkbox"/> Other
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	N/A	N/A

Routines	Description
Data Dictionary (DD) References	N/A
Global References	N/A

6.2.2.3.3 Bulletins

The VSA Phase 2 software does *not* add, modify, or delete any VistA bulletins.

Table 36: VSA Bulletins

Bulletins	Description
Bulletin Name	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change
RTM	N/A

Related Routines	Routines "Called By"	Routines "Called"
	N/A	N/A

N/A	Description
Mail Subject	N/A
Mail Group	N/A
Parameters	N/A
Data Dictionary (DD) References	N/A

6.2.2.3.4 Data Entries Affected by the Design

The VSA Phase 2 software does *not* add, modify, or delete any VistA data entries.

Table 37: VSA Data Entries Affected by the Design

Field Name	Current Value	New Value
N/A	N/A	N/A

6.2.2.3.5 Unique Records

The VSA Phase 2 software does *not* add, modify, or delete any VistA unique record IDs.

Table 38: VSA Unique Record ID

Field Name	Current Value	New Value
N/A	N/A	N/A

6.2.2.3.6 File or Global Size Changes

The VSA Phase 2 software does *not* change any VistA global sizes.

Table 39: VSA File or Global Size Changes

File/Global Name	Estimated Increase	Estimated Decrease
^XTMP	N/A	N/A

6.2.2.3.7 Mail Groups

The VSA Phase 2 software does *not* add, modify, or delete any VistA mail groups.

Table 40: VSA Mail Groups

Mail Groups	Activities
Mail Group Name	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	N/A	N/A

Mail Groups	Instructions
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Mail Group Description	N/A
Self-Enrollment Allowed	<input type="checkbox"/> Yes <input type="checkbox"/> No

Mail Groups	Instructions
Type	<input type="checkbox"/> Public <input type="checkbox"/> Private

6.2.2.3.8 Security Keys

The VSA Phase 2 software does *not* add, modify, or delete any VistA security keys.

Table 41: VSA Security Keys

Security Keys	Activities
Security Key Name	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change
Related Options	N/A

Related Routines	Routines "Called By"	Routines "Called"
	N/A	N/A

Security Keys	Activities
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Security Key Description	N/A
Subordinate Keys	N/A
Mutually Exclusive Keys	N/A
Granting Condition Logic	N/A

Current Logic
N/A

Modified Logic (Changes are in bold)
N/A

Security Keys	Activities
Hierarchical Precedence	N/A

6.2.2.3.9 Options

The VSA Phase 2 software adds the following VistA option:

Table 42: VSA Options

Options	Activities
Option Name	XSA RPC BROKER ACCESS
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Associated Menu Options that will invoke this reference	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference <input checked="" type="checkbox"/> None
Menu Text Description	VSA RPC Broker Access Option
Option Type	<input type="checkbox"/> Edit <input type="checkbox"/> Print <input type="checkbox"/> Menu <input type="checkbox"/> Inquire <input type="checkbox"/> Action <input type="checkbox"/> Run Routine <input checked="" type="checkbox"/> Other Broker (Client/Server)
Associated Routine	Associated RPCs: <ul style="list-style-type: none"> • XSA GET RPC INFO • XSA RPC LIST
Option Definition	This option defines the RPC Broker entries a user can access when firing up a VSA execution environment.

Current Entry Action Logic
N/A

Modified Entry Action Logic (Changes are in bold)
N/A

Current Exit Action Logic
N/A

Modified Exit Action Logic (Changes are in bold)

N/A

6.2.2.3.10 Protocols

The VSA Phase 2 software does *not* add, modify, or delete any VistA protocols.

Table 43: VSA Protocols

Protocols	Activities
Protocol Name	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change
Associated Protocols	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Item Text Description	N/A
Protocol Type	<input type="checkbox"/> Action <input type="checkbox"/> Menu <input type="checkbox"/> Protocol <input type="checkbox"/> Protocol Menu <input type="checkbox"/> Limited Protocol <input type="checkbox"/> Extended Action <input type="checkbox"/> Dialog <input type="checkbox"/> Other
Associated Routine	N/A

Current Entry Action Logic

N/A

Modified Entry Action Logic (Changes are in bold)

N/A

Current Exit Action Logic

N/A

Modified Exit Action Logic (Changes are in bold)

N/A

6.2.2.3.11 Remote Procedure Call (RPC)

The VSA Phase 2 software creates the following VistA RPCs:



REF: VSA also creates web services from existing VistA RPCs. For a description of RPCs, see Section [1.6.1](#).

Table 44: VSA RPCs—XSA GET RPC INFO

RPCs	Activities
Name	XSA GET RPC INFO
TAG^RTN	GETRPC^XSARPC
Input Parameters	RPC NAME
Results Array	<input type="checkbox"/> Single Value <input type="checkbox"/> Array <input type="checkbox"/> Word Processing <input checked="" type="checkbox"/> Global Array <input type="checkbox"/> Global Instance
Description	This RPC call returns a REMOTE PROCEDURE file (#8994) data dictionary profile of the requested RPC name.

Table 45. VSA RPCs—XSA RPC LIST

RPCs	Activities
Name	XSA RPC LIST
TAG^RTN	APILIST^XSARPC
Input Parameters	START
Results Array	<input type="checkbox"/> Single Value <input checked="" type="checkbox"/> Array <input type="checkbox"/> Word Processing <input type="checkbox"/> Global Array <input type="checkbox"/> Global Instance
Description	Returns a list of remote procedures from the REMOTE PROCEDURE file (#8994).

Table 46. VSA RPCs—XSA GET LOGGING STATUS

RPCs	Activities
Name	XSA GET LOGGING STATUS
TAG^RTN	LOGSTAT^XSAUTL
Input Parameters	N/A
Results Array	<input checked="" type="checkbox"/> Single Value <input type="checkbox"/> Array <input type="checkbox"/> Word Processing <input type="checkbox"/> Global Array <input type="checkbox"/> Global Instance
Description	Return a string of enabled logging.

Table 47. VSA RPCs—XSA SET LOGGING

RPCs	Activities
Name	XSA SET LOGGING
TAG^RTN	LOGSET^XSAUTL
Input Parameters	SITE, DEVON, DEVOFF
Results Array	<input checked="" type="checkbox"/> Single Value <input type="checkbox"/> Array <input type="checkbox"/> Word Processing <input type="checkbox"/> Global Array <input type="checkbox"/> Global Instance
Description	Return an error string or a success string.

Table 48. VSA RPCs—XSA GET LOG ENTRY

RPCs	Activities
Name	XSA GET LOG ENTRY
TAG^RTN	LOGSEL^XSAUTL1
Input Parameters	DATESUB, PROCSUB, SRCSUB, XSATIDIX
Results Array	<input checked="" type="checkbox"/> Single Value <input type="checkbox"/> Array <input type="checkbox"/> Word Processing <input type="checkbox"/> Global Array <input type="checkbox"/> Global Instance
Description	Return an array with log data from a single entry in the ^XTMP log.

Table 49. VSA RPCs—XSA LOG LIST RETRIEVAL

RPCs	Activities
Name	XSA LOG LIST RETRIEVAL
TAG^RTN	LOGLIST^XSAUTL1
Input Parameters	XSATIDIX, LOGBEG, LOGEND
Results Array	<input type="checkbox"/> Single Value <input checked="" type="checkbox"/> Array <input type="checkbox"/> Word Processing <input type="checkbox"/> Global Array <input type="checkbox"/> Global Instance
Description	Return array of ^XTMP log entry indexes.

6.2.2.3.12 Constants Defined in Interface

The VSA Phase 2 software does *not* define any constants in the interface.

Table 50: VSA Constants Defined in Interface

Name	Description
N/A	N/A

6.2.2.3.13 Variables Defined in Interface

The VSA Phase 2 software does *not* define any variables in the interface.

Table 51: VSA Variables Defined in Interface

Name	Type	Description
N/A	N/A	N/A

6.2.2.3.14 Types Defined in Interface

The VSA Phase 2 software does *not* define any types in the interface.

Table 52: VSA Types Defined in Interface

Name	Type	Description
N/A	N/A	N/A

6.2.2.3.15 GUI

The VSA Phase 2 software does *not* affect any VistA GUI.



REF: For a description of the VSA GUI, see Sections [3.2.4](#) and [8](#).

Table 53: VSA Wizard GUI

Unit Name	Description
	N/A

6.2.2.3.16 GUI Classes

There are no specific VSA GUI classes, as these classes are autogenerated from the Service Descriptor definition. The VSA Wizard uses Java classes internally. In addition to Java files, the Wizard also has JavaScript, Cascading Style Sheet (CSS), and Java Server Page (JSP) files.

Table 54: VSA Wizard GUI Classes

GUI Classes	Instructions
Class Name	N/A
Derived From Class	N/A
Purpose	N/A

6.2.2.3.17 Current Form

The VSA Service Descriptor form is displayed in [Figure 15](#).

The Service Descriptor xml schema is used to build the web form used in the VSA Wizard, and its corresponding values are captured in corresponding instances of the Java classes autogenerated from the Service Descriptor file (in XML format).

6.2.2.3.18 Modified Form

The VSA Phase 2 software does *not* modify or delete any existing forms.

6.2.2.3.19 Components on Form

For a description of the VSA Service Descriptor form fields and gadgets see [Table 25](#).

Table 55: VSA Components on Form

Name	Type	Description
See Table 25 .		

6.2.2.3.20 Events

The following events occur with the VSA Phase 2 software:

Table 56: VSA Events

Name	Type	Description
Save	Action	Saving the values in a Service Descriptor file.
Generate	Action	Generating the Web Service in a war file.
Deploy	Action	Deploying the war file to a java EE web server.

6.2.2.3.21 Methods

There are no methods associated with the VSA Phase 2 software.

Table 57: VSA Methods

Method Name	Procedure/Function	Description
N/A	N/A	N/A

6.2.2.3.22 Special References

There are no special references with the VSA Phase 2 software.

Table 58. VSA Special References

Special Reference Name	Type	Description
N/A	N/A	N/A

6.2.2.3.23 Class Events

There are no class events with the VSA Phase 2 software.



REF: For a list of events, see Section [6.2.2.3.20](#).

Table 59: VSA Class Events

Name	Type	Description
N/A	N/A	N/A

6.2.2.3.24 Class Methods

There are no class methods associated with the VSA Phase 2 software.

Table 60: VSA Class Methods

Name	Procedure/Function	Description
N/A	N/A	N/A

6.2.2.3.25 Class Properties

There are no class properties associated with the VSA Phase 2 software.

Table 61: VSA Class Properties

Class Properties Name	Type	Visibility	Description
N/A	N/A	N/A	N/A

6.2.2.3.26 Uses Clause

The VSA Phase 2 software uses the following Uses clauses:

- RPC Service for Wizard
- VistaServiceInvoker
- Service Code Generator
- Service Deployer

6.2.2.3.27 Forms

The VSA Phase 2 software does *not* create or modify any VistA ScreenMan forms.

Table 62: VSA Forms

Forms	Description
Form Name	VSA Service Descriptor form
Enhancement Category	<input checked="" type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input type="checkbox"/> No Change
Form Functionality	<p>The VSA Wizard's Service Descriptor form allows the user to:</p> <ul style="list-style-type: none"> • Search for RPCs to be called from VistA. • Retrieve and add operations and parameter mappings. • Save and store service descriptor files in XML format. • Create and deploy WAR files that are necessary in defining and publishing an RPC as a web service. • Test via a link the deployed web service and review the work flow to verify the newly auto-generated web service.

Current Form Layout

See [Figure 15](#).

Modified Form Layout (Changes are in bold)

N/A

6.2.2.3.28 Functions

The VSA Phase 2 software does *not* affect any VistA functions.



REF: For a list and description of VSA internal functions, see [Table 11](#).

Table 63: VSA Functions

Function Name	Activities
Short Description	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change
Related Options	None

Related Routines	Routines "Called By"	Routines "Called"
	N/A	N/A

Function Name	Activities
Data Dictionary (DD) References	N/A
Related Protocols	N/A
Related Integration Control Registrations (ICRs)	N/A
Data Passing	<input type="checkbox"/> Input <input type="checkbox"/> Output <input type="checkbox"/> Both <input type="checkbox"/> Global Reference <input type="checkbox"/> Local Reference
Input Attribute Name and Definition	Name: N/A Definition: N/A
Output Attribute Name and Definition	Name: N/A Definition: N/A

Current Logic
N/A

Modified Logic (Changes are in bold)
N/A

6.2.2.3.29 Dialog

The VSA Phase 2 software does *not* add, modify or delete any entries in the Vista DIALOG file (#.84).



NOTE: The VSA VistA M Routine Calling Adapter (VMRCA) listener may be adding entries in the Vista DIALOG file (#.84) file in a *future* iteration.

Table 64: VSA Dialog

Dialog	Instructions
Dialog Message (Description)	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change
Dialog Message (Description) Condition	N/A
Current Dialog Message (Description)	N/A
Modified Dialog Message (Description) (Changes are in bold)	N/A

6.2.2.3.30 Help Frame

The VSA Phase 2 software does *not* add, modify, or delete any VistA help frames.

Table 65: VSA Help Frame

Help Frame	Description
Help Frame Text	N/A
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change
Help Frame Text Calling Mechanism	N/A

Current Help Frame Text
N/A

Modified Help Frame Text (Changes are in bold)
N/A

6.2.2.3.31 HL7 Application Parameter

The VSA Phase 2 software does *not* add, modify, or delete any HL7 application parameters.

Table 66: VSA HL7 Application Parameter

HL7 Application Parameter Name	Description	
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change	
Enhancement Category	Current	Modified
Application Status	<input type="checkbox"/> Active <input type="checkbox"/> Inactive <input type="checkbox"/> Active <input type="checkbox"/> Inactive	
Facility Name	N/A	
Country Code	N/A	
HL7 Field Separator	N/A	
HL7 Encoding Characters	N/A	
Mail Group	N/A	

6.2.2.3.32 HL7 Logical Link

The VSA Phase 2 software does *not* add, modify, or delete any HL7 logical links.

Table 67: VSA HL7 Logical Link

HL7 Logical Link	Description	
HL7 Logical Link Parameter Name	N/A	
Enhancement Category	<input type="checkbox"/> New <input type="checkbox"/> Modify <input type="checkbox"/> Delete <input checked="" type="checkbox"/> No Change	
Enhancement Category	Current	Modified
Node	N/A	
Institution	N/A	
Domain	N/A	
Autostart	N/A	
Queue Size	N/A	
LLP Type	N/A	

6.2.2.3.33 COTS Interface

The VSA Phase 2 software does *not* have any COTS interfaces external to OI&T

Table 68: VSA COTS Interface

COTS Interface	Description
Communication Method	N/A
Application Interface	N/A

6.3 Network Detailed Design

Authorized VA users access the VSA Wizard through a web front-end via a web browser. The browser connects from within the VA's Intranet and communicates with the Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5 via Hypertext Transfer Protocol Secure (HTTPS) standard Port 443 over Transport Control Protocol/Internet Protocol (TCP/IP).

The following system network communications interfaces are required for VSA.

- Hypertext Transfer Protocol Secure (HTTPS)
- Transport Control Protocol/Internet Protocol (TCP/IP)



REF: For a VSA communications diagram, see [Figure 24](#).

6.4 Service Oriented Architecture / Enterprise Shared Services Detailed Design

This section provides details of provided and consumed services as follows:

- Consumed Services: VSA does *not* consume any external services, so a Service Description Document is *not* required for any internal (*non-public*) services.
- Provided Services: VSA does *not* provide any external (*public*) web services. VSA has only created an internal (*private*) web service that is *not* available to other applications.



REF: For a list and description of VSA RPCs, see the “[Remote Procedure Call \(RPC\)](#)” section.



NOTE (*ProPath SDD Template*): Any consumed or provided services will be uploaded to the Enterprise Shared Services (ESS) Registry and Repository. At some point in the near *future*, we do *not* expect these SOA artifacts (e.g., SLA, Service Description, etc.) to be static documents. They will be dynamically generated from the ESS Registry and Repository tool in the form of reports. Any application and service integration design is also documented here.

6.4.1 Service Description for VsaCommonService

The VSA Wizard uses an internal (*private*) service called VsaCommonService; it is deployed as part of the same EAR file as the VSA Wizard itself. This service is used by the RPC Search functionality in the VSA Wizard to:

1. Obtain a list of the Remote Procedures that start with the specified letters on the development VistA system.
2. Obtain the information (field data) for a selected Remote Procedure Call (RPC).



NOTE: In the *future*, this service will be converted from a SOAP service to a RESTful service.

6.4.2 Service Design for Provided Services

VSA provides the *mechanism* to create services from Remote Procedure Calls (RPCs). It does *not* distribute any services; other than the VsaCommonService (*private*) web service that is consumed internally by the VSA Wizard. This internal service is *not* public. It is a developer tool, as is the VSA Wizard, so it will *not* be available on the Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI) or any other public service registry and repository.

As a test demonstration of the VSA tool, however, the VSA development team is collaborating with other project teams to create specific services based on identified use cases. For example:

- Veteran Point of Service (VPS).
- VistA Evolution/Mobile Development. The Patient Viewer phase 2 will be writing orders. They will be the first application to test the VSA solution that will resolve or come up with a solution for the session management problem.



ATTENTION: These demonstration services are *not* owned or distributed by VSA; they are owned and maintained by the appropriate project teams.

All Web services will eventually be stored, maintained, and available for use at an Enterprise-level, such as through an Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI).



REF: For more information on the ESB/eMI, see the eMI (a.k.a. SOA Suite) SharePoint site: [REDACTED]

6.4.2.1 Introduction

6.4.2.1.1 Purpose and Scope of Service

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.1.2 Links to Other Documents

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.2 Service Details

6.4.2.2.1 Service Identification

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

Table 69. VSA Provided Services—Service Identification

Service Attribute	Value
Name and Alias (if any)	N/A
Overview	
Version	
Latest Status	
Service Type	
Architecture Layer	
Business Domain	
Service Domain	
Business Organization and Owner	
Technical Organization and Owner	
Development Organization and Owner	
Support Organization and Owner	
Target Consumer Organization(s) and Owner(s)	

6.4.2.2.2 Service Versions

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

Table 70. VSA Provided Services—Service Versions

Version Numbers	Current Status of Version	A Brief Description of the change implemented in that version
N/A		

6.4.2.2.3 Summary of Design and Platform Details

6.4.2.2.3.1 SOA Patterns Implemented

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.2.3.2 COTS Platform vendor names and versions for hosting platform

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.3 Dependencies

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4 Service Design Details

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.1 Interface Technical Specs

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.1.1 Service Invocation Type

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.1.2 Service Interface Type

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.1.3 Service Name

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.1.4 Interface

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.1.5 End Points

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.1.6 Operations or Methods

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

Table 71. VSA Provided Services—Operations or Methods

Operation Name	Inputs	Outputs	Transactional Qualities if relevant (Updating?, Atomic?, Can participate in transaction?)	Pre and Post Conditions	Exceptions
N/A					

6.4.2.4.1.7 Message Schemas

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.2 Information Model

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.2.1 Class Diagram and Description of Entities Involved

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.2.2 Mappings from ELDM to Standards Based Schemas

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.3 Behavior Model (a.k.a. Use Case Realization)

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.3.1 Use Cases (Use Case Model)

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.4.3.2 Interaction Diagrams

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.5 Gap Analysis

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

Table 72. VSA Provided Services—Gap Analysis

Design Elements→ Policies / Service Level Definition (SLD) elements etc.↓	Design Element A	Design Element B	Design Element C	Comment for non-conformance
N/A				

6.4.2.5.1 Variances from Enterprise Target Architecture

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.5.2 Variances from SLDs

N/A, no Service Level Definitions (SLDs). VSA does *not* distribute any services, other than those services consumed internally by VSA itself.

6.4.2.5.3 Variances from Standards and Policies

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

6.4.2.5.4 Justification for Exceptions and Mitigation

N/A; VSA does *not* distribute any services; other than those services consumed internally by VSA itself.

7 External Interface Design

External systems are systems that are *not* within the scope of VistA Services Assembler (VSA), regardless of whether the other systems are managed by the vendor or its client.

This section describes the interfaces between VSA and external systems and/or subsystems.

7.1 Interface Architecture

The Federating Platform serves as the registry and repository for storing Service Descriptor and Web Archive (WAR) files (web services). It federates routing of queries from provider and consumer "service" requests to and from Veterans Health Information Systems and Technology Architecture (VistA). It also provides the security interface that controls single sign-on (SSO) access, so a user's authentication token is trusted across multiple VA Information Technology (IT) systems or organizations.

The VSA Federating Platform does the following:

- Stores Service Descriptor files in Extensible Mark-up Language (XML) format and published WAR files (web services). A federated web service contains one or more operations. Each operation corresponds to a Remote Procedure Call (RPC).
- Allows service-Providing applications the ability to expose (publish) VistA functionality (i.e., RPCs) through Simple Object Access Protocol (SOAP) and RESTful web services.
- Allows service-Consuming applications to request and invoke federated web services.

For VSA the federating platforms are as follows:

- VSA Phase 2 software— Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5 on the Enterprise Development Environment (EDE) VMs, which includes the following platforms:
 - Linux: 2 VM.
 - Windows: 8 VM ("Dev-in-a-box;" primarily for contractors without Government Furnished Equipment (GFE) laptops.

The EDE Data Center is managed by:

- VA—Enterprise Operations (EO)
- Contractor—MITRE Corporation
- For national release of VSA, the Federating Platform will be the Enterprise Service Bus (ESB)/Enterprise Messaging Infrastructure (eMI).



REF: For more information on the ESB/eMI, see the eMI (a.k.a. SOA Suite) SharePoint site: [REDACTED]

7.2 Interface Detailed Design

7.2.1 VistaServiceInvoker Application

The VistaServiceInvoker application on the VSA Federating Platform is a Java class that delegates and passes an instruction object to the VistA M Routine Calling Service (VMRCS) listener, which is implemented as a Caché SOAP web service.

It knows how to take the inputs from the generated business service, package them up in a proper instruction object, and calls the VMRCS listener. The VistaServiceInvoker uses a Java web service client (VSA VistA Client) auto-generated from the VMRCS listener Web Service Description Language (WSDL).

7.2.2 VistA M Routine Calling Service

The VMRCS listener is an internal Caché web service written and deployed in a VistA instance. The VMRCS listener is a *private* service that can only be consumed by VSA-generated services.

The VMRCS client listener is used by VSA-generated services to invoke the VMRCS listener web service on the Caché system.

The VMRCS listener does the following:

- Serves as an HTTP listener using Caché objects.
- Employs a two-way SSL connectivity and is responsible for delegating requests to run M routines to the VistA M Routine Calling Adaptor (VMRCA) listener.
- Facilitates the use of M routines as the basis for VistA SOA service application business logic.
- Transforms payloads between M routine compatible syntax and various external formats (e.g., XML and JavaScript Object Notation [JSON]), and type conversions to JSON, etc.

7.2.3 VistA M Routine Calling Adapter

The VMRCA listener is a set of M routines bundled as a VSA package (VSA namespace = "XSA"), which:

- Integrates the VMRCS listener component with the traditional VistA M computing environment.
- Checks a VMRCS instruction for correctly formatted and required legacy process invocation information, which provides for the invocation of M routines.
- Builds the legacy M API signature to execute the legacy process and returns the results to the VMRCS listener.
- Performs Kernel authentication. It uses the same instructions to allow authentication to Kernel.
- Returns the response payload from VistA to the VMRCS listener.

8 Human-Machine Interface

This section describes the VistA Services Assembler (VSA) human-machine interface (i.e., Graphical User Interface [GUI]) relative to the user.

8.1 Overview

VSA provides a GUI of the VSA Wizard utility that is:

- Web-based
- Modular
- Customizable

The GUI consists of a structured *point-and-click* interface. There are also free-text fields on a Service Descriptor form for the user to enter information. VSA complies with VA's Web development standards, including Section 508 of the Rehabilitation Act. It consists of a Service Descriptor form that enables a user to enter and save information that can be used to create a Veterans Health Information Systems and Technology Architecture (VistA) Service Oriented Architecture (SOA) service that invokes a remote procedure identified by a developer or system integrator.

The VSA Wizard web application user interface is a browser-based client that graphically integrates required system functions. It provides the ability for the service developer to search for and display information about a remote procedure on the development VistA system. Certain forms or fields are editable depending on the information already entered into the VSA Wizard form.

8.2 Interface Design Rules

This section identifies the conventions and standards for designing the VSA GUI.

The VSA Wizard GUI application and Service Descriptor form on the Oracle® WebLogic Java EE application server interfaces with VistA environments to help create a web service that exposes selected Remote Procedure Calls (RPCs) defined in the development account.



NOTE: All of the values entered into the data fields on the Service Descriptor form are saved in the Service Descriptor file in Extensible Mark-up Language (XML) format. This file is the key configuration file that is created via the VSA Wizard.

Authorized VA users access the VSA Wizard through a web front-end via a web browser. The browser connects from within the VA's Intranet and communicates with the Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5 via Hypertext Transfer Protocol Secure (HTTPS) standard Port 443 over Transport Control Protocol/Internet Protocol (TCP/IP).

Upon creation of a Service Descriptor file (XML format), part of the VSA Wizard's functionality calls the VsaCommonService web service component used to interrogate VistA for available RPCs and retrieve all the details of a given RPC to help pre-populate the VSA Wizard fields. The user does *not* interact directly with the VsaCommonService component; rather, the VSA Wizard communicates with the RPC web service via HTTPS. The VSA Wizard uses a number of Java classes that are *not* visible to the user but provisions the functionality for the creation and auto-generation of web services. The VSA Wizard uses a Java Enterprise Edition (Java EE) object-oriented platform. Java EE is an open source standard and can be shared with other open source development efforts.

The most important feature of VSA is the advantage for M developers from seeing any web service artifact (e.g., Web Services Description Language [WSDLs], Java code, etc.) generated by VSA utilities. The VSA Wizard utility provides every "producing" application who wants to expose some VistA functionality as a web service. The VSA Wizard prompts the provider with information about the VistA functionality. From this information, a Java API (Java Standard Library) for XML REST Web Services (JAX-RS) or Java API I (Java Standard Library) for XML SOAP Web Services (JAX-WS) annotated Java class is generated, and assembled along with other file artifacts into a deployable WAR file. When the Java web service is deployed, the web service's corresponding WSDL becomes available via a URL. As an example, a VistA Pharmacy M developer can use the VSA Wizard web form to create a Pharmacy web service *without* having to create a WSDL or Java class manually. A "Consuming" application developer (Java service developer), wants to look at the WSDL, or at least be told where the WSDL file from the Pharmacy web service is located. The WSDL file is used to generate the Pharmacy web-service client API and provides the developer the ability to write Java code against the created client Application Programming Interface (API).

There is also the WSDL that represents the VMRCs listener SOAP service on the VistA side. This WSDL is generated by Caché and is used only by the VistaServiceInvoker component of VSA. This web service is necessary for VSA to communicate between WebLogic and Caché; however, it is internal and only accessible to VSA.



NOTE: In subsequent VSA development iterations, several VSA-generated services will be created as a "reference implementation" to demonstrate the capabilities of VSA and the viability of using auto-generated services to execute VistA logic. These "reference implementation" services will interface with the M logic (routines and RPCs) of various VistA domain applications.

8.2.1 Inputs

The VSA Phase 2 software allows the user (e.g., service developer) to use a VA Intranet web browser to access the VSA Wizard on the Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5. The user can modify existing or create new Service Descriptor files, which are used to create VSA web services (WAR files).

For security purposes, the user is prompted to enter their logon credentials to access the VSA Wizard application (GUI) on the Oracle® WebLogic Server 12c (Release 12.x) on Windows Server 2008 or on Red Hat Enterprise Linux release 6.5.

The user is presented with a Service Descriptor form to enter the information to edit or create a Service Descriptor file in XML format. VSA validates the user entries.

8.2.2 Outputs

The VSA Phase 2 software creates the following display screens and output files relative to the user:

- Display Screens:
 - VSA Wizard main page (GUI front-end), see [Figure 25](#).
 - VSA Wizard Service Descriptor form, see [Figure 26](#). Includes display of any VSA system messages.
- Output Files:
 - Service Descriptor XML file, which is used to generate the WAR file.
 - WAR file, which is used to generate the web service.

8.3 Navigation Hierarchy

The following images show how a user moves through the GUI.

8.3.1 VSA Wizard—Main Page

Figure 25. VSA Wizard—Main Page (Screen)

VSA Wizard

Service Descriptor XML File

Locate an existing service descriptor file or create a new one

Navigate to the file path that contains the service descriptor file or type a path in the Path field and press Enter. Once the file is located, click the name of the file to begin working with it.

Press the Create button to create a new service descriptor file.

Path:

Current Location: localhost \ C: \ Oracle \ config \ domains_12.1.3 \ base_domain

- autodeploy
- bin
- config
- console-ext
- generatedWebService
- init-info
- lib
- nodemanager
- pending
- security
- servers
- tmp
- gov.va.med.test.TestService-1.0.vsa.sdf.xml

VistA Services Assembler (VSA)
Version: 1.0.0
Date of build: 20140904-1054



REF: For a definition of all data elements on this screen and subscreens, see [Table 24](#).

8.3.2 VSA Wizard—Service Descriptor Form

Figure 26. VSA Wizard—Service Descriptor Form for RESTful Services

VSA Wizard

Back to Service Descriptor selection page

Service Information

Enter information about this web service

The Service Name is the name of the service as exposed to client applications. The Version is the current version of this web service. The Service Namespace is a URI that corresponds to this web service. The Java Package identifies the package to use for generated Java classes.

Service Type: ☐ SOAP ☒ REST

Service Name:

Version:

Java Package:

REST-Specific Information:

URL Path:

Life Cycle:

Produces:

Consumes:

Vista Remote Procedure Selection

Look up an RPC and autogenerate an operation

Look up an RPC on the Vista system and autogenerate a web service operation that corresponds to the selected RPC.

Enter the first few characters of the RPC and press Search. A list of RPCs that start with the input value appears. Select an item in the list to display information about that RPC. Press Autogenerate to automatically generate an operation that corresponds to that RPC.

RPC Search

Operation Information

Create web service operations

Edit the autogenerated operations or manually create and edit web service operations.

Operations:	RPC Name	Operation Name	Resp Type	HTTP Method	URL Path	Consumes	Produces	
			string					+ -

Actions

Process the web service

Save the service descriptor XML file and create, deploy, and test the web service.

Step 1: Save the form to a service descriptor XML file.

Step 2: Create the web service as a WAR file.

Step 3: Deploy the web service to WebLogic.

Vista Services Assembler (VSA)

Version: 1.0.0

Date of build: 20140904-1054

Figure 27. VSA Wizard—Service Descriptor Form for SOAP Services

VSA Wizard
Back to Service Descriptor selection page

Service Information

Enter information about this web service

The Service Name is the name of the service as exposed to client applications. The Version is the current version of this web service. The Service Namespace is a URI that corresponds to this web service. The Java Package identifies the package to use for generated Java classes.

Service Type: ☒ SOAP ☐ REST

Service Name:

Version:

Java Package:

SOAP-Specific Information:

Service Namespace:

Vista Remote Procedure Selection

Look up an RPC and autogenerate an operation

Look up an RPC on the Vista system and autogenerate a web service operation that corresponds to the selected RPC.

Enter the first few characters of the RPC and press Search. A list of RPCs that start with the input value appears. Select an item in the list to display information about that RPC. Press **Autogenerate** to automatically generate an operation that corresponds to that RPC.

RPC Search

RPC Name:
Search

Operation Information

Create web service operations

Edit the autogenerated operations or manually create and edit web service operations.

Operations:

RPC Name	Operation Name	Resp Type	
<input type="text"/>	<input type="text"/>	string	<input type="button" value="+"/> <input type="button" value="x"/>

Actions

Process the web service

Save the service descriptor XML file and create, deploy, and test the web service.

Step 1: Save the form to a service descriptor XML file.

Step 2: Create the web service as a WAR file.

Step 3: Deploy the web service to WebLogic.

Vista Services Assembler (VSA)
Version: 1.0.0
Date of build: 20140904-1054



REF: For a definition of all data elements on this screen and subscreens, see [Table 25](#).

9 Security and Privacy

9.1 Security

The VistA Services Assembler (VSA) system designers gave consideration to integrity controls in order to restrict the loss, misuse, modification of, or unauthorized access to information that could affect the vendor or its customers.

The VSA Phase 2 software provides minor security and system integrity controls. It also provides improved error messages returned from the VistA M Routine Calling Service (VMRCS) and VistA M Routine Calling Adapter (VMRCA) listeners.

Future VSA versions will implement greater security and system integrity controls to ensure that the following minimum levels of control are included:

- Ability to identify audit information by user identification; network terminal identification; date, time, and data accessed or changed.
- Audit procedures to meet control, reporting, and retention period requirements.
- Controls to restrict access of critical data items.
- Verification processes for additions, deletions, or updates of critical data.
- Debug logging control and query utilities.



NOTE: In addition to "service descriptors," in *future* phases the VSA Wizard will maintain an audit record of VistA SOA service generation actions that have occurred (including the actions taken, developer, date/time, etc.).



REF: For a list of VSA security and privacy-related requirements, see the "Security Specifications" section in the *VSA Requirements Specifications Document (RSD)*.

9.2 Privacy

Future VSA versions of VSA will implement secure socket (Hypertext Transfer Protocol Secure [HTTPS]) messaging and reference VA Identity and Access Management (IAM)-certified user authentication and authorization services *before* access to legacy Veterans Health Information Systems and Technology Architecture (VistA) data is allowed.



REF: For a list of VSA security and privacy-related requirements, see the "Security Specifications" section in the *VSA Requirements Specifications Document (RSD)*.

10 Attachment A—Approval Signatures

The signature below is an acknowledgement that the signatory understands the purpose and content of the VistA Services Assembler (VSA) Phase 2 (VSA-P2) System Design Document (SDD).

Signed:

_____, Business Sponsor Date

_____, Information Technology (IT) Program Manager Date

Office of Information and Technology (OI&T)/Product Development (PD) – Health Care Delivery
Projects Division Date
VSA-P2 Project Manager

_____,
Service Delivery and Engineering (SDE) Program Administration Office (PAO) Project Manager Date
Enterprise Platform Group

_____, Integration Architect Date

_____, Chief Business Office (CBO) Date
Director Veteran Point of Service (VPS) Project

(Reference Implementation—Demonstration of VSA web services created for specific project Use Cases.)

██████████, Co-Director Connected Health

Date

VHA Office of Informatics & Analytics

VistA Evolution/Mobile Development; Patient Viewer Phase 2 Project Representative

(Reference Implementation—Demonstration of VSA web services created for specific project Use Cases.)

11 Appendix A—Additional Information

11.1 RTM

For the VistA Services Assembler (VSA) Requirement Traceability Matrix (RTM), see the *VSA Requirements Specifications Document (RSD)*.

11.2 Package and Installation

Outline any special considerations for software packaging and installation.

11.3 Design Metrics

Describe all metrics to be used during the design activity.

11.4 Acronym List and Glossary

For acronyms and glossary terms see the “[Definition, Acronyms, and Abbreviations](#)” section.

11.5 Required Technical Documents

11.5.1 VSA Documentation

Readers who wish to learn more about VSA should consult the following:

- *VistA Services Assembler (VSA) Project Charter*
- *VistA Services Assembler (VSA) IPT Charter*
- *VistA Services Assembler (VSA) Project Management Plan (PMP)*
- *VistA Services Assembler (VSA) Business Requirements Document (BRD)*
- *VistA Services Assembler (VSA) Requirements Specifications Document (RSD)*
- *VistA Services Assembler (VSA) System Design Document (SDD)* (this manual)
- *VistA Services Assembler (VSA) Master Test Plan*
- *VistA Services Assembler (VSA) Installation Guide*
- *VistA Services Assembler (VSA) Systems Management Guide*
- *VistA Services Assembler (VSA) Developer’s Guide*
- *VistA Services Assembler (VSA) User Guide*
- VistA Services Assembler (VSA) VA SharePoint sites:

- **Public**

Site: [REDACTED]

[.aspx](#)

- **Project Site:** [REDACTED]

11.5.2 VistA Documentation

Nationally released VistA documentation is available in the following formats and can be downloaded from several document repositories.

- **Formats:**
 - Microsoft Word (doc or docx).
 - Adobe Acrobat Portable Document Format (pdf).



REF: The PDF documents *must* be read using the Adobe Acrobat Reader, which is freely distributed by Adobe Systems Incorporated at the following Website: <http://www.adobe.com/>

- **Repositories:**
 - VA Software Document Library (VDL) Website: [REDACTED]
 - Product Support (PS) Anonymous Directories.

11.5.3 Standards and Guidelines

The VSA project team, software, and test servers adhere to the following directives, policies, procedures, standards, and guidelines:

- [Program Management Accountability System \(PMAS\)](#) and [ProPath](#) guidelines and processes—To monitor project progress.
- [Technical Reference Model \(TRM\)](#)—For approved tools and standards.
- MUMPS (M) American National Standards Institute (ANSI) X11.1-1995 standard: <http://71.174.62.16/Demo/AnnoStd> (Annotated).
- VA Standards & Conventions Committee (SACC) Codes Standards and Conventions: [REDACTED] [communities/app_dev/sac/default.aspx](#)
- Java Platform, Enterprise Edition (Java EE) Architecture (Eclipse documentation): <http://help.eclipse.org/indigo/index.jsp?topic=%2Forg.eclipse.jst.j2ee.doc.user%2Ftopics%2Fejarch.html>.
- [World Wide Web Consortium \(W3C\) Simple Object Access Protocol \(SOAP\) Standard](#).
- [World Wide Web Consortium \(W3C\) Extensible Markup Language \(XML\) Standard](#).
- [Federal Guidelines—IAM Service: VA Handbook 6500-Information Security Program](#).

- VA Directive 6102—F or the Enterprise Knowledge Management Solution and to serve the purpose of guaranteeing all VA Internet and Intranet standards for systems operations are considered.
- Section 508 of the Rehabilitation Act (29 U.S.C. § 794d): [VA Section 508 Home Page](#).
- Enterprise Shared Services/Service Oriented Architecture (ESS/SOA):
[REDACTED]ea.oit.va.gov/ess-design-guidance-documents/
- National Institute of Standards and Technology (NIST) and Federal Desktop Core Configuration (FDCC) Guidelines—For servers regarding build-outs and security settings. Servers use standard protocols and ports for communication and network settings:
 - NIST SP 800-44 Checklist to ensure specifications are listed in compliance for securing public Web servers; *Guidelines on Securing Public Web Servers* document: <http://csrc.nist.gov/publications/nistpubs/800-44-ver2/SP800-44v2.pdf>.
 - *NIST Electronic Authentication Guideline* document: <http://csrc.nist.gov/publications/nistpubs/800-63-1/SP-800-63-1.pdf>.
- [Federal Identity, Credential, and Access Management \(FICAM\) Roadmap and Implementation Guidance Roadmap and Implementation Guidance](#) document.
- Electronic and Information Technology Accessibility Standards (36 CFR 1194).
- Federal Information Security Management Act (FISMA) of 2002.
- VAAR 852.273-75 Security Requirements for Unclassified Information Technology Resources (interim Oct 2008).
- FIPS Pub 201, Personal Identity Verification for Federal Employees and Contractors, February 25, 2005.

