# *VISTA* Data Project

Final Report
2019

The VHA Information Systems Technology Architecture (*VISTA*) is the U.S. Department of Veterans Affairs integrated longitudinal clinical, business, and administrative information system. VHA operates 130 *VISTA* instances to support the operations of over 1200 VHA hospitals and clinics nationwide.

VISTA's data model - the roadmap to VHA's institutional, business, and clinical processes and data embedded within all VISTA databases - has evolved organically over the past 35 years but has never been exposed, audited, and leveraged in modern, mainstream, maintainable form.

In the VISTA Data Project (VDP), VISTA's organic data model is comprehensively exposed, transformed, and incrementally normalized to produce a national, standardized Master *VISTA* Data Model (MVDM). An operationalized MVDM provides a single, secure, symmetric read-write interface to every VISTA and, through *emulation* of existing interfaces, enables VISTA's current clients such as CPRS work unchanged but with greatly enhanced security, simplicity, and efficiency.
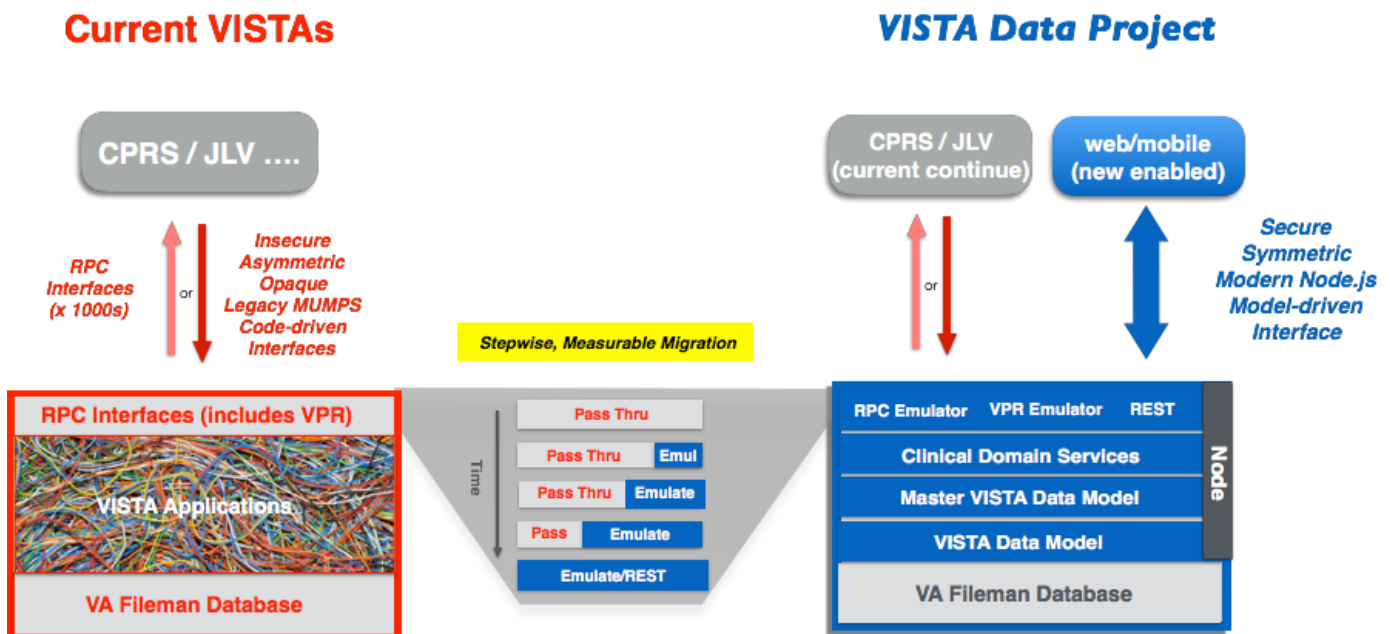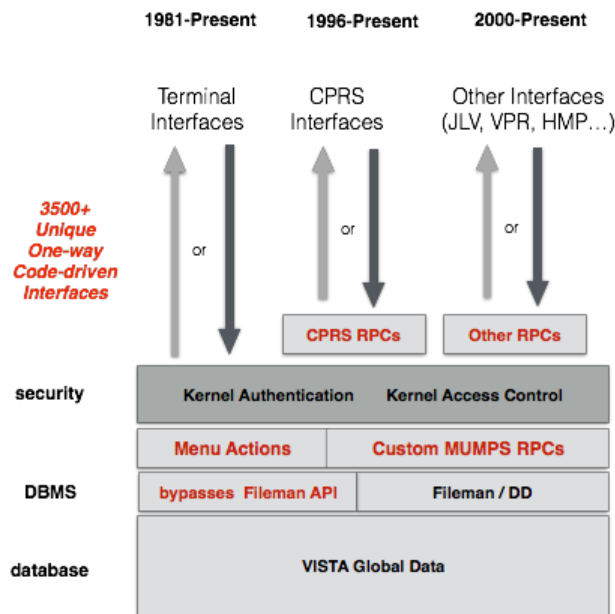

Strategic Benefits:

- New, maintainable veteran care server based on mainstream technology
- New web and mobile clients enabled with mainstream technology
- Current clients (CPRS/JLV) supported and enforce VA Care coverage
- May now safely incrementally retire legacy MUMPS VISTA [spaghetti]
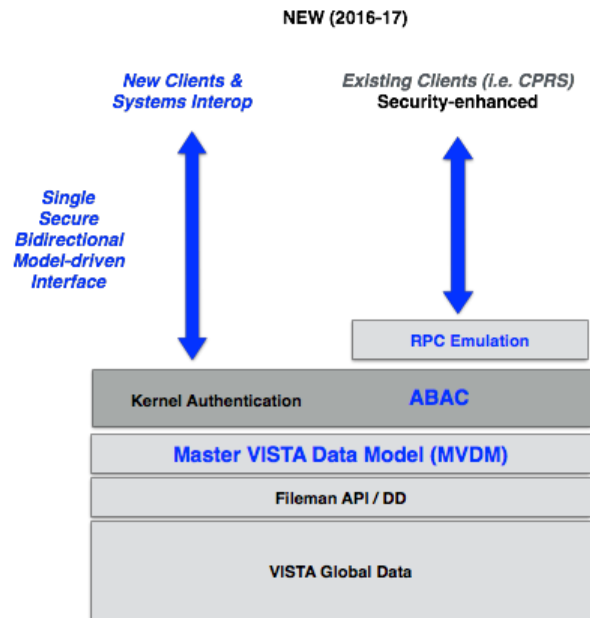- Clinical Domain Services may be implemented over COTS

VDP Feature:
Modern, Secure Model-Driven Interface

VISTA currently has a variety of bespoke and redundant access, auditing, and authentication mechanisms including FileMan and Remote Procedure Call (RPC) permissions. MVDM introduces (1) comprehensive patient-centric security (2) industry standard attribute-based access control (ABAC) and (3) tiered auditing to VISTA. This new modern data-centric security mechanism allows all data to be safely liberated in modern, mainstream, cloud-native environments where all of the *VISTA* systems are now hosted.

Code-centric Interfacing (3500+)

1981-Present	1996-Present	2000-Present
Terminal Interfaces	CPRS Interfaces	Other Interfaces (JLV, VPR, HMP…)

3500+ Unique One-way Code-driven Interfaces

or

or   or

CPRS RPCs	Other RPCs

security

Kernel Authentication	Kernel Access Control

Menu Actions	Custom MUMPS RPCs

DBMS

bypasses Fileman API	Fileman / DD

database

VISTA Global Data

Model-centric Interface (1)

NEW (2016-17)

New Clients & Systems Interop   *Existing Clients (i.e. CPRS)* **Security-enhanced**

Single Secure Bidirectional Model-driven Interface

RPC Emulation

Kernel Authentication	ABAC

Master VISTA Data Model (MVDM)

Fileman API / DD

VISTA Global Data

Current interfacing to VISTA is through over 3500 unique, opaque, one-way (either read or write) legacy (over 20 years old) MUMPS remote procedure calls (RPCs) which are neither documented nor maintained. These are hard-coded in MUMPS for specific clients only and not interchangeable to other clients due to embedded business logic within the custom MUMPS and client code. Security for RPCs is based on the Terminal (roll-and-scroll) interface and its Menu Actions. This is a terminal-specific security mechanism, and not applicable for any external, web-, or GUI-based interface.
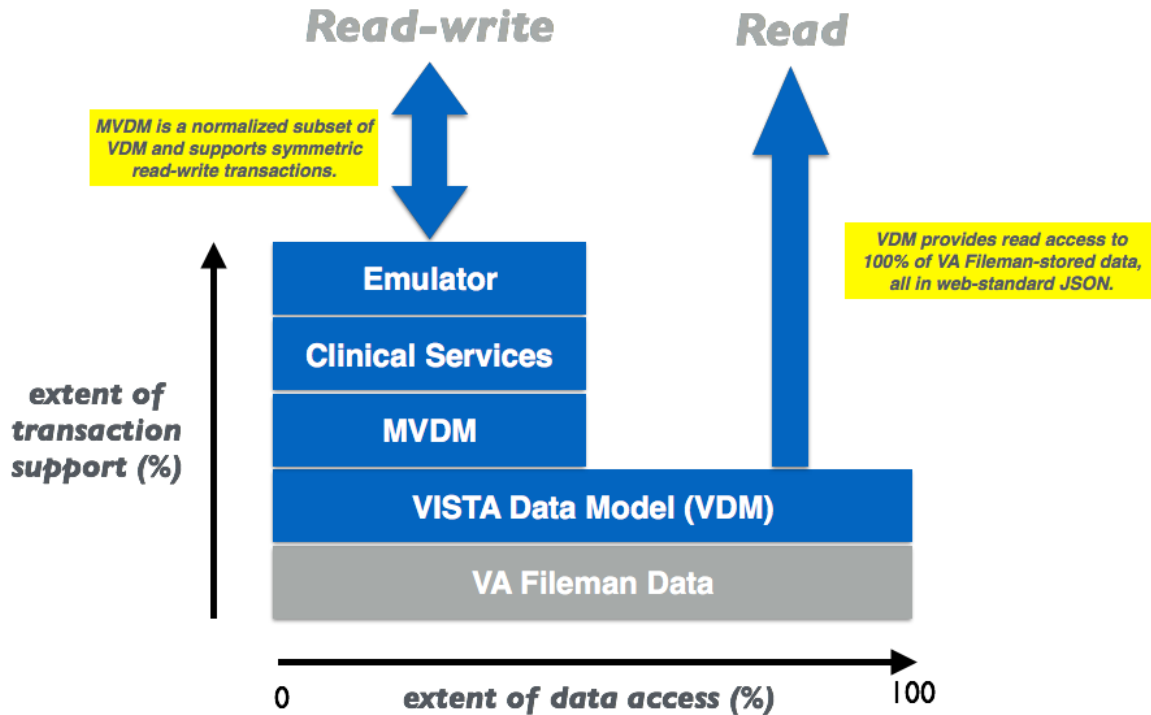
Many of the 3500 RPCs bypass the Fileman API and Data Dictionary, writing direct to MUMPS global storage. Bypassing the Fileman API means that the security and auditing measures of the database are bypassed, creating a significant security gap. In addition, this makes the data inaccessible to any other applications or by any other method other than by writing yet more custom MUMPS RPCs (The read and write RPCs are distinct from each other). The only means to access or interface to new data is to write new MUMPS RPCs using the Terminal-based Actions-centric security, in addition to custom RPC MUMPS security code.

All interfacing to VISTA is through one single, secure, symmetric (bidirectional) read-write Master VISTA Data Model (MVDM). The read data model is identical to the write data model (i.e. symmetric) providing one single universal structured data read/write mechanism. All interfaces and functionality are Model-driven, language-agnostic, client-agnostic, Fileman API compliant, and secured with both existing Kernel authentication, in addition to new modern, industry-standard, patient-centric, Attribute-Based Acess Control (ABAC).

Existing RPC-based clients read/write through the MVDM via RPC emulation, and are ABAC security-enhanced. New clients and external structured data integration directly uses the MVDM with ABAC security. Red= MUMPS code that may be retired; Blue = enhanced replacement

VDP Feature:
Comprehensive Read-Write Data Access

VDP provides 100% coverage of all VISTA data (all 85,000 fields) in web-standard JSON form and – incrementally - is building out a normalized master model for the functionality used by CPRS and other VISTA clients.

Read-write   **Read**

MVDM is a normalized subset of VDM and supports symmetric read-write transactions.

VDM provides read access to 100% of VA Fileman-stored data, all in web-standard JSON.

Emulator

Clinical Services

MVDM

VISTA Data Model (VDM)

VA Fileman Data

extent of transaction support (%)

0   extent of data access (%)   100

VDP Feature:
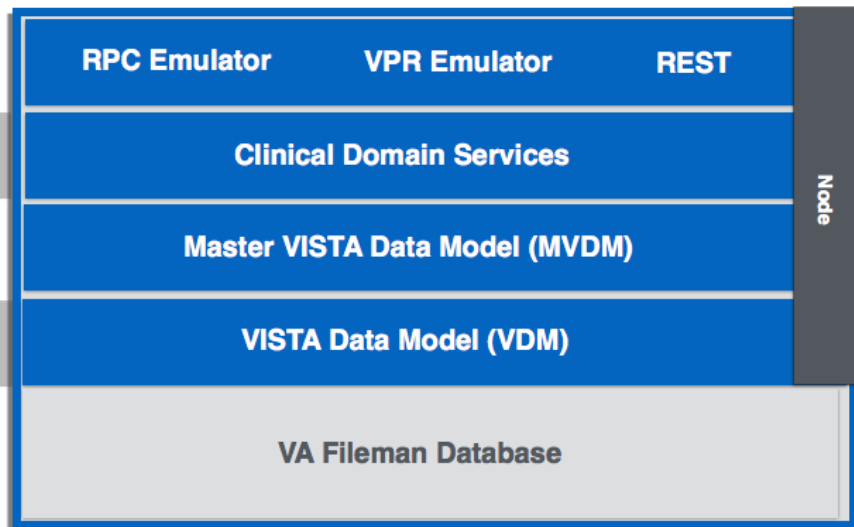# Layered Service-Based Data Architecture

MVDM runs inside VISTA, replacing its spaghetti of M code and interfaces with a modular, model-based, backward-compatible server, making VISTA more maintainable using off-the-shelf tools and technologies.

· Emulation and New Interfaces
· All reduce to same service interactions

· (Problem, Pharmacy …) Services over MVDM
· Patient level selection and security

· Normalizes VDM
· Distinguishes Veteran and Patient/Clinical specifics
· A Clean "CRUD+R"/Events paradigm

· Transparent JSON of the native model
· Read for 100% data in FileMan
· Write Tested for MVDM covered classes

· All interaction through formal FileMan API
· Only FileMan changes fix Data Dictionary (DD)

RPC Emulator	VPR Emulator	REST
Clinical Domain Services		
Master VISTA Data Model (MVDM)		
VISTA Data Model (VDM)		
VA Fileman Database		

Node

**Structured VISTA Server
(mainstream, modular, extensible)**

■ Javascript/Node.js

VDP Deliverables

The technical deliverables enumerated in PWS that have been completed and delivered.

Deliverable	Name	Where
7	Machine Processable VISTA Data Model (VDM)	Code: VDM in Domain Prototypes
8	Date-stamped FileMan Data Model Definitions	Code: otherDeliverables
9	Approach to "Live VDM" Maintenance of	Final Report: Live VDM Maintenance
10	Normalized VISTA Data Model	Code: MVDM in Domain Prototypes
11	Heuristic Code	Code: Domain Prototypes
12	Normalization Reports	Final Report: MVDM normalization of VDM
13	Website that graphically depicts VDM, MVDM, and	Code: Website of Project
14	Report on Exposure of Older Models	Final Report: Exposure of Older
15	Date-stamped Metadata for lab, surgery, TIU notes and other applications in a MVDM compatible	Code: otherDeliverables
18	Machine-processable Annotations	Code: otherDeliverables
19	Software Code	Code: otherDeliverables
23	End to end Demonstration	*Final Report: End To End Emulation Demonstration – assembled from code deliverables*
25	Prototype query access to VISTA Data	Code: otherDeliverables
28	Prototype Patient-centric Data	Code: otherDeliverables
33	Prototype Web-Based Query Interface to	Code: otherDeliverables
35	VISTA Application	Code: Domain Prototypes
36	Meta-model(s)/Prototype(s) (VPR	Code: Domain Prototypes - VPR
41	Security Management Summary Report on	Final Report: Security Management

Note: per agreed contract amendment, deliverables 23 and 41 replaced 39 and 40

VDP
Exposure of Management of VistA Models Report

The Vista Data Model (VDM) is generated directly from the internal data model ("Data Dictionary") of VISTA and reflects different styles of coding adopted in VISTA over thirty years and supports data arrangements that are no longer used in the system.

The Master VistA Data Model (MVDM) is a hand curated, hand coded reframing of the native VISTA Data Model (VDM). The other input to MVDM select procedural logic of VISTA, mostly logic that implements remote procedure calls (RPCs) over the VDM.

MVDM is a machine-processable model that reframes VDM and reimplements CPRS RPC behaviors to capture how VISTA currently behaves in a consistent form

From VDM to MVDM

1. Establishes common patterns for the same operations and data forms. For example, data is removed and marked as removed in one form in MVDM unlike the variety of forms in VDM. See the "removal" operation in the Problem and Allergy Model.
2. Combines disparate data into single classes where appropriate. For example, in Allergy, the VDM has two objects (Allergy and Allergy Reaction) for noting an Allergy. MVDM combines these two objects into one.
3. Doesn't expose deprecated properties or patterns. For example, MVDM Allergy suppresses the notion of "chart marked". With the advent of CPRS, VA no longer distinguishes an electronic and paper chart.
4. Synthesizes properties to expose implicit values only denoted in comments. For example, observation units in Vitals.
5. Groups related properties into sub objects, making classes more manageable. For example, the VDM Patient class has over 450 properties, largely laid out in one flat list. It makes sense to gather related properties such as an address line and zip code into clearly structured, contained classes.
6. Normalizes properties so that a common property is identified and labeled the same way across relevant objects. For example, in MVDM, a property that references the Patient class is always called "patient". VDM can use "patient_name" or "name". This reflects the desire for global property name scope in MVDM while VDM has, at best, per class name scope for its properties.

In addition, where the VDM of one domain seems to lack a desired property, it is noted in the MVDM. This should promote improvements in the VDM and hence in the MVDM.

For MVDM, RPC logic captures permissible combinations of properties and side effects. For example, in allergies, how an historical allergy differs from an observed allergy. These *behaviors* are embedded in MVDM to ensure that old valid combinations of properties and objects may be created, updated, and deleted.

VDP
MVDM Normalization of VDM

Background: MVDM's normalization of VDM is one of the key reports for project progress measurement and scoping.

The following shows the reduction of redundancy between MVDM and VDM.

MVDM Class	Count	VDM Class	Count
Allergy	19	Patient Allergies; Adverse Reaction Reporting	129
Problem	37	Problem-9000011	49
Vital	10	Gmrv_Vital_Measurement-120_5	13

VDP
Live Vista Data Model Maintenance Report

Background

The native Data Model of VISTA (VDM) is a class-centric restatement of VISTA's FileMan Data Dictionary (DD). As such, it is generated from the DD and must change as the DD changes. Keeping VDM up to date means "keeping up with DD changes".

In addition, as the MVDM builds over the VDM, changes to the DD may effect it too.

Keeping VDM up to date

Currently when the DD changes, the statically generated VDM must be regenerated. In Year 2 of VDP, we'll move to a *Dynamic VDM* where VDM is rendered on-demand from the DD.

But VDM test code - a large portion of the VISTA Data Project's code is test code - is static and may make assumptions that no longer hold when the DD and then VDM changes. By running test code nightly, we will catch discrepancies nearly as soon as they happen. As many such discrepancies would effect MVDM, the brittleness of the test code is one key utility for keeping MVDM up to date.

Keeping MVDM up to date

By its nature the Master VISTA Data Model (MVDM) is manually designed and coded. Two sorts of changes to the DD and then VDM can effect the MVDM:

- additions of new properties or aspects of properties ("is it required", "range of values") that need to be explicitly exposed or addressed in MVDM
- new triggers (cross references, computations) in VDM which means changes to the VDM have unaccounted for side effects

As noted above, the static VDM test code will expose some of these discrepancies. Further discrepancies will be exposed in the MVDM Test code.

In addition, a "VDM Change Report" will be created that will isolate how the VDM has changed. This report will not only aid in interpreting test code failures but will also guide changes not addressed in test code.

VDP
Security Management Summary Report

Broadly, Security has three aspects:
1. **authentication** (is this Dr Kildare?)
2. **access control** (should Dr Kildare be able to do this?) and
3. **auditing** (what did Dr Kildare do?).

PIKS (Patient-Institution-Knowledge-System) is a model for decomposing the data in a health-care system with a focus on isolating Patient data from other data. **PIKS-based Security** adds another dimension to access control. In addition to the operation being performed (create-read-update-delete) and the object of the operation (the type of data in scope), a PIKS categorization of data adds the organizing principle of health-care, the patient.

Current RPC Interface to VISTA

In the current RPC interface to VISTA, a user [1] is authenticated using 2 RPCs with the 2 password access/verify mechanism - effectively authentication looks up a user record. Then [2] access control is applied ("action-based") on each subsequent RPC call - the user's record references a list of permitted RPCs. Finally [3] FileMan can be setup to record an audit trail of the data changed by RPC calls. The "outside world" accesses this setup through the RPC Broker, a TCP server with a proprietary protocol.

This current RPC setup has the following access control and auditing issues:

1. RPC-centric access control is largely "action based": it isn't per patient. For example, it controls whether you can "write a prescription", not whether you can "write a prescription for John".
2. There is a concept of patient - users can be assigned a list of patients. But enforcement relies on client- enforcement. In other words, the client asks the VISTA server, "can my user access patient Fred's
record?". If the answer is no, then the client won't touch Fred but the server doesn't enforce this restriction. In effect, patient access control is purely advisory in the current VISTA scheme.
3. FileMan auditing only works if RPCs call the FileMan APIs (there are two) and don't directly read and write the globals that store FileMan data. **As many bypass the API, there is a large auditing hole**.
Changes happen but FileMan auditing doesn't know.
4. Auditing, where it exists, is on FileMan data but access control is on RPCs ("actions"). In other words, the "paradigms" are inconsistent.
5. While in theory, with data mining, you could partition FileMan audit trails per patient, the system doesn't do it.

MVDM [1] fits under whatever authentication mechanism VA wants to use going forward, [2] applies patient- centric, data-centric access control ("should Dr Kildare be able to write a Prescription for John") and [3] through its event mechanism provides comprehensive audit trails for all activity.

PIKS delivers the patient-centric part of access control. The dimensions of operation, time or place are not PIKS - "can John write a Prescription at Facility A at midnight?" PIKS adds the patient - "can John write a Prescription at Facility A at midnight for Fred?".

MVDM not only notes the user, the place (facility), the time and operation involved in a procedure - it also knows and notes the patient. The MVDM Management Client displays MVDM audit events which reflect the nuance exposed ...

| Dec 21st 2016 @ 7:27:01 pm | Vital | LIST | 011a073e-8206-41e8-b0ba-adfd78b7c434 | ALEXANDER,ROBERT (200-61) | CARTER,DAVID (2-25) | VISTA HEALTH CARE (4-2957 / 6100) |
| Dec 21st 2016 @ 7:27:01 pm | Allergy | LIST | d7a94e6e-81fa-4acf-af8b-45000e353def | ALEXANDER,ROBERT (200-61) | CARTER,DAVID (2-25) | VISTA HEALTH CARE (4-2957 / 6100) |

PIKS involves making *patient* a first class concept along side *user*, *place* (facility), *time* and *operation*. It makes security health-care specific.

With the added dimension of patient, all operations from listing to describing to changing, can be restricted for a user to a particular group of patients.

VDP End to End Demonstration
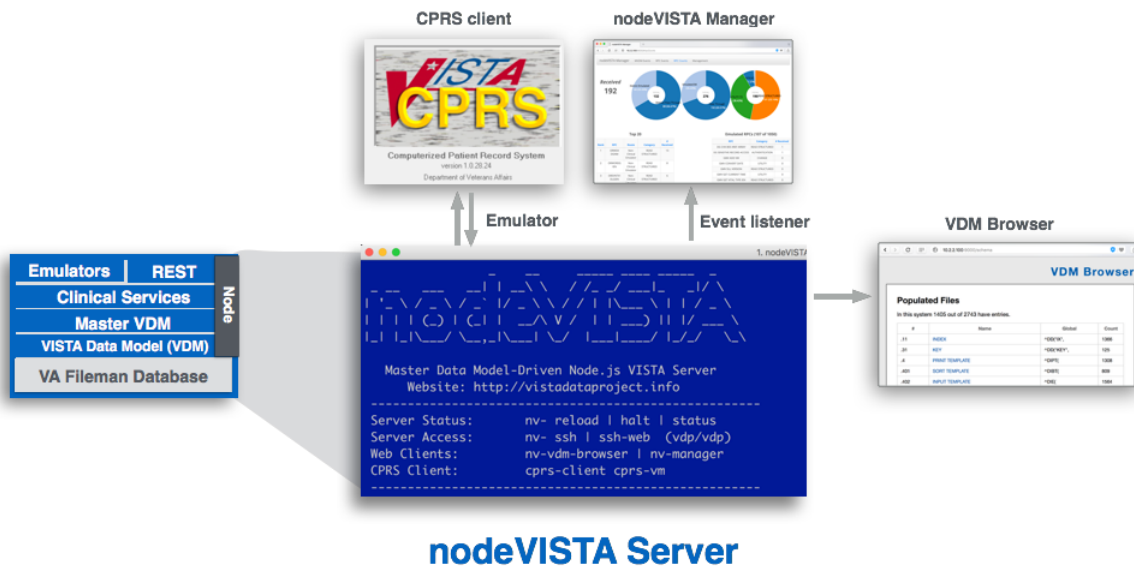
Start CPRS…
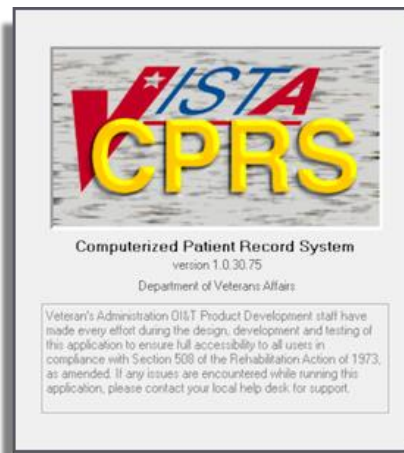
Before logging in, notice that the **nodeVISTA Manager RPC Events** tab shows the first RPC traffic coming from CPRS.

Total: 3	Total No Polling: 3	Pass Through: 1	RPC Emulated: 1	Server: 1	**RPC Emulate:** ✅

| Date	RPC Name	Route	Transaction Id
Apr 11th 2017 @ 12:21:37 am	**XUS INTRO MSG**	**Non-Clinical Emulator**	**8822957a-7082-4ea9-b83f-bf34d12f7921**
Apr 11th 2017 @ 12:21:37 am	XUS SIGNON SETUP	Pass Through	ab08fb31-0b22-4b9f-b606-0da7e1042383
Apr 11th 2017 @ 12:21:37 am	TCPConnect	Server	9814f4c6-1351-4ae4-b2f8-0b752f3fd93d

《 〈 1 〉 》

Login into VISTA as **ALEXANDER,ROBERT** using the following credentials:

Access Code: fakedoc1
Verify Code: 1doc!@#$
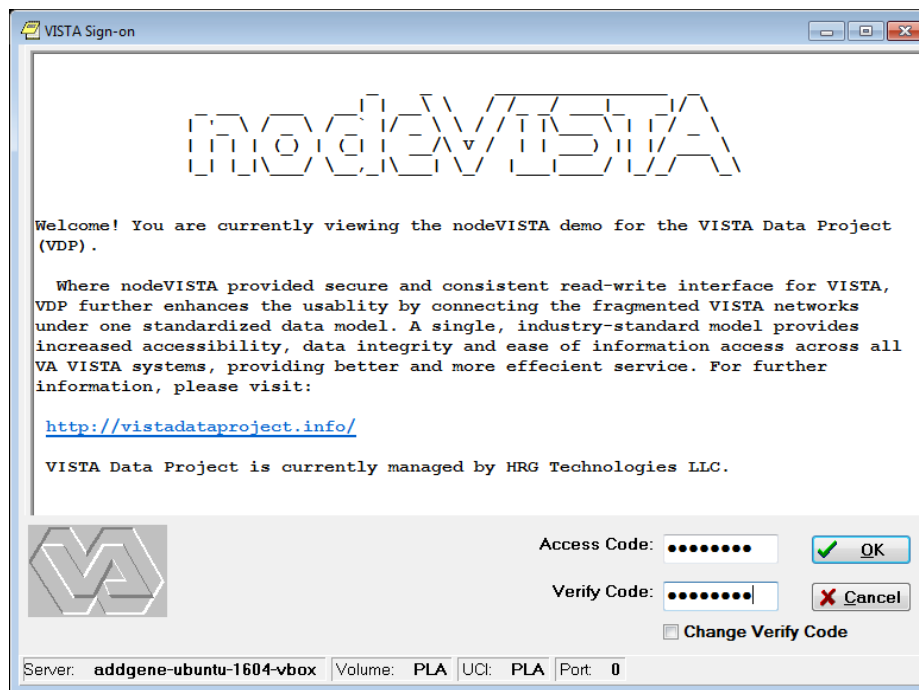
After login, you will be brought to the patient selection dialog…
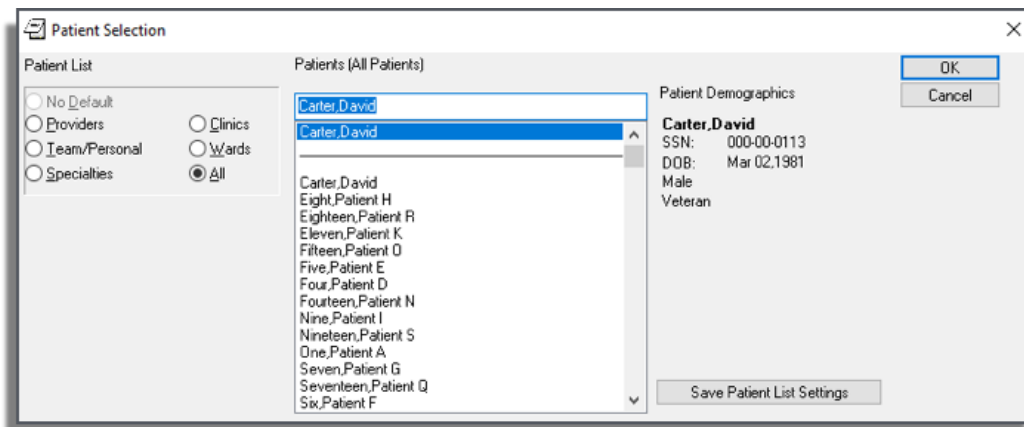
Before choosing patient **Carter,David**, look in the **nodeVISTA Manager** and see just how *chatty* CPRS is. Before any patient is selected, over 80 RPCs have been sent:

Total: 81   Total No Polling: 81   Pass Through: 20   RPC Emulated: 60   Server: 1   **RPC Emulate:** ✔

Date	RPC Name	Route	Transaction Id	IP Ad
Apr 11th 2017 @ 12:24:54 am	**ORWORB SETSORT**	Non-Clinical Emulator	1eab189b-6d4a-4243-8b5f-7b060e...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	**ORWORB GETSORT**	Non-Clinical Emulator	579ae26b-5d05-43c5-9fcb-53e061...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	ORWORB FASTUSER	Pass Through	97a02469-93b1-4fa0-b93e-086032d...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	**ORWPT LIST ALL**	Clinical Emulator	26206f27-879e-4e48-8339-603d9c6...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	ORWPT DFLTSRC	Pass Through	c9682c78-18a0-469a-af09-e73b2ba...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	ORQPT DEFAULT PATIENT LIST	Pass Through	fad8ef30-d9ab-4cfa-b729-f978d595...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	ORQPT DEFAULT LIST SORT	Pass Through	b8f84fa0-ee8a-47bb-ac28-95ee471...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	ORQPT DEFAULT LIST SOURCE	Non-Clinical Emulator	3ae4d277-0d13-4a1b-a5ae-67b869...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	**ORWCH LOADSIZ**	Non-Clinical Emulator	cef3853b-73d2-484c-80fb-ecd3ec3...	::ffff:1
Apr 11th 2017 @ 12:24:54 am	**ORWDBA3 HINTS**	Non-Clinical Emulator	48704adb-6b85-4273-b1eb-8ccd67...	::ffff:1

Back at CPRS, select 'OK' and you will be brought to the (empty) patient chart

and the **nodeVISTA Manager** shows the extra RPCs invoked - the total is now at 133…

Total: 133 Total No Polling: 117 Pass Through: 52 RPC Emulated: 80 Server: 1 **RPC Emulate:** ✓			
Date	**RPC Name**	**Route**	**Transaction Id**
Apr 11th 2017 @ 12:24:47 am	ORWGRPC TYPES	Pass Through	914613bd-6821-44e3-bd06-66b90f...
Apr 11th 2017 @ 12:24:47 am	**ORWGRPC GETPREF**	**Non-Clinical Emulator**	**d0bd5a0c-cb72-4c15-ba34-b4c4a7...**
Apr 11th 2017 @ 12:24:47 am	ORWU HASKEY	Pass Through	7d22e544-bcbb-401f-a80c-887b92d...
Apr 11th 2017 @ 12:24:47 am	ORWU HASKEY	Pass Through	ca32f9a1-0916-484a-900c-83c79ce...
Apr 11th 2017 @ 12:24:47 am	**ORWGRPC GETPREF**	**Non-Clinical Emulator**	**31466e43-105a-4fb5-b20f-fb77c02...**
Apr 11th 2017 @ 12:24:47 am	**ORWU DT**	**JS Utility Emulator**	**7d9ccbaa-ab5c-4626-b23e-9f44ec...**
Apr 11th 2017 @ 12:24:47 am	**ORWSR SHOW SURG TAB**	**Non-Clinical Emulator**	**191aadff-d058-49f3-af0a-56ec6fa...**
Apr 11th 2017 @ 12:24:46 am	**ORWCH LOADALL**	**Non-Clinical Emulator**	**fff2b2b4-4800-47ef-9fb8-f781ad61...**
Apr 11th 2017 @ 12:24:46 am	**ORWORDG IEN**	**Non-Clinical Emulator**	**9df6eac7-e908-4302-bc35-7b523b...**
Apr 11th 2017 @ 12:24:46 am	**ORWORDG IEN**	**Non-Clinical Emulator**	**43af5118-c545-4706-84e6-fcc9b41...**
Apr 11th 2017 @ 12:24:46 am	**ORWORDG IEN**	**Non-Clinical Emulator**	**a51e9e8d-3ca6-4fca-af8b-afb0453...**
Apr 11th 2017 @ 12:24:46 am	**ORWORDG IEN**	**Non-Clinical Emulator**	**bd860e4d-3a56-4241-ab37-caa24c...**

and the **MVDM Events** tab shows MVDM model events for emulated RPCs…

Total: 13 Describe: 9 List: 4 Create: 0 Update: 0 Remove: 0 Unremoved: 0 Delete: 0 **RPC Emulate:** ✓					
Date	**Domain**	**Type**	**Transaction Id**	**User**	**Patient**
Apr 11th 2017 @ 12:26:29 am	Vital	LIST	058883b5-f81c-403b-a9d8-1f803ae78...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:29 am	Allergy	LIST	0c21cdea-f56c-4980-8ca8-b650fcb82d...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:28 am	Patient	DESCRIBE	c0c8e3da-e7dd-475f-bc74-35724f871...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:28 am	Patient	DESCRIBE	d20d22cd-1094-47ef-a4b4-38ce1f3b6...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:28 am	Patient	DESCRIBE	c0260bda-df84-48a8-93a4-a0ceb3452...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:28 am	Allergy	LIST	dc6d2d9f-147b-4218-90d2-8b7afa944...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:28 am	Patient	DESCRIBE	dc6d2d9f-147b-4218-90d2-8b7afa944...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:28 am	Patient	DESCRIBE	5ef8c247-312a-4bed-a21e-03090f257...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:28 am	Patient	DESCRIBE	4736480d-80c1-49cd-9d44-869be555...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:28 am	Patient	DESCRIBE	5d52abbd-7946-44a1-8e92-db8e3332...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:27 am	Patient	DESCRIBE	b08515b2-d7a4-43e1-9f56-3cb79a9cd...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:26:27 am	Patient	DESCRIBE	ca5be4f5-2e55-4847-b0ab-f115f9ba1...	ALEXANDER,ROBERT (200-62)	CARTER,DAVID (2-25)
Apr 11th 2017 @ 12:24:54 am	Patient	LIST	26206f27-879e-4e48-8339-603d9c6bc...	ALEXANDER,ROBERT (200-62)	