

VistA Data Project (VDP) Prototype

Deliverable Summary – End of 2016

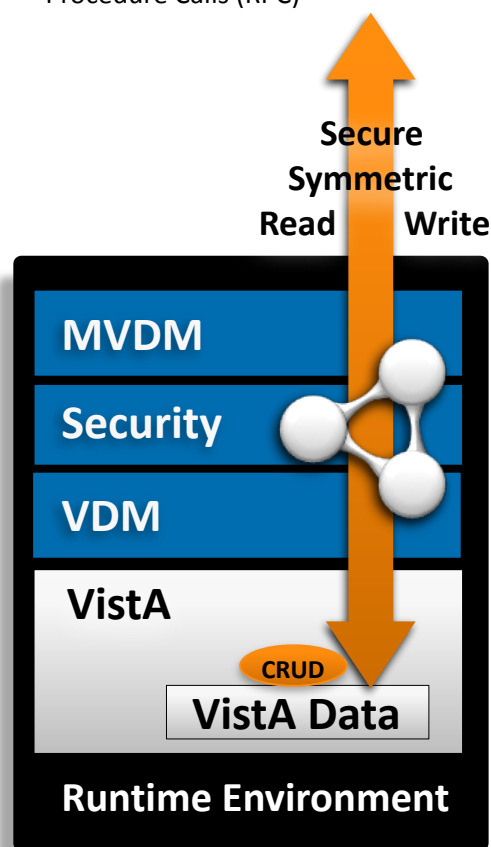
(Does not include post-prototype operating concept)




Purpose:

- Testing feasibility of a Single Secure Symmetric Read-Write model for interfacing to VISTA

What will the prototype test?

- Responsive interfacing to VistA without wading through ~3300 inconsistent, unreliable Remote Procedure Calls (RPC)



-  VistA Data Models (deliverables) (Linked Data / JSON-LD)
-  Existing VistA (MUMPS)
-  Runtime Environment (Javascript / Node.js)



Linked Data Model or metadata:

- Industry-standard, machine-processable, web-centric Linked Data model using JSON-LD serialization
- Enables data from different sources to be connected and queried
- All VDP data models are represented in this form

Local VistA Data Model (VDM):

- Comprehensively exposes full, native operational VistA data model as JSON-LD to enable:
- Quantifiable data access
 - *Read*: 100%; can securely read everything
 - *Write*: % can be quantified; prototype will incrementally expand; includes data update business rules
- Creates symmetric read-write model (read model = write model)
- Abstraction / separation-of-concerns between the implementation complexities of VistA internals and VistA clients
- No persistence, pass-through to VistA internals

Security Model:

- Enrichment layer on the VDM with metadata annotations to support on-the-data, patient-centric attribute-based access control (ABAC) “data centric” security

Master VistA Data Model (MVDM):

- Subset of VDM normalized across VDMs
- Normalization to:
 - Eliminate redundancies (data reduction)
 - Refine representations (data organization)
 - Address distinctions between VistA instances
- Incorporates security model features
- Leverages all features and functionality of the VDM and security layers
- No persistence, pass-through from clients to VDM

API, code, and runtime environment:

- Will use VA approved server-side Node.js / Javascript runtime environment
- Packages are event-driven from the client with asynchronous I/O