

PseudoVet

Developers Guide

Written by Will BC Collins IV

Date: June 26, 2015

VHA Innovations Future Technology Laboratory

Department of Veterans Affairs

Table of Contents

[Introduction 4](#_Toc424146430)

[Background 4](#_Toc424146431)

[Technical Overview 4](#_Toc424146432)

[Purpose of this Manual 5](#_Toc424146433)

[Architectural Overview 5](#_Toc424146434)

[Prerequisites 5](#_Toc424146435)

[Development Environment 5](#_Toc424146436)

[Developer Knowledge 5](#_Toc424146437)

[Development Environment 6](#_Toc424146438)

[System Requirements 6](#_Toc424146439)

[Installing Required Software under CentOS 7 6](#_Toc424146440)

[Install MongoDB Database Server Software 6](#_Toc424146441)

[Install Node.JS and NPM 7](#_Toc424146442)

[Core Reference Database (CRD) 7](#_Toc424146443)

[CRD Development Activities 7](#_Toc424146444)

[Discovery 7](#_Toc424146445)

[Taxonomic Scheme 7](#_Toc424146446)

[Name 8](#_Toc424146447)

[Occupation 10](#_Toc424146448)

[Prepositioning 10](#_Toc424146449)

[Import 11](#_Toc424146450)

[Data Types 11](#_Toc424146451)

[Social Security Number Data 12](#_Toc424146452)

[FACILITY DATA 12](#_Toc424146453)

[Medical Staff DATA 12](#_Toc424146454)

[CLINIC DATA 12](#_Toc424146455)

[PATIENT DEMOGRAPHICS 12](#_Toc424146456)

[SYMPTOMS DATA 12](#_Toc424146457)

[DIAGNOSIS DATA 13](#_Toc424146458)

[CURRENT PROCEDURAL TERMINOLOGY DATA (CPT) 13](#_Toc424146459)

[FORMULARY DATA 13](#_Toc424146460)

[LABORATORY TESTS AND REFERENCE RANGES 13](#_Toc424146461)

[RADIOLOGICAL TESTS AND REFERENCE RANGES 13](#_Toc424146462)

[WHO DISABILITY ASSESSMENT SCHEDULE (WHODAS 2.0) 13](#_Toc424146463)

[SERVICE CONNECTED DISABILITY DATA 13](#_Toc424146464)

[PSEUDOVET INTERFACE (PI) 13](#_Toc424146465)

[AUTHENTICATION AND AUTHORIZATION 13](#_Toc424146466)

[CLINICAL DATA MODIFICATION 14](#_Toc424146467)

[LINK ATTRIBUTION 14](#_Toc424146468)

[GENDER, RACIAL, AND GEOSPACIAL TAGGING 14](#_Toc424146469)

[PROCESS EXECUTION 14](#_Toc424146470)

[LOGGING 14](#_Toc424146471)

[PSEUDOVET DATABASE 14](#_Toc424146472)

[AUTOMATION SERVICES 14](#_Toc424146473)

[ALPHA ENGINE 14](#_Toc424146474)

[FACILITY GENERATOR 14](#_Toc424146475)

[CLINIC GENERATOR 14](#_Toc424146476)

[STAFF GENERATOR 14](#_Toc424146477)

[PSEUDOVET GENERATOR 15](#_Toc424146478)

[ALPHA QUEUE 15](#_Toc424146479)

[OMEGA ENGINE 16](#_Toc424146480)

[PI OMEGA INTERFACE 16](#_Toc424146481)

[SYMPTOMS GENERATOR 16](#_Toc424146482)

[APPOINTMENT GENERATOR 16](#_Toc424146483)

[WORKFLOW GENERATOR 16](#_Toc424146484)

[TROUBLE MAKER 16](#_Toc424146485)

[OMEGA QUEUE 16](#_Toc424146486)

[CLIENT EHR SYSTEM INTEGRATION 16](#_Toc424146487)

[PI EHR INTERFACE 16](#_Toc424146488)

[VISTA ADAPTOR 16](#_Toc424146489)

# Introduction

PseudoVet is an automated patient data fabrication engine.  It’s goal is to provide a set of active synthetic patients that can be used for healthcare software development and testing for applications that are geared towards VA’s VistA and Enterprise Heath Management Platform (eHMP) through the Veterans Health Administrations’ (VHA) Future Technology Laboratory (FTL) a publically accessible development environment. More information on the VHA Innovation Laboratory and FTL can be found here: <http://vaftl.us>

## Background

Development against real patient data unnecessarily exposes patient health information (PHI) and personally identifiable information (PII) and cannot be used by developers outside of the VA network.  Development against current fabricated data is not useful because the data sets are very old which require development teams to spend much time developing data sets to use in lieu of writing code.  Typical fabrication of patient data is typically of little or no medical relevance.  The development of a system that creates and updates synthetic patient data using a set of templates for various diagnosis would provide more relevant patient data for development that could be used both inside and outside of the VA network.  Development outside of the VA network is desirable as it allows more  collaboration with the Open Source community which is in-line with the VA’s Open Source Initiatives.

## Technical Overview

PseudoVet’s fabricated patient records are created by random selection of diagnosis data such as service connected disabilities, symptoms, and thereby provides more clinically relevant fabricated progress notes, laboratory data, as well as surgical procedures, discharge, and other ancillary data.  In addition to common clinical data related to specific diagnosis, PseudoVet also continuously schedules appointments, randomly no-shows patients, generate consults, means tests and other administrative activities that occur in a real patients record.

## Purpose of this Manual

The purpose of this manual is to provide instruction on how PseudoVet is designed and built in order for other developers to extend and improve PseudoVet.

# Architectural Overview

The PseudoVet system is comprised of the following components:

* **Core Reference Database (CRD)** – A MongoDB database containing model and template data used for the automatic generation of synthetic patient data
* **PseudoVet Interface (PI)** - A web based application and services to support the generation of synthetic patient and supporting data
* **PseudoVet Database** – A database where all generated synthetic patient resides
* **Automation Services -** Back-end services that generate and continuously update synthetic patient data
* **Client EHR System Integration –** Integration routines for data synchronization between the PseudoVet system and external Electronic Health Record Systems (EHR’s)

# Prerequisites

### Development Environment

This manual assumes you have a system setup with MongoDB, Node.js, and NPM.

### Developer Knowledge

Developers must have an intimate working knowledge of the following applications/technologies and their dependencies:

* **Node.js** - a modular object oriented event-driven framework;
* **Backbone.js** - an API providing structural features such as models with key-value, views, and even handling over a RESTful JSON interface)
* **Marrionette.js** - a library around backbone.js that simplifies its usage)
* **Underscore.js** - provides a library of 80 functions such as map, select, invoke, and also helpful methods for binding functions, templates, and testing)
* **jQuery** - a JavaScript library that simplifies document traversal, manipulation, event handling, animation, Ajax, and cross-browser capabilities)
* **HTML5** - the latest standard for HTML and CSS (Cascading Style Sheets)
* **Twitter Bootstrap** – HTML, CSS, and JS framework.
* **SPA** – Single Page Applications (a web application where all necessary code – HTML, JavaScript, and CSS is retrieved with a single page load and additional resources are dynamically loaded as a response to user actions)
* **Apache** – HTTP server
* **Vagrant** – build system for creating lightweight portable development environments
* **Chef** – automation system for building, deploying, and managing infrastructure
* **Grunt** – a javascript task runner for task automation
* **Jenkins** – a continuous integration server for software development
* **HighCharts** – javascript api for creating interactive diagrams and charts
* **jasmine** – a behavior-driven development framework for testing javascript code
* **EWD.js** – a Node.js-based Application Framework and Application Server/Container for use with Caché, GlobalsDB, GT.M and MongoDB databases
* **SOLR** – enterprise search platform build on apache lucene
* **Cache’** – Intersystems’ NoSQL database engine
* **JavaScript** – web development language
* **JSON** - (JavaScript Object Notation) is a lightweight data-interchange format.
* **MongoDB** – NoSQL database using JSON/BSON as it’s native data format.
* **Express.JS** – a Node.js extension for use in developing web services
* **SSH** – or ‘Secure Shell’ is a cryptographic network protocol for initiating text-based shell sessions on remote machines in a secure way.

# Development Environment

## System Requirements

To develop PseudoVet, a developer must have a computer having the following features:

* 7.5 MB RAM or greater
* 1 CPU or greater
* 30 GB Hard drive or greater
* Operating System: Microsoft Windows 7 or greater, Apple OS X 10.9 or greater, RHEL 6.x or greater, CentOS 7.x, Ubuntu 14.x

## Installing Required Software under CentOS 7

Create a CentOS 7 64-bit machine with at least 7.5 GB RAM and 30 GB Hard drive

### Install MongoDB Database Server Software

From a console prompt type the following commands:

sudo yum install vim

sudo vim /etc/yum.repos.d/mongodb.repo

Press the i key and paste the following text:

[mongodb]  
name=MongoDB Repository  
baseurl=http://downloads-distro.mongodb.org/repo/redhat/os/x86\_64/  
gpgcheck=0  
enabled=1

Exit and save by typing :wq and pressing the <enter> key.

sudo yum –y update

sudo yum –y install mongodb-org mongodb-org-server

systemctl start mongod

Mongodb listens on 27017 by default

### Install Node.JS and NPM

From a console prompt type the following commands:

sudo yum install nodejs

sudo yum install npm

# Core Reference Database (CRD)

The CRD is concerned with collecting reference data for use in building synthetic patient data. As there is a large body of freely available documents such as comma separated value (CVS) files, Excel spreadsheets, web-services, and other formats of this data freely available, the goal of CRD development is to create tools that will collect this data for use in generating usable synthetic patient data.

## CRD Development Activities

CRD development is split into 3 activities:

* **Discovery** – This activity involves finding types of data that that can be attributed to a patient
* **Schema** – Developing a data organization model for the data
* **Prepositioning** – Identifying or developing methods to reformatting collected discovery data into JSON format using the developed schema
* **Data Import** – Importing the JSON files into the CRD

### Discovery

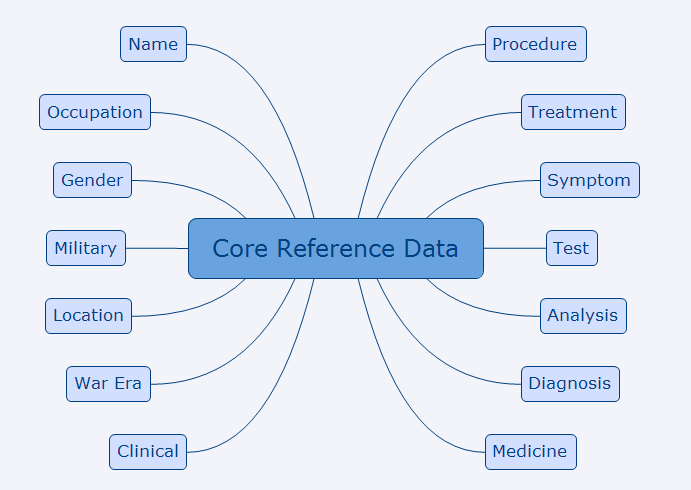
Discovery is simply finding relevant reference data in a format that can be easily obtained such as text files, xml, or comma separated values (CSV). Once a new reference data set is found it needs to be copied to the **reference\_data\_raw** folder.

### Taxonomic Scheme

This is the most difficult task as it requires knowledge of how to make sense of the data and understand how it will be accessed from PseudoVet after it has been imported.

To simplify things we will define how objects are nested in the CRD based upon the types of data that are needed to develop an EMR and all support data to include human (staff and patient) information.

This is the current basic breakdown of data in the CRD:



In order to create a synthetic entity such as a Pseudo Veteran we need Core Reference Data to draw from for any details relating to an entity so we start with these:

### Name

A name is broken down into: Lastname, Firstname, Middle Name, and Suffix

The CRD contains imaginary names in order that they cannot be confused with real people. They are organized within CRD like this:

**{**

**"Name": {**

**"Lastname": [**

**"Blollins",**

**...**

**"Zdohsy"**

**],**

**"Firstname": [**

**"Anblue":**

**...**

**"Zartak"**

**],**

**"Middle": [**

**"Andro",**

**...**

**"Zeeki"**

**],**

**"Suffix": [**

**"I",**

**"II",**

**"III",**

**"IV",**

**"V",**

**"Jr.",**

**"Sr",**

**"Esq"**

**]**

**}**

**}**

#### Lastname

To preposition lastnames from the lastnames.txt file in the reference\_data\_raw folder type the following command from the src\core-reference folder:

X:\PseudoVet\src\core-reference>node crdconverter.js --i reference\_data\_raw\Name-Lastnames.txt --l Name --o reference\_data\Name-Lastnames.json --f Lastname,x,y --d space

The output will be a json formatted file in the reference\_data folder called **Name-Lastnames.json**

#### Firstname

To preposition firstnames, which are also used to generate middle names, there is a file called Name-Firstnames.csv located in the reference\_data\_raw folder. Type the following command from the src\core-reference folder:

Node crdconverter.js –I reference\_data\_raw\Name-Firstnames.csv –l Name –o reference\_data\Name-Firstnames.json –f Name,Gender

### Occupation

Occupations are similarly organized under Occupation but also

Gender

Military

Location

War Era

**Location** – Any location data falls under this category to include, states, street names, types of facilities, etc…

**Drug** – This could be any type of substance that is used to alter an entity

**Clinical** – symptoms, diagnosis, cpt codes, whodas 2.0, service-connected disabilities

Clinical Procedure Terminology (CPT) Codes – These are codes used to describe services related to diagnosis used for scheduling, billing, etc…

#### ICD-9 Procedure Codes

Procedure Code,Long Description, Short Description

node crdconverter.js --i reference\_data\_raw\ICD-9-CMS32\_DESC\_LONG\_SHORT\_SG.csv --o reference\_data\Clinical-ICD9-Procedure.json --d comma --l Clinical

#### ICD-9 Diagnosis Codes

Diagnosis Code,Long Description, Short Description

node crdconverter.js --i reference\_data\_raw\ICD-9-CMS32\_DESC\_LONG\_SHORT\_DX.csv --o reference\_data\Clinical-ICD9-Diagnosis.json --d comma --l Clinical

### Prepositioning

In the src\core-reference folder there are a number of utilities that are useful in prepositioning data prior to import into the CRD.

#### crdrelimiter.js

*crdrelimiter.js* – changes a defined number of delimiters. This is useful if there are delimiters such as spaces or commas in a file and to use that as the delimiter breaks up data incorrectly. An example of a line in a file that would be useful to use this utility against is as follows:

engine v-6 a v-6 engine is powerful but, not as powerful as a v-8

Using a space as the delimiter above would result in a file that has a description broken across several array elements. From line to line this could vary greatly so it might make sense to have the first two spaces converted into a delimiter that is not a comma since there is also a comma in the line. A carrot ‘^’ or tilde ‘~’ may be a good choice.

#### crdconverter.js

*crdconverter.js*  - This utility can be used to take a list from a text file and turn it into a JSON file that can be imported into the CRD.

node crdconverter.js --d space --i

reference\_data\_raw\lastnames.txt --o lastnames.json --f lastname,x,y,z --d space --l Name

### Import

In the src\core-reference folder there is a utility called *crdimport.js.*  This utility can be used to take a prepositioned JSON file and import the data into the CRD.

It is simply a convenience utility that ultimately performs the following operations:

jsonlint name-of-reference-data.json

mongoimport --db crd --collection diagnosis --drop --file primer-dataset.json

## Data Types

There are many types of data that are needed for the development of a patient record. To date, the following types of data have been identified as being necessary to have in order to develop data for PseudoVet:

* States
* Territories
* Islands
* Cities
* Streets
* Zip/Postal Codes
* Facility Types
* Clinic Types
* Medical Staff Types
* Sexes
* Races
* War Eras
* Branches of Armed Services
* Occupations and Incomes
* Last names
* First names
* Symptoms
* Diagnosis from ICD-9 DSM-5
* Current Procedural Terminology Data (CPT) codes
* Formulary data such as VA Class, Restriction, Generic Medication Names, Dosage Form
* Laboratory Tests and Reference Ranges
* Radiological test types and Reference Ranges
* World Health Organization’s (WHO) Disability Assessment Schedule (WHODAS 2.0)
* Service-Connected Disability Types
* Social Security Numbers

## Social Security Number Data

In order to avoid using real Social Security Numbers (SSN’s), only invalid ranges of SSN’s will be used in the PseudoVet project. The first three digits (once used to denote the ‘area’ of assignment) of the SSN will be 999 which is invalid. The second 2 digits which make up the ‘group’ portion of the Identification Number (ID) will be 00 which is invalid. This leaves 0000-9999 as the numbers assigned to synthetic patients. If all 9999 are generated the group number can be incremented to 01 and so on. Because the SSN will be completely fabricated, it will not be stored as reference data in the CRD.

### FACILITY DATA

A collection of geographical locations such as states, outlying territories, islands, cities, and streets are used to generate Medical Center, Outpatient Clinic, and Community-Based Outpatient Clinic (CBOC) facility data.

### Medical Staff DATA

A collection of clinical disciplines and common first and last names used for the automatic generation of a panel of providers for each facility. Clinical disciplines include doctors, nurses, social workers, laboratory, pharmacy as well as administrative staff.

### CLINIC DATA

A collection of clinic types are used for the automatic generation of clinics within facilities. Providers and other staff are assigned randomly to clinics based upon facility and clinic type.

Investigate the types of clinics present at different facility types in order to populate realistic clinic types, counts, and services that could be available.

As an example, Dental services are not usually present in a Community-Based Outpatient Clinic (CBOC), nor are Emergency, Radiology, or Nuclear Medicine services.

### PATIENT DEMOGRAPHICS

A collection of sex, race, war eras, branches of United States armed services, states, outlying territories, islands, cities, streets, occupations, incomes, and names used for the automatic generation of a pool of synthetic patient demographic data to be used in the creation of PseudoVet patient records.

### SYMPTOMS DATA

A collection of symptoms that contain complementary measurable criteria relating to reference ranges for blood, stool, behavioral, physiological, pathological attributes that are measurable.

### DIAGNOSIS DATA

Clinical diagnosis codes to include both International Statistical Classification of Diseases and Related Health Problems (ICD-9) and the Diagnosis and Statistical Manual of Mental Disorders, Fifth Edition (DSM-5).

### CURRENT PROCEDURAL TERMINOLOGY DATA (CPT)

A collection of Current Procedural Terminology Data (CPT) codes with references that link the data with diagnosis and symptoms data.

### FORMULARY DATA

A collection of current Medication Formulary Data to include VA Class, Restriction, Generic Medication Names, Dosage Form {i.e., Liquid, Oral, Tab, Injection, Suppository, etc.}, Comments, Clinical Guidance and provide references that link the data with diagnosis and symptoms data.

### LABORATORY TESTS AND REFERENCE RANGES

A collection of laboratory tests and reference ranges for laboratory test results indicating low, normal, and high ranges.

Include mapping between test results and symptoms attributed to reference ranges within the CRD.

### RADIOLOGICAL TESTS AND REFERENCE RANGES

A collection of radiological tests using a variety of imaging techniques such as X-ray radiography ultrasound, computed tomography (CT), nuclear medicine including positron emission tomography (PET), and magnetic resonance imaging (MRI).

Include reference ranges for radiological examination results indicating resultant ranges and mapping to symptoms and diagnosis within the CRD.

### WHO DISABILITY ASSESSMENT SCHEDULE (WHODAS 2.0)

A collection of assessment measures and ranges of values with references to behavioral diagnosis from the World Health Organization’s (WHO) Disability Assessment Schedule (WHODAS 2.0).

### SERVICE CONNECTED DISABILITY DATA

A collection of Service Connected Disabilities with referenced diagnosis data.

### PSEUDOVET INTERFACE (PI)

A web-based Single Page Application (SPA) as a clinical and administrative data interface that both clinical subject matter experts and administrators use to view and manipulate core reference data.

### AUTHENTICATION AND AUTHORIZATION

A role-based authentication and authorization system for the PI system. Functionality must be developed that permit the assignment of roles to users for administration of the system and/or clinical data modification.

### CLINICAL DATA MODIFICATION

Develop functionality in the PI to add, edit, view, and delete data in the CRD for symptom, diagnosis, laboratory, radiological, CPT, formulary, whodas 2.0, demographics, clinic, provider, and common last/first name data.

### LINK ATTRIBUTION

A mechanism to link and modify links between laboratory, CPT, formulary, radiological, whodas 2.0, data to symptoms, symptoms to diagnosis, and diagnosis to service connected disability data.

### GENDER, RACIAL, AND GEOSPACIAL TAGGING

Gender, racial, and geospacial tagging to diagnosis and symptoms wherever diagnosis are only relevant to a specific gender, race, or geospacial attribute.

### PROCESS EXECUTION

A mechanism to spawn routines to generate facilities, clinics, staff, and patients at will.

### LOGGING

A mechanism for logging and displaying the output of activities that occur within the PI as well as from the Automation Engine.

### PSEUDOVET DATABASE

The PseudoVet database is developed using MongoDB to contain all synthetic data that is automatically generated by the Automation Services. The dataset includes facility, clinic, staff, patient, and all clinical data.

# AUTOMATION SERVICES

The automation services generate, process and queue data relating to PseudoVet facility, staff, patient, and clinical data workflow processes.

## ALPHA ENGINE

A mechanism to seed the PseudoVet Database with synthetic data using the CRD as the source for possible values.

### FACILITY GENERATOR

A mechanism to generate a configurable number of healthcare facilities. The facilities yielded from executing the facility generation utility must be attributed to random cities from the CRD Facility Data.

### CLINIC GENERATOR

A mechanism to generate clinics within every generated facility based upon facility type. Clinic names will be assigned by specialty and primary care provider names.

### STAFF GENERATOR

A mechanism to generate staff and assign them to clinics based on clinic type.

### PSEUDOVET GENERATOR

A means to programmatically generate a synthetic patient (PseudoVet) utilizing data populated within the CRD having the following key identifiers;

* First Name
* Last Name
* Middle Initial (optional)
* Suffix (optional)
* Date of Birth
* Birth Place
* Sex
* Race
* Social Security Number
* Address Line 1
* Address Line 2
* City
* State
* Zip Code
* Email Address

Generate random next of kin, spouses, and all ancillary data that represents the entries required to complete a VA Form 10-10EZ for registration in a Veterans Health Information System Architecture (VistA) system.

Utilize the CRD to randomly generate occupations, incomes, military service duty dates, service-connected disabilities, and other initial diagnosis and symptom data. Military service must be divided between all of the United States Military branches {Air Force, Air Force Reserve, Air National Guard, Army, Army Reserve, Army National Guard, Coast Guard, Coast Guard Reserve, Marine Corps, Marine Corps Reserve, Navy, and Navy Reserve}.

Assign each PseudoVet to a facility primary care team and provider.

### ALPHA QUEUE

Develop a mechanism that queues and triggers the delivery of generated data from the Alpha Engine and generators to client EHR’s as described in Section 5.4.4.

## OMEGA ENGINE

### PI OMEGA INTERFACE

Develop a sub-interface within PI that permits modification to all functionality within the Omega Engine.

### SYMPTOMS GENERATOR

Develop random generating algorithms that assign realistic laboratory, radiological, and WHODAS 2.0 values, physiological/pathological attributes, and generate synthetic result entries such as progress notes.

### APPOINTMENT GENERATOR

Develop a mechanism to make appointments, re-schedule appointments, cancel appointments, no-show patients from making appointments, and automate a random occurrence of these activities on patients that have been created.

### WORKFLOW GENERATOR

Develop programmatic workflows such as blood draw, X-ray, appointment to review lab results, prescription of medication, surgeries, common outcomes, prescription pick-up, and automate a random occurrence of these activities on patients that have been created based upon symptoms and diagnosis that have been generated for each patient.

### TROUBLE MAKER

Develop a process that randomly assigns pathological and traumatic conditions to random percentages of patients within random geospacial areas in the PseudoVet database. The assignment of a pathological or traumatic condition will result in a system activity such as an emergency room visit for a traumatic or behavioral condition, or lab results that indicate a random pathological condition.

### OMEGA QUEUE

Develop a mechanism that queues and triggers the delivery of generated data from the Omega Engine, generators, and trouble maker to client EHR’s as described in Section 5.4.4.

## CLIENT EHR SYSTEM INTEGRATION

### PI EHR INTERFACE

Develop a sub-interface within PI that enables PseudoVet administrators to add/edit/modify/delete Electronic Health Record Systems (EHRs) client system entries. Client systems are EHRs that receive synthetic patient data from PseudoVet. The default EHR type must be the VistA Adaptor.

### VISTA ADAPTOR

Develop a mechanism that writes all medical data into Veterans Health Information Systems and Technology Architecture (VistA) systems. The VistA Adaptor must provide entries fields to include; access/verify codes, system names, IP addresses, and port to communicate with.

The VistA Adaptor must be a type of EHR Adaptor that can be selected when adding Client EHRs to the PseudoVet system.

Develop the VistA Adaptor to communicate to VistA systems via SSH using Expect library to traverse the menu options within VistA, EWD.js, VistA.js, RPC broker calls, or a combination of these methods of communication.