VistA Services Assembler Phase 2 (VSA-P2)

VistA.js Project 3.0.10

Installation, Back-Out, and Rollback Guide

**

October 2016

Version 2.3

Department of Veterans Affairs (VA)

Office of Information and Technology (OI&T)

Product Development (PD)

Revision History

 **NOTE**: The revision history cycle begins once changes or enhancements are requested after the Installation, Back-out, and Rollback Guide has been base-lined.

Table : Revision History

| Date | Version | Description | Author |
| --- | --- | --- | --- |
| 10/17/2016 | 2.3 | Various updates based on production release 3.0.10 (with vjs-environment 3.0.5), including starting VSA Servers in “clustered mode.” | Apex Data Solutions Technical Writer |
| 08/22/2016 | 2.2 | Revised based on VA feedback. Reflects installation steps for VSA VistA.js comprised of vjs-environment 3.0.5 and Maestro 3.0.8, 3.0.8.1, and 3.0.9. Also updated sample configuration files. | Apex Data Solutions Technical Writer |
| 07/29/2016 | 2.1 | Updated release 3.0.5 to 3.0.8. | Apex Data Solutions Technical Writer |
| 06/08/2016 | 2.0 | Various updates based on release 3.0.5. | Apex Data Solutions Technical Writer |
| 05/02/2016 | 1.9 | Updated to reflect shared storage settings in the “vsa-gen config” utility. | Apex Data Solutions Technical Writer |
| 04/19/2016 | 1.8 | Updated Web server installation steps to align with Federator installation steps. | Apex Data Solutions Technical Writer |
| 04/14/2016 | 1.7 | Added dos2unix command for VistA, Federator, and Web servers to convert certificate files from DOS to UNIX file format. | Apex Data Solutions Technical Writer |
| 04/11/2016 | 1.6 | Updates based on testing Federator installation. | Apex Data Solutions Technical Writer |
| 04/10/2016 | 1.5 | Final review pass prior to Federator installations in IOC on 4/11/2016. | Apex Data Solutions Technical Writer |
| 04/09/2016 | 1.4 | Incorporated VA feedback regarding directory structure and related naming conventions. | Apex Data Solutions Technical Writer |
| 04/07/2016 | 1.3 | Incorporated VA feedback regarding directory structure and related naming conventions. | Apex Data Solutions Technical Writer |
| 04/05/2016 | 1.2 | Revised based on interactive install in the IOC environment, and accompanying VA feedback. | Apex Data Solutions Technical Writer |
| 03/28/2016 | 1.1 | Revised based on VA feedback. | Apex Data Solutions Technical Writer |
| 03/15/2016 | 1.0 | Initial Installation, Back-Out, and Rollback Guide for VSA VistA.js Pilot release.  Based on Property Template Version 2.1, released on February 2016. | VSA VistA.js VA Project Team |

Table of Contents

[1 Introduction 1](#_Toc464484282)

[1.1 Disclaimers 1](#_Toc464484283)

[1.1.1. Documentation Disclaimers 1](#_Toc464484284)

[2 Pre-Installation and System Requirements 1](#_Toc464484285)

[2.1 Platform Installation and Preparation 1](#_Toc464484286)

[2.1.1 General 2](#_Toc464484287)

[2.1.2 Pre-Installation Checklist 2](#_Toc464484288)

[2.1.2.1 Maestro Linux Server 2](#_Toc464484289)

[2.1.2.2 VistA Linux Server 2](#_Toc464484290)

[2.1.2.3 Federator Linux Server 3](#_Toc464484291)

[2.1.2.4 Web Linux Server 4](#_Toc464484292)

[2.1.3 Installation Users 5](#_Toc464484293)

[2.2 Download and Extract Files 6](#_Toc464484294)

[2.2.1 Release Files 6](#_Toc464484295)

[2.2.2 On the Target Maestro Linux Server 6](#_Toc464484296)

[2.2.3 On the Target VistA, Federator, and Web Linux Servers 7](#_Toc464484297)

[2.3 Database Creation 8](#_Toc464484298)

[2.4 Installation Scripts 8](#_Toc464484299)

[2.5 Cron Scripts 9](#_Toc464484300)

[2.6 Access Requirements and Skills Needed for the Installation 9](#_Toc464484301)

[3 Installation Procedure 10](#_Toc464484302)

[3.1 On the Target Maestro Linux Server 10](#_Toc464484303)

[3.2 On the Target VistA Linux Server 12](#_Toc464484304)

[3.3 On the Target Federator Linux Server 20](#_Toc464484305)

[3.4 On the Target Web Linux Server 28](#_Toc464484306)

[4 Implementation Procedure 37](#_Toc464484307)

[4.1 System Configuration 37](#_Toc464484308)

[4.2 Database Tuning 37](#_Toc464484309)

[5 Back-Out Procedure 38](#_Toc464484310)

[5.1 Back-out Strategy 38](#_Toc464484311)

[5.2 Back-out Considerations 38](#_Toc464484312)

[5.2.1 Load Testing 39](#_Toc464484313)

[5.2.2 User Acceptance Testing 39](#_Toc464484314)

[5.3 Back-out Criteria 39](#_Toc464484315)

[5.4 Back-out Risks 39](#_Toc464484316)

[5.5 Authority for Back-out 39](#_Toc464484317)

[5.6 Back-out Procedure 39](#_Toc464484318)

[5.6.1 Stopping the VSA VistA.js Servers 39](#_Toc464484319)

[5.6.1.1 Maestro Server 39](#_Toc464484320)

[5.6.1.2 Web VSA Server 40](#_Toc464484321)

[5.6.1.3 VistA VSA Server 40](#_Toc464484322)

[5.6.1.4 Federator VSA Server 40](#_Toc464484323)

[5.6.2 Removing all VSA VistA.js Platform Components 41](#_Toc464484324)

[5.6.2.1 Maestro Server 41](#_Toc464484325)

[5.6.2.2 VistA, Federator, and Web VSA Servers 41](#_Toc464484326)

[6 Rollback Procedure 41](#_Toc464484327)

[6.1 Rollback Considerations 42](#_Toc464484328)

[6.2 Rollback Criteria 42](#_Toc464484329)

[6.3 Rollback Risks 42](#_Toc464484330)

[6.4 Authority for Rollback 42](#_Toc464484331)

[6.5 Rollback Procedure 42](#_Toc464484332)

[7 Approval Signatures 43](#_Toc464484333)

[8 Appendix A – Sample Server Configuration Files 44](#_Toc464484334)

[8.1 VistA VSA Server 44](#_Toc464484335)

[8.1.1 baypines-config.js 44](#_Toc464484336)

[8.2 Federator VSA Server 45](#_Toc464484337)

[8.2.1 Federator Config File: federator-config.js 45](#_Toc464484338)

[8.2.2 VistA VSA Server Router File: vjs-vista-router.js 46](#_Toc464484339)

[8.3 Web VSA Server 49](#_Toc464484340)

[8.3.1 Web VSA Server Config File: web-config.js 49](#_Toc464484341)

[8.3.2 API Browser Config File: vsa-api-browser-application.js 51](#_Toc464484342)

[8.3.3 RPC Wizard Config File: vsa-rpc-wizard-application.js 51](#_Toc464484343)

[9 Appendix B – Importing the MUMPS Components 53](#_Toc464484344)

[9.1 XSA KIDS Patch Components 53](#_Toc464484345)

[9.2 Software and Documentation Retrieval Instructions 54](#_Toc464484346)

[9.2.1 Pre-Installation Instructions 54](#_Toc464484347)

[9.2.2 Installation Instructions 55](#_Toc464484348)

[9.2.3 Post-Installation Instructions 55](#_Toc464484349)

[9.3 XSA Routines 56](#_Toc464484350)

[9.3.1 Importing the XSAPXROUTINES.xml File 56](#_Toc464484351)

[10 Appendix C – Installation Scripts 61](#_Toc464484352)

[10.1 Maestro 61](#_Toc464484353)

[10.2 vjs-environment 62](#_Toc464484354)

List of Tables

[Table 1: Revision History ii](#_Toc464484270)

[Table 2: VSA VistA.js Storage Requirements 1](#_Toc464484271)

[Table 3: Default Caché Installation Users 5](#_Toc464484272)

[Table 4: Default Installation Directories 5](#_Toc464484273)

[Table 6: VistA VSA Server Configuration 14](#_Toc464484274)

[Table 7: Federator VSA Server Configuration 23](#_Toc464484275)

[Table 8: Web VSA Server Configuration 31](#_Toc464484276)

[Table 9: Log File Locations 37](#_Toc464484277)

[Table 10: Patch Components 53](#_Toc464484278)

[Table 11: Download Servers 54](#_Toc464484279)

[Table 12: Maestro Scripts 61](#_Toc464484280)

[Table 13: vjs-environment Scripts 62](#_Toc464484281)

List of Figures

Figure 1: Maestro Server Landing Page 11

Figure 2: VSA Server Landing Page 19

Figure 3: VSA Server Landing Page 27

Figure 4: VSA Server Landing Page 37

Figure 5: Sample VistA VSA Server Configuration File: baypines-config.js 45

Figure 6: Sample Federator VSA Server Configuration File: federator-config.js 46

Figure 7: Sample Federator VSA Server Router File: vjs-vista-router.js 47

Figure 8: Sample Web VSA Server Configuration File: web-config.js 50

Figure 9: Sample Web VSA Server Configuration File: API Browser 51

Figure 10: Sample Web VSA Server Configuration File: RPC Wizard 52

Figure 11: View Routines - Import 57

Figure 12: Browse Import Routines 57

Figure 13: Select XSAPXROUTINES.ro 58

Figure 14: Import Routines - Next Screen 59

Figure 15: Review Routines to Import 60

# Introduction

This document provides installation instructions for Veterans Health Information System and Technology Architecture (VistA) Services Assembler (VSA) JavaScript (VistA.js) Platform as managed through the VSA Phase 2 (VSA-P2) project.

 **NOTE:** Throughout this document, commands that need to be entered on the command line are represented in a **bold, *non*-proportional** font. Output from those commands is represented in the same *non*-bold, *non*-proportional font.

## Disclaimers

### Documentation Disclaimers

This manual provides an overall explanation of configuring VSA VistA.js and its functionality; however, no attempt is made to explain how the overall VistA programming system is integrated and maintained. Such methods and procedures are documented elsewhere. We suggest you look at the various VA Internet and Intranet SharePoint sites and websites for a general orientation to VistA. For example, visit the Office of Information and Technology (OI&T) Product Development (PD) Intranet Website.

 DISCLAIMER: The appearance of any external hyperlink references in this manual does *not* constitute endorsement by the Department of Veterans Affairs (VA) of this Website or the information, products, or services contained therein. The VA does *not* exercise any editorial control over the information you find at these locations. Such links are provided and are consistent with the stated purpose of this VA Intranet Service.

# Pre-Installation and System Requirements

System requirements will vary based on the type of VSA VistA.js server being installed. This information is provided in the remainder of this section.

Table : VSA VistA.js Storage Requirements

| Component | Location | Archived | Unarchived | Installed | Notes |
| --- | --- | --- | --- | --- | --- |
| Maestro Server | Enterprise Data Center | 48MB | 118MB | 85MB | Total: 203MB required for a complete installation. |
| vjs-environment | VistA site | 9.8MB | 61.5MB | 61.5MB | Size-wise, unarchiving this file is equivalent to installing it. |
| **Federator**  VSA Server | Enterprise Data Center | N/A | N/A | 29.6MB | Refers to the size of each generated instance. |
| **Web**  VSA Server | Enterprise Data Center | N/A | N/A | 66.1MB | Refers to the size of each generated instance. |
| **VistA**  VSA Server | VistA site | N/A | N/A | 28.3MB | Refers to the size of each generated instance. |

## Platform Installation and Preparation

### General

* No exceptional resources required.
* Sites can install the software in test/pre-production and production accounts independently. There is no reliance on having the software installed in a previous environment.
* No backup procedures need to be performed before installing this software.
* For an initial installation, users do not need to be logged off the system.
* The software does not need to be installed during off-peak hours.
* The average amount of time required to install VSA VistA.js on a given Linux server is about 20 minutes.
* Verifying checksum values (e.g., MD5) is *not* required.
* While all of the VistA.js VSA Servers can run together on a single virtual machine, this document assumes that the Maestro server, **VistA** VSA Server, **Federator** VSA Server, and **Web** VSA Server will all be installed on separate virtual machines**.** The **Web** VSA Server should not be installed in a production environment.

### Pre-Installation Checklist

The following software and software versions are required before the VSA VistA.js platform can be installed on a target Linux server:

#### Maestro Linux Server

* A VA-installed version of Red Hat Enterprise Linux (v6.8).
* Port 4873 must be open. If this port is not open on the server’s firewall, it can be by using the following command:

**sudo /sbin/iptables -A VistA -p tcp -m state --state NEW -m tcp --dport <*port*> -j ACCEPT**

* Maestro Linux server SSL certificate and key.
* Installing user must be able to start and stop a service installed locally to that user.
* Server must be able to access the target VSA VistA.js server(s) at their fully qualified domain name reliably. External internet access is not assumed.
* Node.js is not installed on the server (the installer will provide).
* The installer will not install Maestro as a service. A suggested cron job/script implementation is noted in Section 2.5, Cron Scripts.

#### VistA Linux Server

* A VA-installed version of Red Hat Enterprise Linux (v6.8).
* The following ports must be open. If these port are not open on the server’s firewall, they can be by using the following command:

**sudo /sbin/iptables -A VistA -p tcp -m state --state NEW -m tcp --dport <*port*> -j ACCEPT**

* + For HTTPS connections:
    - Pre-Production: 187<*Magic-Number*>
    - Production: 186<*Magic-Number*>
  + For socket connections:
    - Pre-Production: 189<*Magic-Number*>
    - Production: 188<*Magic-Number*>
* InterSystems’ Caché version 2014.1.3 AH14809 with Jul15SV1Rev2 patch and Engine Link libraries installed.
* The Caché instance must be configured with Operating System Authentication enabled.
* The Caché instance must have the %Service\_CallIn service enabled and set to allow operating system authentication.
* The latest version of cache0120.node (v1.1.123 or higher) must exist in the Caché instance’s bin directory.
* VistA Kernel patch XU\*8.0\*659 must be installed.
* The XSA\_1\_0.KID patch must be applied to the target VistA Caché database prior to starting a **VistA** VSA Server.
* VistA Linux server SSL certificate and key.
* Installing user must be able to start and stop a service installed locally to that user.
* The VistA Linux server must be able to access the target Maestro Linux server reliably on port 4873. External internet access is not assumed.
* Node.js is not installed on the server (the installer will provide).
* The VSA VistA.js installer does not install VSA VistA.js as a service. A suggested cron job/script implementation is noted in [Section 2.5 “Cron Scripts.”](#_Cron_Scripts_1)

#### Federator Linux Server

* A VA-installed version of Red Hat Enterprise Linux (v6.8).
* Port 8080 must be open. If this port is not open on the server’s firewall, it can be by using the following command:

**sudo /sbin/iptables -A VistA -p tcp -m state --state NEW -m tcp --dport <port> -j ACCEPT**

* InterSystems’ Caché version 2014.1.3 AH14809 with Jul15SV1Rev2 patch and Engine Link libraries installed.
* The Caché instance must be configured with Operating System Authentication enabled.
* The Caché instance must have the %Service\_CallIn service enabled and set to allow operating system authentication.
* VistA Kernel patch XU\*8.0\*659 must be installed.
* A FEDERATOR namespace must exist before the routines in the XSAPEXROUTINES.xml file can be imported.
* The XSAPXROUTINES.xml file must be imported into the FEDERATOR namespace before a **Federator** VSA Server can be started.
* The latest version of cache0120.node (v1.1.123 or higher) must exist in the Caché instance’s bin directory.
* Federator Linux server SSL certificate and key.
* Installing user must be able to start and stop a service installed locally to that user.
* The Federator Linux server must be able to access the target Maestro Linux server reliably on port 4873. External internet access is not assumed.
* Node.js is not installed on the server (the installer will provide).
* The VSA VistA.js installer does not install VSA VistA.js as a service. A suggested cron job/script implementation is noted in Section [2.5 “Cron Scripts.”](#_Cron_Scripts_2)

#### Web Linux Server

* A VA-installed version of Red Hat Enterprise Linux (v6.8).
* Port 8080 must be open. If this port is not open on the server’s firewall, it can be by using the following command:

**sudo /sbin/iptables -A VistA -p tcp -m state --state NEW -m tcp --dport <port> -j ACCEPT**

* InterSystems’ Caché version 2014.1.3 AH14809 with Jul15SV1Rev2 patch and Engine Link libraries installed.
* The Caché instance must be configured with Operating System Authentication enabled.
* The Caché instance must have the %Service\_CallIn service enabled and set to allow operating system authentication.
* The latest version of cache0120.node (v1.1.123 or higher) must exist in the Caché instances bin directory.
* Web Linux server SSL certificate and key.
* Installing user must be able to start and stop a service installed locally to that user.
* The Federator Linux server must be able to access the target Maestro Linux server reliably on port 4873. External internet access is not assumed.
* Node.js is not installed on the server (the installer will provide).
* The VSA VistA.js installer does not install VSA VistA.js as a service. A suggested cron job/script implementation is noted in [Section 2.5 “Cron Scripts.”](#_Cron_Scripts_3)

### Installation Users

Table : Default Caché Installation Users

| Server / Environment | Caché User |
| --- | --- |
| Maestro Linux server | vsacacheusr |
| Federator Linux server | vsacacheusr |
| Web Linux server | vsacacheusr |
| VistA VSA Server: General format | <*scd*>cacheusr |
| VistA VSA Server: Boise | boicacheusr |
| VistA VSA Server: South Texas | stxcacheusr |
| VistA VSA Server: Hudson Valley | hvhcacheusr |
| VistA VSA Server: Memphis | memcacheusr |
| VistA VSA Server: Hampton | hamcacheusr |

Table : Default Installation Directories

| Server / Environment | Directory |
| --- | --- |
| Maestro Linux server | /opt/vsa/Maestro |
| Maestro installer | /opt/vsa/MaestroInstaller |
| vjs-environment | /opt/vsa/vjs-environment |
| Federator VSA Server | /app/vsa/user/vistajs/vjs-federator |
| Web VSA Server | /app/vsa/user/vistajs/web |
| VistA VSA Server: Pre-Prod General format | /srv/vista/<*scd*>/user/vistajs/<*SCD*>TST-vista |
| VistA VSA Server: Prod: General format | /srv/vista/<*scd*>/user/vistajs/<*SCD*>-vista |
| VistA VSA Server, Pre-Prod: Boise | /srv/vista/boi/user/vistajs/BOITST-vista |
| VistA VSA Server, Prod: Boise | /srv/vista/boi/user/vistajs/BOI-vista |
| VistA VSA Server, Pre-Prod: South Texas | /srv/vista/stx/user/vistajs/STXTST-vista |
| VistA VSA Server, Prod: South Texas | /srv/vista/stx/user/vistajs/STX-vista |
| VistA VSA Server, Pre-Prod: Hudson Valley | /srv/vista/hvh/user/vistajs/HVHTST-vista |
| VistA VSA Server, Prod: Hudson Valley | /srv/vista/hvh/user/vistajs/HVH-vista |
| VistA VSA Server, Pre-Prod: Memphis | /srv/vista/mem/user/vistajs/MEMTST-vista |
| VistA VSA Server, Prod: Memphis | /srv/vista/mem/user/vistajs/MEM-vista |
| VistA VSA Server, Pre-Prod: Hampton | /srv/vista/ham/user/vistajs/HAMTST-vista |
| VistA VSA Server, Prod: Hampton | /srv/vista/ham/user/vistajs/HAM-vista |

## hostnameDownload and Extract Files

### Release Files

Release files are located in Jazz under CCM > Source Control > VSA VistA.js code, consisting of:

* vsa-vistajs-maestro-3.0.*n*.tgz, where *n* represents the latest release(s) referenced in the most recent entry in [Table 1: Revision History](#COL001_TBL001), and on this document’s cover page.
* vsa-vistajs-environment-3.0.5.tgz

 **NOTE:** For IOC, Maestro will only be installed at the Austin Information Technology Center (AITC).

### On the Target Maestro Linux Server

1. Log into the target Maestro Linux server.
2. Create an /opt/vsa/ directory if one does not already exist:

**sudo install –m755 -d /opt/vsa**

1. Move the vsa-vistajs-maestro-3.0.*n*.tgz to the /opt/vsa directory:

**sudo mv vsa-vistajs-maestro-v3.0.*n*.gtar /opt/vsa**

1. Change to the /opt/vsa directory:

**cd /opt/vsa**

1. Extract the Maestro server archive:

**sudo tar xvf vsa-vistajs-maestro-v3.0.*n*.gtar**

### On the Target VistA, Federator, and Web Linux Servers

 **NOTE:** The vsa-vistajs-environment-3.0.5.tgz file *must* be copied to each target **VistA**, **Federator**, and **Web** Linux server.

 **NOTE:** A VistA.js **Web** VSA Server should not be installed in a production environment. The VistA.js **Web** VSA Server contains developer tools (specifically the RPC Wizard and API Browser) than can be used to modify or delete existing RESTified RPCs outside of the existing VA production promotion process. As such, all references to the **Web** VSA Server in the following steps can be skipped.

1. Log into the target VistA, Federator, or Web Linux server.
2. Create an /opt/vsa directory if one does not already exist:

**sudo install -m755 -d /opt/vsa**

1. Move the vsa-vistajs-environment-3.0.5.tgz file to the /opt/vsa directory.

**sudo mv vsa-vistajs-environment-3.0.5.tgz /opt/vsa**

1. Change to the /opt/vsa directory:

**cd /opt/vsa**

1. Extract the VSA VistA.js environment archive:

**sudo tar xzvf vsa-vistajs-environment-3.0.5.tgz**

1. Verify in the top level of the vjs-environment directory that the following contents are there:

**sudo ls -1 /opt/vsa/vjs-environment**

create-servers.sh

README.md

resources/

setup.sh

use\_node.sh

1. Verify the following directory exists:

**sudo ls -1 /opt/vsa/vjs-environment/resources**

node

## Database Creation

This section assumes that a database already exists, and that the required MUMPS components will be imported into the target VistA database.

 **REF:** For the associated Kernel Installation & Distribution System (KIDS) patch procedure, see Appendix B—Importing the MUMPS Components. This procedure can be done after completing the remainder of this installation process.

## Installation Scripts

VSA VistA.js is a “generative” platform, which allows it to create new server instances without the need to download or install additional software. VSA VistA.js provides the components necessary to fully create **VistA**, **Federator**, and **Web** VSA Servers on-demand, which are included in their respective installation archive files. Outside of an installation that is performed by following these instructions, no other installation scripts are required.

The Maestro installation script, maestro-installer, is required for the Maestro server only.

The use\_node.sh script, included with vjs-environment, is used to add the VSA VistA.js-specific version of Node.js in the installing user’s path.

After issuing the “npm install vista-services-assembler -g” command, VSA VistA.js server configurations and installations are handled via the VSA VistA.js vsa-gen command.

 **REF:** For more information, see “[Appendix C—Installation Scripts](#_Appendix_D:_Installation).”

## Cron Scripts

The sinopia.init.d file can be moved to the /etc./init.d directory by a System Administrator (e.g., mv sinopia.init.d /etc/init.d/sinopia) to run as a service when the server is booted.

For starting VSA VistA.js instances on boot, add the following to the appropriate boot files on the following servers:

**Maestro** Linux server:

nohup /opt/vsa/MaestroInstaller/scripts/start.sh &

**VistA**, **Federator**, and **Web** Linux servers:

nohup /opt/vsa/vjs-environment/source use\_node.sh &

nohup /opt/vsa/vjs-environment/setup.sh -i <*Maestro-Linux-Server-Fully-Qualified-Domain-Name>* &

**VistA** Linux server:

nohup /srv/vista/<*scd*>/user/vistajs/<*SCD*>-vista/start-<*SCD*>-vista.sh &

**Federator** Linux server:

nohup /app/vsa/user/vistajs/vjs-federator/start-vjs-federator.sh &

**Web** Linux server:

nohup /app/vsa/user/vistajs/web/start-web.sh &

## Access Requirements and Skills Needed for the Installation

The installing user should possess basic Linux command-line skills such as:

* Logging in to a target system via **ssh**.
* Using SFTP to download files.
* Listing directories/files via **ls**.
* Navigating around the file system via **cd**.
* Moving files via **mv**.
* Extracting files via **tar**.
* Modifying file attributes via **chmod** and **chown**.
* Starting and stopping processes via **ps** and **kill**.
* Run processes and scripts as another user via **sudo**.

# Installation Procedure

## On the Target Maestro Linux Server

1. Change to the MaestroInstaller directory:

**cd /opt/vsa/MaestroInstaller**

1. Execute the installer script:

**sudo ./maestro-installer install /opt/vsa**

1. Confirm that the Maestro directory has been created:

**ls -1 /opt/vsa**

Maestro

1. Check that sinopia (Maestro) is running on port 4873:

**sudo netstat -plnt | grep sinopia**

tcp 0 0 0.0.0.0:4873 0.0.0.0:\* LISTEN 2964/Sinopia

1. Verify that the following generated files are in the MaestroInstaller/scripts directory:

**ls -a1 scripts:**

sinopia.init.d

start.sh

stop.sh

 **NOTE:** Once installed, the generated scripts for stopping and starting the sinopia service can be invoked directly via the following scripts in the scripts directory:

* Starting the sinopia service:

./start.sh

* Stopping and starting the sinopia service:

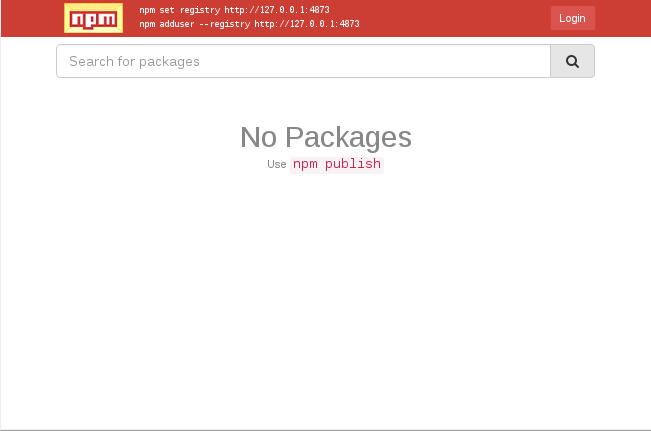
./stop.sh

1. In a browser from a machine with access to the Maestro Linux server, navigate to:

https://*<Maestro-Linux-Server-Fully-Qualified-Domain-Name>:*4873

1. If the Maestro server is running, you should see the Maestro server landing page:

Figure : Maestro Server Landing Page



## On the Target VistA Linux Server

1. Set the PATH variable for current user to include the VSA VistA.js /node/bin directory:

**export PATH=$PATH:/opt/vsa/vjs-environment/resources/node/bin**

 **NOTE:** Perform steps 1 - 7 *once* per server, then perform steps 8 - 19 for each **VistA** instance on that server.

1. Change to the vjs-environment directory:

**cd /opt/vsa/vjs-environment**

1. Execute the setup script to connect to the Maestro server:

**sudo env PATH=$PATH ./setup.sh -i <*Maestro-Linux-Server-Fully-Qualified-Domain-Name*>**

You should see output similar the following:

[vjs] Setting up local environment and link to Maestro

[vjs] Connected to Maestro at IP: <Maestro-Server-Fully-Qualified-Domain-Name>

[vjs] Setting up npm registry.

[vjs] Logging in to npm.

[vjs] Finished setup.

1. Configure npm:

**sudo env PATH=$PATH npm -g config set registry http://<*Maestro-Linux-Server-Fully-Qualified-Domain-Name*>:4873**

**sudo env PATH=$PATH npm -g config set user root**

1. Clear the NPM cache:

**sudo env PATH=$PATH npm cache clear**

1. Install VistA Services Assembler:

**sudo env PATH=$PATH npm install -g vista-services-assembler**

You should see output similar to the following:

/opt/vsa/vjs-environment/resources/node/bin/ -> /opt/vsa/vjs-environment/resources/node/lib/node\_modules/vista-services-assembler/bin/vjs.js

vista-services-assembler@0.7.0 /opt/vsa/vjs-environment/resources/node/lib/node\_modules/vista-services-assembler

├── figures@1.3.5

├── vjs-globals@1.0.2

├── telnet-client@0.1.0

├── through@2.3.8

├── mustache@2.1.3

├── ent@2.2.0

├── chalk@1.1.0 (escape-string-regexp@1.0.3, supports-color@2.0.0, ansi-styles@2.1.0, has-ansi@2.0.0, strip-ansi@3.0.0)

├── nomnom@1.8.1 (underscore@1.6.0, chalk@0.4.0)

├── shelljs@0.4.0

├── inquirer@0.8.5 (ansi-regex@1.1.1, cli-width@1.0.1, readline2@0.1.1)

├── mu2@0.5.20

├── fs-extra@0.18.4 (jsonfile@2.2.1, graceful-fs@3.0.8, rimraf@2.4.2)

├── lodash@3.10.0

└── rx@2.5.3

1. Confirm that vsa-gen is installed:

**sudo env PATH=$PATH vsa-gen --version**

You should see output similar to the following:

VSA Generator Version x.x.x

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file

Except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

1. Create a vistajs directory:

**sudo install -m775 -d /srv/vista/<*scd*>/user/vistajs**

1. Change directory to /srv/vista/*<scd>*/user/vistajs/:

**cd /srv/vista/<*scd*>/user/vistajs**

1. Create a VistA VSA Server configuration file by running:

**sudo env PATH=$PATH vsa-gen config <*namespace*>**

You will be prompted to enter values for each of the configuration items *specific to the selected VSA Server profile*.

For **VistA** VSA Servers, *<namespace>* should be the name of the target namespace in Caché where VistA is running. Use the following values, substituting the particular **VistA** VSA Server’s information as required when prompted by vsa-gen:

Table : VistA VSA Server Configuration

| Question | Answer |
| --- | --- |
| To which platform should this VSA Web Server instance be targeted? | node |
| To which target architectural profile should this VSA Web Server instance be configured? | vista |
| Which platform framework would you like to use to target this VSA Web Server instance? | vsa |
| What should this VSA Web Server instance be called? | Pre-Prod: <*SCD*>TST-vista  Prod: <*SCD>*-vista |
| Run this VSA Web Server in which environment? | production |
| Which port do you wish to use for HTTP? | Pre-Prod: 187<*Magic-Number*>  Prod: 186<*Magic-Number*>  See [Table 4](#_Installation_Users) for more information. |
| Use SSL (i.e., HTTPS)? | Yes |
| Use default certificates? (not recommended for production) | Yes |
| When clustering, how many workers (processes) should be used? | **Clustered**: (# of available CPU cores) - 1  **Non-clustered:** Single CPU with single core, or to use a single core on a multi-core CPU: 1 |
| What type of database should this instance connect to? | cache |
| Where is the target database located? | General: /srv/vista/<*scd*>/cache/scdr*#*tpr*XX*/mgr  Where:   * **scd**: Three letter site code * **r***:* ***R***egional systems * **#:** Region number (1-4) * **t** -*or*- **p***:* ***t***est/pre-production, or ***p***roduction * **r** : ***d***ata server, or ***a***pp server * **XX**: Instance number (typically 01- 03 for app servers) |
| To which database file or namespace should this instance connect? | Pre-Prod: <*SCD*>TST  Prod: <*SCD*> |
| Which user account should be used to access this database? | <*scd*>cacheusr |
| What password should be used to validate the user account? | None required; use default. |

 **REF:** For more information, see [Appendix A—Sample Server Configuration Files](#_Appendix_A:_Sample_2).

1. Once the instance has been configured, generate the VistA VSA Server:

**sudo env PATH=$PATH npm\_config\_group=<*scd*>cacheusr vsa-gen server <*namespace*>-config.js**

You should see output similar to the following:

Loading assembler module...

Assembly process starting now

About to assemble baypines-vista

✓ Installing vsa-server

✓ Generating Base Server

✓ Generating package.json

✓ Generating Start Scripts

✓ Installing Modules

✓ Generating shared storage config

✓ Disabling pub/sub

✓ Installing SSL Certs

✓ Disabling websocket Server

✓ Disabling metadata information

✓ Disabling Development Mode

✓ Removing default actions

✓ Configuring HTTP(S) Port

✓ Configuring Instance Name

✓ Forcing browser to IE11 mode

✓ Installing Default Config Files

✓ Installing Actions

✓ Copying MCode

✓ Setting Server Type to API

✓ Generating database config

✓ Configuring Filtered Params

✓ Installing Default Public Files

✓ Copying cache.node

Generation of baypines-vista completed.

Start the instance in single threaded mode by typing:

cd baypines-vista

./start-baypines-vista.sh

Start the instance in clustered mode by typing:

cd baypines-vista

./start-baypines-vista-cluster.sh

Successfully assembled baypines-vista server.

1. After the **VistA** VSA Server is generated, you should have a directory structure similar to the following:

+-- baypines-config.js

+-- baypines-vista/

| +-- actions/

| +-- config/

| +-- gruntfile.js

| +-- initializers/

| +-- log/

| +-- mcode/

| +-- node\_modules/

| +-- package.json

| +-- pids/

| +-- public/

| +-- README.md

| +-- servers/

| +-- ssl/

| +-- start-baypines-vista.sh

| +-- tasks/

| +-- test/

1. Change ownership of the <*namespace*>-vista directory and its contents to the site specific user:

**sudo chown -R <*scd*>cacheusr \ /srv/vista/<*scd*>/user/vistajs/<*namespace*>-vista**

1. Change to the <*namespace*>-vista directory:

**cd /srv/vista/<*scd*>/user/vistajs/<*namespace*>-vista**

1. Make a copy of the default vistajs.key file:

**sudo -u <*scd*>cacheusr mv ssl/vistajs.key ssl/vistajs.key.org**

1. Replace the default vistajs.key file with a server-specific vistajs.key file:

**sudo -u <*scd*>cacheusr cp <*Directory*>/vistajs.key ssl/**

 **NOTE:** Be sure that the “-----BEGIN RSA PRIVATE KEY-----” line appears at the top of the vistajs.key file, and the “-----END RSA PRIVATE KEY-----” line appears at the bottom. If these two lines are missing in their respective locations, the server may fail to start.

1. If required, run dos2unix to convert the vistajs.key DOS text file format to UNIX:

**sudo -u <*scd*>cacheusr dos2unix ssl/vistajs.key**

1. Make a copy of the default vistajs.crt file:

**sudo -u <*scd*>cacheusr mv ssl/vistajs.crt ssl/vistajs.crt.org**

1. Replace the default vistajs.crt file with a server-specific vistajs.crt file:

**sudo -u <*scd*>cacheusr cp <*Directory*>/vistajs.crt /ssl**

 **NOTE:** Be sure that the “-----BEGIN CERTIFICATE-----” line appears at the top of the vistajs.crt file, and the “-----END CERTIFICATE-----” line appears at the bottom. If these two lines are missing in their respective locations, the server may fail to start.

1. If required, run dos2unix to convert the vistajs.crt DOS text file format to UNIX:

**sudo –u <*scd*>cacheusr dos2unix ssl/vistajs.crt**

1.  **NOTE:** Prior to starting a **VistA** VSA Server, ensure that the XSA\_1\_0.KID patch has been applied on the target VistA system. See [Appendix B – Importing the MUMPS Components](#_Appendix_B_–) for more information.
2. Start the **VistA** VSA Server:

*Non-clustered:*

**sudo -b -u <*scd*>cacheusr env PATH=$PATH nohup ./start-<*namespace*>-vista.sh &**

*Clustered:*

**sudo -b -u <*scd*>cacheusr env PATH=$PATH nohup ./start-<*namespace*>-vista-cluster.sh &**

1. Server output can be reviewed in nohup.out:

**sudo less nohup.out**

1. In a browser from a machine with access to the **VistA** Linux server, navigate to:

https://*<VistA-Linux-Server-Fully-Qualified-Domain-Name>:*8080

1. If the **VistA** VSA Server is running, the VSA Server landing page will be displayed:

Figure : VSA Server Landing Page



## On the Target Federator Linux Server

1. Set the PATH variable for current user to include node bin directory:

**export PATH=$PATH:/opt/vsa/vjs-environment/resources/node/bin**

1. Change to the vjs-environment directory:

**cd /opt/vsa/vjs-environment**

1. Execute the setup script to connect to the Maestro server:

**sudo env PATH=$PATH ./setup.sh -i *<Maestro-Linux-Server-Fully-Qualified-Domain-Name>***

You should see output similar the following:

[vjs] Setting up local environment and link to Maestro

[vjs] Connected to Maestro at IP: *<Maestro-Server-Fully-Qualified-Domain-Name>*

[vjs] Setting up npm registry.

[vjs] Logging in to npm.

[vjs] Finished setup.

1. Configure npm:

**sudo env PATH=$PATH npm -g config set registry http*://<Maestro-Linux-Server-Fully-Qualified-Domain-Name>:*4873**

**sudo env PATH=$PATH npm -g config set user root**

1. Clear the NPM cache:

**sudo env PATH=$PATH npm cache clear**

1. Install VistA Services Assembler:

**sudo env PATH=$PATH npm install -g vista-services-assembler**

You should see output similar to the following:

/opt/vsa/vjs-environment/resources/node/bin/vjs -> /opt/vsa/vjs-environment/resources/node/lib/node\_modules/vista-services-assembler/bin/vjs.js

vista-services-assembler@0.7.0 /opt/vsa/vjs-environment/resources/node/lib/node\_modules/vista-services-assembler

├── figures@1.3.5

├── vjs-globals@1.0.2

├── telnet-client@0.1.0

├── through@2.3.8

├── mustache@2.1.3

├── ent@2.2.0

├── chalk@1.1.0 (escape-string-regexp@1.0.3, supports-color@2.0.0, ansi-styles@2.1.0, has-ansi@2.0.0, strip-ansi@3.0.0)

├── nomnom@1.8.1 (underscore@1.6.0, chalk@0.4.0)

├── shelljs@0.4.0

├── inquirer@0.8.5 (ansi-regex@1.1.1, cli-width@1.0.1, readline2@0.1.1)

├── mu2@0.5.20

├── fs-extra@0.18.4 (jsonfile@2.2.1, graceful-fs@3.0.8, rimraf@2.4.2)

├── lodash@3.10.0

└── rx@2.5.3

1. Confirm that vsa-gen is installed:

**sudo env PATH=$PATH vsa-gen --version**

You should see output similar to the following:

VSA Generator Version x.x.x

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file

Except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

1. Create a vistajs directory:

**sudo install -m775 -g vsacacheusr -d /app/vsa/user/vistajs**

1. Change directory to /app/vsa/user/vistajs/:

**cd /app/vsa/user/vistajs**

1. Create a Federator VSA Server configuration file by running:

**sudo env PATH=$PATH vsa-gen config federator**

Use the following values, substituting the particular **Federator** VSA Server’s information as required when prompted by vsa-gen:

Table : Federator VSA Server Configuration

| Question | Answer |
| --- | --- |
| To which platform should this VSA Web Server instance be targeted? | node |
| To which target architectural profile should this VSA Web Server instance be configured? | federator |
| Which platform framework would you like to use to target this VSA Web Server instance? | vsa |
| What should this VSA Web Server instance be called? | vjs-federator |
| Run this VSA Web Server in which environment? | production |
| Which port do you wish to use for HTTP? | 8080  See [Table 4](#_Installation_Users_1) for more information. |
| Use SSL (i.e., HTTPS)? | Yes |
| Use default certificates? (not recommended for production) | Yes |
| When clustering, how many workers (processes) should be used? | **Clustered**: (# of available CPU cores) - 1  **Non-clustered:** Single CPU with single core, or to use a single core on a multi-core CPU: 1 |
| Where is the shared storage database located? | General:  /app/vsa/cache/vsae*X*tf4*X*/mgr  Pre-Prod:  DNS saweb40: /app/vsa/cache/vsaeatf40/mgr  DNS saweb41:  /app/vsa/cache/vsaeatf41/mgr  vaphivsaweb40:  /app/vsa/cache/vsaeptf40/mgr  vaphivsaweb41:  /app/vsa/cache/vsaeptf41/mgr  Prod:  TBD |
| To which namespace should this instance connect to on the shared storage DB? | FEDERATOR |
| Which user account should be used to access the shared storage DB? | vsacacheusr |
| What password should be used to access the shared storage DB? | None required; use default. |

 **REF:** For more information, see [[Appendix A—Sample Server Configuration Files](#_Appendix_A:_Sample_2)s](#_Appendix_A:_Sample_1).

1. Once the instance has been configured, generate the Federator VSA Server:

**sudo env PATH=$PATH npm\_config\_group=vsacacheusr vsa-gen server federator-config.js**

You should see output similar to the following:

Loading assembler module...

Assembly process starting now

About to assemble vjs-federator

✓ Installing vsa-server

✓ Generating Base Server

✓ Generating package.json

✓ Generating Start Scripts

✓ Installing Modules

✓ Disabling pub/sub

✓ Installing SSL Certs

✓ Disabling websocket Server

✓ Disabling metadata information

✓ Disabling Development Mode

✓ Disabling Param Scrubbing

✓ Allowing Session Token Headers

✓ Removing default actions

✓ Configuring HTTP(S) Port

✓ Configuring Instance Name

✓ Forcing browser to IE11 mode

✓ Installing Default Config Files

✓ Installing Actions

✓ Copying MCode

✓ Copying cache.node

✓ Generating shared storage config

✓ Configuring Filtered Params

✓ Installing Default Public Files

✓ Setting Server Type to API

Generation of vjs-federator completed.

Start the instance in single threaded mode by typing:

cd vjs-federator

./start-vjs-federator.sh

Start the instance in clustered mode by typing:

cd vjs-federator

./start-vjs-federator-cluster.sh

Successfully assembled vjs-federator server.

1. After the **Federator** VSA Server is generated, you should have a directory structure similar to the following:

+-- federator-config.js

+-- vjs-federator/  
| +-- actions/

| +-- config/

| +-- gruntfile.js

| +-- initializers/

| +-- log/

| +-- node\_modules/

| +-- package.json

| +-- pids/

| +-- public/

| +-- README.md

| +-- servers/

| +-- ssl/

| +-- start-vjs-federator.sh

| +-- tasks/

| +-- test/

1. Change ownership of the vjs-federator directory and its contents to the site specific user:

**sudo chown -R vsacacheusr /app/vsa/user/vistajs/vjs-federator**

1. Change the attributes of the vjs-federator directory:

**sudo chmod 755 /app/vsa/user/vistajs/vjs-federator**

1. Change to the vjs-federator directory:

**cd vjs-federator**

1. Make a copy of the default vjs-vista-router.js file:

**sudo -u vsacacheusr mv config/plugins/vjs-vista-router.js config/plugins/vjs-vista-router.js.org**

1. Replace the default vjs-vista-router.js file with an IOC-specific vjs-vista-router.js file:

**sudo -u vsacacheusr cp <*Directory*>/vjs-vista-router.js config/plugins/**

 **REF:** For more information, see [Appendix A—Sample Server Configuration Files](#_Appendix_A:_Sample_2). An example vjs-vista-router.js file is provided in this appendix for reference.

1. Make a copy of the default vistajs.key file:

**sudo -u vsacacheusr mv ssl/vistajs.key ssl/vistajs.key.org**

1. Replace the default vistajs.key file with a server-specific vistajs.key file:

**sudo -u vsacacheusr cp <*Directory*>/vistajs.key ssl/**

 **NOTE:** Be sure that the “-----BEGIN RSA PRIVATE KEY-----” line appears at the top of the vistajs.key file, and the “-----END RSA PRIVATE KEY-----” line appears at the bottom. If these two lines are missing in their respective locations, the server may fail to start.

1. If required, run dos2unix to convert the vistajs.key DOS text file format to UNIX:

**sudo -u vsacacheusr dos2unix ssl/vistajs.key**

1. Make a copy of the default vistajs.crt file:

**sudo -u vsacacheusr mv ssl/vistajs.crt ssl/vistajs.crt.org**

1. Replace the default vistajs.crt file with a server-specific vistajs.crt file:

**sudo -u vsacacheusr cp <*Directory*>/vistajs.crt /ssl**

 **NOTE:** Be sure that the “-----BEGIN CERTIFICATE-----” line appears at the top of the vistajs.crt file, and the “-----END CERTIFICATE-----” line appears at the bottom. If these two lines are missing in their respective locations, the server may fail to start.

1. If required, run dos2unix to convert the vistajs.crt DOS text file format to UNIX:

**sudo -u vsacacheusr dos2unix ssl/vistajs.crt**

 **NOTE:** Prior to starting a **Federator** VSA Server, ensure that the routines in the XSAPXROUTINES.xml have been imported into the FEDERATOR namespace. See [Appendix B – Importing the MUMPS Components](#_Appendix_B_–_1) for more information.

1. Start the **Federator** VSA Server:

*Non-clustered:*

**sudo -b -u vsacacheusr env PATH=$PATH nohup ./start-vjs-federator.sh \ &**

*Clustered:*

**sudo -b -u vsacacheusr env PATH=$PATH nohup ./start-vjs-federator-cluster.sh \ &**

1. Server output can be reviewed in nohup.out:

**sudo less nohup.out**

1. In a browser from a machine with access to the **Federator** Linux server, navigate to:

https://*<Federator-Linux-Server-Fully-Qualified-Domain-Name>:*8080

1. If the **Federator** VSA Server is running, the VSA Server landing page will be displayed:

Figure : VSA Server Landing Page



## On the Target Web Linux Server

 **NOTE:** A VistA.js **Web** VSA Server should not be installed in a production environment. The VistA.js **Web** VSA Server contains developer tools (specifically the RPC Wizard and API Browser) than can be used to modify or delete existing RESTified RPCs outside of the existing VA production promotion process. As such, this section can be skipped for a production installation.

1. Set the PATH variable for current user to include node bin directory:

**export PATH=$PATH:/opt/vsa/vjs-environment/resources/node/bin**

1. Change to the vjs-environment directory:

**cd /opt/vsa/vjs-environment**

1. Execute the setup script:

**sudo env PATH=$PATH ./setup.sh -i *<Maestro-Linux-Server-Fully-Qualified-Domain-Name>***

You should see output similar the following:

[vjs] Setting up local environment and link to Maestro

[vjs] Connected to Maestro at IP: *<Maestro-Server-IP-Address>*

[vjs] Setting up npm registry.

[vjs] Logging in to npm.

[vjs] Finished setup.

1. Configure npm:

**sudo env PATH=$PATH npm -g config set registry http://*<Maestro-Server-Fully-Qualified-Domain-Name>*:4873**

**sudo env PATH=$PATH npm -g config set user root**

1. Clear the NPM cache:

**sudo env PATH=$PATH npm cache clear**

1. Install VistA Services Assembler:

**sudo env PATH=$PATH npm install -g vista-services-assembler**

You should see output similar to the following:

/opt/vsa/vjs-environment/resources/node/bin/vjs -> /opt/vsa/vjs-environment/resources/node/lib/node\_modules/vista-services-assembler/bin/vjs.js

vista-services-assembler@0.7.0 /opt/vsa/vjs-environment/resources/node/lib/node\_modules/vista-services-assembler

├── figures@1.3.5

├── vjs-globals@1.0.2

├── telnet-client@0.1.0

├── through@2.3.8

├── mustache@2.1.3

├── ent@2.2.0

├── chalk@1.1.0 (escape-string-regexp@1.0.3, supports-color@2.0.0, ansi-styles@2.1.0, has-ansi@2.0.0, strip-ansi@3.0.0)

├── nomnom@1.8.1 (underscore@1.6.0, chalk@0.4.0)

├── shelljs@0.4.0

├── inquirer@0.8.5 (ansi-regex@1.1.1, cli-width@1.0.1, readline2@0.1.1)

├── mu2@0.5.20

├── fs-extra@0.18.4 (jsonfile@2.2.1, graceful-fs@3.0.8, rimraf@2.4.2)

├── lodash@3.10.0

└── rx@2.5.3

1. Confirm that vsa-gen is installed:

**sudo env PATH=$PATH vsa-gen --version**

You should see output similar to the following:

VSA Generator Version x.x.x

Licensed under the Apache License, Version 2.0 (the "License"); you may not use this file

Except in compliance with the License. You may obtain a copy of the License at

http://www.apache.org/licenses/LICENSE-2.0

1. Create a vistajs directory:

**sudo install -m775 -g vsacacheusr -d /app/vsa/user/vistajs**

1. Change directory to /app/vsa/user/vistajs/:

**cd /app/vsa/user/vistajs**

1. Create the **Web** VSA Server configuration file by running:

**sudo env PATH=$PATH vsa-gen config web**

Use the following values, substituting the particular **Web** VSA Server’s information as required when prompted by vsa-gen:

Table : Web VSA Server Configuration

| Question | Answer |
| --- | --- |
| To which platform should this VSA Web Server instance be targeted? | node |
| To which target architectural profile should this VSA Web Server instance be configured? | web |
| Which platform framework would you like to use to target this VSA Web Server instance? | vsa |
| What should this VSA Web Server instance be called? | web |
| Run this VSA Web Server in which environment? | production |
| Which port do you wish to use for HTTP? | 8080  See [Table 4](#_Installation_Users_2) for more information. |
| Use SSL (i.e., HTTPS)? | Yes |
| Use default certificates? (not recommended for production) | Yes |
| When clustering, how many workers (processes) should be used? | **Clustered**: (# of available CPU cores) - 1  **Non-clustered:** Single CPU with single core, or to use a single core on a multi-core CPU: 1 |
| Which directory should the cache node binary (cache0120.node) be copied from? | General:  /app/vsa/cache/vex/bin  Pre-Prod:  /app/vsa/cache/vsaewd20/bin  /app/vsa/cache/vsaetw55/bin |

 **REF:** For more information, see [Appendix A—Sample Server Configuration Files](#_Appendix_A:_Sample_2).

1. Once the instance has been configured, generate the **Web** VSA Server:

**sudo env PATH=$PATH npm\_config\_group=vsacacheusr vsa-gen server web-config.js**

You should see output similar to the following:

Loading assembler module...

Assembly process starting now

About to assemble web

✓ Installing vsa-server

✓ Generating Base Server

✓ Generating package.json

✓ Generating Start Scripts

✓ Installing Modules

✓ Disabling pub/sub

✓ Installing SSL Certs

✓ Disabling metadata information

✓ Disabling Development Mode

✓ Removing default actions

✓ Disabling Param Scrubbing

✓ Configuring HTTP(S) Port

✓ Installing Binary Modules.

✓ Configuring Instance Name

✓ Forcing browser to IE11 mode

✓ Installing Default Config Files

✓ Installing Actions

✓ Copying cache.node

✓ Installing Scripts

✓ Generating database config

✓ Configuring Filtered Params

✓ Installing Default Public Files

Generation of web completed.

Start the instance in single threaded mode by typing:

cd web

./start-web.sh

Start the instance in clustered mode by typing:

cd web

./start-web-cluster.sh

Successfully assembled web server.

1. After the **Web** VSA Server is configured, you should have a directory structure similar to the following:

+-- web-config.js

+-- web/

| +-- actions/

| +-- config/

| +-- gruntfile.js

| +-- initializers/

| +-- log/

| +-- node\_modules/

| +-- package.json

| +-- pids/

| +-- public/

| +-- README.md

| +-- scripts/

| +-- servers/

| +-- ssl/

| +-- start-web.sh

| +-- tasks/

| +-- test/

1. Change ownership of the web directory and its contents to the site specific user:

**sudo chown -R vsacacheusr /app/vsa/user/vistajs/web**

1. Change the attributes of the vjs-federator directory:

**sudo chmod 755 /app/vsa/user/vistajs/web**

1. Change to the web directory:

**cd web**

1. Edit the config/plugins/vsa-rpc-wizard-application.js file to ensure the details for the **Federator** VSA Server are configured correctly. Specifically, for the federatorServers entry:

* The name (vjs-federator)
* The Federator Linux server fully qualified domain name
* The port (8080)

1. Edit the config/plugins/vsa-api-browser-application.js file to ensure the details for the **Federator** VSA Server are configured correctly. Specifically, for the federatorServers entry:

* The name (vjs-federator)
* The Federator Linux server fully qualified domain name
* The port (8080)

 **REF:** For more information, see [Appendix A: Sample Server Configuration Files](#_Appendix_A:_Sample_3).

1. Make a copy of the default vistajs.key file:

**sudo -u vsacacheusr mv ssl/vistajs.key ssl/vistajs.key.org**

1. Replace the default vistajs.key file with a server-specific vistajs.key file:

**sudo -u vsacacheusr cp <*Directory*>/vistajs.key ssl/**

 **NOTE:** Be sure that the “-----BEGIN RSA PRIVATE KEY-----” line appears at the top of the vistajs.key file, and the “-----END RSA PRIVATE KEY-----” line appears at the bottom. If these two lines are missing in their respective locations, the server may fail to start.

1. If required, run dos2unix to convert the vistajs.key DOS text file format to UNIX:

**sudo -u vsacacheusr dos2unix ssl/vistajs.key**

1. Make a copy of the default vistajs.crt file:

**sudo -u vsacacheusr mv ssl/vistajs.crt ssl/vistajs.crt.org**

1. Replace the default vistajs.crt file with a server-specific vistajs.crt file:

**sudo -u vsacacheusr cp <*Directory*>/vistajs.crt ssl/**

 **NOTE:** Be sure that the “-----BEGIN CERTIFICATE-----” line appears at the top of the vistajs.crt file, and the “-----END CERTIFICATE-----” line appears at the bottom. If these two lines are missing in their respective locations, the server may fail to start.

1. If required, run dos2unix to convert the vistajs.crt DOS text file format to UNIX:

**sudo -u vsacacheusr dos2unix ssl/vistajs.crt**

1. Start the **Web** VSA Server:

*Clustered:*

**sudo -b -u vsacacheusr env PATH=$PATH nohup ./start-web.sh &**

*Non-clustered:*

**sudo -b -u vsacacheusr env PATH=$PATH nohup ./start-web-cluster.sh &**

1. Server output can be reviewed in nohup.out:

**sudo less nohup.out**

1. In a browser from a machine with access to the target **Web** server, navigate to:

https://*<Web-Linux-Server-Fully-Qualified-Domain-Name>:*8080

1. If the **Web** VSA Server is running, the VSA Server landing page will be displayed:

Figure : VSA Server Landing Page



# Implementation Procedure

There are no post-installation procedures required. To monitor server traffic, log files can be examined in the following directory for each server created via vsa-gen server:

Table : Log File Locations

| VSA Server | Log File Location |
| --- | --- |
| VistA | /srv/vista/<*scd>/*user/vistajs/*<server-name>*/log |
| Federator | /app/vsa/user/vistajs/*<server-name>*/log |
| Web | /app/vsa/user/vistajs/*<server-name>*/log |

## System Configuration

If all three VSA Servers (**VistA**, **Federator**, and **Web**) are run on the same physical or virtual machine where Caché is also installed, the user starting the **VistA** and **Federator** VSA Servers must be a member of that Linux server’s cacheusr group.

During testing, the installation was performed under the cacheusr account to create individual **VistA**, **Federator**, and **Web** VSA Servers on the front-end machine as described in the steps above. The same is true when installing VSA VistA.js on separate physical or virtual machines.

## Database Tuning

N/A.

# Back-Out Procedure

Back-out pertains to a return to the last known good operational state of the software and appropriate platform settings.

## Back-out Strategy

VSA VistA.js is a platform. Currently, all existing VSA VistA.js servers (Maestro server, **VistA**, **Federator**, and **Web** VSA Servers) should be replaced when installing a new VSA VistA.js version. Therefore, the back-out strategy currently pertains to the *entire* platform rather than the platform’s individual servers and components.

The back-out strategy applies to both the files installed on the file server, and the routines/KIDS patches applied to the MUMPS database.

 **REF:** For more information on the associated database components, see the *VSA VistA.js Technical Manual* and [Appendix B, Importing the MUMPS Components.](#_Appendix_B_–_2)

In general, it is *recommended* to install a new version of the VSA VistA.js platform and ensure that it is working as expected *before* replacing the current, running version. If for some reason the new version does not function as expected, there should be no impact to the current, running version, or the services or applications running on it.

If rolling back to a prior version, the current version of the VSA VistA.js platform *must* be completely removed, and the desired version installed in its place. The files of the current version can be archived in any number of ways before installing a new version, including:

Renaming the current directory that the VSA VistA.js server is installed in before installing a new version.

Installing each new version of a VSA VistA.js server in a separate directory, and making a symlink to the current version. This allows multiple versions to co-exist, so that rolling back to a prior version is simply a matter of changing a symlink.

Keeping archival copies of the Maestro and vjs-environment files accompanying each VSA VistA.js version.

Maintaining prior versions of the KIDS build file is useful so it can be reapplied in conjunction with a prior release of the VSA VistA.js platform. As of VSA VistA.js 3.0.2, an XSA\_1\_0.KID file is located in each VSA VistA.js VistA server’s mcode directory, which is created when a new VSA VistA.js VistA server is configured and generated via vsa-gen.

## Back-out Considerations

This section *must* be addressed by the VA’s existing “chain-of-command” as it relates to starting, stopping, installing, and removing servers and server instances/processes.

### Load Testing

Load Testing will be addressed in Increment 3.

 **REF:** For more information, consult the VSA VistA.js Performance Engineering Plan.

### User Acceptance Testing

Apex Data Solutions will assist with back-out during UAT if necessary.

## Back-out Criteria

Backing out an installed version of VSA VistA.js includes, but is not limited to, the most recently installed version of the VSA VistA.js platform is not functioning as expected.

## Back-out Risks

On the target **Web** Linux server, when starting and stopping the **Web** VSA Server process, users of the VSA VistA.js RPC Wizard and VSA VistA.js API Browser applications will be affected and data will be lost.

On the target **VistA** or **Federator** Linux severs, all **VistA** VSA Server-bound transactions, and all transactions through the **Federator** VSA Server will be affected, and data will be lost.

As such, all traffic *must* be routed to alternate **VistA, Federator, or Web** VSA Servers *before* starting or stopping these servers.

## Authority for Back-out

Systems Administrators with the requisite permissions to modify server settings and install/remove software.

## Back-out Procedure

The back-out procedure will vary based on need. However, if a running instance of the VSA VistA.js platform *must* be removed, the following steps *must* be used to stop all servers and remove the associated files (as shown in the steps that follow).

### Stopping the VSA VistA.js Servers

#### Maestro Server

1. On the connected **VistA**, **Federator**, and **Web** Linux servers, connect to an alternate Maestro server:

**cd /opt/vsa/vjs-environment**

**./setup.sh -i *<New-Maestro-Linux-Server-Fully-Qualified-Domain-Name>***

1. On the target Maestro Linux server:

**cd /opt/vsa/MaestroInstaller/scripts**

**./stop.sh**

#### Web VSA Server

1. On the Web Linux server, alert all VSA VistA.js RPC Wizard and VSA VistA.js API Browser application users that the **Web** VSA Server will be stopped.
2. Stop the **Web** VSA Server process:

**ps -ef | grep start**

3097 2900 0 09:20 pts/0 00:00:00 bash ./start-webux.sh

3098 3097 0 09:20 pts/0 00:00:03 node node\_modules/vsa-server/bin/vsa-server start

1. Kill the Web VSA Server process and its parent:

**kill 3098**

**kill 3097**

#### VistA VSA Server

1. Reroute the traffic to an alternate **VistA** VSA Server.
2. Configure the **Federator** VSA Server to connect to the new **VistA** VSA Server.
3. Stop the **VistA** VSA Server process:

**ps -ef | grep start**

3330 2900 0 09:32 pts/0 00:00:00 bash ./start-baypines-vista.sh

3331 3330 0 09:32 pts/0 00:00:01 node node\_modules/vsa-server/bin/vsa-server start

1. Kill the VistA VSA Server process and its parent:

**kill 3331**

**kill 3330**

#### Federator VSA Server

1. Reroute the traffic to an alternate **Federator** VSA Server.
2. Configure a new **Federator** VSA Server to connect to the required **VistA** VSA Servers.
3. Stop the **Federator** VSA Server process:

**ps -ef | grep start**

3090 2900 0 09:20 pts/0 00:00:00 bash ./start-vjs-federator.sh

3091 3090 0 09:20 pts/0 00:00:01 node node\_modules/vsa-server/bin/vsa-server start

1. Kill the Federator VSA Server process and its parent:

**kill 3091**

**kill 3090**

### Removing all VSA VistA.js Platform Components

#### Maestro Server

1. On the Maestro Linux server, remove the Maestro server and installation file from its installation directory:

**cd /opt/vsa**

**cd MaestroInstaller**

**./maestro-installer uninstall**

**cd ..**

**rm -rf MaestroInstaller**

**rm vsa-vistajs-maestro-3.0.n.tgz**

#### VistA, Federator, and Web VSA Servers

1. On the VistA, Federator, and/or Web Linux server, remove the associated **VistA, Federator,** and/or **Web** VSA Server directories:

**cd /srv/vista/*<scd>*/user/vistajs/**

**rm -rf *<namespace>*-vista *or***

**rm -rf vjs-federator *or***

**rm -rf web**

1. Remove the VSA VistA.js environment directory and installation file from its installation directory:

**cd /opt/vsa**

**rm -rf vjs-environment**

**rm vsa-vistajs-environment-3.0*.n.*tgz**

# Rollback Procedure

Rollback pertains to data.

## Rollback Considerations

If VSA VistA.js-related data is removed from Caché, the associated routines will need to be re-added as per “[Appendix B—Importing the MUMPS Components](#_Appendix_C:_Importing_1).” In addition, any RESTified RPCs that are stored in Caché will be lost.

## Rollback Criteria

Backing out data associated with an installed version of VSA VistA.js includes, but is not limited to, reverting to a known good Caché configuration that supports the VSA VistA.js platform.

## Rollback Risks

If RESTified RPCs are removed for the Caché database, those associated routines will fail to function when called from their associated **VistA** and **Federator** VSA Servers.

## Authority for Rollback

Systems Administrators with the requisite permissions to modify server settings and install/remove software.

## Rollback Procedure

If the removal of database-specific VSA VistA.js components is desired, the following routines and data dictionaries can be safely removed from the associated VistA system/database without causing harm to other applications.

1. Routines: All routines in the XSA namespace can be deleted
2. Remote Procedures: All RPCs in the XSA namespace can be deleted
3. Data Dictionaries: The XSA DEFINITIONS file (17.01) can be deleted

 **REF:** For more information on the XSA namespace, see the *VSA VistA.js Technical Manual*.

# Approval Signatures

REVIEW DATE:

SCRIBE: Apex Data Solutions

Signed:

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Integrated Project Team (IPT) Chair Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Business Sponsor Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Information Technology (IT) Program Manager Date

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Project Manager Date

# Appendix A – Sample Server Configuration Files

This section includes sample configuration files taken from the testing environment that were generated via “vsa-gen config.” They will be similar in IOC, PPR, and PRD, assuming:

* The target installation directory of each VSA VistA.js VSA Servers (**VistA**, **Federator**, and **Web**) is /srv/vista/*<scd>*/user/vistajs.
* The database configuration files reside in /srv/vista/*<scd>*/cache/<*instance*>/mgr.
* This configuration information is captured when running “vsa-gen config” to configure a new VSA VistA.js VSA Server of type **VistA**, **Federator**, or **Web**. If the values in these sample files differ than what is required in IOC (Pre-Prod and/or Prod), the requisite configuration information should be provided interactively when issuing “vsa-gen config.” These files can also be edited by hand after running “vsa-gen config” (and *before* running “vsa-gen server”) as required.

## VistA VSA Server

### baypines-config.js

The figure below reflects the contents of a sample baypines-config.js file.

Figure : Sample VistA VSA Server Configuration File: baypines-config.js

module.exports = {

"platform" : "node",

"framework" : "vsa",

"profile" : "vista",

"instanceName" : "baypines-vista",

"environment" : "production",

"port" : "xxxx",

"websocketsPort" : "",

"useSSL" : "Yes",

"useDefaultCertificates" : "Yes",

"sslKey" : "",

"sslCert" : "",

"workers" : "1",

"appsDirectory" : "",

"servicesDirectory" : "",

"webRootDirectory" : "",

"database\_type" : "cache",

"database\_path" : "/opt/cache/mgr",

"database\_namespace": "BAYPINES",

"database\_user" : "b86fbd7ad6ae6a",

"database\_password" : "b465b7",

"cache\_node\_path" : "",

"shared\_store\_path" : "",

"shared\_store\_ip" : "",

"shared\_store\_port" : "",

"shared\_store\_namespace" : "",

"shared\_store\_username" : "",

"shared\_store\_password" : ""

};

## Federator VSA Server

### Federator Config File: federator-config.js

The figure below reflects the contents of a sample federator-config.js file.

Figure : Sample Federator VSA Server Configuration File: federator-config.js

module.exports = {

"platform" : "node",

"framework" : "vsa",

"profile" : "federator",

"instanceName" : "vjs-federator",

"environment" : "production",

"port" : "xxxx",

"websocketsPort" : "",

"useSSL" : "Yes",

"useDefaultCertificates" : "Yes",

"sslKey" : "",

"sslCert" : "",

"workers" : "1",

"appsDirectory" : "",

"servicesDirectory" : "",

"webRootDirectory" : "",

"database\_type" : "",

"database\_path" : "",

"database\_namespace": "",

"database\_user" : "",

"database\_password" : "",

"cache\_node\_path" : "",

"shared\_store\_path" : "/opt/cache/mgr",

"shared\_store\_ip" : "",

"shared\_store\_port" : "",

"shared\_store\_namespace" : "FEDERATOR",

"shared\_store\_username" : "b86fbd7ad6ae6a",

"shared\_store\_password" : "b465b7"

};

### VistA VSA Server Router File: vjs-vista-router.js

The figure on the following page reflects the contents of a sample vjs-vista-router.js file.

Figure : Sample Federator VSA Server Router File: vjs-vista-router.js

"use strict";

exports.default = {

"vjs-vista-router": function(api){

return {

name: 'VistA Router',

description: 'This route handler manages calls to VistA instances',

rootResource: 'vistas',

version: '0.0.1',

config: {

// Partial results disabled until client libraries handle them

partialResponses: {

enabled: false,

maximumResponseTime: 1000,

cacheExpiryTimeMS: "10m"

},

patientVisitedCache: {

expiryHour24: 21,

expiryMinute: 23

},

vistas: {

endpoints: {

"baypines": {

"host": "127.0.0.1",

"port": xxxx,

"secretKey": "federator-secret-key",

"clientID": "federator-access-id",

"stationID": 516,

"siteName": "BAY PINES VAMC"

},

"HALEY": {

"host": "127.0.0.1",

"port": xxxx,

"secretKey": "federator-secret-key",

"clientID": "federator-access-id",

"stationID": 673,

"siteName": "TAMPA (JAH VAH)"

},

"westpalm": {

"host": "127.0.0.1",

"port": xxxx,

"secretKey": "federator-secret-key",

"clientID": "federator-access-id",

"stationID": 548,

"siteName": "WEST PALM BEACH VAMC"

}

},

"scopes": {

"all": ["baypines", "HALEY", "westpalm"],

"region-1": ["baypines", "HALEY"],

"region-2": ["westpalm"]

}

}

}

}

}

};

## Web VSA Server

### Web VSA Server Config File: web-config.js

The figure below reflects the contents of a sample web-config.js file.

Figure : Sample Web VSA Server Configuration File: web-config.js

module.exports = {

"platform" : "node",

"framework" : "vsa",

"profile" : "web",

"instanceName" : "web",

"environment" : "production",

"port" : "xxxx",

"websocketsPort" : "",

"useSSL" : "Yes",

"useDefaultCertificates" : "Yes",

"sslKey" : "",

"sslCert" : "",

"workers" : "1",

"appsDirectory" : "",

"servicesDirectory" : "",

"webRootDirectory" : "",

"database\_type" : "",

"database\_path" : "",

"database\_namespace": "",

"database\_user" : "",

"database\_password" : "",

"cache\_node\_path" : "/opt/cache/bin",

"shared\_store\_path" : "",

"shared\_store\_ip" : "",

"shared\_store\_port" : "",

"shared\_store\_namespace" : "",

"shared\_store\_username" : "",

"shared\_store\_password" : ""

};

### API Browser Config File: vsa-api-browser-application.js

The figure below reflects the contents of a sample vsa-api-browser-application.js file.

Figure : Sample Web VSA Server Configuration File: API Browser

"use strict";

exports.default = {

"vsa-api-browser-application": function(api){

return {

name : "vsa-api-browser-application",

title : "API Definition Browser",

description : "Permits viewing of API swagger definitions.",

basepath : "vsa-api-browser",

version : "0.0.1",

config : {

federationServers: [

{ name: "vjs-federator", host: '127.0.0.1', port: 'xxxx' }

]

}

}

}

};

### RPC Wizard Config File: vsa-rpc-wizard-application.js

The figure below reflects the contents of a sample vsa-rpc-wizard-application.js file.

Figure : Sample Web VSA Server Configuration File: RPC Wizard

"use strict";

exports.default = {

"vsa-rpc-wizard-application": function(api){

return {

name : "vsa-rpc-wizard-application",

title : "VSA RPC Wizard",

description : "An application for restification of VistA RPCs",

basepath : "vsa-rpc-wizard",

version : "0.0.1",

config : {

federationServers: [

{ name: "vjs-federator", host: '127.0.0.1', port: '8083' }

]

}

}

}

};

# Appendix B – Importing the MUMPS Components

The VSA M components distributed by patch XSA\_1\_0.KID provide supporting extrinsic functions called by the VSA VistA.js platform through the Caché Node.js (cache.node) interface. All functions are intended to be called only by VSA VistA.js. No other software should reference any routines in the XSA namespace.

 **NOTE:** VistA Kernel patch XU\*8.0\*659 must be installed prior to installing the XSA 1.0 KIDS patch at each target VistA site.

## XSA KIDS Patch Components

Table : Patch Components

| Component | Description/Value |
| --- | --- |
| File Name | VSA SWAGGER DEFINITION |
| File Number | 17.01 |
| New/Modified/Deleted: | New |
| Forms Associated: | N/A |
| Functions Associated: | N/A |
| HL Logical Link: | N/A |
| HL7 Application Parameters: | N/A |
| HLO Application Registry: | N/A |
| Mail Groups Associated: | N/A |
| Options Associated: | N/A |
| Parameter Definitions: | N/A |
| Protocols Associated: | N/A |
| Remote Procedure Calls: | N/A |
| Security Keys Associated: | N/A |
| Templates Associated: | N/A |
| Additional Information: | N/A |
| New Service Requests (NSRs): | N/A |
| Patient Safety Issues (PSIs): | N/A |
| Remedy Ticket(s) & Overviews: | N/A |
| Test Sites: | TBD |

## Software and Documentation Retrieval Instructions

Software being released as a host file and/or documentation describing the new functionality introduced by this patch are available for download. The preferred method is to download the files from download.vista.med.DNS.

This transmits the files from the first available server. Sites may also elect to retrieve files directly from a specific server. Sites can retrieve the software and/or documentation directly using Secure File Transfer Protocol (SFTP) from the ANONYMOUS.SOFTWARE directory at the following Office of Information and Technology (OI&T) Field Offices:

Table : Download Servers

| Site Location | Download Server |
| --- | --- |
| Albany: | DNS .med.DNS |
| Hines: | DNS .med.DNS |
| Salt Lake City: | DNS .med.DNS |

The documentation will be in the form of Adobe Acrobat files.

### Pre-Installation Instructions

This patch can be installed with users on the system. This patch should take about 20 minutes to install.

The KIDS files can be located in the following directory that was created when generating a **VistA** VSA Server:

/srv/vista/*<scd>*/user/vistajs/*<namespace>*-vista/mcode/XSA\_1\_0.KID

Optionally, this file can be copied to another location to make it easier to locate during installation, such as /tmp.

### Installation Instructions

1. Use the Load a Distribution option on the Kernel Installation and Distribution System (KIDS) installation menu. When prompted for a host file enter ***<directory>*: XSA\_1\_0.KID**, where ***<directory>*** is the local directory where the host file was copied, for example: /tmp /XSA\_1\_0.KID.
2. From the Kernel Installation and Distribution System Menu, select the Installation Menu. From this menu, you can elect to use the following options. When prompted for the INSTALL enter the patch ID (**XSA 1.0**):
3. Backup a Transport Global—This option creates a backup message of any routines exported with this patch. It does not back up any other changes, such as DD's or templates.
4. Compare Transport Global to Current System—This option allows you to view all changes that will be made when this patch is installed. It compares all components of this patch (e.g., routines, DD's, templates, etc.).
5. Verify Checksums in Transport Global—This option allows you to ensure the integrity of the routines that are in the transport global.
6. From the Installation Menu, select the Install Package(s) option and choose the patch to install. Enter **XSA 1.0**.
7. At the “Want KIDS to Rebuild Menu Trees Upon Completion of Install? NO//” prompt, enter **NO**.
8. At the “Want KIDS to INHIBIT LOGONs during the install? NO//” prompt, enter **NO**.
9. At the “Want to DISABLE Scheduled Options, Menu Options, and Protocols? NO//” prompt, enter **NO**.
10. If prompted “Delay Install (Minutes): (0 - 60): 0//”, enter **0**.

### Post-Installation Instructions

N/A

## XSA Routines

As of VSA VistA.js release 3.0.8, routines housed in an XSAPXROUTINES.xml file must be imported into each FEDERATOR namespace.

 **NOTE:** If a “FEDERATOR” namespace does not exist in the Federator’s Caché instance, one must be created before the XSA routines can be imported, and before a Federator VSA Server can be started.

The XSAPXROUTINES.xml file is located in the mcode directory when a new VSA Server of profile type “vista” or “federator” is generated (created) via the “vsa-gen server” command.

This file can be copied to any directory where it will be convenient to importing it via the Caché Admin Console, such as the /tmp directory.

### Importing the XSAPXROUTINES.xml File

1. Copy the XSAPXROUTINES.xml file to /tmp, renaming the extension to “.ro.”

VistA Linux servers:

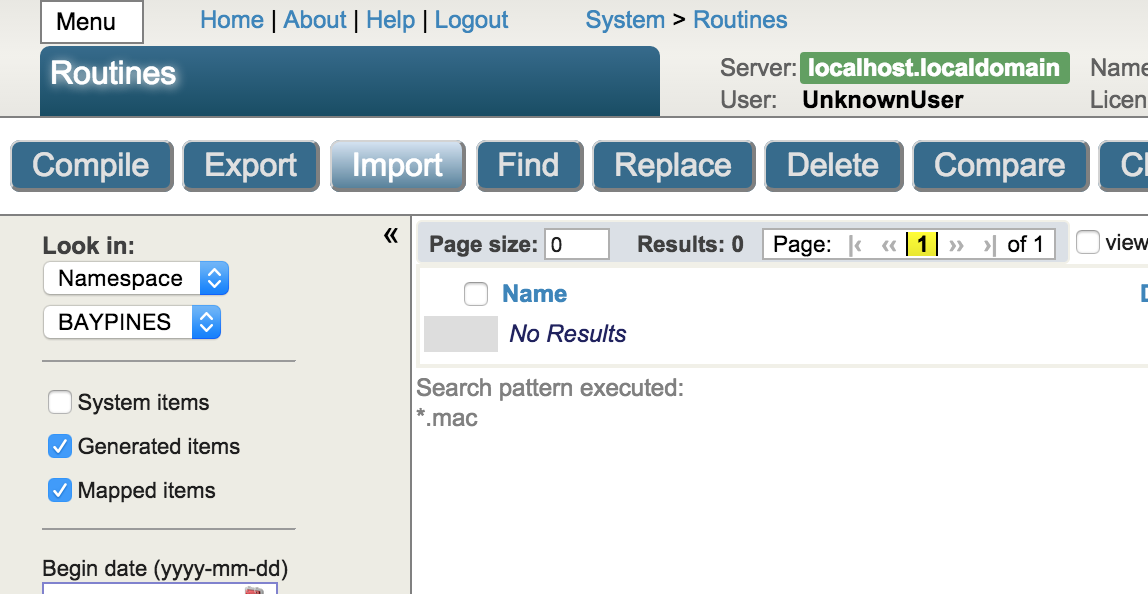
**scudo cp /srv/vista/<*scd*>/user/vistajs/<*namespace*>-vista/mcode/ XSAPXROUTINES.xml /tmp/XSAPXROUTINES.ro**

Federator Linux servers:

**sudo cp /app/vsa/user/vistajs/vjs-federator/mcode/XSAPXROUTINES.xml /tmp/XSAPXROUTINES.ro**

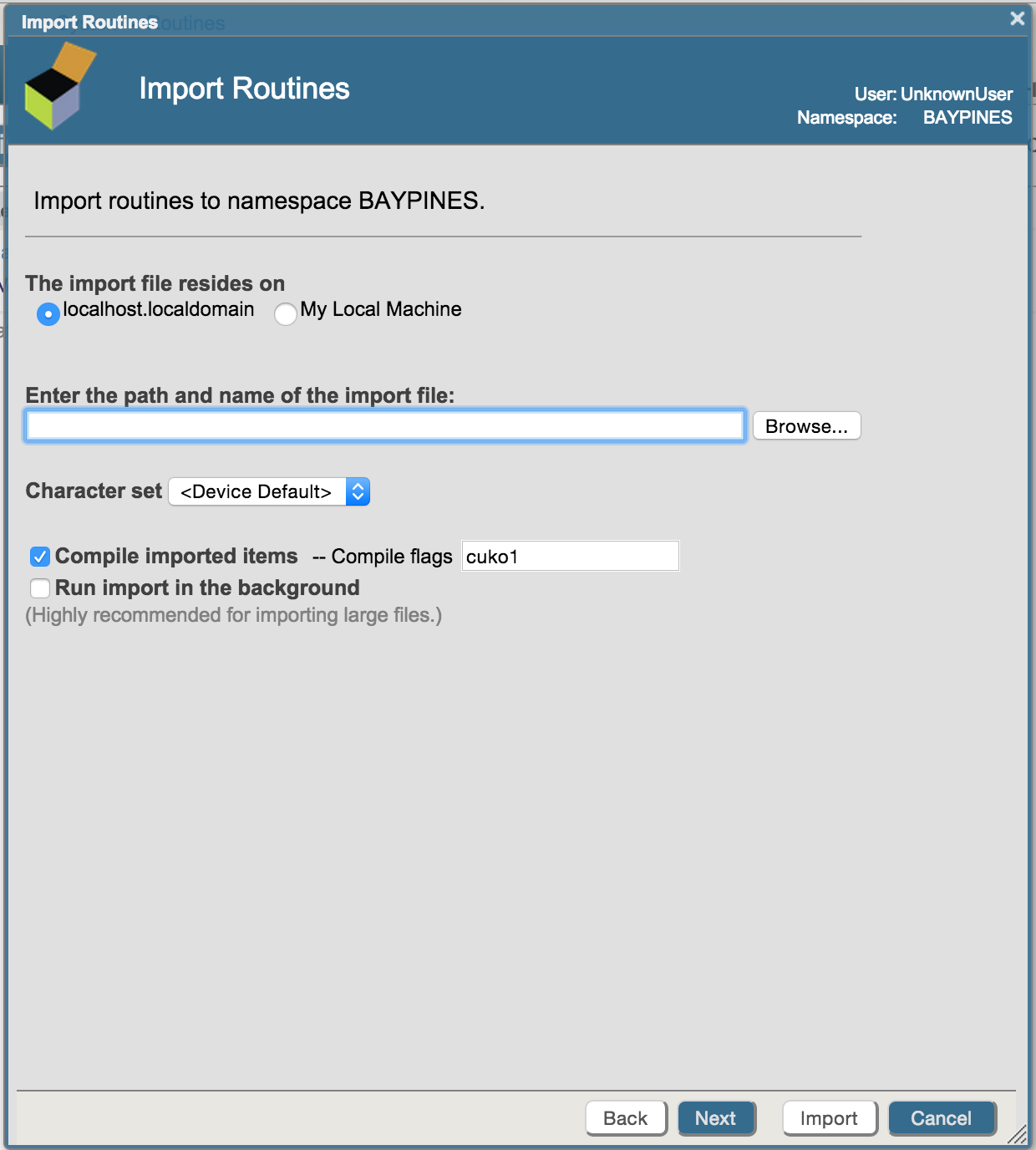
1. In the Caché administration console, select Menu > View Routines.
2. Leave the **Look in** drop-down as the default (Namespace) and select the target VistA namespace from the drop-down and click **Import**.

Figure : View Routines - Import



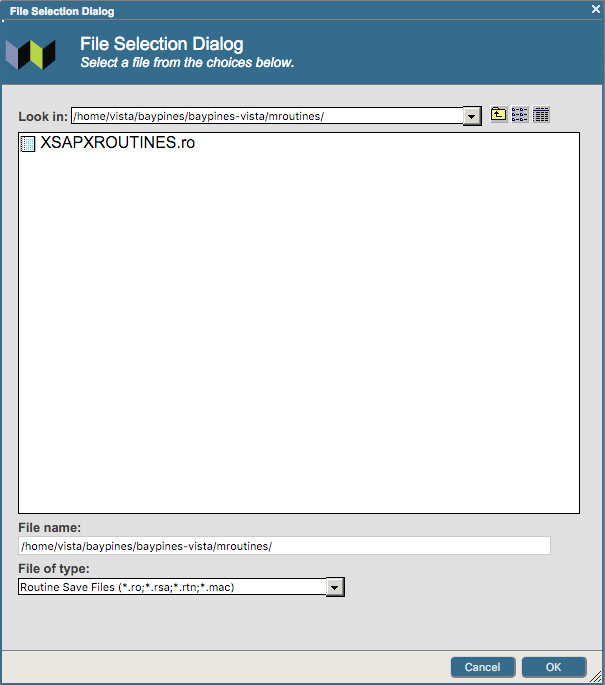
1. Click **Browse.**

Figure : Browse Import Routines



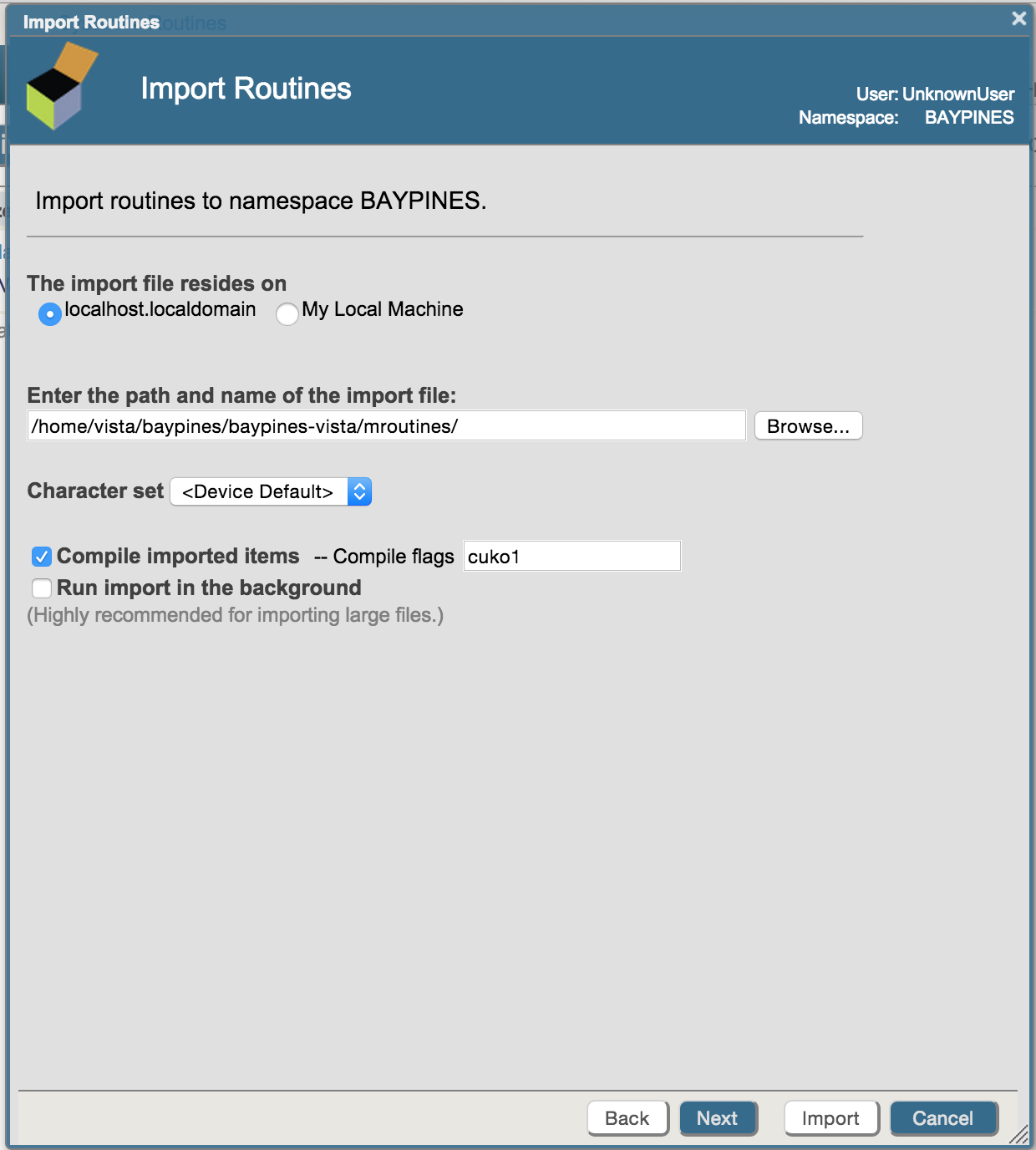
1. Using the **File Selection** dialog, navigate to where the XSAPXROUTINES.ro file is located.
2. Select the **XSAPXROUTINES.ro** file and click **OK**.

Figure : Select XSAPXROUTINES.ro



1. Click **Next**.

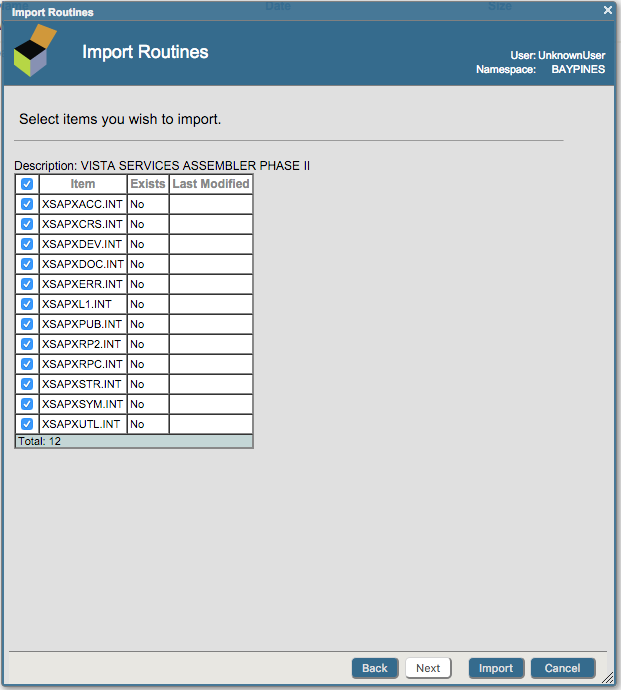
Figure : Import Routines - Next Screen



1. Review the items being imported and click **Import**.

NOTE: The number of routines and their names may vary from release to release. The screenshot below is for illustrative purposes only.

Figure : Review Routines to Import



1. The **Import result** log will be displayed. Click **Done**.

# Appendix C – Installation Scripts

This section lists the scripts used during the installation of the VSA VistA.js platform.

## Maestro

“Maestro” is the name given to the installer for VSA VistA.js that creates an instance of a Node Package Manager (NPM) package repository. This NPM package repository is implemented on the open source Sinopia project, and provides a given server with the ability to pull down the VSA components required when building new **VistA**, **Federator**, or **Web** servers.

Table : Maestro Scripts

| Script | Function |
| --- | --- |
| maestro-installer | Installs the Maestro server. During installation, the Maestro server is automatically started. |
| start.sh | Starts the Maestro server. |
| stop.sh | Stops the Maestro server. |
| sinopia.init.d | This file can be moved to the /etc./init.d directory by a System Administrator to run as a service when the server is booted.  For example:  mv sinopia.init.d /etc/init.d/sinopia |

## vjs-environment

The vjs-environment contains supporting environment variables and scripts.

Table : vjs-environment Scripts

| Script | Function |
| --- | --- |
| setup.sh | Establishes communication with the target Maestro server. |
| use\_node.sh | Adds the included VSA VistA.js Node.js software in the installing user’s path. |
| npm install -g vista-services-assembler | Installs VistA Services Assembler “globally” on the target server, so new **VistA**, **Federator**, and/or **Web** VSA Servers can be configured and created via vsa-gen. |
| vsa-gen | The main VistA Services Assembler “assembly” command that allows new **VistA**, **Federator**, and **Web** VSA Servers to be created on-demand. |
| vsa-gen config | Configures a **VistA**, **Federator**, or **Web** VSA Server, allowing the installing user to specify information about the type of server desired, and information about the machine on which that server will run. This command generates a config.js file for use by “vsa-gen server.” |
| vsa-gen server | Creates a server that has been configured via “vsa-gen config,” using the specified config.js file to build the new server. |