Medication Order Check Healthcare Application (MOCHA) Server 4.0

Installation Guide

June 2024

Department of Veterans Affairs (VA)

Office of Information and Technology (OIT)

Revision History

Date	Version	Description	Author
06/28/2024	3.0	PREM*4*1 • Updated the VIP • Updated FDB_DIF to FDB45_DIF • Updated Figures: 1, 2, 3, 8, 10, and 11 • Updated Tables: 1, 4, 5, 6 • Updated versions of WebLogic, Java, Oracle, and RHELS • Updated section 1.2 • Updated section 3.2.2 • Updated section 3.3.2 • Updated section 3.3.3 • Updated section 3.3.4 • Updated section 4.2 • Updated section 4.5 • Updated section 4.6 • Added section 4.7 • Updated section 4.8 • Updated section 4.9 • Updated section 4.9.1.1 • Updated section 4.9.2 • Updated section 7.2 • Added section 7.3 • Updated Title page, Revision History, and Footers	Liberty IT Solutions
07/06/2022	2.1	PREM*3*4: • Updated section **4.9.2** and the version references in step 1 for the jar files to 2.17.1 • Updated embedded MOCHA Server Specifications, provided by AITC, in section **3.3.2** • Updated Title page, Revision History, Table of Contents, and Footers	Booz Allen Hamilton
10/22/2020	2.0	PREM*3*2: • Updated site information in section 3.2.2 • Updated log4j2 in section 4.8.2 • Updated the sample log4j2.xml file in section 7.1 • Updated Title page, Revision History, and Footers	Liberty ITS
3/13/17	1.1	Added Section 4.8.3 ESAPI, Section 7.2 Appendix B, Section 7.3 Appendix C; Updated Section 5 Back-Out Procedure	REDACTED
1/31/17	1.0	Initial draft of Deployment Plan and Installation Guide	REDACTED

Artifact Rationale

The Veteran-focused Integrated Process (VIP) 4.0 Guide indicates the VA Product (Line) Accountability and Reporting System (VA PARS) reporting tool requires a Gateway Review that will move the project from the Planning Stage and to the Build Stage and will require Release Approval before deploying into production. The Product Line Manager will ensure necessary documents are made available for the release approval process.

Table of Contents

List of Tables

List of Figures

1 Introduction

This document describes how to deploy and install the Pharmacy Reengineering (PRE) Medication Order Check Healthcare Application (MOCHA) Server 4.0, as well as how to back-out the product and rollback to a previous version or data set. This document is a companion to the project charter and management plan for this effort. In cases where a non-developed Commercial Off-The-Shelf (COTS) product is being installed, the vendor provided User and Installation Guide may be used, but the Back-Out Recovery strategy still needs to be included in this document.

1.1 Purpose

The purpose of this plan is to provide a single, common document that describes how, when, where, and to whom the MOCHA Server 4.0 will be deployed and installed, as well as how it is to be backed out and rolled back, if necessary. The plan also identifies resources, communications plan, and rollout schedule. Specific instructions for installation, back-out, and rollback are included in this document.

1.2 Dependencies

MOCHA Server requires the administration of the Oracle First Databank (FDB)-MedKnowledge database and Oracle WebLogic application server.

Table 1: Dependencies

System Name	Location	Description
Pharmacy Enterprise Customization System (PECS) FDB MedKnowledge	AITC	• The FDB MedKnowledge (Formerly known as FDB-DIF) database contains standard drug data and VA Customization for pharmaceutical drug concepts. MOCHA Server maintains its own copy of FDB MedKnowledge that is copy from the PECS instance.
Web Application Server	AITC	• Target location for MOCHA application
MOCHA Build Ear file		• Up-to-date build of MOCHA application from Jenkins Release branch
Oracle Database	AITC	• Exported FDB Fwk v4.5 file • Import / Export Materialized View Scripts

1.3 Constraints

Not Applicable (N/A)

2 Roles and Responsibilities

Table 2: Deployment, Installation, Back-out, and Rollback Contact List

Type of Contact	Contact Name	Department	Phone #	Email address
Austin Information Technology Center (AITC) Operations	PRE-Distribution List	AITC Build/ Application Managers	(512) 981-4792	VA IT SDE EO EAS MOC SUSTAINMENT REDACTED
AITC Operations	REDACTED	WebLogic Administration Group	REDACTED	REDACTED
AITC Operations	REDACTED	Database Administration Group	REDACTED	REDACTED
PRE PD Operations Team	REDACTED	PRE-program WebLogic Administration Group	REDACTED	REDACTED
PRE PD Operations Team	REDACTED	PRE-program Database Administration Group	REDACTED	REDACTED
MOCHA Server Development Lead	REDACTED	MOCHA Server Development Group	REDACTED	REDACTED

Table 3: Deployment, Installation, Back-out, and Rollback Roles and Responsibilities

ID	Team	Phase / Role	Tasks	Project Phase (See Schedule)
	Enterprise Operations/OIT	Deployment	Plan and schedule deployment (including orchestration with vendors)	
	OIT	Deployment	Determine and document the roles and responsibilities of those involved in the deployment.	
	Enterprise Operations	Deployment	Test for operational readiness	
	Enterprise Operations	Deployment	Execute deployment	
	Enterprise Operations	Installation	Plan and schedule installation	
	OIT	Installation	Ensure authority to operate and that certificate authority security documentation is in place	
	Enterprise Operations	Installation	Validate through facility POC to ensure that IT equipment has been accepted using asset inventory processes	
	OIT	Installations	Coordinate training	
	OIT	Back-out	Confirm availability of back-out instructions and back-out strategy (what are the criteria that trigger a back-out)	
	Enterprise Operations/OIT	Post Deployment	Hardware, Software and System Support	

3 Deployment

The deployment is scheduled via AITC on the Enterprise Operation (EO) calendar.

3.1 Timeline

The software release process will take at least 2 hours. This duration includes the time for deployment and testing to ensure that it is in conformance to the release requirements. There will not be any interruption of service.

Refer to the Project schedule for scheduling details.

3.2 Site Readiness Assessment

This section discusses the locations that will receive the MOCHA deployment.

The deployment will be coordinated with AITC in accordance with Veterans Affairs (VA) standard release procedures. For AITC installation scheduling will be done via the EO release calendar. Requests for a release will be made through the AITC Request for Change Order

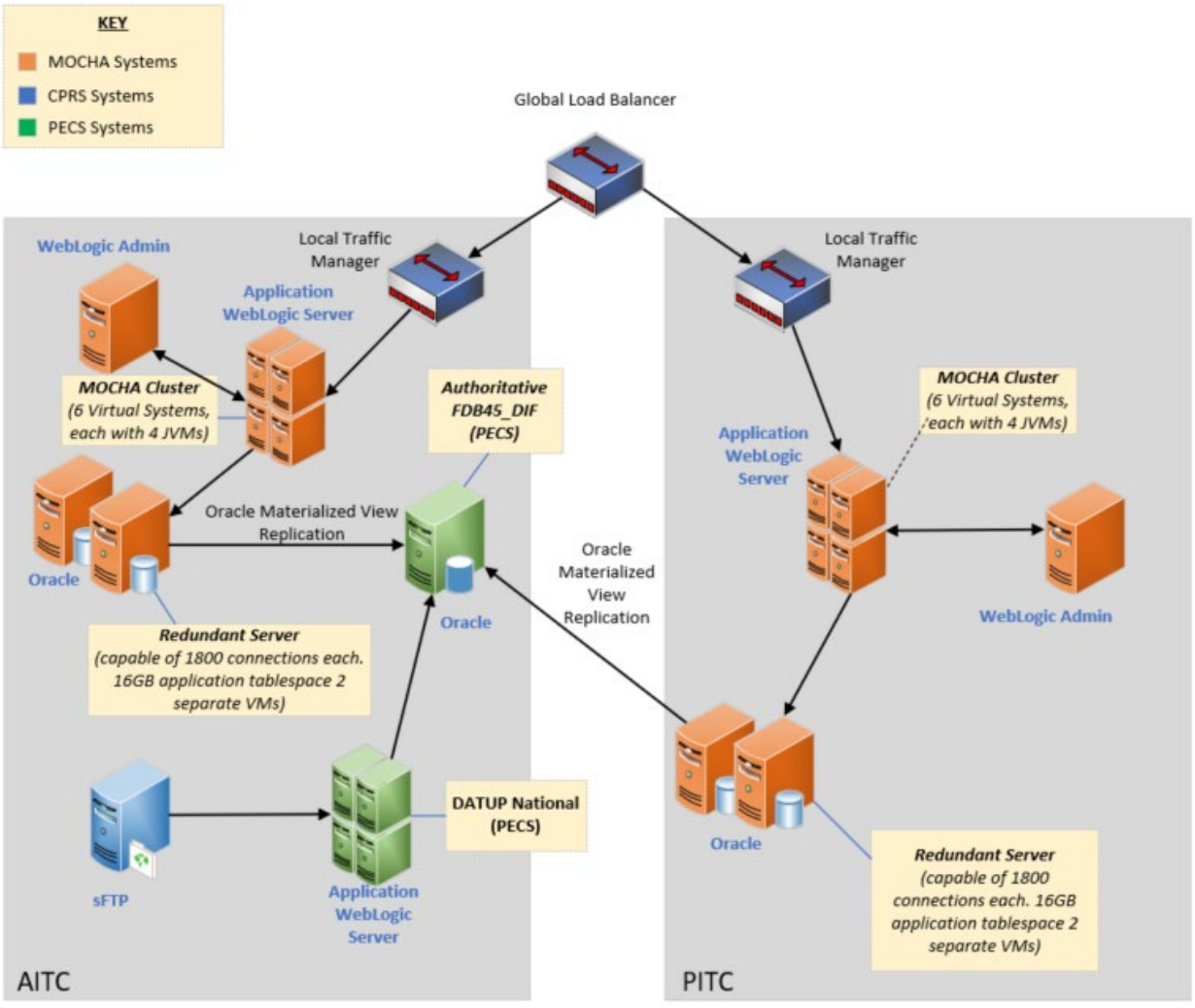
(RFCO) Form and submitted at least 2 days prior to the deployment date. This process will ensure that resources are available to deploy changes. All changes are deployed and tested in the lower environments (development, Computer Emergency Response Team (CERT), and Pre-Prod) before being deployed to production.

The next sections discuss the locations that will receive the MOCHA Server 4.0 deployment.

3.2.1 Deployment Topology (Targeted Architecture)

The diagram below shows the target deployment topology of MOCHA Server in the AITC EO Cloud.

Figure 1: MOCHA Deployment



3.2.2 Site Information (Locations, Deployment Recipients)

The MOCHA Server 4.0 deployment is planned to take place at the VA Austin Information Technology Center (AITC) utilizing the Enterprise Operations (EO) Cloud. MOCHA Server will be tested by the three test sites through the MOCHA application.

Test Sites for the MOCHA Server release:

- VA Central Western Massachusetts Healthcare System (Northampton), Leeds, MA
- Denver Veterans Administration Medical Center (VAMC), Denver, CO
- Shreveport VAMC, Shreveport, LA

3.2.3 Site Preparation

Not Applicable (N/A)

3.3 Resources

The following section describes the hardware, software, facilities required for the deployment and installation of MOCHA Server.

Additional information may be found in the MOCHA Server Production Operations Manual (POM) located in the MOCHA Server Documentation stream under the PHARM project area in Return to Clinic (RTC).

3.3.1 Facility Specifics

N/A

3.3.2 Hardware

The detailed hardware configuration for MOCHA Server is contained in the RHEL 7 specifications.

3.3.3 Software

The following table describes software specifications required at each site prior to deployment.

Table 4: Software Specifications

Required Software	Make	Version	Configuration	Manufacturer	Other
WebLogic	Application Server	12.2.1.4.1	Cluster	Oracle	
Java	Application Server Runtime	1.8.0_391	N/A	Oracle	
Oracle Database	Database	19c	Stand-alone	Oracle	
RHEL	Operating System	7.x	OS	RedHat	

Please see the Roles and Responsibilities table in Section 2 above for details about who is responsible for preparing the site to meet these software specifications.

3.3.4 Communications

The stakeholders are informed about the successful release of the product or other information pertaining to the release via an AITC managed Automated Notification Reporting (ANR) email distribution list. Change Orders and deployment calendars are also updated with the deployment status.

- Notify business owner of production deployment.
- The Release Manager will schedule activities and identify the required personnel for each activity.
- Meetings will be scheduled for deployment personnel to work through the deployment steps.

3.3.4.1 Deployment/Installation/Back-Out Checklist

The Deployment/Installation/Back-Out Checklist will be completed and maintained by the AITC EO team and will align with the EO calendar.

Table 5: Deployment/Installation/Back-Out Checklist

Activity	Day	Time	Individual who completed task
Deploy	TBD	TBD	Infrastructure Operations (IO)
Install	TBD	TBD	IO
Back-Out	TBD	TBD	IO

4 Installation

4.1 Pre-installation and System Requirements

The table below details the environment necessary to deploy MOCHA Server.

Table 6: Software Specifications

Required Software	Make	Version	Configuration	Manufacturer	Other
WebLogic	Application Server	12.2.1.4.1	Cluster	Oracle	
Java	Application Server Runtime	1.8.0_391	N/A	Oracle	

Required Software	Make	Version	Configuration	Manufacturer	Other
Oracle Database	Database	19c	Stand-alone	Oracle	
RHEL	Operating System	7.x	OS	RedHat	

4.2 Platform Installation and Preparation

The MOCHA v3.0 platform will be used. AITC will follow EO standard operating procedures to install and prepare the environment prior to the installation of MOCHA Server 4.0.

4.3 Download and Extract Files

Software configuration and deployment artifacts will be provided by the Product Development (PD) team with the Request for Change Order (RFCO). The file(s) will be provided electronically copied to an appropriate location and the path communicated to the EO team.

4.4 Database Creation

The installation of scripts noted in the next section assumes that Oracle 19c Database Server is configured and running. Proper installation of the Oracle Relational Database Management System (RDBMS) is one in which the Oracle Universal Installer and DBCA were used to perform an error-free installation and a general-purpose instance was created. A properly configured Oracle RDBMS is one in which the associated Oracle application development and configuration tools, namely Structured Query Language (SQL)*Plus, can be used to connect to the instance through a Transparent Network Substrate alias.

The installation of scripts noted in the next section will create a set of materialized view artifacts in order to enable FDB data to be read from another database/environment. Prior to this solution being completely implemented, there must also be a database link created to this database which houses FDB45_DIF schema objects.

Execute the following scripts on the target database which contains the FDB related schema objects to create the materialized view artifacts and permissions needed by any sourcing database. After executing the scripts noted below, the FDB data will be accessed via database link created for that purpose.

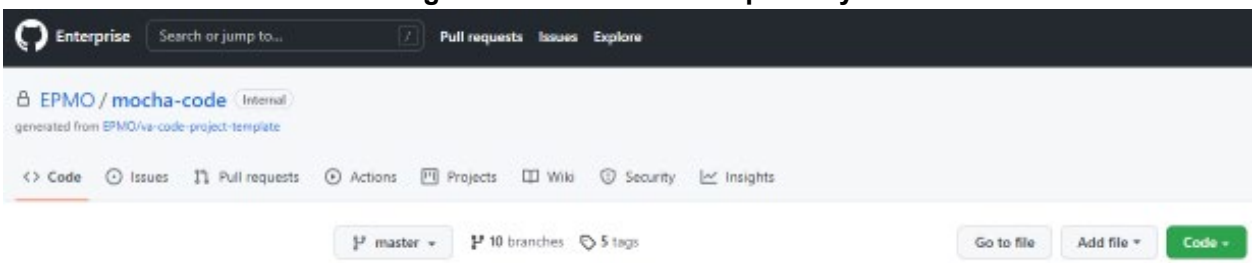
On the sourcing database(s) through the database link and its authorized account, you may now access the FDB data through the aforementioned database link and account.

4.5 Installation Scripts

Installation scripts needed for the database installation, as well as the procedure on how to set up FDB Fwk v4.5, are provided in VA GitHub EC, located in the mocha-code repository docs folder. DBA should make sure to disable the archive mode before setting up 4.5 schema and then enable it after schema import.

The selection of data from the FDB data housed in the database using the authorized account specified using the database link is now enabled.

Figure 2: MOCHA Code Repository



4.6 Cron Scripts

N/A

4.7 Oracle Scheduler

The MOCHA Materialized Views refresh from the Oracle Scheduler. The MOCHA Server pulls the data from PECS incrementally.

4.8 Access Requirements and Skills Needed for the Installation

Linux System Administrator will need:

- Access to the Linux console of the server where MOCHA WebLogic is running
- Access to the WebLogic web-based Console
- Access to the location indicated in section 4.5 Installation Scripts

Database Administrator will need:

- Access to the Linux console of the server where FDB Oracle Database is running
- Access to the location indicated in section 4.5 Installation Scripts

4.9 Installation Procedure

The installation instructions found within this guide are intended to be performed on a clean installation of WebLogic 12.2.1.4.1, with a separate managed server to act as the Deployment Server. For details on completing the installation of the following items, please refer to each item's installation and configuration documentation supplied by Oracle.

For successful deployment of the MOCHA Server software, the following assumptions must be met:

- The Deployment Server is configured and running.
- WebLogic is configured to run with the Java™ Standard Edition Development Kit, Version 1.8+.
- Access to the WebLogic console is by means of any valid administrative username and password.

- The proper Oracle 19c database driver libraries for the chosen deployment environment are present on the class path for the respective Deployment Servers.
- Red Hat Enterprise Linux 7.x operating system is properly installed.
- Domain Name Server (DNS) resolution is configured for the MOCHA Server.
- The installation instructions are followed in the order that the sections are presented within this Installation Guide.
- A PECS database is available for replication.
- Cron job is run on the PECS database server to load MOCHA Materialized Views.

4.9.1 801WebLogic Installation Instructions

The following sections detail the steps required to configure and deploy MOCHA Server onto WebLogic.

4.9.1.1 WebLogic Server Startup Configuration

MOCHA Server requires additional arguments added to the WebLogic Server's Server Start properties. This section details the steps to add the arguments to the server

1. Open and log into the WebLogic console, using an administrative username and password. The WebLogic console is located at: `http://<Deployment Machine>:<Deployment Server Port>/console`.
2. Within the `Domain Structure` panel found in the left column of the WebLogic console, click on the `Environment > Servers` node.
3. Within the `Change Center` panel found in the left column of the WebLogic console, click **Lock & Edit**.
4. Click on the server name corresponding to the deployment server in the `Summary of Servers` panel found in the right column of the WebLogic console. WebLogic will display the panel `Settings for Deployment Server` in the right column of the console.
5. Click on the `Server Start` tab. WebLogic will display the panel `Server Start` tab in the `Settings for Deployment Server` in the right column of the console.
6. Insert the following text in the Arguments box, separated by spaces:

   ```
   -Xms1024m
   -Xmx1024m
   -XX:MaxPermSize=256m
   -Dweblogic.client.socket.ConnectTimeout=10000
   -Dspring.profiles.active=FDBCloud
   ```

Also add arguments for Log4j2 file and other Log files. (For reference, see the examples below, modify path per your server configuration) :-

```
- log4j2.xml file locate at
Dlog4j2.configurationFile=/u01/app/oracle/user_projects/dom
ains/moc-preprod/moc-config/log4j2.xml
```

7. Click the Save **Button.**
8. Within the `Change Center` panel in the left column of the WebLogic console, click **Activate Changes**.

4.9.1.2 FDB MedKnowledge Data Source Configuration

MOCHA Server uses a database connection by means of a data source to FDB MedKnowledge. Complete the following steps to create a new connection pool and data source for FDB MedKnowledge.

Note: The current MOCHA Server production environment leverages multiple database instances. This configuration requires a multiple connection data source.

Creating the MOCHA Server FDB MedKnowledge Java Naming Directory Interface (JNDI):

1. Follow the steps below in the "Creating the individual database connections" for each of the database instances MOCHA Server will connect.

   Example: if MOCHA Server will connect to two separate database instances, two JNDI connections would be created and names appropriately (datasource/FDB-DIF-1 and datasource/FDB-DIF-2)

2. As when creating the multi data sources, in the WebLogic Server administration console, click **Lock & Edit** to lock the configuration. Then, in the Domain Structure area, click Data Sources under Services
3. On the Summary of Data Sources page, click **New** and select **Multi Data** Source.
4. On the Configure the Multi Data Source page, enter the multi data source Name and JNDI Name. Below are the values for the JNDI (quotes should be omitted in the entered value)
   - Name: useful label that describes the connection. For example, "MOCHA Server FDB MedKnowledge database connection".
   - JNDI Name: "*datasource/FDB45_DIF"*
   - Algorithm Type: select Load-Balancing
5. Click **Next** to continue**.**
6. Under Select Targets, select the server or servers that the previously created generic data sources were targeted to. Then click **Next**.
7. Select the XA driver type for the multi data source. Click **Next**.
8. Select the data sources that will be part of this multi data source and click the right arrow to move them from Available to Chosen (*datasource/FDB-DIF-1 and datasource/FDB-DIF-2for example)*. Click **Finish**.

9. In the Change Center, click **Activate Changes**.

Creating JDBC generic data sources:

1. Open and log into the WebLogic console, using an administrative username and password. The WebLogic console is located at: `http://<Deployment Machine>:<Deployment Server Port>/console`.
2. Within the `Domain Structure` panel found in the left column of the WebLogic console, click on the `Services> Data Sources` node.
3. Within the `Change Center` panel found in the left column of the WebLogic console, click **Lock & Edit**.
4. Click **New** and select **Generic Data Source** found in the `Summary of JDBC Data Sources` panel found in the right column of the WebLogic console. WebLogic will display the panel `Create a New JDBC Data Source` in the right column of the console, where details of the new data source are set.
5. For the `Name`, type a useful label that describes the connection. For example, "`FDB45_DIF DB Connection 1`".
6. For the `JNDI Name`, type `datasource/FDB45_DIF-(1,2, etc)`.
7. For the `Database Type`, select `Oracle`.
8. Click **Next**.
9. For the `Database Driver`, verify that `Oracle's Driver (Thin) for Instance Connections; Versions:9.0.1 and later` is selected.
10. Click **Next**. WebLogic will now display the panel `Transaction Options` in the right column of the console, where the transaction attributes for this data source are set.
11. Click **Next**. WebLogic will display the panel `Connection Properties` in the right column of the console, where the data source attributes are set.
12. For `Database Name`, type the name of the Oracle database to which MOCHA Server will connect.
13. For `Host Name`, type the name of the machine on which Oracle is running. For example, `exampleappserver.abc.va.gov`.
14. For `Port`, type the port on which Oracle is listening. For example, `1234`.
15. For `Database Username`, type the user to connect to the FDB database. For example, `FDB45_DIF`.
16. For `Password` and `Confirm Password`, type the password for the user given previously.
17. Click `Next`. WebLogic will display the panel `Test Database Connection` in the right column of the console, where the new data source can be tested.

18. Leave all values as set by default, with the exception of `Test Table Name`. For this attribute, type `fdb_version`.
19. Click **Next**.
20. WebLogic will now display the panel `Select Targets` in the right column of the console, where the target server is selected for the new data source.
21. Select the Deployment Server as the target. For example, `mocha_ms01`.
22. Click **Finish**.
23. Click **Activate Changes**.
24. WebLogic will now display the panel `Summary of JDBC Data Sources` in the right column of the console, where the newly created data source is displayed.

4.9.2 Log4j2

MOCHA Server uses Log4j2 to provide debug and error logs. Log4j2 is a dependency of MOCHA Server and must be configured prior to the deployment of the application.

To install Log4j2, the log4j2 jar must be placed on the Deployment Server's class path and the, log4j2.xml, must be edited to include the MOCHA appenders and loggers. Complete the following instructions to place the Log4j2 library on the Deployment Server's class path.

1. Copy **log4j-api-2.17.1.jar** and **log4j-core-2.17.1.jar** to server/lib folder where WebLogic is installed, **/u01/app/Oracle_Home/wlserver/server/lib**, for example.
   Note: If log4j2 is already installed, the jar file will already be on the server.
2. Configure WebLogic to include the Log4j2 library in the Deployment Server's class path. Please refer to the WebLogic documentation provided by BEA for completing this step.
3. Create the **log4j2.xml** file that is located in the path specified in the Deployment Server arguments.
4. Configure the **log4j2.xml** file using Appendix A as a reference.
5. Restart the Deployment Server to load the Log4j2 configuration.

4.9.3 ESAPI

MOCHA Server uses OWASP ESAPI to provide security for and consistency when logging incoming MOCHA Server requests. ESAPI is a dependency of MOCHA Server and must be configured prior to the deployment of the application.

ESAPI is packaged with the MOCHA Server build but must have required configuration files on the WebLogic class path for the MOCHA Server service to operate. The specific files needed are ESAPI.properties and validation.properties. Complete the following instructions to place the ESAPI properties files on the Deployment Server's class path.

1. Create the ESAPI.properties and validation.properties files using section 7.2 Appendix B and section 7.3 Appendix C as a reference.
2. Place the ESAPI properties files in the root domain directory of the WebLogic server.

- Note: You can also modify the WebLogic classpath to include the configuration files by modifying the CLASSPATH in the setDomainEnv file in the *DOMAIN_HOME*/bin directory.
- Note: Ensure the WebLogic server has sufficient privileges to access the esapi properties file. For example, if coping the file from another server, use the chown Linux command to make the WebLogic process owner the owner of the esapi properties files.

3. Restart the Deployment Server to load the ESAPI configuration.

4.9.4 Deploy New MOCHA Server Build

The following steps detail the deployment of the MOCHA Server application build on WebLogic application Server.

1. Open and log into the WebLogic console. This is located at: http://<Deployment Machine>:<Deployment Server Port>/console.
2. Within the Domain Structure panel in the left column of the WebLogic console, click the **Deployments** node. For reference, see Figure 3: Domain Structure.

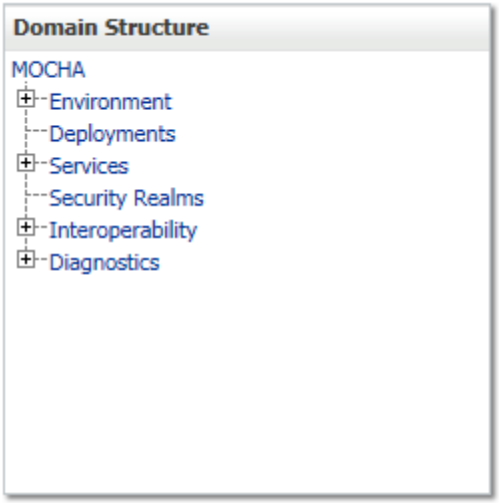


Figure 3: Domain Structure

3. Within the Change Center panel in the left column of the WebLogic console, click **Lock & Edit**. For reference, see Figure 4: Change Center.

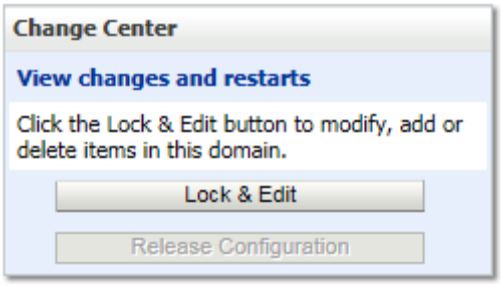


Figure 4: Change Center

4. Click **Install** found in the Deployments panel in the right column of the WebLogic console. For reference, see Figure 5: Deployments.

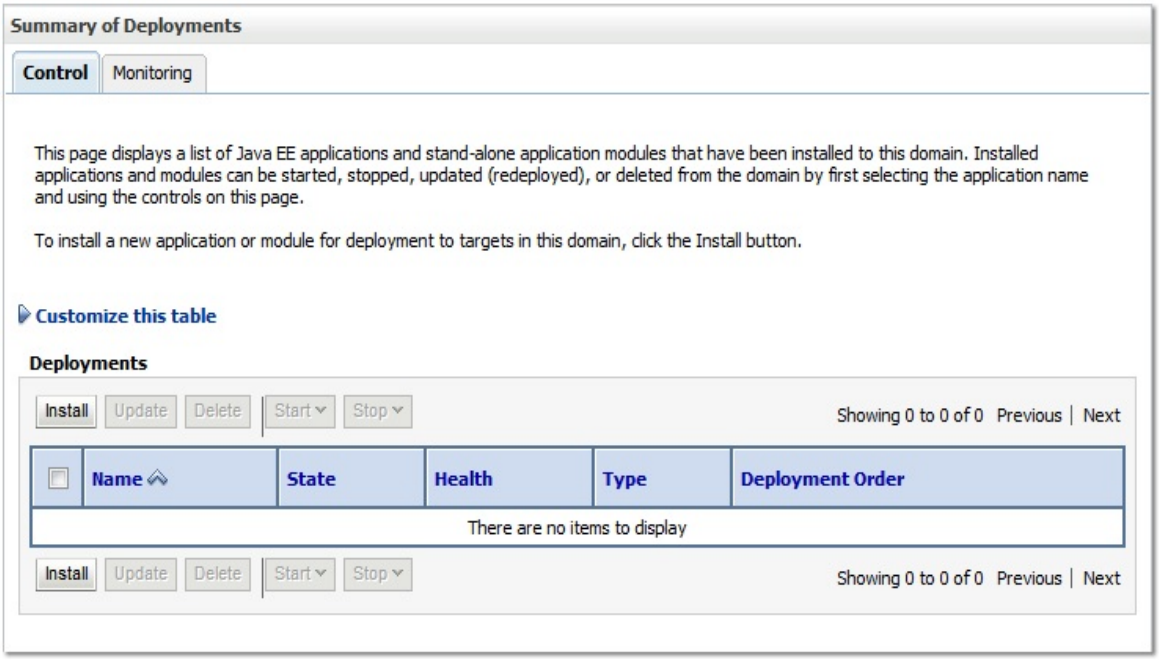


Figure 5: Deployments

5. WebLogic will now display the panel Install Application Assistant in the right column of the console, where the location of the MOCHA Server deployment will be found.
6. If the MOCHA Server build for deployment has already been transferred to the Deployment Machine, navigate to the deployment file location using the links and file structure displayed within the Install Application Assistant window.
7. Select the mocha-server-X.X.XX.XXXX.ear file. (Version number will depend on the current MOCHA Server version. For example, for MOCHA Server 4.0 build 0000, the ear file is mocha-server-4.0.01.0000).
8. Once the MOCHA Server build for deployment is located and selected, click **Next**.

9. WebLogic will now display the panel Choose targeting style within the Install Application Assistant in the right column of the console. Leave the default value selected. Install this deployment as an application and click **Next**. For reference, see Figure 6: Choose Targeting Style.

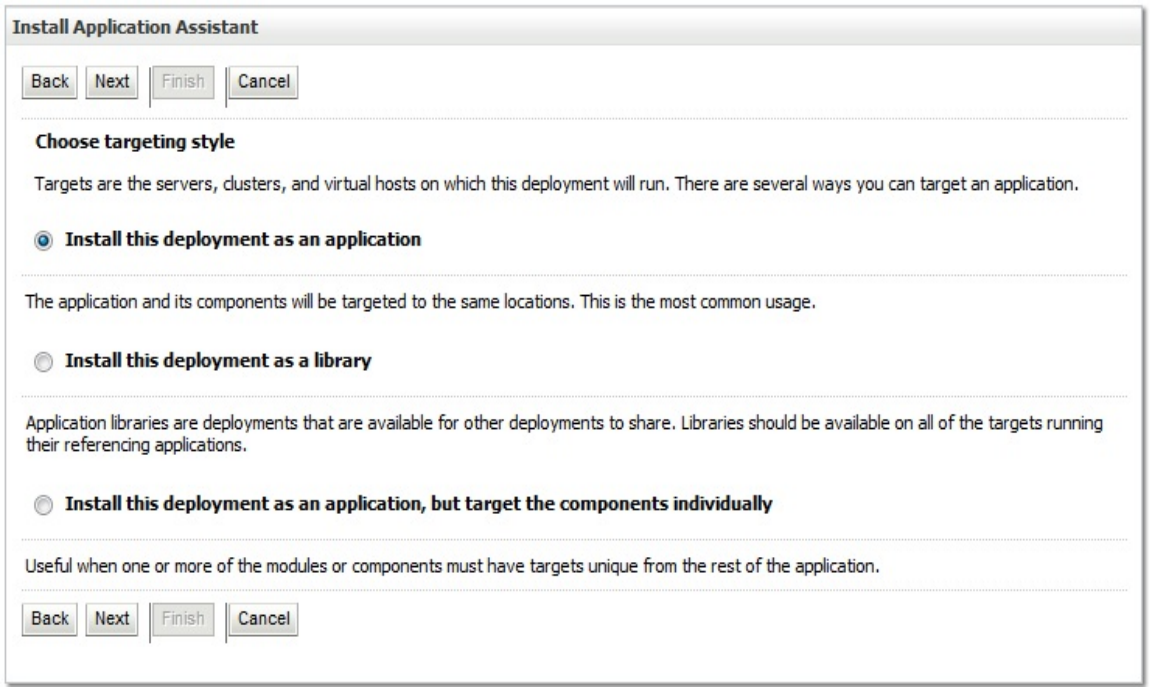


Figure 6: Choose Targeting Style

10. Within the Install Application Assistant in the right column of the console, WebLogic will now display the panel Select deployment targets, where the Deployment Server will be selected as the target in the next step.
11. For the Target, select the Deployment Server. For example, LocalPharmacyServer.
12. Select Part of the cluster, and select MOCHASrv1, MOCHASrv2, MOCHASrv3.
13. Click **Next**.
14. Within the Install Application Assistant, WebLogic will now display the panel Optional Settings in the right column of the console, where the name of the deployment and the copy behavior are chosen. For reference, see Figure 7: Optional Settings.

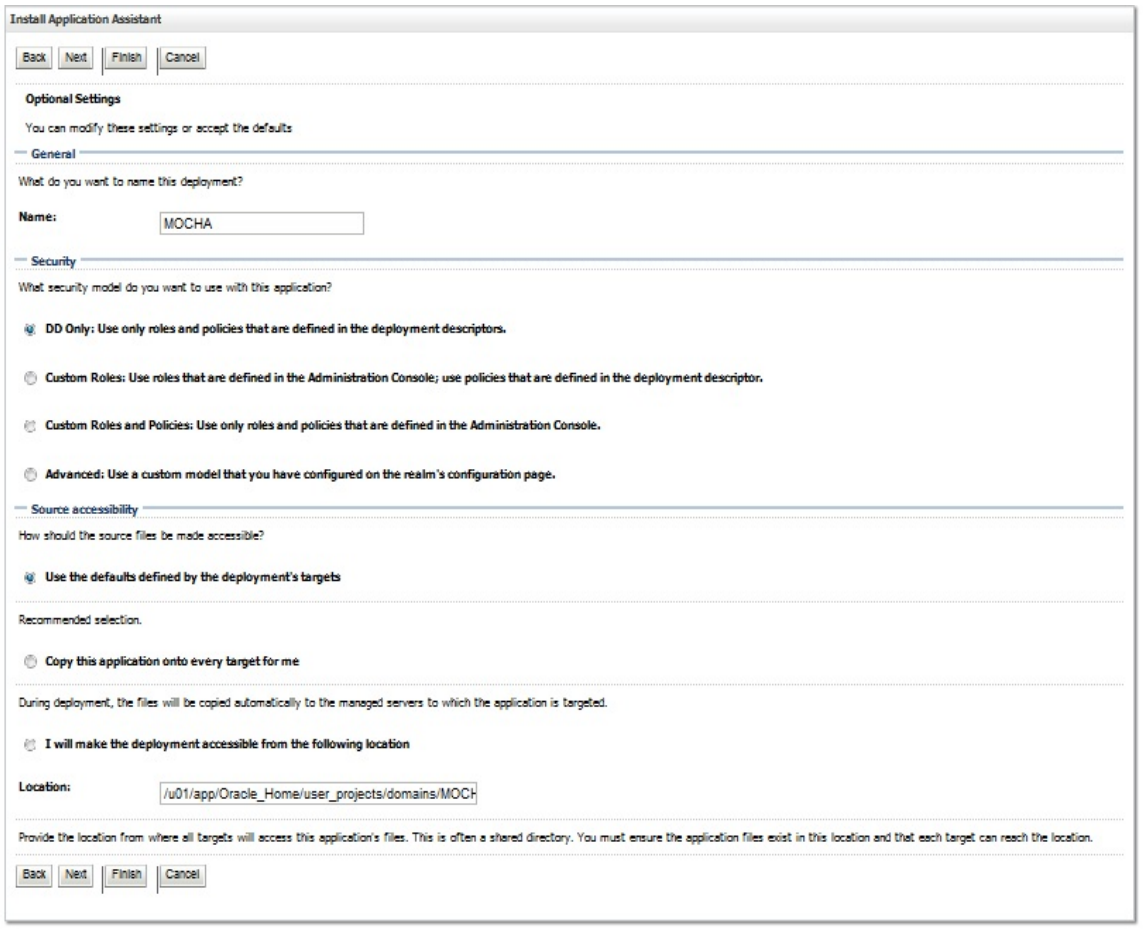


Figure 7: Optional Settings

15. Enter the `Name` for the deployment. For example, MOCHA.
16. Verify that the following default option for Security is selected:

    DD Only: Use only roles and policies that are defined in the deployment descriptors.
17. Verify that the following default option for Source accessibility is selected:

    Use the defaults defined by the deployment's targets.
18. Click **Next**.
19. Within the Install Application Assistant in the right column of the console WebLogic will now display the panel Review your choices and click **Finish**, which summarizes the steps completed above. For reference, see Figure 8: Review Your Choices and Click Finish (Note: Version number will depend on the current MOCHA Server version. For example, for MOCHA Server 4.0 build 0000, the ear file is mocha-server-4.0.08.26-bfb85b1-1).

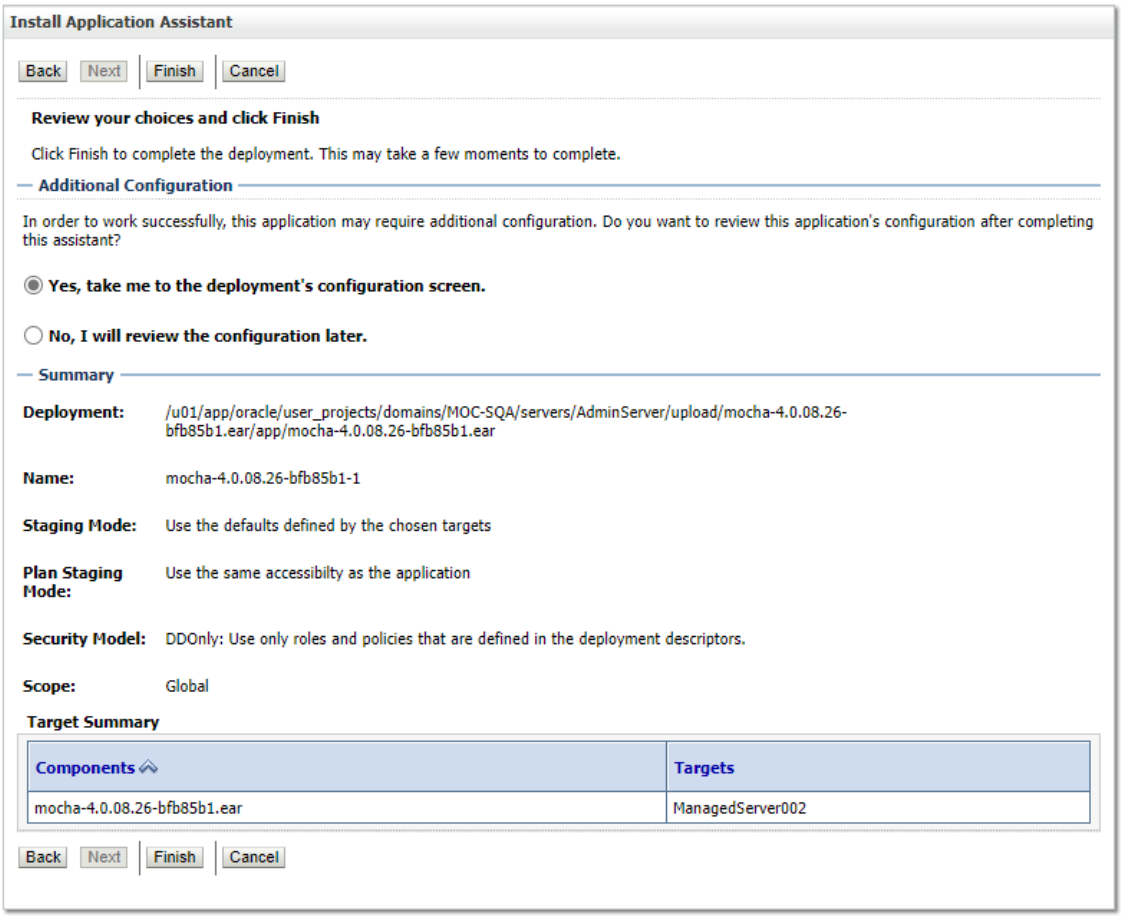


Figure 8: Review Your Choices and Click Finish

20. Verify that the values match those entered in Steps 7 through 19 and click **Finish**.

21. WebLogic will now display the panel Settings for MOCHA, in the right column of the console, where the values previously entered are available as well as a setting to change the deployment order. For reference, see Figure 9: Settings for MOCHA.

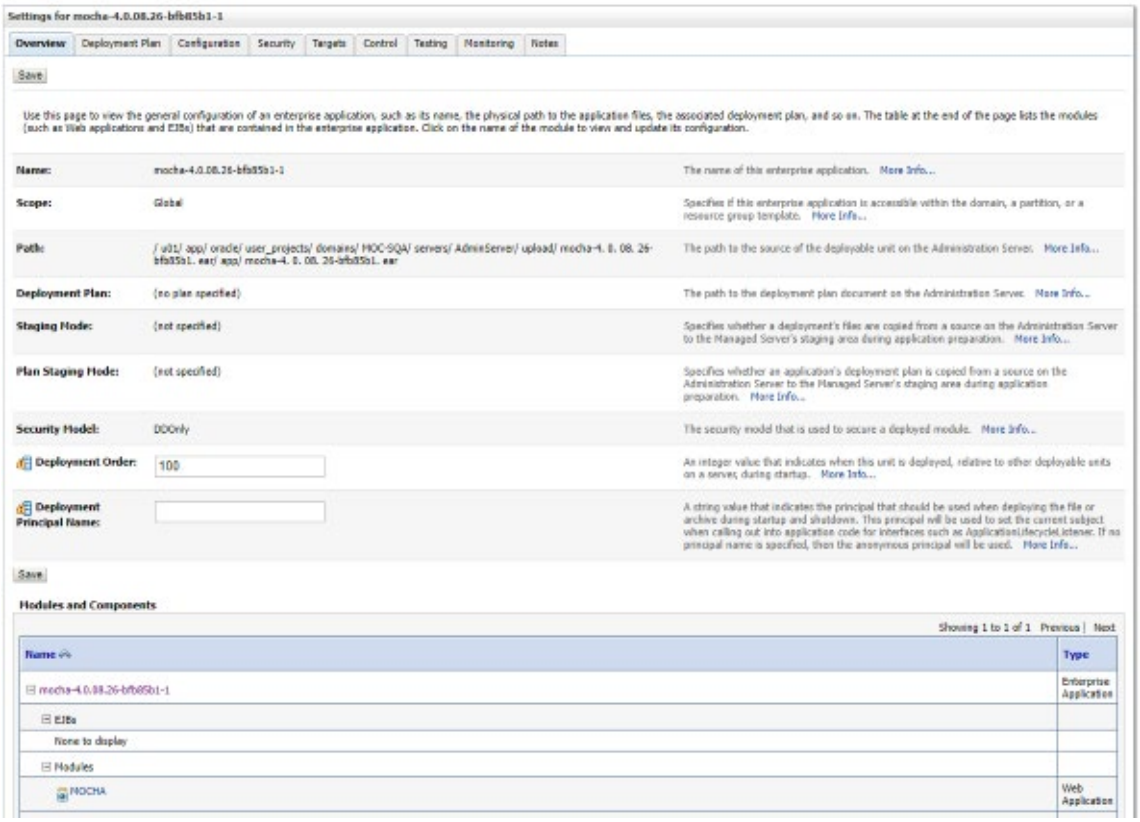


Figure 9: Settings for MOCHA

22. Leave all the values as defaulted by WebLogic and click **Save**.

23. Within the Change Center panel in the left column of the WebLogic console, click **Activate Changes**. For reference, see Figure 10: Activate Changes.

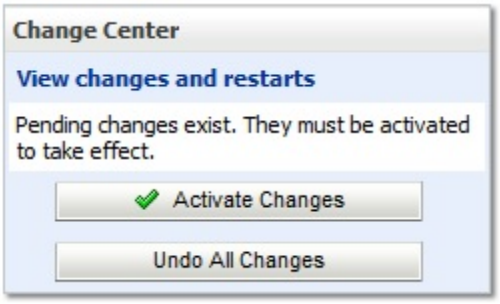


Figure 10: Activate Changes

24. Within the Domain Structure panel in the left column of the WebLogic console, click the **Deployments** node. For reference, see Figure 11: Domain Structure.

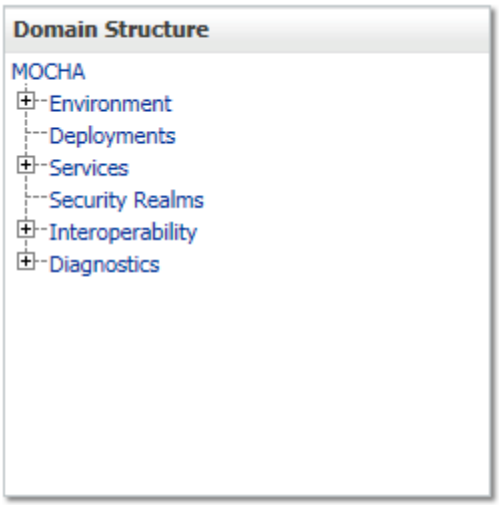


Figure 11: Domain Structure

25. WebLogic will now display the panel Summary of Deployments in the right column of the console, where all deployments for the WebLogic domain are listed. For reference, see Figure 12: Summary of Deployments.

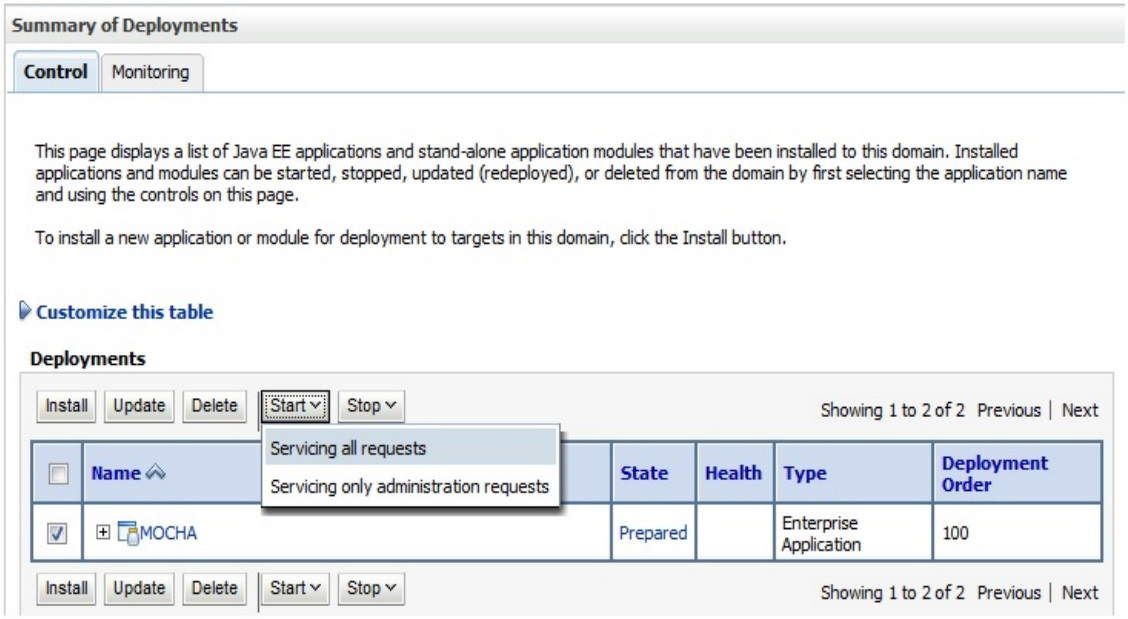


Figure 12: Summary of Deployments

26. Select the previously deployed MOCHA deployment, click **Start**, and then select "Servicing all requests" from the drop-down list box.

27. WebLogic will now display the panel Start Application Assistant in the right column of the console for confirmation to start servicing requests. For reference, see Figure 13: Start Application Assistant.

Figure 13: Start Application Assistant

28. Click **Yes** in the Start Application Assistant panel in the right column of the WebLogic console.

29. WebLogic now returns to the Summary of Deployments panel in the right column of the console. For reference, see Figure 14: Summary of Deployments – MOCHA Deployment Active. Note: Verify that the state of MOCHA Deployment is Active.

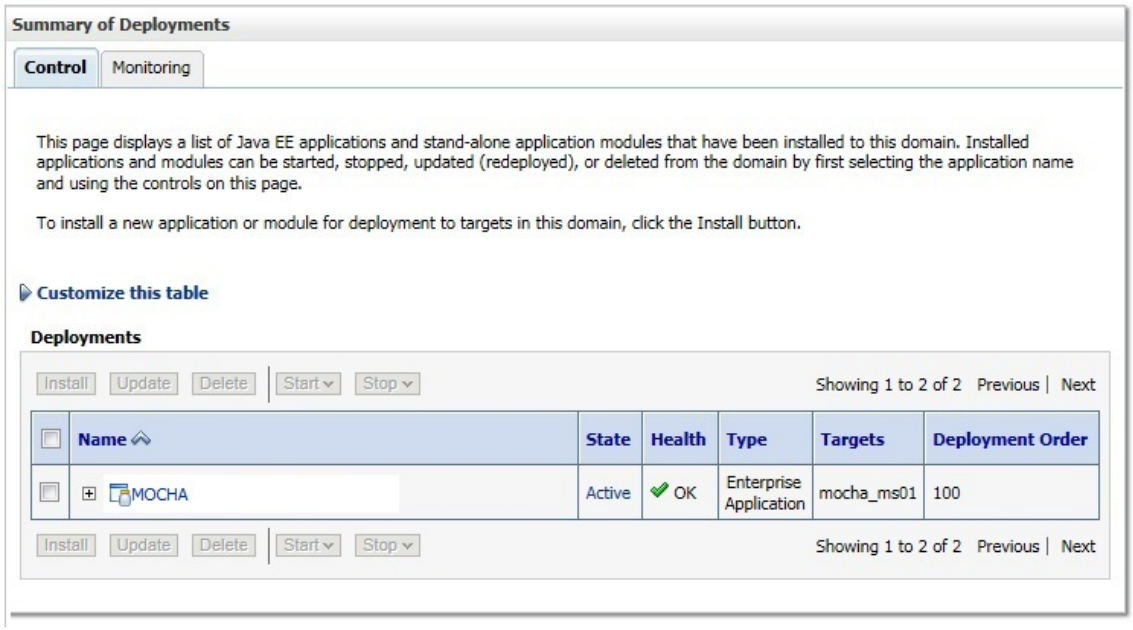


Figure 14: Summary of Deployments – MOCHA Deployment Active

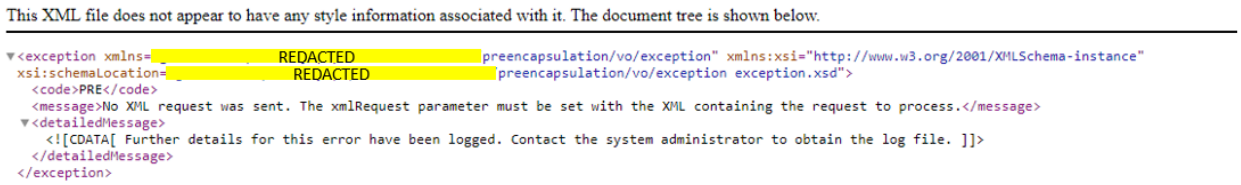
4.10 Installation Verification Procedure

When the MOCHA Server ear file has been deployed successfully, it can be verified by accessing an xml page from a web browser.

Open a web browser, such as Microsoft Edge or Chrome, and enter the URL in the following format:

http://*mochaserverhostname*:*port*/MOCHA/ordercheck, where the *mochaserverhostname* is the fully qualified domain name of the Linux server running MOCHA Server, and *port* is the port number where the ear file was deployed in WebLogic. Here is a sample URL for reference: http://exampleserver.abc.va.gov:<Deployment Server Port>/MOCHA/ordercheck

If the MOCHA Server app is up, the browser should return xml that looks similar to this:



4.11 System Configuration

Not Applicable

4.12 Database Tuning

Not Applicable

5 Back-Out Procedure

Specific back-out procedures for a release are provided in the Request for Change (RFC) that is submitted to EO.

5.1 Back-Out Strategy

Not Applicable

5.2 Back-Out Considerations

Not Applicable

5.2.1 Load Testing

Not Applicable

5.2.2 User Acceptance Testing

Not Applicable

5.3 Back-Out Criteria

Not Applicable

5.4 Back-Out Risks

Not Applicable

5.5 Authority for Back-Out

Not Applicable

5.6 Back-Out Procedure

Not Applicable

5.7 Back-out Verification Procedure

Not Applicable

6 Rollback Procedure

Enterprise Operations (EO) should follow.

6.1 Rollback Considerations

Not Applicable

6.2 Rollback Criteria

Not Applicable

6.3 Rollback Risks

Not Applicable

6.4 Authority for Rollback

Not Applicable.

6.5 Rollback Procedure

Not Applicable

6.6 Rollback Verification Procedure

Not Applicable

7 Appendix

7.1 Appendix A: Sample log4j2.xml file

```
<?xml version="1.0" encoding="UTF-8"?>

<!-- Configuration has an attribute named status that you can set to trace or debug to get configuration messages from Log4J2. -->
<Configuration>
	<Properties>
		<Property name="logDir">PPSLogs</Property>
		<Property name="maxFileSize">10 MB</Property>
		<Property name="maxRolloverFiles">10</Property>
		<Property name="logPattern">%d{DEFAULT} %-5p [%t] [%c:%M] %m%n</Property>
	</Properties>

	<Appenders>
		<Console name="ConsoleAppender" target="SYSTEM_OUT">
			<PatternLayout>
				<Pattern>${logPattern}</Pattern>
			</PatternLayout>
		</Console>

		<RollingFile name="ApacheAppender" filename="${logDir}/apache.log" filePattern="${logDir}/apache-%i.log">
```

```
                        <PatternLayout>
                                <Pattern>${logPattern}</Pattern>
                        </PatternLayout>
                        <Policies>
                                <OnStartupTriggeringPolicy />
                                <SizeBasedTriggeringPolicy size="${maxFileSize}" />
                        </Policies>
                        <DefaultRolloverStrategy max="${maxRolloverFiles}"/>
                </RollingFile>

                <RollingFile name="PPSNAppender" fileName="${logDir}/ppsn.log"
filePattern="${logDir}/ppsn-%i.log">
                        <PatternLayout>
                                <Pattern>${logPattern}</Pattern>
                        </PatternLayout>
                        <Policies>
                                <OnStartupTriggeringPolicy />
                                <SizeBasedTriggeringPolicy size="${maxFileSize}" />
                        </Policies>
                        <DefaultRolloverStrategy max="${maxRolloverFiles}"/>
                </RollingFile>

                <RollingFile name="HibernateAppender" fileName="${logDir}/hibernate.log"
filePattern="${logDir}/hibernate-%i.log">
                        <PatternLayout>
                                <Pattern>${logPattern}</Pattern>
                        </PatternLayout>
                        <Policies>
                                <OnStartupTriggeringPolicy />
                                <SizeBasedTriggeringPolicy size="${maxFileSize}" />
                        </Policies>
                        <DefaultRolloverStrategy max="${maxRolloverFiles}"/>
                </RollingFile>

                <RollingFile name="HTMLAppender" fileName="${logDir}/ppsn-all.html"
filePattern="${logDir}/ppsn-all-%i.html">
                        <HTMLLayout>
                                <LocationInfo>true</LocationInfo>
                                <Title>PPS-N Log</Title>
                        </HTMLLayout>
                        <Policies>
                                <OnStartupTriggeringPolicy />
                                <SizeBasedTriggeringPolicy size="1000 MB" />
                        </Policies>
                        <DefaultRolloverStrategy max="${maxRolloverFiles}"/>
                </RollingFile>

                <RollingFile name="SpringAppender" fileName="${logDir}/spring.log"
filePattern="${logDir}/spring-%i.log">
                        <PatternLayout>
                                <Pattern>${logPattern}</Pattern>
```

```
                        </PatternLayout>
                        <Policies>
                                <OnStartupTriggeringPolicy />
                                <SizeBasedTriggeringPolicy size="${maxFileSize}" />
                        </Policies>
                        <DefaultRolloverStrategy max="${maxRolloverFiles}"/>
                </RollingFile>

        <RollingFile name="VAAppender" fileName="${logDir}/other_va_software.log"
filePattern="${logDir}/other_va_software-%i.log">
                        <PatternLayout>
                                <Pattern>${logPattern}</Pattern>
                        </PatternLayout>
                        <Policies>
                                <OnStartupTriggeringPolicy />
                                <SizeBasedTriggeringPolicy size="${maxFileSize}" />
                        </Policies>
                        <DefaultRolloverStrategy max="${maxRolloverFiles}"/>
        </RollingFile>

                <RollingFile name="PPSNPerformanceAppender"
fileName="${logDir}/ppsPerformance.log" filePattern="${logDir}/ppsPerformance-%i.log">
                        <PatternLayout>
                                <Pattern>${logPattern}</Pattern>
                        </PatternLayout>
                        <Policies>
                                <OnStartupTriggeringPolicy />
                                <SizeBasedTriggeringPolicy size="${maxFileSize}" />
                        </Policies>
                        <DefaultRolloverStrategy max="${maxRolloverFiles}"/>
                </RollingFile>
        </Appenders>
        <Loggers>
                <logger name="gov.va.med.pharmacy" level="error" additivity="false">
                        <AppenderRef ref="PPSNAppender" />
                        <AppenderRef ref="HTMLAppender" />
                </logger>

                <logger name="gov.va.med" level="error" additivity="false">
                        <AppenderRef ref="VAAppender" />
                        <AppenderRef ref="HTMLAppender" />
                </logger>

                <logger name="org.apache" level="error" additivity="false">
                        <AppenderRef ref="ApacheAppender"/>
                        <AppenderRef ref="HTMLAppender" />
                </logger>

                <logger name="org.aspectj" level="error" additivity="false">
                        <AppenderRef ref="SpringAppender" />
                        <AppenderRef ref="HTMLAppender" />
```

```
            </logger>

            <logger name="org.dbunit" level="error" additivity="false">
                <AppenderRef ref="PPSNAppender" />
                <AppenderRef ref="HTMLAppender" />
            </logger>

            <logger name="org.hibernate" level="error" additivity="false">
                <AppenderRef ref="HibernateAppender" />
                <AppenderRef ref="HTMLAppender" />
            </logger>

            <logger name="org.springframework" level="error" additivity="false">
                <AppenderRef ref="SpringAppender" />
                <AppenderRef ref="HTMLAppender" />
            </logger>

            <logger name="net.sf.navigator" level="error" additivity="false">
                <AppenderRef ref="PPSNAppender" />
                <AppenderRef ref="HTMLAppender" />
            </logger>

            <logger name="net.sf.ehcache" level="error" additivity="false">
                <AppenderRef ref="PPSNAppender" />
                <AppenderRef ref="HTMLAppender" />
            </logger>

            <!-- Special performance metrics Loggers -->
            <!-- AOP-based -->
            <logger
                name="org.springframework.aop.interceptor.PerformanceMonitorInterceptor"
                level="trace"
                additivity="false">
                <appender-ref ref="PPSNPerformanceAppender" />
            </logger>
            <logger

        name="org.springframework.aop.interceptor.JamonPerformanceMonitorInterceptor"
                level="trace"
                additivity="false">
                <appender-ref ref="PPSNPerformanceAppender" />
            </logger>
            <Root level="error">
                <AppenderRef ref="ConsoleAppender"/>
                <AppenderRef ref="HTMLAppender" />
            </Root>
        </Loggers>
</Configuration>
```

7.2 Appendix B: Sample ESAPI.properties file

```
#
# OWASP Enterprise Security API (ESAPI) Properties file -- TEST Version
#
# This file is part of the Open Web Application Security Project (OWASP)
# Enterprise Security API (ESAPI) project. For details, please see
# http://www.owasp.org/index.php/ESAPI.
#
# Copyright (c) 2008,2009 - The OWASP Foundation
#
# DISCUSS: This may cause a major backwards compatibility issue, etc. but
#                       from a name space perspective, we probably should have prefaced
#                       all the property names with ESAPI or at least OWASP. Otherwise
#                       there could be problems is someone loads this properties file into
#                       the System properties. We could also put this file into the
#                       esapi.jar file (perhaps as a ResourceBundle) and then allow an external
#                       ESAPI properties be defined that would overwrite these defaults.
#                       That keeps the application's properties relatively simple as usually
#                       they will only want to override a few properties. If looks like we
#                       already support multiple override levels of this in the
#                       DefaultSecurityConfiguration class, but I'm suggesting placing the
#                       defaults in the esapi.jar itself. That way, if the jar is signed,
#                       we could detect if those properties had been tampered with. (The
#                       code to check the jar signatures is pretty simple... maybe 70-90 LOC,
#                       but off course there is an execution penalty (similar to the way
#                       that the separate sunjce.jar used to be when a class from it was
#                       first loaded). Thoughts?
###############################################################################
#
# WARNING: Operating system protection should be used to lock down the .esapi
# resources directory and all the files inside and all the directories all the
# way up to the root directory of the file system. Note that if you are using
# file-based implementations, that some files may need to be read-write as they
# get updated dynamically.
#
# Before using, be sure to update the MasterKey and MasterSalt as described below.
# N.B.: If you had stored data that you have previously encrypted with ESAPI 1.4,
#                       you *must* FIRST decrypt it using ESAPI 1.4 and then (if so desired)
#                       re-encrypt it with ESAPI 2.0. If you fail to do this, you will NOT be
#                       able to decrypt your data with ESAPI 2.0.
#
```

```
#                       YOU HAVE BEEN WARNED!!! More details are in the ESAPI 2.0 Release Notes.
#
#==========================================================================
# ESAPI Configuration
#
# If true, then print all the ESAPI properties set here when they are loaded.
# If false, they are not printed. Useful to reduce output when running JUnit tests.
# If you need to troubleshoot a properties related problem, turning this on may help,
# but we leave it off for running JUnit tests. (It will be 'true' in the one delivered
# as part of production ESAPI, mostly for backward compatibility.)
ESAPI.printProperties=false

# ESAPI is designed to be easily extensible. You can use the reference implementation
# or implement your own providers to take advantage of your enterprise's security
# infrastructure. The functions in ESAPI are referenced using the ESAPI locator, like:
#
#    String ciphertext =
#                       ESAPI.encryptor().encrypt("Secret message");   // Deprecated in 2.0
#    CipherText cipherText =
#                       ESAPI.encryptor().encrypt(new PlainText("Secret message")); // Preferred
#
# Below you can specify the classname for the provider that you wish to use in your
# application. The only requirement is that it implement the appropriate ESAPI interface.
# This allows you to switch security implementations in the future without rewriting the
# entire application.
#
# ExperimentalAccessController requires ESAPI-AccessControlPolicy.xml in .esapi directory
ESAPI.AccessControl=org.owasp.esapi.reference.DefaultAccessController
# FileBasedAuthenticator requires users.txt file in .esapi directory
ESAPI.Authenticator=org.owasp.esapi.reference.FileBasedAuthenticator
ESAPI.Encoder=org.owasp.esapi.reference.DefaultEncoder
ESAPI.Encryptor=org.owasp.esapi.reference.crypto.JavaEncryptor

ESAPI.Executor=org.owasp.esapi.reference.DefaultExecutor
ESAPI.HTTPUtilities=org.owasp.esapi.reference.DefaultHTTPUtilities
ESAPI.IntrusionDetector=org.owasp.esapi.reference.DefaultIntrusionDetector
# Log4JFactory Requires log4j.xml or log4j.properties in classpath - http://www.lalilуna.de/log4j-tutorial.html
#ESAPI.Logger=org.owasp.esapi.reference.Log4JLogFactory
ESAPI.Logger=org.owasp.esapi.logging.java.JavaLogFactory
#ESAPI.Logger=org.owasp.esapi.reference.ExampleExtendedLog4JLogFactory
ESAPI.Randomizer=org.owasp.esapi.reference.DefaultRandomizer
ESAPI.Validator=org.owasp.esapi.reference.DefaultValidator
#==========================================================================
```

```
#These new lines added recommendation by 2.4.0.0 version
#                    ~~~~~ Important Note ~~~~~
# This is a workaround to make sure that a commit to address GitHub issue #509
# doesn't accidentally break someone's production code. So essentially what we
# are doing is to reverting back to the previous possibly buggy (by
# documentation intent at least), but, by now, expected legacy behavior.
# Prior to the code changes for issue #509, if invalid / malicious HTML input was
# observed, AntiSamy would simply attempt to sanitize (cleanse) it and it would
# only be logged. However, the code change made ESAPI comply with its
# documentation, which stated that a ValidationException should be thrown in
# such cases. Unfortunately, changing this behavior--especially when no one is
# 100% certain that the documentation was correct--could break existing code
# using ESAPI so after a lot of debate, issue #521 was created to restore the
# previous behavior, but still allow the documented behavior. (We did this
# because it wasn't really causing a security issues since AntiSamy would clean
# it up anyway and we value backward compatibility as long as it doesn't clearly
# present security vulnerabilities.)
# More defaults about this are written up under GitHub issue #521 and
# the pull request it references. Future major releases of ESAPI (e.g., ESAPI 3.x)
# will not support this previous behavior, but it will remain for ESAPI 2.x.
# Set this to 'throw' if you want the originally intended behavior of throwing
# that was fixed via issue #509. Set to 'clean' if you want the HTML input
# sanitized instead.
#
# Possible values:
#   clean -- Use the legacy behavior where unsafe HTML input is logged and the
#            sanitized (i.e., clean) input as determined by AntiSamy and your
#            AntiSamy rules is returned. This is the default behavior if this
#            new property is not found.
#   throw -- The new, presumably correct and originally intended behavior where
#            a ValidationException is thrown when unsafe HTML input is
#            encountered.
#
#Validator.HtmlValidationAction=clean
Validator.HtmlValidationAction=throw

# With the fix for #310 to enable loading antisamy-esapi.xml from the classpath
# also an enhancement was made to be able to use a different filename for the configuration.
# You don't have to configure the filename here, but in that case the code will keep looking for antisamy-esapi.xml.
# This is the default behavior of ESAPI.
#
#Validator.HtmlValidationConfigurationFile=antisamy-esapi.xml
```

```
# This is the _minimum_ key size (in bits) that we allow with ANY symmetric
# cipher for doing encryption. (There is no minimum for decryption.)
#
# Generally, if you only use one algorithm, this should be set the same as
# the Encryptor.EncryptionKeyLength property.
Encryptor.MinEncryptionKeyLength=128

# Maximum size of HTTP header key--the validator regex may have additional values.
HttpUtilities.MaxHeaderNameSize=256
# Maximum size of HTTP header value--the validator regex may have additional values.
HttpUtilities.MaxHeaderValueSize=4096
# Maximum size of JSESSIONID for the application--the validator regex may have additional values.
HttpUtilities.HTTPJSESSIONIDLENGTH=50
# Maximum length of a URL (see https://stackoverflow.com/questions/417142/what-is-the-maximum-length-of-a-url-in-different-browsers)
HttpUtilities.URILENGTH=2000
# Maximum length of a redirect
HttpUtilities.maxRedirectLength=512
# Maximum length for an http scheme
HttpUtilities.HTTPSCHEMELENGTH=10
# Maximum length for an http host
HttpUtilities.HTTPHOSTLENGTH=100
# Maximum length for an http path
HttpUtilities.HTTPPATHLENGTH=150
#Maximum length for a context path
HttpUtilities.contextPathLength=150
#Maximum length for an httpServletPath
HttpUtilities.HTTPSERVLETPATHLENGTH=100
#Maximum length for an http query parameter name
HttpUtilities.httpQueryParamNameLength=100
#Maximum length for an http query parameter -- old default was 2000, but that's the max length for a URL...
HttpUtilities.httpQueryParamValueLength=500

#Sets whether or not we will overwrite http status codes to 200.
HttpUtilities.OverwriteStatusCodes=true
#Sets the application's base character encoding.  This is forked from the Java Encryptor property.
HttpUtilities.CharacterEncoding=UTF-8
Logger.UserInfo=false
Logger.ClientInfo=false
#======================End of lines added for v2.4
```

```
#===========================================================================
# ESAPI Authenticator
#
Authenticator.AllowedLoginAttempts=3
Authenticator.MaxOldPasswordHashes=13
Authenticator.UsernameParameterName=username
Authenticator.PasswordParameterName=password
# RememberTokenDuration (in days)
Authenticator.RememberTokenDuration=14
# Session Timeouts (in minutes)
Authenticator.IdleTimeoutDuration=20
Authenticator.AbsoluteTimeoutDuration=120

#===========================================================================
# ESAPI Encoder
#
# ESAPI canonicalizes input before validation to prevent bypassing filters with encoded attacks.
# Failure to canonicalize input is a very common mistake when implementing validation schemes.
# Canonicalization is automatic when using the ESAPI Validator, but you can also use the
# following code to canonicalize data.
#
#     ESAPI.Encoder().canonicalize( "%22hello world&#x22;" );
#
# Multiple encoding is when a single encoding format is applied multiple times. Allowing
# multiple encoding is strongly discouraged.
Encoder.AllowMultipleEncoding=false

# Mixed encoding is when multiple different encoding formats are applied, or when
# multiple formats are nested. Allowing multiple encoding is strongly discouraged.
Encoder.AllowMixedEncoding=false

# The default list of codecs to apply when canonicalizing untrusted data. The list should include the codecs
# for all downstream interpreters or decoders. For example, if the data is likely to end up in a URL, HTML, or
# inside JavaScript, then the list of codecs below is appropriate. The order of the list is not terribly important.
Encoder.DefaultCodecList=HTMLEntityCodec,PercentCodec,JavaScriptCodec



#===========================================================================
# ESAPI Encryption
#
# The ESAPI Encryptor provides basic cryptographic functions with a simplified API.
# To get started, generate a new key using java -classpath esapi.jar org.owasp.esapi.reference.crypto.JavaEncryptor
# There is not currently any support for key rotation, so be careful when changing your key and salt as it
```

```
# will invalidate all signed, encrypted, and hashed data.
#
# WARNING: Not all combinations of algorithms and key lengths are supported.
# If you choose to use a key length greater than 128, you MUST download the
# unlimited strength policy files and install in the lib directory of your JRE/JDK.
# See http://java.sun.com/javase/downloads/index.jsp for more information.
#
# Backward compatibility with ESAPI Java 1.4 is supported by the two deprecated API
# methods, Encryptor.encrypt(String) and Encryptor.decrypt(String). However, whenever
# possible, these methods should be avoided as they use ECB cipher mode, which in almost
# all circumstances a poor choice because of its weakness. CBC cipher mode is the default
# for the new Encryptor encrypt / decrypt methods for ESAPI Java 2.0.  In general, you
# should only use this compatibility setting if you have persistent data encrypted with
# version 1.4 and even then, you should ONLY set this compatibility mode UNTIL
# you have decrypted all of your old encrypted data and then re-encrypted it with
# ESAPI 2.0 using CBC mode. If you have some reason to mix the deprecated 1.4 mode
# with the new 2.0 methods, make sure that you use the same cipher algorithm for both
# (256-bit AES was the default for 1.4; 128-bit is the default for 2.0; see below for
# more details.) Otherwise, you will have to use the new 2.0 encrypt / decrypt methods
# where you can specify a SecretKey. (Note that if you are using the 256-bit AES,
# that requires downloading the special jurisdiction policy files mentioned above.)
#
#                     ***** IMPORTANT: These are for JUnit testing. Test files may have been
#                                                         encrypted using these values so do not change these or
#                                                         those tests will fail. The version under
#                                                                 src/main/resources/.esapi/ESAPI.properties
#                                                         will be delivered with Encryptor.MasterKey and
#                                                         Encryptor.MasterSalt set to the empty string.
#
#                                                         FINAL NOTE:
#                     If Maven changes these when run, that needs to be fixed.
#       256-bit key... requires unlimited strength jurisdiction policy files
### Encryptor.MasterKey=pJhlri8JbuFYDgkqtHmm9s0Ziug2PE7ovZDyEPm4j14=
#       128-bit key
Encryptor.MasterKey=a6H9is3hEVGKB4Jut+lOVA==
Encryptor.MasterSalt=SbftnvmEWD5ZHHP+pX3fqugNysc=
# Encryptor.MasterSalt=

# Provides the default JCE provider that ESAPI will "prefer" for its symmetric
# encryption and hashing. (That is it will look to this provider first, but it
# will defer to other providers if the requested algorithm is not implemented
# by this provider.) If left unset, ESAPI will just use your Java VM's current
# preferred JCE provider, which is generally set in the file
```

```
# "$JAVA_HOME/jre/lib/security/java.security".
#
# The main intent of this is to allow ESAPI symmetric encryption to be
# used with a FIPS 140-2 compliant crypto-module. For details, see the section
# "Using ESAPI Symmetric Encryption with FIPS 140-2 Cryptographic Modules" in
# the ESAPI 2.0 Symmetric Encryption User Guide, at:
# http://owasp-esapi-java.googlecode.com/svn/trunk/documentation/esapi4java-core-2.0-symmetric-crypto-user-guide.html
# However, this property also allows you to easily use an alternate JCE provider
# such as "Bouncy Castle" without having to make changes to "java.security".
# See Javadoc for SecurityProviderLoader for further details. If you wish to use
# a provider that is not known to SecurityProviderLoader, you may specify the
# fully-qualified class name of the JCE provider class that implements
# java.security.Provider. If the name contains a '.', this is interpreted as
# a fully-qualified class name that implements java.security.Provider.
#
# NOTE: Setting this property has the side-effect of changing it in your application
#       as well, so if you are using JCE in your application directly rather than
#       through ESAPI (you wouldn't do that, would you? ;-), it will change the
#       preferred JCE provider there as well.
#
# Default: Keeps the JCE provider set to whatever JVM sets it to.
Encryptor.PreferredJCEProvider=

# AES is the most widely used and strongest encryption algorithm. This
# should agree with your Encryptor.CipherTransformation property.
# By default, ESAPI Java 1.4 uses "PBEWithMD5AndDES" and which is
# very weak. It is essentially a password-based encryption key, hashed
# with MD5 around 1K times and then encrypted with the weak DES algorithm
# (56-bits) using ECB mode and an unspecified padding (it is
# JCE provider specific, but most likely "NoPadding"). However, 2.0 uses
# "AES/CBC/PKCSPadding". If you want to change these, change them here.
# Warning: This property does not control the default reference implementation for
#                       ESAPI 2.0 using JavaEncryptor. Also, this property will be dropped
#                       in the future.
# @deprecated
Encryptor.EncryptionAlgorithm=AES
#                     For ESAPI Java 2.0 - New encrypt / decrypt methods use this.
Encryptor.CipherTransformation=AES/CBC/PKCS5Padding

# Applies to ESAPI 2.0 and later only!
# Comma-separated list of cipher modes that provide *BOTH*
# confidentiality *AND* message authenticity. (NIST refers to such cipher
# modes as "combined modes" so that's what we shall call them.) If any of these
```

```
# cipher modes are used then no MAC is calculated and stored
# in the CipherText upon encryption. Likewise, if one of these
# cipher modes is used with decryption, no attempt will be made
# to validate the MAC contained in the CipherText object regardless
# of whether it contains one or not. Since the expectation is that
# these cipher modes support message authenticity already,
# injecting a MAC in the CipherText object would be at best redundant.
#
# Note that as of JDK 1.5, the SunJCE provider does not support *any*
# of these cipher modes. Of these listed, only GCM and CCM are currently
# NIST approved. YMMV for other JCE providers. E.g., Bouncy Castle supports
# GCM and CCM with "NoPadding" mode, but not with "PKCS5Padding" or other
# padding modes.
Encryptor.cipher_modes.combined_modes=GCM,CCM,IAPM,EAX,OCB,CWC

# Applies to ESAPI 2.0 and later only!
# Additional cipher modes allowed for ESAPI 2.0 encryption. These
# cipher modes are in _addition_ to those specified by the property
# 'Encryptor.cipher_modes.combined_modes'.
# Note: We will add support for streaming modes like CFB & OFB once
# we add support for 'specified' to the property 'Encryptor.ChooseIVMethod'
# (probably in ESAPI 2.1).
#
#       IMPORTANT NOTE:     In the official ESAPI.properties we do *NOT* include ECB
#                           here as this is an extremely weak mode. However, we *must*
#                           allow it here so we can test ECB mode. That is important
#                           since the logic is somewhat different (i.e., ECB mode does
#                           not use an IV).
# DISCUSS: Better name?
#       NOTE: ECB added only for testing purposes. Don't try this at home!
Encryptor.cipher_modes.additional_allowed=CBC,ECB

# 128-bit is almost always sufficient and appears to be more resistant to
# related key attacks than is 256-bit AES. Use '_' to use default key size
# for cipher algorithms (where it makes sense because the algorithm supports
# a variable key size). Key length must agree to what's provided as the
# cipher transformation, otherwise this will be ignored after logging a
# warning.
#
# NOTE: This is what applies BOTH ESAPI 1.4 and 2.0. See warning above about mixing!
Encryptor.EncryptionKeyLength=128

# Because 2.0 uses CBC mode by default, it requires an initialization vector (IV).
```

```
# (All cipher modes except ECB require an IV.) There are two choices: we can either
# use a fixed IV known to both parties or allow ESAPI to choose a random IV. While
# the IV does not need to be hidden from adversaries, it is important that the
# adversary not be allowed to choose it. Also, random IVs are generally much more
# secure than fixed IVs. (In fact, it is essential that feed-back cipher modes
# such as CFB and OFB use a different IV for each encryption with a given key so
# in such cases, random IVs are much preferred. By default, ESAPI 2.0 uses random
# IVs. If you wish to use 'fixed' IVs, set 'Encryptor.ChooseIVMethod=fixed' and
# uncomment the Encryptor.fixedIV.
#
# Valid values:                 random|fixed|specified                  'specified' not yet implemented; planned for 2.1
Encryptor.ChooseIVMethod=random
# If you choose to use a fixed IV, then you must place a fixed IV here that
# is known to all others who are sharing your secret key. The format should
# be a hex string that is the same length as the cipher block size for the
# cipher algorithm that you are using. The following is an example for AES
# from an AES test vector for AES-128/CBC as described in:
# NIST Special Publication 800-38A (2001 Edition)
# "Recommendation for Block Cipher Modes of Operation".
# (Note that the block size for AES is 16 bytes == 128 bits.)
#
Encryptor.fixedIV=0x000102030405060708090a0b0c0d0e0f

# Whether or not CipherText should use a message authentication code (MAC) with it.
# This prevents an adversary from altering the IV as well as allowing a more
# fool-proof way of determining the decryption failed because of an incorrect
# key being supplied. This refers to the "separate" MAC calculated and stored
# in CipherText, not part of any MAC that is calculated as a result of a
# "combined mode" cipher mode.
#
# If you are using ESAPI with a FIPS 140-2 cryptographic module, you *must* also
# set this property to false.
Encryptor.CipherText.useMAC=true

# Whether or not the PlainText object may be overwritten and then marked
# eligible for garbage collection. If not set, this is still treated as 'true'.
Encryptor.PlainText.overwrite=true

# Do not use DES except in a legacy situations. 56-bit is way too small key size.
#Encryptor.EncryptionKeyLength=56
#Encryptor.EncryptionAlgorithm=DES

# TripleDES is considered strong enough for most purposes.
```

```
#       Note:   There is also a 112-bit version of DESede. Using the 168-bit version
#                       requires downloading the special jurisdiction policy from Sun.
#Encryptor.EncryptionKeyLength=168
#Encryptor.EncryptionAlgorithm=DESede

Encryptor.HashAlgorithm=SHA-512
Encryptor.HashIterations=1024
Encryptor.DigitalSignatureAlgorithm=SHA1withDSA
Encryptor.DigitalSignatureKeyLength=1024
Encryptor.RandomAlgorithm=SHA1PRNG
Encryptor.CharacterEncoding=UTF-8
# Currently supported choices for JDK 1.5 and 1.6 are:
#       HmacSHA1 (160 bits), HmacSHA256 (256 bits), HmacSHA384 (384 bits), and
#       HmacSHA512 (512 bits).
# Note that HmacMD5 is *not* supported for the PRF used by the KDF even though
# these JDKs support it.
Encryptor.KDF.PRF=HmacSHA256

#===========================================================================
# ESAPI HttpUtilties
#
# The HttpUtilities provide basic protections to HTTP requests and responses. Primarily these methods
# protect against malicious data from attackers, such as unprintable characters, escaped characters,
# and other simple attacks. The HttpUtilities also provides utility methods for dealing with cookies,
# headers, and CSRF tokens.
#
# Default file upload location (remember to escape backslashes with \\)
HttpUtilities.UploadDir=C:\\ESAPI\\testUpload
# let this default to java.io.tmpdir for testing
#HttpUtilities.UploadTempDir=C:\\temp
# Force flags on cookies, if you use HttpUtilities to set cookies
HttpUtilities.ForceHttpOnlySession=false
HttpUtilities.ForceSecureSession=false
HttpUtilities.ForceHttpOnlyCookies=true
HttpUtilities.ForceSecureCookies=true
# Maximum size of HTTP headers
HttpUtilities.MaxHeaderSize=4096
# File upload configuration
HttpUtilities.ApprovedUploadExtensions=.zip,.pdf,.doc,.docx,.ppt,.pptx,.tar,.gz,.tgz,.rar,.war,.jar,.ear,.xls,.rtf,.properties,.java,.class,.t
xt,.xml,.jsp,.jsf,.exe,.dll
HttpUtilities.MaxUploadFileBytes=500000000
# Using UTF-8 throughout your stack is highly recommended. That includes your database driver,
# container, and any other technologies you may be using. Failure to do this may expose you
```

```
# to Unicode transcoding injection attacks. Use of UTF-8 does not hinder internationalization.
HttpUtilities.ResponseContentType=text/html; charset=UTF-8
# This is the name of the cookie used to represent the HTTP session
# Typically this will be the default "JSESSIONID"
HttpUtilities.HttpSessionIdName=JSESSIONID



#===========================================================================
# ESAPI Executor
# CHECKME - Not sure what this is used for, but surely it should be made OS independent.
Executor.WorkingDirectory=C:\\Windows\\Temp
Executor.ApprovedExecutables=C:\\Windows\\System32\\cmd.exe,C:\\Windows\\System32\\runas.exe



#===========================================================================
# ESAPI Logging
# Set the application name if these logs are combined with other applications
Logger.ApplicationName=PRE
# If you use an HTML log viewer that does not properly HTML escape log data, you can set LogEncodingRequired to true
Logger.LogEncodingRequired=false
# Determines whether ESAPI should log the application name. This might be clutter in some single-server/single-app environments.
Logger.LogApplicationName=true
# Determines whether ESAPI should log the server IP and port. This might be clutter in some single-server environments.
Logger.LogServerIP=true
# LogFileName, the name of the logging file. Provide a full directory path (e.g., C:\\ESAPI\\ESAPI_logging_file) if you
# want to place it in a specific directory.
Logger.LogFileName=ESAPI_logging_file
# MaxLogFileSize, the max size (in bytes) of a single log file before it cuts over to a new one (default is 10,000,000)
Logger.MaxLogFileSize=10000000



#===========================================================================
# ESAPI Intrusion Detection
#
# Each event has a base to which .count, .interval, and .action are added
# The IntrusionException will fire if we receive "count" events within "interval" seconds
# The IntrusionDetector is configurable to take the following actions: log, logout, and disable
#  (multiple actions separated by commas are allowed e.g. event.test.actions=log,disable
#
# Custom Events
# Names must start with "event." as the base
# Use IntrusionDetector.addEvent( "test" ) in your code to trigger "event.test" here
```

```
# You can also disable intrusion detection completely by changing
# the following parameter to true
#
IntrusionDetector.Disable=false
#
IntrusionDetector.event.test.count=2
IntrusionDetector.event.test.interval=10
IntrusionDetector.event.test.actions=disable,log

# Exception Events
# All EnterpriseSecurityExceptions are registered automatically
# Call IntrusionDetector.getInstance().addException(e) for Exceptions that do not extend EnterpriseSecurityException
# Use the fully qualified classname of the exception as the base

# any intrusion is an attack
IntrusionDetector.org.owasp.esapi.errors.IntrusionException.count=1
IntrusionDetector.org.owasp.esapi.errors.IntrusionException.interval=1
IntrusionDetector.org.owasp.esapi.errors.IntrusionException.actions=log,disable,logout

# for test purposes
# CHECKME: Shouldn't there be something in the property name itself that designates
#                       that these are for testing???
IntrusionDetector.org.owasp.esapi.errors.IntegrityException.count=10
IntrusionDetector.org.owasp.esapi.errors.IntegrityException.interval=5
IntrusionDetector.org.owasp.esapi.errors.IntegrityException.actions=log,disable,logout

# rapid validation errors indicate scans or attacks in progress
# org.owasp.esapi.errors.ValidationException.count=10
# org.owasp.esapi.errors.ValidationException.interval=10
# org.owasp.esapi.errors.ValidationException.actions=log,logout

# sessions jumping between hosts indicates session hijacking
IntrusionDetector.org.owasp.esapi.errors.AuthenticationHostException.count=2
IntrusionDetector.org.owasp.esapi.errors.AuthenticationHostException.interval=10
IntrusionDetector.org.owasp.esapi.errors.AuthenticationHostException.actions=log,logout



#===========================================================================
# ESAPI Validation
#
# The ESAPI Validator works on regular expressions with defined names. You can define names
# either here, or you may define application specific patterns in a separate file defined below.
# This allows enterprises to specify both organizational standards as well as application specific
```

```
# validation rules.
#
Validator.ConfigurationFile=validation.properties

# Validators used by ESAPI
Validator.AccountName=^[a-zA-Z0-9]{3,20}$
Validator.SystemCommand=^[a-zA-Z\\-\\/]{1,64}$
Validator.RoleName=^[a-z]{1,20}$
Validator.Redirect=^\\/PRE.*$

# Global HTTP Validation Rules
# Values with Base64 encoded data (e.g. encrypted state) will need at least [a-zA-Z0-9\/+=]
Validator.HTTPScheme=^(http|https)$
Validator.HTTPServerName=^[a-zA-Z0-9_.\\-]*$
Validator.HTTPCookieName=^[a-zA-Z0-9\\-_]{1,32}$
Validator.HTTPCookieValue=^[a-zA-Z0-9\\-\\/+=_ ]*$
Validator.HTTPHeaderName=^[a-zA-Z0-9\\-_]{1,32}$
Validator.HTTPHeaderValue=^[a-zA-Z0-9()\\-=\\*\\.\\?;,+\\/:&_ ]*$
Validator.HTTPServletPath=^[a-zA-Z0-9.\\-\\/_]*$
Validator.HTTPPath=^[a-zA-Z0-9.\\-_]*$
Validator.HTTPURL=^.*$
Validator.HTTPJSESSIONID=^[A-Z0-9]{10,30}$

# Contributed by Fraenku@gmx.ch
# Googlecode Issue 116 (http://code.google.com/p/owasp-esapi-java/issues/detail?id=116)
Validator.HTTPParameterName=^[a-zA-Z0-9_\\-\\[\\]\\.]{1,32}$
Validator.HTTPParameterValue=^[\\p{L}\\p{N}.\\-/+=_ !$*?@]{0,1000}$
Validator.HTTPContextPath=^/[a-zA-Z0-9.\\-_]*$
Validator.HTTPQueryString=^([a-zA-Z0-9_\\-]{1,32}=[\\p{L}\\p{N}.\\-/+=_ !$*?@%]*&?)*$
Validator.HTTPURI=^/([a-zA-Z0-9.\\-_]*/?)*$



# Validation of file related input
Validator.FileName=^[a-zA-Z0-9!@#$%^&{}\\[\\]()_+\\-=,.~'` ]{1,255}$
Validator.DirectoryName=^[a-zA-Z0-9:/\\\\!@#$%^&{}\\[\\]()_+\\-=,.~'` ]{1,255}$

# Validation of dates. Controls whether or not 'lenient' dates are accepted.
# See DataFormat.setLenient(boolean flag) for further details.
Validator.AcceptLenientDates=false
```

7.3 Appendix C: Sample esapi-java-logging.properties

```
handlers=java.util.logging.ConsoleHandler
.level=INFO
```

java.util.logging.ConsoleHandler.level=INFO

java.util.logging.ConsoleHandler.formatter = java.util.logging.SimpleFormatter

java.util.logging.SimpleFormatter.format=[%1$tF %1$tT] [%3$-7s] %5$s %n

#https://www.logicbig.com/tutorials/core-java-tutorial/logging/customizing-default-format.html

7.4 Appendix D: Sample validation.properties file

```
# The ESAPI validator does many security checks on input, such as canonicalization
# and whitelist validation. Note that all of these validation rules are applied *after*
# canonicalization. Double-encoded characters (even with different encodings involved,
# are never allowed.
#
# To use:
#
# First set up a pattern below. You can choose any name you want, prefixed by the word
# "Validation." For example:
#   Validation.Email=^[A-Za-z0-9._%-]+@[A-Za-z0-9.-]+\\.[a-zA-Z]{2,4}$
#
# Then you can validate in your code against the pattern like this:
#     ESAPI.validator().isValidInput("User Email", input, "Email", maxLength, allowNull);
# Where maxLength and allowNull are set for you needs, respectively.
#
# But note, when you use boolean variants of validation functions, you lose critical
# canonicalization. It is preferable to use the "get" methods (which throw exceptions) and
# and use the returned user input which is in canonical form. Consider the following:
#
# try {
#    someObject.setEmail(ESAPI.validator().getValidInput("User Email", input, "Email", maxLength, allowNull));
#
Validator.SafeString=^[.\\p{Alnum}\\p{Space}]{0,1024}$
Validator.Email=^[A-Za-z0-9._%'-]+@[A-Za-z0-9.-]+\\.[a-zA-Z]{2,4}$
Validator.IPAddress=^(?:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)$
Validator.URL=^(ht|f)tp(s?)\\:\\/\\/[0-9a-zA-Z]([-.\\w]*[0-9a-zA-Z])*(:(0-9)*)*(\\/?)([a-zA-Z0-9\\-\\.\\?\\,\\:\\'\\/\\\\\\+=&amp;%\\$#_]*)?$
Validator.CreditCard=^(\\d{4}[- ]?){3}\\d{4}$
Validator.SSN=^(?!000)([0-6]\\d{2}|7([0-6]\\d|7[012]))([ -]?)(?!00)\\d\\d\\3(?!0000)\\d{4}$

Validator.accessControlDb=^[^\*]*$
Validator.commandInjection=^[^\r\n]*$
Validator.crossSiteScriptingPersistent=^[^\r\n]*$
Validator.crossSiteScriptingReflected=^((?!<script).)*$
Validator.denialOfServiceRegExp=^[^\r\n]*$
Validator.jsonInjection=^[^\*]*$
Validator.logForging=^[^\r\n]*$
Validator.openRedirect=^[^\|]*$
```

```
Validator.pathManipulation=^[^\r\n]*$
Validator.portabilityFlawFileSeparator=^[^\*]*$
Validator.portabilityFlawLocale=^[^\r\n]*$
Validator.privacyViolation=^[^\r\n]*$
Validator.sqlInjection=^[^\r\n]*$
Validator.systemInformationLeakExternal=^[^\r\n]*$
Validator.xmlExtEntityInj=^[^\|]*$
```