Oracle Financial Services Analytical Applications Infrastructure

Security Guide

Release 8.0.x

May 2025

ORACLE
Financial Services

ORACLE

Oracle Financial Services Analytical Applications Infrastructure Security Guide

Document Control

Version Number	Revision Date	Change Log
1.0	January 2015	Created the document for security-related configurations in the OFSAA 8.0.0.x.x Release versions.
1..1	April 2015	Added section 3.3 Configuration to restrict file uploads for the Ngan Hang SR 3-10413030421.
2.0	November 2015	Added section 3.4 based on Bug 21810721.
3.0	December 2015	Updated Web Application Server Security Configuration section based on Bug 22070501.
4.0	June 2016	Added content based on Bug 23603150.
5.0	December 2016	• Rectified the broken link in the TLS Configuration for WebLogic section. • Modified the 'Configuration to restrict HTTP methods other than GET/POST' section based on Bug 25308546.
6.0	June 2017	Added section 'Configuring Application Security' for Bug 25957230, 25990244 and 25957206.
7.0	August 2017	Updated for Bug 26568700.
8.0	September 2017	Removed the list of filter servlet keywords and created a MOS document.
9.0	May 2018	Updated for security enhancements in 8.0.6.0.0.
10.0	August 2018	Added back Filter Servlet chapter for Doc 28542034.
11.0	October 2018	Updated for Doc 28672747 and Doc 28771653.
12.0	February 2019	Updated for Doc 29288736, 29352320, and 29352863.
13.0	May 2019	Added a new chapter for Secure Database Connection.
14.0	Aug 2019	• Added generic system configuration information in *Security Configurations* for Doc 30204166. • Added tip to configure from SSLV3 to TLSV1.2 in *Enabling HTTPS Configuration for OFSAA* for Doc 30171443.
15.0	Dec 2019	Updated section *Configuration to set Content Security Policy* with information for validation of web.xml file (Doc 30622153).
16.0	May 2020	Added the *Configure CORS Header* and *Configure HSTS in Response Header* sections.
17.0	June 2020	Updated the *Configure Security for WebLogic* section and added the *Configure X-XSS-Protection Header* section (Doc 31465088).
18.0	October 2021	Added the *Configure Oracle Drivers Recommended in the CPU Releases* section (Doc 33516621).

Version Number	Revision Date	Change Log
19.0	December 2021	Added the *Configure Apache HTTP Server to Stop DDoS, Slowloris, and DNS Injection Attacks* Section (Doc 29115193 and Doc 29113822).
20.0	July 2024	Note added for *Missing Samesite attribute in unused cookie FIC_SESSION* (36738473)
21.0	March 2025	Added configuration for sameSiteCookies in section *Configure Security for Tomcat* (37415835)
21.1	May 2025	Added section *Configure HSTS in Oracle WebLogic Server*.

Table of Contents

Confidential - Oracle Restricted

1 Preface

Oracle Financial Services Analytical Applications Infrastructure (OFSAAI) provides for configuration of security parameters and this guide provides information about the configurations required and how to set it.

The information contained in this document is intended to give you quick exposure and understanding of the security configurations required after the installation of the OFSAAI.

1.1 Audience

This guide is intended for System Administrators (SA) who are instrumental in installing and performing secure configurations for the OFSAAI. It is assumed that the SAs are technically sound and proficient in UNIX, Database Administration, and Web Application Administration to install and configure OFSAAI in the released environment.

1.1.1 Prerequisites for the Audience

This document assumes that you have experience in installing Enterprise components and have a basic knowledge of the following:

- OFSAAAI pack components
- OFSAA Architecture
- UNIX Commands
- Database Concepts
- Web server and web application server

1.2 Reference Documents

This section identifies the following additional documents related to the OFSAA Infrastructure:

- *Oracle Financial Services Advanced Analytical Applications Infrastructure Application Pack Installation and Configuration Guide*
- *Oracle Financial Services Analytical Applications Environment Check Utility Guide*
- *Oracle Financial Services Analytical Applications Infrastructure Administration Guide*
- *Oracle Financial Services Analytical Applications Infrastructure User Guide*

2 Secure Configurations

Configure a set of security parameters to have a secure environment for the OFSAA installation. The required configurations are presented in the following list:

- **Oracle Data Redaction**: Enable protection of data by setting the Oracle Database Advanced Security option. Mask (redact) sensitive data shown to the user in real-time. To enable this option during installation, see *Enabling Data Redaction* section in the *OFS Analytical Applications Infrastructure Installation and Configuration Guide*. To enable this option post-installation, see the *Data Redaction* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **Transparent Data Encryption (TDE)**: Enable this option to secure the data at rest when stored in the Oracle database. To configure TDE during installation, see the *Transparent Data Encryption (TDE)* section in the *OFS Analytical Applications Infrastructure Installation and Configuration Guide*. If you want to configure after installation, see the *Transparent Data Encryption (TDE)* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **Key Management**: The OFSAA configuration schema (CONFIG) is the repository to store passwords for users and application database schemas centrally. These values are AES-256 bit encrypted using an encryption key uniquely generated for each OFSAA instance during the installation process. The OFSAA platform provides a utility (`EncryptC.sh`) to rotate or generate a new encryption key. The *Key Management* section in the *OFS Analytical Applications Infrastructure Administration Guide* explains how to generate and store this key in a Java Key Store.

> **NOTE** Integration with any other Key Management solution is out of the scope of this release.

- **File Encryption**: OFSAA supports file encryption using the AES-256 Bit format. For more information, see the *File Encryption* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **Database Password Reset**: Change the database password for the Config schema and Atomic schema periodically. For more information, see the *Database Password Reset/ Change* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **Password Reset**: Reset passwords for users, if required. For more information, see the *Database Password Reset/ Change* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **Enable and Disable Users**: For more information, see the *Enable and Disable Users* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **SSO Authentication (SAML) Configuration**: For more information, see the *SSO Authentication (SAML) Configuration* section in the *OFS Analytical Applications Infrastructure Administration Guide*.

- **Public Key Authentication**: Configure the Public Key Authentication on UNIX. For more information, see the *Setting Up Public Key Authentication on Client Server* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **Data Security and Data Privacy**: Configure to protect data against unauthorized access and data theft. For more information, see the *Data Security and Data Privacy* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **Input and Output Encoding**: OFSAAI is enabled with input validation and output encoding to protect from various types of security attacks.
- **Password rotation every 30 days**: For more information, see the *Changing Password* section in the relevant version of the *OFS Analytical Applications Infrastructure User Guides*.
- **Additional Cross-Origin Resource Sharing (CORS)**: Configure CORS. For more information, see the *Knowing Additional Cross-Origin Resource Sharing (CORS)* section in the *OFS Analytical Applications Infrastructure Administration Guide*.
- **System Configuration and Identity Management**: Configure the following parameters from the information in the *System Configuration and Identity Management* section in the relevant version of the *OFS Analytical Applications Infrastructure User Guides*:
  - Set session timeout
  - Enable CSRF
  - Set frequency of password change
  - Configure password restriction details
  - Configure password history
  - Configure security questions for a password reset
  - Configure the activation period by setting Dormant Days, Inactive Days, and Working Hours

3 Secure Header Configurations

Secure header configurations protect you from website attacks such as XSS and Clickjacking. The subsections in this topic describe the various methods that you can configure on your OFSAAI system to make it secure from such attacks.

Topics:

3.1 Configure X-XSS-Protection Header

Configure the X-XSS-Protection HTTP header to turn on the XSS Filter and prevent certain categories of XSS attacks. The value `1; mode=block` in the `<param-value>` tag enables the XSS Filter.

To configure the X-XSS-Protection Header, follow these steps:

1. Navigate to the `web.xml` file in the `$FIC_HOME/ficweb/webroot/WEB-INF/`directory.
2. Modify the value in the <param-value> tag to `1;mode=block` as shown in the following:

```
<init-param>
<param-name>XSSProtection</param-name>
<param-value>1;mode=block</param-value>
</init-param>
```

3.2 Configure X-Frame-Options

Configure X-Frame-Options to protect against external agencies creating attacks by embedding content similar to your content and steal user data.

To configure X-Frame-Options, set the following security filters configuration for response header:

`web.xml` file found in the `$FIC_HOME/ficweb/webroot/WEB-INF/` directory is by default configured to set **X-Frame-Options** and header for response header. Add **ALLOW-FROM** for **X-Frame-Options** to limit the domains.

X-Frame-Options

```
<filter>
<filter-name>FilterServlet</filter-name>
<filter-class>com.iflex.fic.filters.FilterServlet</filter-class>
  <init-param>
       <param-name>mode</param-name>
```

```
        <param-value>ALLOW-FROM <URL1>/ <URL2>/</param-value>
    </init-param>
</filter>
```

NOTE

- If ALLOW-FROM is not configured, then the SAMEORIGIN attribute is set in response, where URL1 and URL2 refer to different domain URLs.
- X-Frame-Options is supported only on the Internet Explorer browser.
- Separate <URL1>/ and <URL2>/ with a single space. Adding the URLs without a space between them, or adding two or more spaces between them, results in errors. Make sure that <URL> ends with a forward slash (**/**).

3.3 Configure CORS Header

Configure Cross Origin Request (CORS) to use additional HTTP headers to communicate with browsers to allow a web application running at an origin, access to selected resources from another origin.

Set the Access-Control-Allow-Origin header in the `web.xml` file. For more information, see the *Setting Access-Control-Allow-Origin header* section in the *OFS Analytical Applications Infrastructure Administration Guide*.

3.4 Set Content Security Policy

Content Security Policy (CSP) adds a layer of security to detect and avert website attacks such as Cross-Site Scripting (XSS) and data injection attacks.

NOTE

- This section is applicable for release 8.0.6.3.0 and higher versions in the 8.0.6.x series, and release 8.0.7.1.0 and higher versions in the 8.0.7.x series.
- The configurations to set the Content Security Policy are supported only on Mozilla Firefox and Google Chrome browsers.

To configure CSP, follow these steps:

1. Navigate to the `web.xml` file in the `$FIC_HOME/ficweb/webroot/WEB-INF/` directory.
2. Find the following tag:

```
<context-param>
    <param-name>DOCSERVICE</param-name>
    <param-value>ExternalWSManager</param-value>
```

```
</context-param>
```

3. Add the following tags after the tag in Step 2:

   a. Use the following tag to maintain the default configuration:

```
<context-param>
<param-name>default-src</param-name>
<param-value>default-src 'self'</param-value>
</context-param>
<context-param>
<param-name>script-src</param-name>
<param-value>script-src 'self' 'unsafe-inline' 'unsafe-eval'</param-
value>
</context-param>
<context-param>
<param-name>img-src</param-name>
<param-value>img-src 'self' data:</param-value>
</context-param>
<context-param>
<param-name>style-src</param-name>
<param-value>style-src 'self' 'unsafe-inline'</param-value>
</context-param>
```

WARNING Validate the `web.xml` file and remove any existing duplicate tags to avoid configuration issues.

If you want to maintain the default configuration, retain the tags as shown in the preceding list. However, if you want to custom configure the tags, see the following example and modify as required:

   b. Use the following tag to custom configure the default configuration:

```
<context-param>
<param-name>default-src</param-name>
<param-value>default-src 'self'</param-value>
</context-param>
<context-param>
<param-name>script-src</param-name>
<param-value>script-src <SCRURL> 'self' 'unsafe-inline' 'unsafe-
eval'</param-value>
</context-param>
<context-param>
```

```
<param-name>img-src</param-name>
<param-value>img-src <IMGURL> 'self' data:</param-value>
</context-param>
<context-param>
<param-name>style-src</param-name>
<param-value>style-src <CSSURL> 'self' 'unsafe-inline'</param-value>
</context-param>
```

In the previous example, define the policy by replacing:

- default-src: with no value. This value sets to self.
- <SCRURL>: with the URL of the script that you want to allow to run, which prevents any other script from running.
- <IMGURL>: with the image URLs from trusted sources from which you want to load images and prevent images from untrusted sources.
- <CSSURL>: with the URL of the stylesheet to allow styles from the specified stylesheet and to prevent styles from other sources.

3.5 Configure Referrer Header Validation

Referrer Header Validation protects against CSRF attacks by allowing validated host URLs.

To configure Referrer Header validation, follow these steps:

1. Navigate to the `web.xml` file in the `$FIC_HOME/ficweb/webroot/WEB-INF/` directory.
2. Add the following tag:

```
<filter>
<filter-name>FilterServlet</filter-name>
<filter-class>com.iflex.fic.filters.FilterServlet</filter-class>
<init-param>
  <param-name>AllowHosts</param-name>
  <param-value><URL1>/ <URL2>/</param-value>
</init-param>
</filter>
```

> **NOTE** Separate <URL1> and <URL2> with a single space. Adding the URLs without a space between them or adding two or more spaces between them results in errors. Make sure that <URL> ends with a forward slash (**/**).

3.6 Configure HSTS in Response Header

Set the HTTP Strict Transport Security (HSTS) in the response header to allow server application interaction with only the client over Hypertext Transfer Protocol Secure (HTTPS). Configure the response header field `Strict-Transport-Security` through the Oracle HTTP Server (OHS).

To configure HSTS, follow these steps:

1. Open the OHS conf file `httpd.conf` in the `$INSTANCE_HOME/INSTANCE_NAME/config/OHS/INSTANCE_NAME/` directory.
2. Add the following in the file and save it:

   ```
   Header set Strict-Transport-Security: max-age=63072000;
   includeSubdomains;
   ```

3. Restart OHS.

3.7 Configure HSTS in Oracle WebLogic Server

For information on how to configure HSTS in Oracle WebLogic Server, see the My Oracle Support Document (Doc ID *2146367.1*).

4 Web Application Server Security Configurations

Depending on your configured web application server, see the following sections. Alternatively, you can see your web application server-specific administration guide for additional details.

Topics:

4.1 Enable HTTPS Configuration for OFSAA

HTTPS is recommended during OFSAA installation, by default. This configuration creates an encrypted environment and functions as a secure environment for client-server communication.

> **TIP** See the *HTTPS Protocol* section in the relevant version of the *OFS Analytical Applications Infrastructure Administration Guides* on the *OHC*.
>
> - To enable **HTTPS** post-installation.
> - To view configurations related to **SSLv3** and **TLS1.2**.

4.2 Configure Security for Tomcat

To set security configurations for Tomcat, follow these steps:

1. Add the preferred cipher list to Tomcat and update the value of **sslProtocol** to **TLS 1.2** in the SSL Connector tag of the `$CATALINA_HOME/conf/server.xml` file.
2. Ciphers attribute can be added to the Connector tag in the `server.xml` file as shown in the following example.

> **TIP** Multiple cipher suites must be comma-separated.

For example,

```
ciphers="TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384”
```

For more details on TLS1.2 supported ciphers and recommendations, see the following links:

- *https://www.owasp.org/index.php/Securing_tomcat*
- *https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet#Rule_-_Only_Support_Strong_Cryptographic_Ciphers*

3. Add the following session attributes under the 'Context' tag of the $CATALINA_HOME/conf/server.xml file.

```
sessionCookiePath= "<context>"
sessionCookieDomain= "<domain>"
```

> **NOTE** <context> is OFSAAI context and <domain> is the domain name of the server that must receive the cookie. For example, if the application is accessed through the URL app.mysite.com, then it should be set to app.mysite.com and not mysite.com.

4. Configure for secure and HttpOnly using the following procedure:
   a. In the $CATALINA_HOME/conf/context.xml file, add the 'useHttpOnly=true' attribute to 'Context' tag.
   b. Add the secure="true" attribute to the 'Connector' tag section of the $CATALINA_HOME/conf/server.xml file.
   c. Add the following tags to the session-config section of the $CATALINA_HOME/conf/web.xml file.

```
<cookie-config>
        <http-only>true</http-only>
        <secure>true</secure>
</cookie-config>
```

5. Configure for sameSiteCookies using the following procedure:

   In the $CATALINA_HOME/conf/context.xml file, add CookieProcessor element to 'Context' tag on following line for setting sameSiteCookies in HTTP response header's set-cookie.

```
<CookieProcessor
className="org.apache.tomcat.util.http.LegacyCookieProcessor"
sameSiteCookies="strict" />
```

6. Open the $CATALINA_HOME/conf/server.xml file and add a tag as shown in the following:
   a. Search for the "<Host name=" parameter in the file.
   b. Add the following tag:

```
<Valve className="org.apache.catalina.valves.ErrorReportValve"
showReport="false" showServerInfo="false" />
```

7. Disable the directory listing in the $CATALINA_HOME/conf/web.xml file. Add the following lines to the servlet section:

```
<init-param>
      <param-name>listings</param-name>
      <param-value>false</param-value>
</init-param>
```

8. Restart the Tomcat service.

4.3 Configure Security for WebSphere

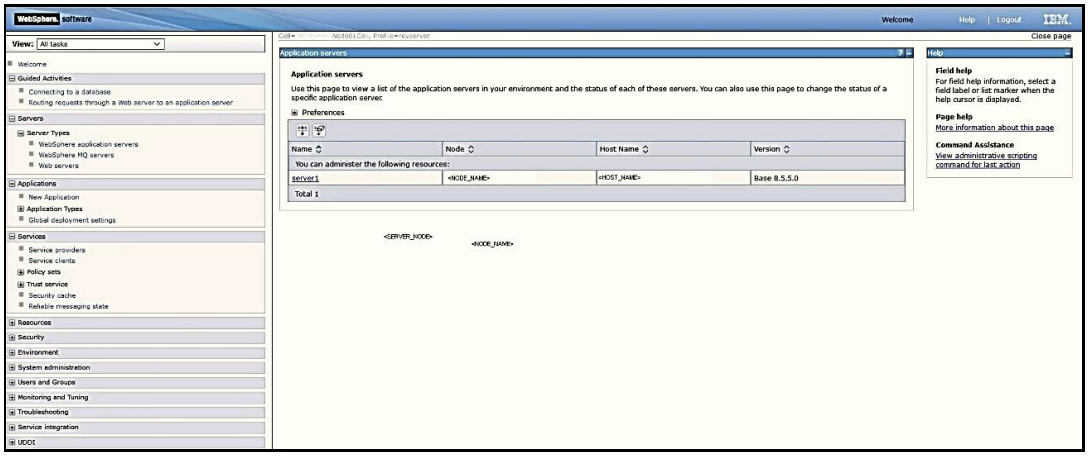
In the WebSphere Admin console, restrict cookies to HTTPS sessions in Sessions Management Configuration, specify the JSESSIONID variable in the Web Container Settings, set TLS configuration, and configure the application security. The subsections describe the procedures in detail.

4.3.1 Configure "Secure and HttpOnly" in Session Management

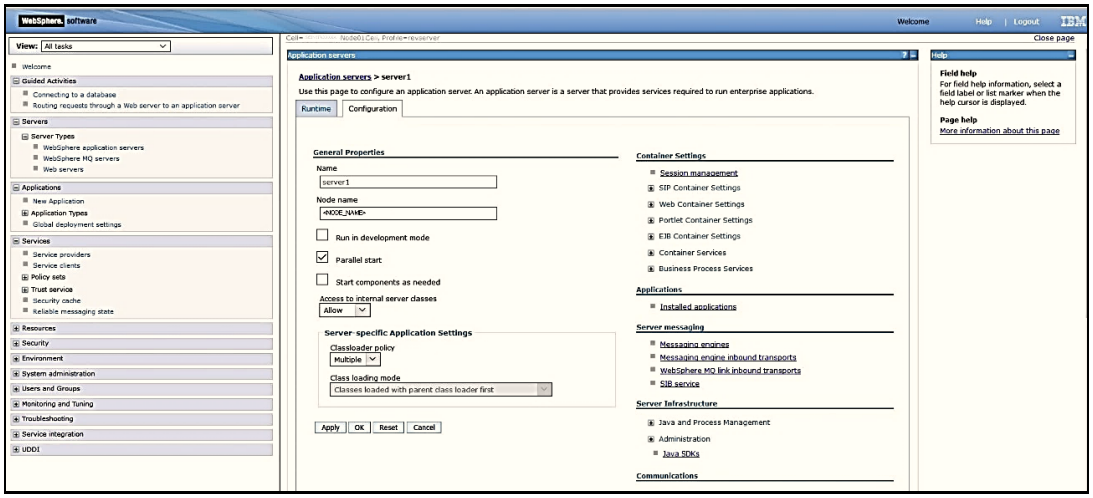
In Session Management Configuration, restrict cookies to HTTPS Sessions.

To set the session management configuration, follow these steps:

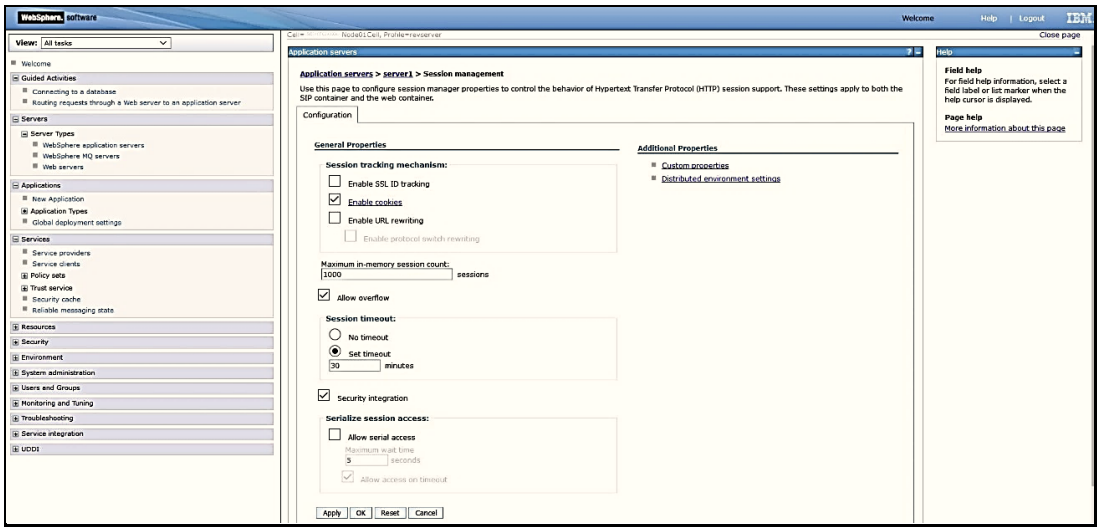
1. Navigate to the WebSphere Admin console and in the Navigation Tree, select **Server**, select **Server Types** and then select **WebSphere application servers**.
2. Select the configured Application Server from the list by clicking on the **Server Name**.



3. Select **Configuration** and then select **Session Management** from **Container Settings**.



4. In **General Properties**, select **Enable Cookies**.



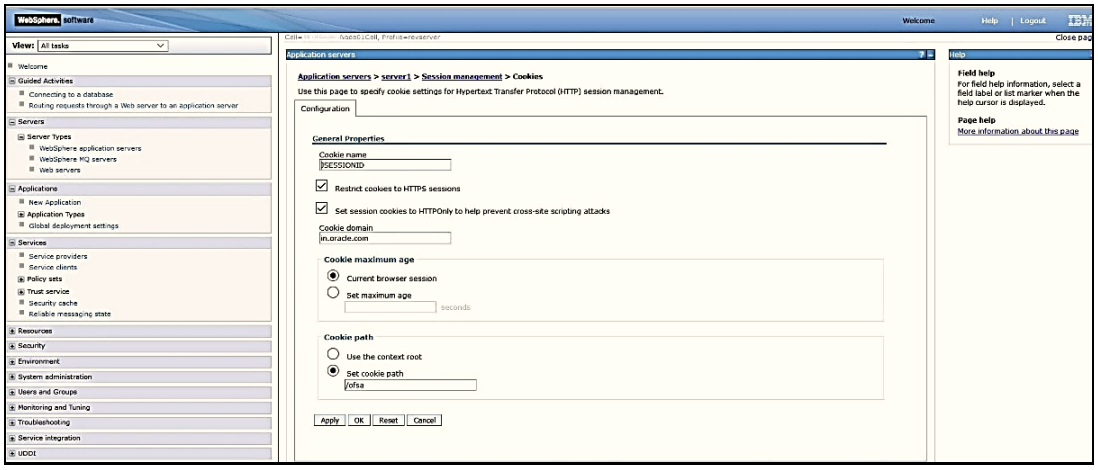
5. Enter the following details:

   `Cookie Name - JSESSIONID`

   `Cookie domain - <domain>`

   `Cookie Path - /<context>/`

> **NOTE** <context> is OFSAAI context and <domain> is the domain name of the server that receives the cookie. For example, if the application is accessed through the URL `app.mysite.com`, then set it to `app.mysite.com` and not `mysite.com`.



6. Ensure the following checkboxes are selected:
   - **Restrict Cookies to HTTPS Sessions**
   - **Set session cookies to HTTPOnly to prevent cross-site scripting attacks**
7. Click **Apply** and save changes.
8. Restart the Application Server through the console.

4.3.2 Configure TLS for WebSphere

To configure the TLS protocol in WebSphere, follow these steps:

1. Log in to the console (*http://host:adminport/ibm/console*).
2. In the **Security** menu, select **SSL certificate and key management**, select **SSL configurations**, select **NodeDefaultSSLSettings**, and then select **Quality of protection (QoP)** settings.
3. Change the **Protocol** value to `TLSv1.2`.

The preceding configuration ensures that the WebSphere server accepts only TLSv1.2 connections. That is, when the web server acts as a server (inbound) or as a client (outbound) the SSL connections are established through the TLSv1.2 protocol. When testing from a browser, ensure to check that the browser settings are set to initiate only TLS handshakes.

For more information, see *Configuring WebSphere Application Server to support TLS 1.2*.

For cipher suite configuration, see *https://www.ibm.com/support/knowledgecenter/en/linuxonibm/liaag/wascrypt/l0wscry00_wasciphersuite.htm*

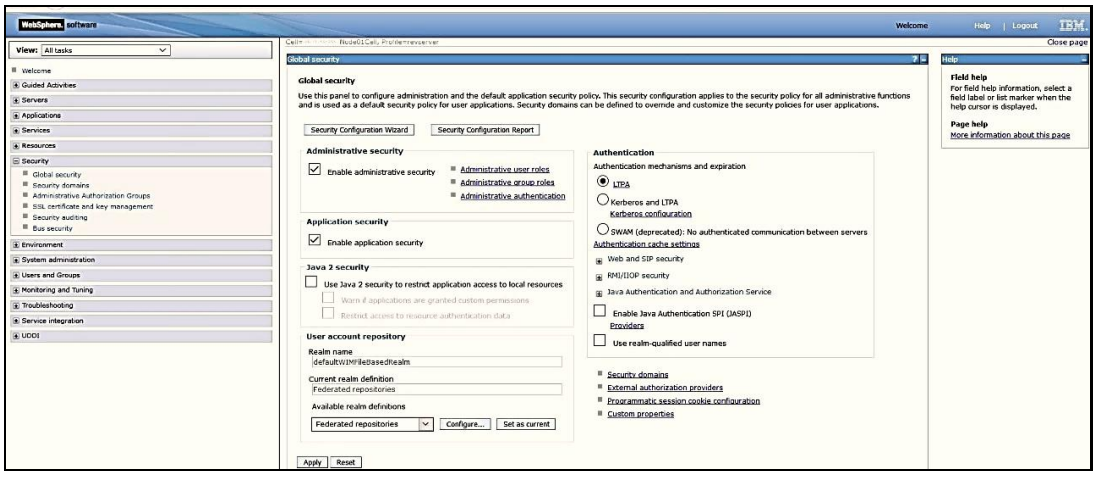
For more details about strong cipher configuration, see *https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet#Rule_-_Only_Support_Strong_Cryptographic_Ciphers*.

4.3.3 Configure Application Security

Enable Application security to secure your server from unauthorized users and allow access only to authenticated users. This prevents unauthorized access to configuration files in directories.

To enable Application security, follow these steps:

1. Log in to WebSphere with administrator credentials.
2. Select **Security** from the tree and then select **Global security** to display the **Global security** window.
3. Select **Enable administrative security** and **Enable application security**.



4. Click **Apply** and save the configuration.

4.3.4 Disable Directory Listing

> **NOTE** This section is applicable for release 8.0.6.0.0 and later.

Directory listing is disabled by default. In other words, **directoryBrowsingEnabled** is set to **false**.

For detailed information, see the IBM WebSphere User Documentation.

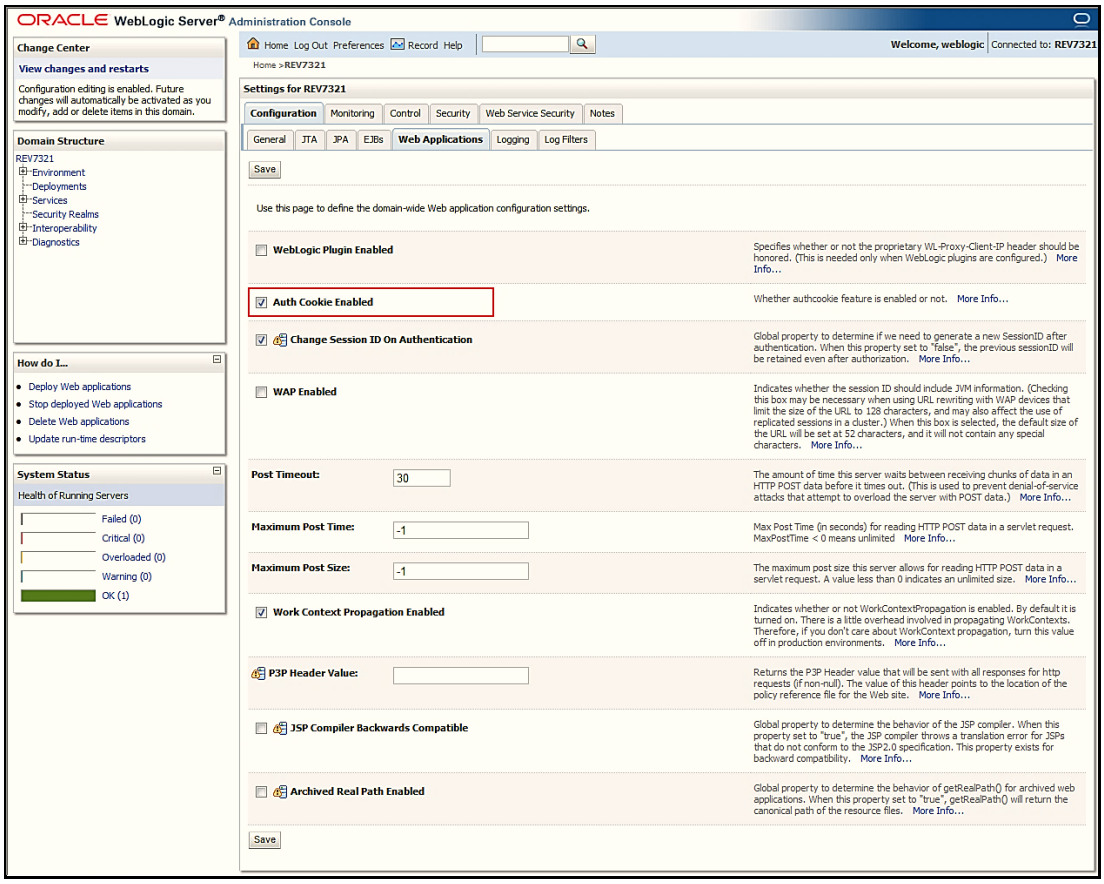
4.4 Configure Security for WebLogic

In the WebLogic Server, though the **Auth Cookie Enabled** option is selected by default, the cookies are not secure. To ensure this, you must toggle the "Auth Cookie Enabled" option in WebLogic console by disabling it first and then re-enabling it for secure cookies. After that, create a `weblogic.xml` file in the `$FIC_HOME/ficweb` directory and the `deploy.ear` file in your WebLogic server.

> **NOTE** In case customer getting additional cache called `FIC_SESSION` contact Oracle support.

To configure security for WebLogic, follow these steps:

1. Log in to WebLogic Server Administrative Console.
2. Select **Domain Structure** and then select **Domain** from the tree.
3. Select **Configurations** (selected by default), and then select the **Web Application**.



4. Scroll through the configurations options within the page and locate the **Auth Cookie Enabled** option. By default, the checkbox is selected.
5. Unselect **Auth Cookie Enabled** and click **Save**.
6. Select **Auth Cookie Enabled** and click **Save**.
7. Configure session **Secure and HttpOnly**.
   - For OFSAAI versions earlier to 8.0.2.0.0, configure as shown in the following:
     - Create the `weblogic.xml` file in the `$FIC_HOME/ficweb/webroot/WEB- INF/` directory and add the following tags.

```
<weblogic-web-app xmlns="http://www.bea.com/ns/weblogic/90">
<session-descriptor>
<cookie-name>JSESSIONID</cookie-name>
<cookie-domain><domain></cookie-domain>
<cookie-path>/<context></cookie-path>
<cookie-http-only>true</cookie-http-only>
<cookie-secure>true</cookie-secure>
</session-descriptor>
</weblogic-web-app>
```

- For OFSAAI versions 8.0.2.0.0 or higher, modify the `weblogic.xml` file in the `$FIC_HOME/ficweb` directory and add the following tag under the root element:

```
<session-descriptor>
<cookie-name>JSESSIONID</cookie-name>
<cookie-domain><domain></cookie-domain>
<cookie-path>/<context></cookie-path>
<cookie-http-only>true</cookie-http-only>
<cookie-secure>true</cookie-secure>
</session-descriptor>
```

> **NOTE** <context> is OFSAAI context and <domain> is the domain name of the server that must receive the cookie. For example, if the application is accessed through URL `app.mysite.com`, then it should be set to `app.mysite.com` and not `mysite.com`.

8. Configure TLS protocol for WebLogic using the following steps:
   a. Add the following parameters in `setDomainEnv.sh` present under `/domains/<DomainName>/bin` as arguments for JAVA_OPTIONS: `-Dweblogic.security.disableNullCipher=true -Dweblogic.security.SSL.protocolVersion=TLS1.2`
   b. Add the preferred cipher suite to the `config.xml` file as shown in the following example. Use only strong cryptographic ciphers recommended for TLS 1.2.

      Example:

```
<ssl>
<name><servername></name>
<enabled>true</enabled>
<ciphersuite> TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384</ciphersuite>
...
</ssl>
```

      For more information, see *https://docs.oracle.com/middleware/1213/wls/SECMG/standards.htm#SECMG743*

      For more details about strong cipher configuration, see *https://www.owasp.org/index.php/Transport_Layer_Protection_Cheat_Sheet#Rule_-_Only_Support_Strong_Cryptographic_Ciphers*.

9. Disable directory listing. Add the following tag under `<container-descriptor>` in `$FIC_HOME/ficweb/weblogic.xml`:

```
<index-directory-enabled>false</index-directory-enabled>
```

10. Enable REST API authorization by OFSAA. Follow these steps:

a. Open the config.xml file located in the domain where OFSAA is deployed, that is, `<domain_home>/config/config.xml`.

b. Add the following in the security-configuration tag:

```
<enforce-valid-basic-auth-credentials>false</enforce-valid-basic-auth-credentials>
```

11. Build the `.ear` file and deploy it onto the WebLogic server.
12. Restart the services.

5 Additional Security Configurations

This section explains how to perform additional security configurations.

Topics:

5.1 Configure to Restrict Access to the Default Web Server Pages

To set the configurations to restrict access to default web server pages in the Apache Tomcat server, follow these steps:

1. Start the Apache Tomcat server by executing the command `startup.sh`.
2. Log in to the **Tomcat Web Application Manager**.
3. Undeploy the **Examples** application from Tomcat:

   Go to the **Tomcat Web Application Manager** window and select **Remove** corresponding to the Tomcat Examples application.
4. Shut down the Apache Tomcat Server by executing the `shutdown.sh` file.
5. Comment the following sections from the `%CATALINA_HOME%/conf/server.xml` file (if available).

   Section I

```
<Context path="/examples" docBase="examples" debug="0"
     reloadable="true" crossContext="true">
<Logger className="org.apache.catalina.logger.FileLogger"
     prefix="localhost_examples_log." suffix=".txt"
     timestamp="true"/>
<Ejb   name="ejb/EmplRecord" type="Entity"
      home="com.wombat.empl.EmployeeRecordHome"
      remote="com.wombat.empl.EmployeeRecord"/>
```

   Section II

```
<Environment name="maxExemptions" type="java.lang.Integer"
        value="15"/>
<Parameter name="context.param.name" value="context.param.value"
```

```
        override="false"/>
<Resource name="jdbc/EmployeeAppDb" auth="SERVLET"
        type="javax.sql.DataSource"/>
<ResourceParams name="jdbc/EmployeeAppDb">
      <parameter><name>user</name><value>sa</value></parameter>
      <parameter><name>password</name><value></value></parameter>
      <parameter><name>driverClassName</name>
      <value>org.hsql.jdbcDriver</value></parameter>
      <parameter><name>driverName</name>
      <value>jdbc:HypersonicSQL:database</value></parameter>
</ResourceParams>
<Resource name="mail/Session" auth="Container"
      type="javax.mail.Session"/>
<ResourceParams name="mail/Session">
      <parameter>
      <name>mail.smtp.host</name>
      <value>localhost</value>
      </parameter>
</ResourceParams>
      <ResourceLink name="linkToGlobalResource"
              global="simpleValue"
              type="java.lang.Integer"/>
</Context>
```

6. Delete the `%CATALINA_HOME%\webapps\ROOT\index.jsp` file.
7. Create a blank `%CATALINA_HOME%\webapps\ROOT\index.html` file.
8. Comment the following tags in the `%CATALINA_HOME%\conf\web.xml` file:

   ```
   <welcome-file>index.htm</welcome-file>
   <welcome-file>index.jsp</welcome-file>
   ```

9. Change the default passwords of Tomcat users in the `%CATALINA_HOME%\conf\tomcat-users.xml` file.

   Following are some examples:

   ```
   <user username="both" password="b$12" roles="tomcat,role1"/>
   <user username="tomcat" password="t$12" roles="tomcat"/>
   <user username="admin" password="a$12" roles="admin,manager"/>
   <user username="role1" password="r$12" roles="role1"/>
   ```

5.2 Configure to Restrict Display of the Web Server Details

To set the configurations to restrict the display of the web server details from http responses, follow these steps:

- Modify the `/httpd.conf` file and set:
  - "ServerTokens" parameter to "Prod"
  - "ServerSignature" parameter to "off"

5.3 Configure to Restrict File Uploads

The Restrict File Uploads configuration restricts the upload of files for certain file types. This configuration is applicable for all OFSAAI UIs and applications that are rendered out of the platform's OJET component.

The following is an example of the Restrict File Uploads configuration:

DOCUMENT_ALLOWED_EXTENSION: txt, pdf, doc, html, htm, xls, zip, jar, xml, jpg, bmp, and jpeg.

The parameter DOCUMENT_ALLOWED_EXTENSION in the CONFIGURATION table of the "configuration" schema holds the list of file extensions for valid file types that are allowed to be attached and uploaded into the OFSAA applications. Attached files that do not have an extension as listed in this parameter value are blocked. This list is extendable.

5.4 Configure to Restrict HTTP Methods Other Than GET and POST

To set the configuration required to restrict HTTP methods other than GET and POST, follow these steps:

1. Modify the `httpd.conf file` of HTTP Server (Apache HTTP Server/Oracle HTTP Server/IBM HTTP Server)

```
RewriteEngine On
RewriteCond %{REQUEST_METHOD} !^(GET|POST)
RewriteRule .* - [R=405,L]
```

2. If the application is not configured with HTTP Server for WebLogic and WebSphere application servers, follow these steps:
   a. Add the following snippet to the `$FIC_HOME/ficweb/webroot/WEB-INF/web.xml` file.

```
<security-constraint>
     <web-resource-collection>
     <web-resource-name>restricted methods</web-resource-name>
         <url-pattern>/*</url-pattern>
         <http-method>PUT</http-method>
         <http-method>PATCH</http-method>
         <http-method>HEAD</http-method>
```

```
        <http-method>DELETE</http-method>
        <http-method>OPTIONS</http-method>
        <http-method>TRACE</http-method>
       <http-method>CONNECT</http-method>
      </web-resource-collection>
     <auth-constraint/>
</security-constraint>
```

b. Navigate to the `$FIC_WEB_HOME` directory in the OFSAA installed server.

c. Execute the `./ant.sh` command to regenerate the `<CONTEXTNAME>.ear/.war` file.

d. Redeploy the EAR/WAR file onto your configured web application server. For more information on deploying the EAR / WAR file, refer to the *Post Installation Configuration* section in *OFS Analytical Applications Infrastructure Installation and Configuration Guide*.

5.5 Configure Enable Unlimited Cryptographic Policy for Java

Enable unlimited cryptographic policy for Java to use AES-256 keys for encryption. For more information, see the *Enabling Unlimited Cryptographic Policy* section from the *OFS Analytical Applications Infrastructure Administration Guide*.

5.6 Configure Apache HTTP Server to Stop DDoS, Slowloris, and DNS Injection Attacks

To prevent Distributed Denial of Service (DDoS), Slowloris, and DNS Injection attacks on Apache HTTP Servers, you must implement specific techniques such as Request Timeout and Quality of Service Extension.

For details, see the following sections.

- *Configure Request Timeout*
- *Configure Quality of Service Extension to Mitigate Slow HTTP DoS Attacks*

5.6.1 Configure Request Timeout

You can configure the Request Timeout (set timeouts to receive HTTP Request Header and HTTP Request Body from a Client) values in the **mod_reqtimeout** module that is included by default in the Apache HTTP Server v2.2.15 and later versions. If the Client does not send the Header or Body data within the configured time, the Server displays a 408 Request Timeout error message.

The following example shows the configuration to set to allow the Client a maximum time of 30 seconds to start sending the Header data and the maximum time limit set is 45 seconds. The example also shows that the Client must transfer the Header data at the rate of 600 bytes per second. Similarly, the Client must start the transfer the Body data within 40 seconds, transfer within 60 seconds, and at a rate of 700 bytes per second.

```
<IfModule mod_reqtimeout.c>
```

```
  RequestReadTimeout header=30-45,MinRate=600 body=40-60,MinRate=700
</IfModule>
```

5.6.2 Configure Quality of Service Extension to Mitigate Slow HTTP DoS Attacks

You can configure the Quality of Service Extension in the mod_qos module of the Apache HTTP Server to set the priorities for specific HTTP Requests.

The following example shows the configuration to set to mitigate slow HTTP DoS attacks. The configuration settings allows the Server to handle up to 110000 connections and limits each IP address to a maximum of 60 connections. It limits the requests to a maximum of 300 connections and disables the HTTP KeepAlive parameter when 240 connections are in use. It also shows that the configuration requires a minimum of 100 bytes per second per connection and limits the connection to 1500 bytes per second when MaxClients reaches its set limit.

```
<IfModule mod_qos.c>
  # handle connections from up to 110000 different IPs
  QS_ClientEntries 110000
  # allow only 60 connections per IP
  QS_SrvMaxConnPerIP 60
  # limit the maximum number of active TCP connections to 300
  MaxClients 300
  # disables keep-alive when 240 (80%) TCP connections are occupied
  QS_SrvMaxConnClose 240
  # minimum request/response speed
  # (deny clients that keep connections open without requesting
anything)
  QS_SrvMinDataRate 100 1500
</IfModule>
```

6 Configure Oracle Drivers Recommended in the CPU Releases

Oracle strongly recommends that you apply the Critical Patch Update (CPU) Security Patches as soon as possible. Apply the Security Patches mentioned in the July 2021 CPU and later CPUs to ensure that the Database Servers and Clients use Enhanced Network Encryption.

For information about the encryption enhancement, see the My Oracle Support Document (Doc ID *2791571.1*).

> **NOTE** For details about Critical Patch Updates, Security Alerts, and Bulletins, see *https://www.oracle.com/security-alerts/*.

After applying the CPUs, configure the Oracle Drivers as follows:

1. Remove all occurrences of the **ojdbc8.jar** file from the `$FIC_HOME` directory and replace it with the **ojdbc8.jar** file from the $ORACLE_HOME/jdbc/lib directory.

   To find all occurrences of the **ojdbc8.jar** file from the `$FIC_HOME` directory, execute the following command:

   ```
   find $FIC_HOME \( -name "ojdbc8.jar" \) -print
   ```

2. Remove the **oraclepki.jar**, **osdt_cert.jar**, and **osdt_core.jar** files from the following directory-locations:

   ```
   $FIC_HOME/ficapp/common/FICServer/lib
   $FIC_HOME/ficapp/icc/lib
   $FIC_HOME /realtime_processing/WebContent/WEB-INF/lib
   $FIC_HOME/ficweb/webroot/WEB-INF/lib
   $FIC_HOME/ficdb/lib
   ```

3. Copy the **oraclepki.jar**, **osdt_cert.jar**, and **osdt_core.jar** from the `$ORACLE_HOME/jlib` directory to the following directory-locations:

   ```
   $FIC_HOME/ficapp/common/FICServer/lib
   $FIC_HOME/ficapp/icc/lib
   $FIC_HOME /realtime_processing/WebContent/WEB-INF/lib
   $FIC_HOME/ficweb/webroot/WEB-INF/lib
   $FIC_HOME/ficdb/lib
   ```

4. *Redeploy the EAR/WAR File*.

7 Redeploy the EAR/WAR File

The redeployment of the EAR/WAR is required if there is any modification in the `$FIC_HOME/ficweb/webroot/WEB-INF/web.xml` file.

To redeploy, follow these steps:

1. Navigate to the `$FIC_WEB_HOME` directory in the OFSAA installed server.
2. Execute the `./ant.sh` command to regenerate the `<CONTEXTNAME>.ear/.war` file.
3. Redeploy the EAR/WAR file onto your configured web application server.

   For more information on deploying the EAR / WAR file, refer to the Post Installation Configuration section in *OFS Analytical Applications Infrastructure Installation and Configuration Guide*.

8 Secure Database Connection Configurations

The Oracle database product supports SSL/TLS connections in its standard edition (since 12c). The Secure Sockets Layer (SSL) protocol provides network-level authentication, data encryption, and data integrity. When a network connection over SSL is initiated, the client and server perform a handshake that includes:

- Negotiating a cipher suite for encryption, data integrity, and authentication
- Authenticating the client by validating its certificate
- Authenticating the server by verifying that its Distinguished Name (DN) is expected
- Client and server exchange key information using public key cryptography

To establish an SSL connection, the Oracle database sends its certificate, which is stored in a wallet. Therefore, on the server, the configuration requires a wallet. And on the client, the JDBC thin driver can use different formats to store the client's certificate and key. For example, JKS, Wallet, or PKCS12.

This document provides details about the steps to establish an SSL connection over TLSv1.2 using the JDBC thin driver with Oracle wallet having storetype as SSO with OraclePKIProvider.

8.1 Configure to Connect OFSAA to the Oracle Database Using a Secure Database Connection (TCPS)

For the documentation, see the *Configurations for Connecting OFSAA to Oracle Database using Secure Database Connection (TCPS)* section in the *OFS Analytical Applications Infrastructure Administration Guide*.

9 Appendix A - Servlet Filter Configurations

Servlet Filter is a controller in the web-container with the Servlet Filter required configurations. This section also lists out the Keywords and Key Characters.

> **NOTE** This section is applicable for releases 8.0.0.x.x to 8.0.5.x.x.

Topics:

9.1 Security and Access

When users try to access a web page, this functionality checks whether they have the required permissions and rights to access it.

9.2 Vulnerability Checks

This function checks for intrusion and Cross-site Scripting vulnerability. Currently, this check is for the following group of keywords/key characters:

- JavascriptKeyWords - paramname in configuration table XSS_JS_KEYWORDS1 to XSS_JS_KEYWORDS13
- JavascriptKeyChars - paramname in configuration table XSS_JS_METACHARS1 to XSS_JS_METACHARS10
- SQLKeyWords - paramname in configuration table XSS_SQL_KEYWORDS1 to XSS_SQL_KEYWORDS23.
- SQLWords - paramname in configuration table XSS_SQL_WORDS1 to XSS_SQL_WORDS4
- SQLTOKENS- XSS_SQL_TOKENS1 to XSS_SQL_TOKENS8
- SQLOPERATORS- XSS_SQL_OPERATORS1 to XSS_SQL_OPERATORS5 are present in the Configuration table.

9.3 Cross-Site Scripting

A Cross-Site Scripting vulnerability check is triggered if the HTTP request contains a combination of any `JavascriptKeyWords` with the `JavascriptKeyChars`.

For example, if an HTTP request contains a combination of any of the `JavascriptKeyWords` (such as **Return**, **Alert**, **Script**, **JavaScript**, or **VBScript**) along with any of `JavascriptKeyChars` (Meta Chars) such as “, ‘, (,), ;, <, >, {, or }, then the request is blocked, and an error message is displayed.

See the My Oracle Support Document (Doc ID *2311605.1*) containing the **PARAMNAME**, **PARAMVALUE**, and **DESCRIPTION** to view the list of keywords and key characters scoped for filtering.

9.4 SQL Injection

An SQL Injection vulnerability check filters multiple combinations of SQLKeyWords and SQLWords.

For example, if an HTTP request contains a combination of any of the disallowed SQLWords (such as From, Into, Where, table) with any of the SQLKeyWords (such as **Alter**, **Insert**, **Select**, **Create**, **Update**, **Delete**, **Drop**, **Truncate**) for XSS check, then the request is blocked, and an error message is displayed.

To view the list of keywords and key characters scoped for filtering, see the My Oracle Support Document (Doc ID *2311605.1*) containing the **PARAMNAME**, **PARAMVALUE**, and **DESCRIPTION**.

9.5 Configure Servlet Filter

Configure the Servlet Filter to process requests and responses after checks for vulnerability, exclude certain keywords and characters, and modify the debug and log directories.

9.5.1 Check for XSS Vulnerability

The following entry is available in the configuration table present in the Configuration Schema. The **Cross-site Checks** are not performed if the entry is not present or the **PARAMVALUE** is **FALSE**. By default, **PARAMVALUE** is set to “**TRUE**”.

PARAMNAME	PARAMVALUE	DESCRIPTION
XSS_IS_CHECK_REQUIRED	TRUE	The parameter to decide whether the XSS check is to be enabled or not.

9.5.2 Exclude Keywords and Key Characters

Exclude the evaluation of a keyword by adding a new **PARAMNAME** with **PARAMVALUE** and a **DESCRIPTION** (optional) to the configuration table. The ending numeral in the new **PARAMNAME** should be higher than any other number in the group.

For example, to exclude the evaluation of JS keyword “return”, which has the PARAMNAME XSS_JS_KEYWORDS**1**, you must update the keyword numeral to XSS_JS_KEYWORDS**12** considering the table has 11 other keywords listed under this category. Ensure that the updated number is higher than any other number in the group.

9.5.3 Modify Debug and Logs Directories

When the application detects a vulnerability, a message is displayed on the front-end and it is logged in the `CSSLogger.log` file. By default, the `CSSLogger.log` file is generated in the `<deployed context>/logs` directory. It contains details of date, time, URL, and user.

You can modify the configuration to create the `CSSLogger.log` file in a directory of your choice. Enter the directory path for the CSS Logger file in the place holder **CSS_LOGGER_PATH** given in the `$FIC_WEB_HOME/webroot/conf/FICWeb.cfg` file.

OFSAA Support

Raise a Service Request (SR) in *My Oracle Support (MOS)* for queries related to OFSAA applications.

Send Us Your Comments

Oracle welcomes your comments and suggestions on the quality and usefulness of this publication. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most about this manual?

If you find any errors or have any other suggestions for improvement, indicate the title and part number of the documentation along with the chapter/section/page number (if available) and contact the Oracle Support.

Before sending us your comments, you might like to ensure that you have the latest version of the document wherein any of your concerns have already been addressed. You can access My Oracle Support site that has all the revised/recently released documents.