KERNEL AUTHENTICATION & AUTHORIZATION FOR J2EE (KAAJEE) VERSION 1.2.0

FOR WEBLOGIC (WL) VERSIONS 10.3.6 AND HIGHER

RELEASE NOTES

August 2020

Department of Veterans Affairs (VA)
Office of Information and Technology (OI&T)
Product Development

This page is left blank intentionally.

Contents

This page is left blank intentionally.

1. Introduction

1.1. Orientation

Significant KAAJEE and Kaajee Security Provider release objects (JARs, EARs, WARs, etc.) contain a build number in the filenames (e.g., in "kaajee-1.2.0.xxx.jar" where "xxx" is the build number). Build numbers are increments with each new Developer Preview. Consult the following table for KAAJEE build increment details.

1.2. About KAAJEE, KAAJEE Classic

Kernel Authentication & Authorization for Java 2 Enterprise Edition (KAAJEE) for Web-based Health*e*Vet applications on WebLogic 10.3.6 and higher, such as Pharmacy Re-Engineering (PRE), provides user authentication to grant access to applications, and retrieves VistA security keys for application authorization. KAAJEE takes advantage of the current user store in VistA to authenticate users in new Health*e*Vet applications. Considered an interim solution, KAAJEE will be enhanced to provide the connection to the Department of Veterans Affairs (VA) Enterprise user authentication (sign-on) system in the future. KAAJEE and Fat-client Kernel Authentication and Authorization (FatKAAT) use VistALink as a connectivity solution from Java 2 Platforms, Enterprise Edition (J2EE) applications to Mumps (M)/VistA.WebLogic Updates Project.

KAAJEE 1.2 is the successor to KAAJEE 1.1, which was originally released to support the BEA WebLogic Server 9.2 platform. Version 1.2 of KAAJEE now supports Oracle WebLogic Server 10.3.6 and higher, and is fully Fortify and TRM compliant.

Like KAAJEE 1.1, KAAJEE 1.2 provides a custom Authentication Provider "plug-in" for BEA WebLogic J2EE servers, including a set of web forms for web applications that allow web applications to authenticate and authorize an end-user using a Kernel system as the source of authentication and authorization. A form is presented upon protected resource request, gathering user A/V codes and presenting a list of pre-determined stations. Such functionality will be referred to as “KAAJEE Classic”, as to denote original implementation of a complete authentication process tied to VistA. Since VA has moved towards two-factory authentication mechanisms, a successor in KAAJEE family has sprung – Single Sign-On Web Application Plugin, SSOWAP. It is important to denote the differences between KAAJEE Classic and KAAJEE SSOWAP, notwithstanding User Interface similarities, the authentication mechanisms and deployment requirements are very different.

1.3. Distribution

Version 1.2 of KAAJEE Classic implements all current features of KAAJEE 1.1. Included in the distribution are the following:

- KAAJEE 1.2 jar files
- Folder of Login web forms for Forms Authentication to drop into a J2EE Web application
- JavaDoc Application Programming Interface (API) documentation for the user manager classes
- Deployment Descriptor examples
- Complete source code for the KAAJEE Classic implementation

- Sample Web Application

  /kaajee-1.2.0.xxx contains the following sub-folders:

  - /dd_examples -- This folder contains sample deployment descriptors.
  - /doc -- This folder contains a readme.txt file.
  - /jars -- This folder contains the KAAJEE JAR file and related JSP files.
  - /javadoc -- This folder contains the KAAJEE javadocs.
  - /samples -- This folder contains the KAAJEE Sample Web application and its dependencies
  - /samples
  - /sso-ccow -- This folder contains elements to make the KAAJEE Sample Web application Single Sign-On (SSO CCOW) enabled.
  - /src -- This folder contains the KAAJEE Java source code.

1.4. For More Information

For more information on any new feature discussed in the Release Notes, please see the KAAJEE documentation set.

This page is left blank intentionally.

2. New Features / Changes for KAAJEE Version 1.2

2.1. Upgraded Deployment Descriptors:

All KAAJEE deployment descriptors have been upgraded as recommended by the WebLogic 10.3.6 and higher documentation. It is recommended that consuming applications do the same.

For KAAJEE, the sample WebLogic deployment descriptors were changed from referencing DTD based document to a Schema based document.

KAAJEE product line is fully Fortify and TRM compliant. All the dependencies have been updated with the latest TRM-approved versions.

2.2. Design Changes to Better Support Additional SSL Use Cases

Previously, KAAJEE only supported the Secure Sockets Layer (SSL) use case where SSL was used on the KAAJEE login page and controller servlet only. Therefore, applications needing other SSL use cases such as SSL on their protected application pages and/or welcome page would have required other means to provide SSL (e.g., a load balancer such as Big-IP from F5 with the SSL Acceleration Module, which can provide encrypting to SSL and decrypting to non-SSL).

In addition to supporting only limited use cases for SSL, previous KAAJEE versions' redirect code to submit j_username and j_password to j_security_check was limited. The protocol (http or https) used for the targeted application page was dependent on whether the redirect was performed with a relative link vs. a complete URL reference. Also, the target page's protocol was dependent on how the security-constraints were defined in web.xml.

The former redirect code that redirected to the "magic servlet" j_security_check has been replaced with a call to BEA WebLogic's ServletAuthentication.authenticate API. Upon successful authentication of the user, KAAJEE redirects to the original targeted URL (including original protocol) of the protected application page. The protocol (http or https) for this page is only subject to modification by any security-constraints defined in web.xml.

Furthermore, the previous SSL-related redirects have been removed from login.jsp. In addition, the <ssl-listen-port-number> tag from the KAAJEE configuration file (i.e., kaajeeConfig.xml) has been removed. Therefore, there is no longer a need to edit the KAAJEE configuration file for configuring SSL. Instead, use the '<transport-guarantee>' sub-element in web.xml to configure SSL. See the following for details.

Listed below are the SSL use cases that KAAJEE now supports:

- **Supported SSL Use Case** 1: No SSL. The welcome/non-protected page (if any), login page, and target protected page are in non-SSL mode.
- **Supported SSL Use Case 2**: Welcome/non-protected page is in SSL. All subsequent pages including KAAJEE login page remain be in SSL by default. All referenced pages will remain in SSL as long as relative links are used while logged in that don't specify/change the protocol back to http. Downgrade to non-SSL is possible only if a link explicitly specifies a protocol change to

http, and if the page is not protected with 'CONFIDENTIAL' in the transport-guarantee tag in web.xml.

- **Supported SSL Use Case 3**: KAAJEE login page only will be in SSL and target application protected page will be in non-SSL.
- **Supported SSL Use Case 4:** Regardless of the protocol used for the welcome/non-protected page, KAAJEE login page and target application protected page are now in SSL. Once logged into SSL, all subsequent pages after the target 'CONFIDENTIAL' page remain be in SSL by default. As long as relative links are used while logged in that don't specify/change the protocol back to http, all referenced pages will remain in SSL. Downgrade to non-SSL is possible only if a link explicitly specifies a protocol change to http, and if the page is not protected with 'CONFIDENTIAL' in the transport-guarantee tag in web.xml.

Listed below is the use case that is NOT supported:

- KAAJEE login page is in non-SSL and target application protected page is in SSL.

2.3. Configure Web.xml to Enable SSL

To enable SSL for a particular protected page, in web.xml, for the <security-constraint>' element whose <web-resource-collection> contains that page, set the value of the '<transport-guarantee>' sub-element to CONFIDENTIAL, within the '<user-data-constraint>' sub-element.Generally, form-based authentication would handle both authentication and authorization. Beginning with this build 005, KAAJEE only implements the user interface part of form-based authentication. The back-end security check is replaced with the ServletAuthentication.authenticate API. Therefore, all authorization failures are handled solely by container security. As such all users who are not authorized to access the targeted page after login will receive an http '403' error. To provide a more user-friendly error message, KAAJEE now distributes a 'loginerror403.jsp' file. The consuming application may use this page or another of their choosing. To use this page, add an '<error-page>' entry in web.xml similar to the one listed below:

```
<error-page>
    <error-code>403</error-code>
    <location>/login/loginerror403.jsp</location>
</error-page>
```

2.4. Integration of Kernel Patch XU*8*451: KAAJEE Login Page—Removal of Refresh Button.

Kernel Patch XU*8*451 provides the following functionality and bug fixes:

- Enhanced Login Functionality:
  - Removed Refresh button from KAAJEE login page.
  - Added JavaScript code for client-side sorting of Institutions.
  - Provided Access and Verify code capability in one line.
  - Added support for parameter passing of Default Institution and Institution sorting preferences. This addresses the issues of persistent cookies when using Thin Clients and Terminal Servers.
  - Made the KAAJEE Login Web page more Section 508 friendlier.

- Added KAAJEE Sample Web Application.
- Updated Software Version Support:
  - Compiled and tested KAAJEE against Standard Data Service (SDS) 18.0.
  - Compiled and tested KAAJEE against VistALink 1.6.1.xxx.
- Fixed Response already committed error. The code that was fixed was associated with processing the persistent cookie information on the Application Server. This fix should also fix the extra M process that was created.

2.5. Administrator Security Role for LoginController

The security role assignment has been upgraded in both web.xml and weblogic.xml. Change the weblogic.xml file in the KAAJEE server application to use KAAJEE in the run-as-principal-name element. It is important that the "KAAJEE" user account (principal) has administrative privileges. If a different principal is used for this purpose, the weblogic.xml file will have to reflect this change.

The format of identifying the cookie name in the <session-descriptor> in weblogic.xml has changed for WebLogic 10.3.6. The changes to each document are as follows:

- web.xml:

```
<servlet>
  <servlet-name>LoginController</servlet-name>
  <servlet-
class>gov.va.med.authentication.kernel.servlet.LoginController
  </servlet-class>
  <run-as>
    <role-name>adminuserrole</role-name>
  </run-as>
</servlet>

<security-role>
  <role-name>adminuserrole</role-name>
</security-role>
```

- weblogic.xml:

```
<run-as-role-assignment>
<role-name>adminuserrole</role-name>
<run-as-principal-name>KAAJEE</run-as-principal-name>
</run-as-role-assignment>

<security-role-assignment>
<role-name>AUTHENTICATED_KAAJEE_USER</role-name>
<principal-name>AUTHENTICATED_KAAJEE_USER</principal-name>
</security-role-assignment>

<session-descriptor>
  <cookie-name>kaajeeJSESSIONID</cookie-name>
</session-descriptor>
```

NOTE: The Classpath on WebLogic 10.3.6 and higher Managed Servers, which is located on the Start Server tab, must now include the classpath to the KAAJEE SSPI JAR file. (i.e., wlKaajeeSecurityProviders-1.2.0.005.jar) in case of 1.2 implementation. KAAJEE SSPI 1.3 does not have that requirement.

2.6. Sign-On/User Context (SSO/UC)-Enabled

Developers now have the option to make KAAJEE-based applications Single Sign-On/User Context (SSO/UC)-enabled. See the KAAJEE Deployment Guide for the details on how to enable your application. The current implementation of single sign-on is based on the user context of the Clinical Context Object Workgroup (CCOW) standard.

KAAJEE architecture will now allow users to authenticate and sign on to multiple applications that are CCOW-enabled and Single Sign-On/User Context (SSO/UC)-aware using a single set of credentials, which will reduce the need for multiple IDs and passwords.

There is an SSO two-factor authentication implementation - SSO Web Application Plugin– KAAJEE SSOWAP. It is a separate distribution and is not part of this package.

2.7. Section 508 Issue Regarding Session Timeouts

KAAJEE now displays an alert dialogue box, warning the end-user how much time remains before the login session expires. This warning is displayed 30 seconds prior to the expiration of the user's login session. In order to provide this warning, KAAJEE utilizes JavaScript. Therefore, KAAJEE distributes a login.js file, which is exported as part of the login\javascript\ folder.

2.8. Dependency Upgrades

- WebLogic: KAAJEE is now dependent on WebLogic 10.3.6 and higher.
- VistALink: KAAJEE is now dependent on VistaLink 1.6.1
- Java: KAAJEE is now dependent on Java 1.7

This page is left blank intentionally.