ellucian.

Application Navigator
Installation Guide

Release 3.0
March 2018

Ellucian
2003 Edmund Halley Drive
Reston, VA 20191
United States of America

Contents

Introduction

This installation guide details the steps that are required to install the following components of Application Navigator 3.0. It includes the configuration of seamless navigation functionality for Banner 8.x forms and Banner 9.x web applications.

Before you install any components of the system, you should review this introduction thoroughly so you have a better understanding of what you are installing and where you will install it.

Seamless Navigation and Application Navigator

Application Navigator is a software component that facilitates seamless navigation between Banner 8.x and Banner 9.x Administrative and Self-Service applications. It includes a unified menu, unified search, and other common mechanisms to provide a consistent, integrated user experience for disparate applications. Seamless navigation is not a product or even a specific software component. It is a user experience facilitated by common software components and enhancements to existing applications.

Use Ellucian Solution Manager to install your product upgrades

Ellucian recommends that you use Ellucian Solution Manager to perform Banner product upgrades, rather than using a manual installation process.

With Solution Manager, you can:

- Identify dependency information within and between products.
- View the latest version numbers for the Banner products you have installed, along with all other version numbers installed in your environment.
- Use the **Get New Releases** feature to identify available upgrades and download them immediately.
- Identify and install product prerequisites, along with any upgrades you have selected.

Solution Manager currently supports most Banner 8 and 9 products. For more information on the Banner product versions currently supported by Solution Manager, see the *Banner Upgrades Support Status* guide. For detailed instructions on how to install and configure Solution Manager, see the latest *Solution Manager User Guide*.

Integration of Banner with Application Navigator

Applications are integrated with Application Navigator by exposing an endpoint for menu data and by participating in cross-document messaging. The menu endpoint is used to retrieve menu data to display to users as a hierarchical, navigable structure, or as the result of a user-initiated search. Cross-document messaging is used to keep the integrated applications and Application Navigator synchronized as users open and close forms and pages, make changes to data, navigate to another page, or attempt to sign out.

Banner administrative menu items maintained on Menu Maintenance Form (GUTGMNU) are exposed to Application Navigator through a single menu service. This menu service is embedded in the `applicationNavigator.war` file, which must be configured to connect to the appropriate database. Instructions for configuring the WAR file are provided later in this document.

The data retrieved using the menu service notifies the Application Navigator on how to open a requested page or form. For Banner 9.x Administrative applications, this information is derived from settings maintained on Object Maintenance Form (GUAOBJS) and Banner9 Module and Page Maintenance Form (GUAPAGE). Instructions on updating these settings and other configuration settings in the database are documented in the subsequent sections of this guide.

Cross-document messaging requires configuration of the applications that will be integrated with Application Navigator. Instructions on configuring the Banner forms environment, enhancing Banner 8.x forms, and configuring Banner 9.x modules are documented in the subsequent sections of this guide.

For Banner 9.x Self Service applications, the menu data is derived from the URL values that are configured in Web Tailor and in the Application Navigator configuration file. This 1-1 relationship ensures the correct self service menu data is populated for the user’s role. Instructions on configuring Banner 9.x Self Service applications are documented in the subsequent sections of this guide.

Hardware and software requirements

Application Navigator is a stand-alone Grails web application that must be deployed on an application server. This section documents the hardware and software requirements for the Application Navigator.

Hardware requirements

The application has the following hardware requirements.

CPU and memory

The number of CPU cores, memory, and configuration for application servers running Banner 9 applications is dependent on the number of expected concurrent users working on the system. Optimal performance sizing and configuration vary per institution, but the following information provides a good starting point:

Quad Core CPU with 6 to 8 GB of memory for the application server.

Please refer to the current version of the *Banner 9 Sizing and Configuration Guide* in the Banner 9 Sizing and Configuration document library in the Ellucian Support Center for more information.

Screen resolution

The minimum PC screen resolution for the application is 1024 x 768.

Software requirements

The application has the following software requirements.

Oracle Database

Supported versions of the Oracle Database depend on multiple factors, including third-party support time lines. For a complete list of supported Oracle technologies, refer to the Ellucian Oracle Support Calendar. The calendar is available in the *Interactive Banner Compatibility Guide*, which can be accessed from the Ellucian Download Center.

Application server

The application is supported on the following application servers:

- Oracle Fusion Middleware 11gR2 using WebLogic 10.3.4, 10.3.5, 10.3.6, and 12.1.3
- Apache Tomcat 7x and 8x

Oracle Fusion Middleware (OFM) consists of several software products, including WebLogic Server. WebLogic Server is required for an Oracle Banner 9.x application server environment.

No other OFM products are required. However, if an SSL-enabled Oracle HTTP Server (OHS) port is used, the Oracle Web Tier should also be installed in order to use the `mod_wl_ohs`.

Middle Tier (application server) platforms

The application is supported on the following application server and operating system combinations:

Tomcat (64 bit)	WebLogic (64 bit)
Red Hat Linux 5.x, 6.x, 7.x	Red Hat Linux 5.x, 6.x, 7.x
Windows Server 2008, 2012	Windows Server 2008, 2012
Solaris 10	Solaris 10
AIX 6.1 (JDK 1.7)	AIX 6.1 (JDK 1.7)
HP-UX	HP-UX 11iV3 (11.31)

Note: Application Navigator was tested on WebLogic using both the Classic Domain template and the Basic Domain template.

For WebLogic server environments, JPA 2.0 support must be enabled. WebLogic server does not enable JPA by default. To enable JPA, use the steps in the appropriate Oracle documentation:

WebLogic 10.3.4:
http://docs.oracle.com/cd/E17904_01/web.1111/e13720/using_toplink.htm#i1221315

WebLogic 10.3.5:
http://docs.oracle.com/cd/E21764_01/web.1111/e13720/using_toplink.htm#EJBAD1309

WebLogic 10.3.6:
http://docs.oracle.com/cd/E23943_01/web.1111/e13720/using_toplink.htm#autoId2

WebLogic 12.1.3
JPA 2.0 is supported by default in Weblogic 12c Server. However, you can install a patch that provides support for the Java Persistence Architecture (JPA) 2.1 when you use Oracle TopLink as the persistence provider.
http://docs.oracle.com/middleware/1213/wls/WLJCA/EJBAD/using_toplink.htm#CIHBAGDH

Ellucian software

The following product upgrade must be applied: Banner General 8.9.

For Banner 8.x Self-Service support with Application Navigator, Web Tailor 8.8.3 is required.

Banner 8.x Form Changes

The Seamless Navigation Enhancement Utility for forms must be run against all the Banner 8.x forms (`.fmb` files). This utility enables the global variables and values to communicate between Banner 8.x forms and Banner 9.x applications. The instructions on how to run the utility are included in the `SeamlessNavigationEnhancementUtility.zip` file that is available for download from the Ellucian Support Center.

Single sign on (SSO) support

Application Navigator requires an external authentication service to validate user credentials and provide access to its capabilities and to integrated applications.

Currently, CAS and SAML 2.0 are the Single Sign-on (SSO) protocols supported by Application Navigator. All integrating applications must be configured to use either of these SSO protocols for authentication services, with the help of a supported centralized Identity Management System.

Note: Banner administrative and Self-Service applications (8.x and 9.x) require CAS or SAML 2.0 for Single Sign-on. Application Navigator has been certified to run on CAS and SAML 2.0 SSO protocols with Ellucian Ethos Identity.

Note: SSO for Banner 8.x INB forms and 8.x Self Service Pages also requires SSO Manager, a component of Banner Enterprise Identity Services (BEIS).

Ellucian provides a centralized Identity Management System called Ellucian Ethos Identity that is available for download from the Ellucian Support Center. Ellucian Ethos Identity supports many industry-standard SSO protocols, such as CAS, SAML 2.0, OAuth, OpenID, and so on. Ellucian Ethos Identity provides a single authentication page for end users and single sign on (SSO) for applications that recognize the supported protocols. Application Navigator has been tested and certified with Ellucian Ethos Identity as the primary Identity Management System for SAML 2.0 support.

We recommend that you first establish the single sign on environment before implementing Application Navigator. For information on establishing a single sign on environment, refer to the *CAS Single Sign On Handbook* or the *Setting Up Ellucian Ethos Identity* document, available for download from the Ellucian Support Center.

Supported browsers

The following browsers are supported with the application:

- Chrome
- Firefox
- Internet Explorer 11 and Edge
- Safari 11 and higher

For more information about supported browsers, refer to the Ellucian Browser Support Calendar. The calendar is available in the Interactive Banner Compatibility Guide on the Ellucian Download Center.

Browser support for NPAPI (technology required for Java applets)

The Java plug-in for web browsers relies on the cross platform plugin architecture NPAPI, which has been supported by all major web browsers for over a decade. Google's Chrome version 45 (scheduled for release in September 2015) dropped support for NPAPI, impacting plugins for Silverlight, Java, Facebook Video and other similar NPAPI based plugins.

Java applications are offered through web browsers as either a web start application (which do not interact with the browser once they are launched) or as a Java applet (which might interact with the browser). This change does not affect Web Start applications, it only impacts applets.

If you have problems accessing Java applications using Chrome, Oracle recommends using Internet Explorer (Windows) or Safari (Mac OS X) instead.

Mozilla announced that by the end of 2016, Firefox will no longer support NPAPI plugins. For as long as Firefox does support NPAPI, Ellucian will continue to test and support Firefox for our Banner INB releases. However, once Firefox drops support for NPAPI, Ellucian will no longer be able to support it. Ellucian advises customers to use Internet Explorer 11 for accessing Oracle Forms based applications (Banner 8.x INB).

Chrome Compatibility Mode

When using Chrome, users must disable Compatibility Mode. If Compatibility Mode is not disabled, errors may occur.

To disable Compatibility Mode in Chrome, perform the following steps:

1. Right-click the Chrome shortcut and select **Properties**.
2. Select the **Compatibility** tab.
3. Clear the **Run this program in compatibility mode for:** check box.

4. Click **Apply**.
5. Click **OK**.

Supported screen readers

The following screen readers are supported with the application:

- JAWS 18 (18.0.4321) – IE11, Firefox on Windows
- JAWS 17 (17.0.221) – IE11, Firefox on Windows
- VoiceOver – Safari on OS X High Sierra (10.13.3)
- VoiceOver – Safari on iPad iOS (11.2.x)

Supported handheld devices

The following handheld devices were tested for minimum support with the application:

- Apple iPad and iPad Mini with iOS 11.2.x
- Google Nexus 9 with Android OS 7.1.1
- Apple iPhone 7/8 with iOS 11.2.x
- Samsung Galaxy S7/S8 with Android OS 7.0

Java dependencies

Install the Java 7 or Java 8 JDK (64-bit version) on the application server before you install the application.

The JDK bin directory must be defined in the PATH system property.

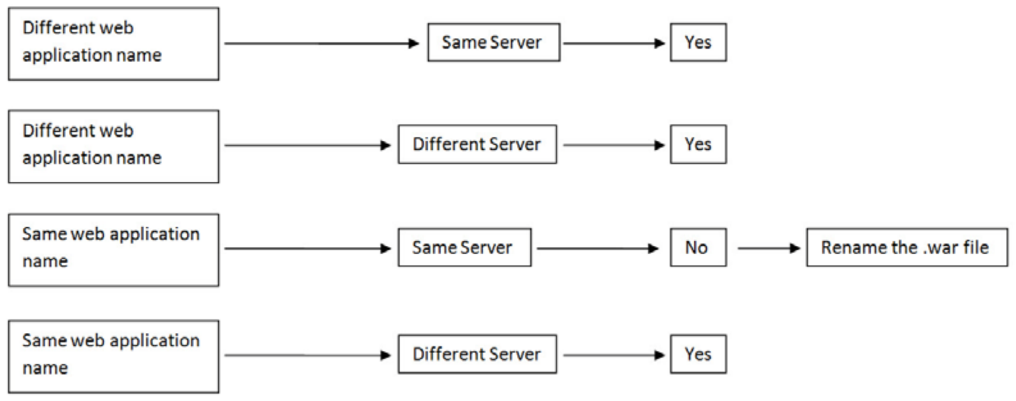
The same version of Java must be used to customize and deploy the WAR file.

Note: Do not install Java 6 JRE. The application does not support Java 6 JRE. The application supports Java 7 or Java 8 JDK and JRE in run time.

Note: Java 7 or Java 8 includes security restrictions for Rich Internet Applications. Refer to Article 000030656 on the Ellucian Support Center for details on Java security restrictions with Liveconnect calls to Oracle Forms Applet.

Deployment of multiple web applications

The following diagram describes various scenarios of deploying multiple web applications:


Different web application name
Same Server
Yes
Different web application name
Different Server
Yes
Same web application name
Same Server
No
Rename the .war file
Same web application name
Different Server
Yes

In the first and second scenarios, you can deploy multiple web applications with different WAR file names on the same or different servers.

In the third scenario, if you want to deploy multiple web applications on the same server, the WAR file names must be different.

In the fourth scenario, you can deploy multiple web applications with the same WAR file name on different servers.

Navigation among 9.x applications

Seamless navigation between Banner 8.x and Banner 9.x Administrative applications and Self-Service applications can be accomplished by using Application Navigator.

For integrating Administrative applications, you must enable this functionality on the GUAPAGE page. This page is used to update the GUBMODU_URL field on the GUBMODU table. This field provides the context information that is associated with a menu object. The GUBMODU_URL must be in the following format:

```
http://<host name>:<port>/<application context name>/
```

For integrating Self-Service applications, you must configure the applications' URLs in Web Tailor. Instructions on how to update the configuration entries in the Application Navigator configuration file are documented in the subsequent sections of this guide.

With CAS or SAML2 SSO

If CAS or SAML2 Single Sign On (SSO) is enabled, a user is authenticated only for the first log in. Subsequently, the user can access and navigate among any Banner 9.x applications. When SSO is enabled and configured between Banner administrative and Self-Service applications, when a user logs out of one application, the user is logged out of all applications that are open.

Use the following procedure to navigate among Banner administrative and Self-Service applications when integrated with Application Navigator using CAS or SAML2 SSO:

1. Log in to Application Navigator using a valid SSO user name and password.
2. Access any Banner administrative or self-service application.
3. Navigate to another Banner 9.x administrative or self-service application. You can navigate among these applications without prompting for a second login.

Refer to the *CAS Single Sign On Handbook* or the *Setting Up Ellucian Ethos Identity* document, available on the Ellucian Support Center, for details on configuring Banner Applications with Single Sign-on for multiple protocols (SAML 2.0, CAS, etc.).

Without SSO

If Single Sign On (SSO) is not enabled, a user must authenticate before accessing each Banner 9.x application. If SSO is not enabled when a user logs out of an application, the user is logged out of that current application. The user is still logged in to all other applications that are currently open.

F5 load balancer configuration

The application was tested using an F5 load balancer configured with the following settings:

```
Load Balancing type = Round Robin

Persistence = Cookie
```

Note: Other configurations may be supported depending on Network Load Balancing (NLB).

Checklists

Detailed installation checklists are available as appendices to this guide. Refer to the "Appendix - CAS Installation Checklist" on page 112 or the "Appendix - SAML 2.0

Installation Checklist” on page 114.

Upgrade the Database

Update login.sql

You must edit the `login.sql` script to update the schema owner's default password and to specify the path to create log files. To update the delivered `login.sql` script, perform the following steps:

1. Replace the #UPDATEME# string with the value of a particular schema owner's password in your environment. Make this update in your environment for each Banner schema owner.
2. Set the value that gets assigned to `splpref`. The value can be set to the `ORACLE_SID` or to a directory name. Your options depend on the operating system.

   The `splpref` variable defines the file prefix that the installation process uses to generate listings or intermediate SQL routines. This feature allows you to segregate the generated output when the stage must be applied to more than one instance.

Verify that the required products are applied

To check that all prerequisite products are applied to the environment, perform the following steps:

1. Invoke SQL*Plus and run the following procedure:

   ```
   sqlplus /nolog @ruappready
   ```

2. Review the `ruappready` listing.

Run the Database Extension Utility

Note: If the Database Extension Utility (DBEU) has been executed during the Banner DB Upgrade 9.x installation, you can skip this section.

If you have not executed the Banner DB Upgrade 9.x on your Banner database, perform the following steps:

1. Invoke SQL*Plus and run the following procedure:

   ```
   sqlplus dbeu_owner/<dbeu_owner_password>
   ```

```
start dbeu_ext_AppNav
```

2. Review the `dbeu_ext_AppNav` listing.

Verify the banproxy database account

The `banproxy` account is used for database connections for administrative applications. The database upgrade process grants the `BAN_DEFAULT_M` role to `banproxy`. If this role is revoked, the application will not start successfully.

Verify Oracle user accounts to connect through banproxy

All Internet Native Banner (INB) or Oracle user accounts must connect using the `banproxy` privilege. To verify that Oracle user accounts can connect through `banproxy`, perform the following steps:

1. Access the Security Maintenance (GSASECR) page.
2. Enter a valid user name.
3. Click **Alter**.
4. Select the **Authorize banproxy** check box.
5. Click **Save**.

Migrate staged files to the permanent directories

This release provides migration scripts for Unix and Windows platforms. These scripts expect your directory structure to match the directory structure created by the Banner installation process. If you choose a different directory structure, you must modify the scripts. The release does not include migration scripts for other platforms due to their highly customized structures. You can, however, use the file `GENMIGR.TXT` as a starting point for writing your own migration scripts.

Unix

The file `GENMIGR.TXT` lists all files that must be deleted from your permanent directories, and all files that should be copied from the staging directory to your permanent directories. The destination is indicated in UNIX format. The format is different on other platforms.

The file `genmigr.shl` does the appropriate removes, copies, and links. The local LN variable at the top of `genmigr.shl` determines the type of links that are used in the migration. If you want to use symbolic links, set `LN='ln -s'` so that the command `${LN} file $BANNER_HOME/links` is translated to `ln -s file $BANNER_HOME/links`. If you want to force the removal of any existing targets before linking files, set `LN='ln -f'`.

To run the migration script in background on a Unix platform, perform the following steps:

1. Ensure that the directory path names in `genmigr.shl` are correct.
2. Ensure that the environment variable `$BANNER_HOME` in `genmigr.shl` is set to the appropriate directory.
3. Sign on to an operating system account that has write permission into the target Banner directories.
4. If you are a cshell user (your operating system prompt is a percent sign), enter `sh` and press Enter to enter the Bourne shell.
5. Navigate to the staging directory for the product.
6. Run the migration script as follows:

   ```
   sh genmigr.shl >genmigr.log 2>&1 &
   ```
7. If you were a cshell user and want to return to that mode, press CTRL-D or enter `exit`. Then press Enter.
8. Review `genmigr.log`. This file contains the results of the migration.

   Note: Even if your directory structure matches the baseline perfectly, some link commands will fail (that is, where the link currently exists). Other link errors might indicate that you had two copies of an object when the migration script was executed. This condition must be corrected. The duplication is probably between links and the product subdirectory.

Windows

The file `genmigr.pl` does the appropriate deletes and copies. To run the migration script on a Windows platform, perform the following steps:

1. Check the value of the BANENV environment variable by executing the SET command from the DOS prompt.
   - If the value of BANENV is REG, the value used for `BANNER_HOME` will be taken from the registry entry:

     ```
     HKEY_LOCAL_MACHINE\SOFTWARE\BANNER\BANNER_HOME
     ```
   - If the value of `BANENV` is ENV, the value used for `BANNER_HOME` will be taken from the environment variable `BANNER_HOME`.
2. Ensure that the directory path names in `genmigr.pl` are correct.

3. Sign on to an operating system account that has write permission into the target Banner directories.
4. Navigate to the staging directory for the product.
5. Run the migration script as follows:

   ```
   perl genmigr.pl >genmigr.log 2>&1
   ```
6. Review `genmigr.log`. This file contains the results of the migration.

Update the version number

To insert the release version number into the Web Application Table (GURWAPP), perform the following steps:

1. Invoke SQL*Plus and run the following procedure:

   ```
   sqlplus general/password
   start versionupdate.sql
   ```
2. Review the `versionupate` listing.

Enable Seamless Navigation for Banner 8.x Forms

Run the Seamless Navigation Enhancement Utility

Run the *Seamless Navigation Enhancement Utility* for forms against all the Banner 8.x forms (`.fmb` files). This utility enables the global variables and values for communication between Banner 8.x forms and Banner 9.x applications. The instructions for running the utility are included in the `SeamlessNavigationEnhancementUtility.zip` file that is available for download from the Ellucian Support Center.

Note: Banner 8.x form changes noted in the following sections are specifically referenced from an Oracle Fusion Middleware (OFM) server (11.1.1.4). If you are using a higher version of Oracle Fusion Middleware server (11gR1 or 11gR2), the directory paths and file locations are subject to change based on your OFM installation.

Update the Oracle WebUtil HTML files

Change the webutilbase.htm file

To make the required change to the `webutilbase.htm` file, perform the following steps:

1. Locate the `webutilbase.htm` file in the following directory:

   `$MIDDLEWARE_HOME/asinst_1/config/FormsComponent/forms/server`

2. Backup the `webutilbase.htm` file.

   Note: This creates a copy of the original file before any modifications are performed.

3. In the `webutilbase.htm` file, add the lines indicated by (`<----ADD`):

```
.....
.....
.....
```

```
<HEAD>
<script type="text/javascript" src="%integrationXE%"></script> <----ADD
<script type="text/javascript" src="%integrationBannerXE%"></script> <----ADD
<TITLE>%pageTitle% - WebUtil</TITLE></HEAD>
.....
.....
.....
```

Change the webutiljpi.htm file

To make the required change to the `webutiljpi.htm` file, perform the following steps:

1. Locate the `webutiljpi.htm` in the following directory:

   `$MIDDLEWARE_HOME/asinst_1/config/FormsComponent/forms/server`

2. Backup the `webutiljpi.htm` file.

   Note: This creates a copy of the original file before any modifications are performed.

3. In the `webutil.jpi.htm file`, add the lines indicated by (`<----ADD`).

```
.....
.....
.....
<HEAD>
<script type="text/javascript" src="%integrationXE%"></script> <----ADD
<script type="text/javascript" src="%integrationBannerXE%"></script> <----ADD
<TITLE>%pageTitle% - WebUtil</TITLE></HEAD>
.....
.....
.....
```

Change the forms_base_ie.js file

To make the required change to the `forms_base_ie.js file`, perform the following steps:

1. Locate the `forms_base_ie.js` file in the following directory:

   `$MIDDLEWARE_HOME/user_projects/domains/ClassicDomain/servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/e18uoi/war/frmjscript`

   Note: The folder location `e18uoi` specified in the directory path may change to a different location such as `spn4wj` or `wb1h9f` based on your OFM installation.

2. Make a backup of `forms_base_ie.js` file. For example, `forms_base_ie.js.ORIG`.

3. In the `forms_base_ie.js` file, add the lines indicated by (`<----ADD`):

```
.....
.....
.....
var jheight = plugin_info.getAttribute("appheight");
var name = plugin_info.getAttribute("appname");

document.write('<APPLET CODEBASE="' + jcodebase + '"\n');
document.write('CODE="oracle.forms.engine.Main"\n');
document.write('ARCHIVE="' + jarchive + '"\n');
document.write('WIDTH="' + jwidth + '"\n');
document.write('HEIGHT="' + jheight + '"\n');
document.write('ID="' + name + '"\n');          <----ADD
document.write('NAME="' + name + '" MAYSCRIPT>\n');
.....
.....
.....
```

Change the banner8_integration.js file

To make the required change in the `banner8_integration.js` file, perform the following steps:

1. Locate the `banner8_integration.js` file.

   Note: The `banner8_integration.js` file is included in the latest supported Banner General release.

2. Copy the `banner8_integration.js` file to the following directory:

   ```
   $MIDDLEWARE_HOME/user_projects/domains/ClassicDomain/
   servers/WLS_FORMS/tmp/_WL_user/formsapp_11.1.1/e18uoi/
   war/frmjscript
   ```

   Note: The folder location `e18uoi` specified in the directory path may change to a different location such as `spn4wj` or `wb1h9f` based on your OFM installation.

Configure the Forms Server

The prerequisite for Forms Server configuration is to have a previously established Banner 8.x INB.

Update Forms Web Configuration

To perform Forms web configuration, perform the following steps:

1. Clone and create a new Web Configuration from your original INB configuration in the WebLogic console. For example, clone a new `SEAM.env` (for seamless navigation) from the existing `PROD.env`.

   1.1. Login to EM Console. For example, `http://your.oracle-forms.server:7001/em`.

   1.2. Open the Forms folder and then click **Forms**.

   1.3. Click **Environment Configuration**.

   1.4. Locate the active ENV setting.

   1.5. Click **Duplicate File**.

   1.6. Set the **Environment File** to `SEAM.env`.

   1.7. Set the **Name** to `SEAM.env`.

   1.8. Click **Duplicate**.

2. Make the following changes in the Forms web configuration:

   2.1. Click **Forms**.

   2.2. Click **Web Configuration**.

   2.3. Locate the INB Web Configuration. For example, `PROD.`

   2.4. Click **Create Like**.

   2.5. Set the **Section To Duplicate** to `PROD.`

   2.6. Set the **New Section Name** to `SEAM.`

   2.7. Click **Create.**

   2.8. Locate the INB Web Configuration. For example, `SEAM.`

   2.9. Set **Show** to `all.`

   2.10. Click on the **Override** tab.

   2.11. Change the following settings:

```
width=100%

height=100%

separateFrame=false

baseHTMLjpi=webutiljpi.htm

baseHTML=webutilbase.htm

webutilArchive=frmwebutil.jar, jacob.jar

legacy_lifecycle=true
```

```
applet_name=forms_applet

JavaScriptBlocksHeartBeat=true

enableJavascriptEvent=true

otherparams=

  obr=%obr% record=%record% tracegroup=%tracegroup%
  log=%log% term=%term%
  ssoProxyConnect=%ssoProxyConnect%
  iamticket=%iamticket%

envFile=SEAM.env
```

Note: Adjust the forms web configuration to a screen resolution of 1024x768 or higher for displaying INB Form without scroll bars.

Note: When copying and pasting the value for the `otherparams` configuration make sure the entire value is entered in one line.

2.12. Add the following new parameters:

2.12.1. Setting parameter `jvmcontroller`

– Set **Name** = `jvmcontroller`

– Set **Value** = `INBSSOController`

– Set **Comment** = `JVM Pool for INB SSO Controller`

2.12.2. Setting parameter `integrationXE`

– Set **Name** = `integrationXE`

– Set **Value** = `http://<application navigator host name>:<port>/applicationNavigator/static/dist/m.js`

– Set **Comment** = `Location of Application Navigator Integration JS library`

Note: Edit the `HOST` and `PORT` information where Application Navigator has been deployed.

2.12.3. Setting parameter `integrationBannerXE`

– Set **Name** = `integrationBannerXE`

– Set **Value** = `/forms/frmjscript/banner8_integration.js`

– Set **Comment** = `Location of Seamless Navigation client integration JS file`

2.12.4. Setting parameter `heartbeatIntegrationBannerXE`

  - Set **Name** = `heartbeatIntegrationBannerXE`
  - Set **Value** = `288000`
  - Set **Comment** = `This parameter is used to set the frequency at which Banner INB sends a keep-alive message to Application Navigator indicating that it is still running. The default is 8 hours.`

  2.13. Click **Apply**.

3. Make the following changes in the Forms Environment configuration:

  3.1. Click **Forms**.

  3.2. Click **Environment Configuration**.

  3.3. Make the required changes to the `FORMS_PATH` entry to point to the directory location for the new Forms environment.

  3.4. Click **Apply**.

Configure JVM Pooling

Note: JVM Pooling is optional for accessing Banner 8x INB Forms through Application Navigator. However, enabling this feature does offer performance improvements when experiencing very high volumes of requests for launching Forms via SSO Manager. JVM Pooling may have already been established by previous SSO configurations with Banner Enterprise Identity Services (BEIS). If not, refer to Chapter 11 of the Banner Enterprise Identity Services Handbook 8.1.5 or later, available in the Ellucian Support Center, for detailed instructions to deploy jar files to the appropriate locations.

To perform JVM Pooling configuration, perform the following steps:

1. Copy the file `ssoclient.jar` (created during Banner Enterprise Identity Services SSO Manager installation) on your Oracle Forms server in a directory where you have read and execute permissions. This location is referred to as <Banner SSO client directory>.

Note: Do not place `ssoclient.jar` in the `<WebLogic_home>/forms/java` directory. Doing so creates a security risk.

2. Under JVM Configuration, create the `INBSSOController` configuration.

  2.1. Click on the **Forms** dropdown list in the upper left corner in main frame.

  2.2. Select **JVM Configuration.**

  2.3. Click **Create**.

  2.4. Set **Name** to `INBSSOController.`

  2.5. Click **Create**.

2.6. Add the following setting:

```
classpath = <Banner SSO client directory>/
ssoclient.jar
```

Note: This is the location of the `ssoclient.jar` outlined in step 1.

Start the INBSSOController

After you configure JVM pooling, you must start the INBSSOController.

1. In the **Forms** drop-down list, select **JVM Controllers**.
2. Select **INBSSOController**.
3. Select the **Start** tab to start the process.

Configure INB Accelerator Keys for unified menu shortcuts

You can assign INB accelerator keys for the unified menu options displayed under the **File>Menu** option for an INB Form when integrated with Application Navigator.

The INB accelerator keys assigned for the unified menu options are displayed as follows:

unified menu Browse	Ctrl+M
unified menu Open Application	Ctrl+Y
unified menu Search	Ctrl+Shift+Y
unified Sign Out	Ctrl+Shift+F

To implement the INB accelerator keys, perform the following steps:

1. Verify the NLS_LANG setting for your Banner 8.x Forms Web Configuration. For example; when using the en-us locale, `NLS_LANG=AMERICAN_AMERICA.AL32UTF8`
2. Edit the resource file `fmrweb_utf8.res`. The default location of this resource file (for en-us locale) under the UNIX and Windows operating systems are listed below.

   a. Unix File Location - `$MIDDLEWARE_HOME/asinst_1/config/FormsComponent/forms/admin/resource/US/`

   b. Windows File Location - `%MIDDLEWARE_HOME%\asinst_1\config\FormsComponent\forms\`

3. Locate the following line in the file:

```
68      : 2 : "Ctrl+D"          : 11022 : "Accelerator1"
```

4. Append the following 4 lines below that line:

```
77      : 2 : "Ctrl+M"          : 11023 : "Accelerator2"
89      : 2 : "Ctrl+Y"          : 11024 : "Accelerator3"
89      : 3 : "Ctrl+Shift+Y"    : 11025 : "Accelerator4"
70      : 3 : "Ctrl+Shift+F"    : 11026 : "Accelerator5"
```

   It is important to understand the syntax of the accelerator keys.

   For example, **89**: is the key code of the “Y” character
   2: indicates that it will be accessed with the “Ctrl” key
   1 is used for the “Shift” key
   0 is used to access it stand alone

   Ctrl+Y: is the key used to access.

5. Save the file changes and restart the Oracle Application Server.
6. Close all browsers, log back into Application Navigator and access an INB 8.x form. Confirm that the new accelerator keys are displayed and are accessible.

There are different settings and directory locations where the resource file is read by the Oracle Application Server based on the values of NLS_LANG defined in the Forms Web Configuration.

If `NLS_LANG=AMERICAN_AMERICA.AL32UTF8`, then the resource file used will be referenced from `“$MIDDLEWARE_HOME/asinst_1/config/FormsComponent/forms/admin/resource/US/fmrweb_utf8.res”`.

If `NLS_LANG=ARABIC_EGYPT.AR8MSWIN1256`, then the resource file used will be referenced from `“$MIDDLEWARE_HOME/asinst_1/config/FormsComponent/forms/admin/resource/AR/fmrweb.res”`.

Update GUAUPRF settings for the unified menu

You can update the General User Preferences Maintenance Form (GUAUPRF) setting for the INB URL that will be used with the seamless navigation solution. To do this, perform the following steps:

1. Login to Banner with the `BASELINE` user ID to make this change on GUAUPRF.

   Note: When you make changes on GUAUPRF, to set the newly updated URL for all users, you must be logged into Banner with the `BASELINE` user ID.

2. Access the GUAUPRF form.
3. Click **Directory Options**.
4. Locate the **Description** field that contains *Enter the name of your unified Banner menu integrated Oracle Forms Server*.
5. In the **User Value** field, enter the URL of your Oracle forms server. For example,

   ```
   User Value = http://your.oracle-forms.server/ssomanager/
   c/INB
   ```

   Note: The server URL is the location of the deployed SSOManager for CAS implementation.

6. Select **Propagate: Copy to All Users** and save your changes.

Enable Seamless Navigation for Banner 8.x Self-Service Pages

Banner 8.x Self-Service application can be configured to integrate with Application Navigator. When configured, a Banner Self-Service menu item will appear in the Application Navigator menu to access all the Banner 8.x Self-Service pages available to the user. The Banner 8.x Self-Service pages can also be launched from the search results displayed in Application Navigator.

Configure Banner 8.x Self-Service

To enable seamless navigation for Banner 8.x Self-Service, you must follow the setup steps outlined below.

Change the banner8ss_integration.js file

To make the required change in the `banner8ss_integration.js` file, perform the following steps:

1. Locate the `banner8ss_integration.js` file in the `$BANNER_HOME/wtlweb/js` folder.
2. Copy the `banner8ss_integration.js` file to the Banner 8.x SSB server where Web Tailor is deployed under the JS directory.
   For example, the SSB directory containing the JS directory:
   `/u01/app/ellucian/banapps/SMPL/ssb/js/`

Verify Web Tailor Parameters

As part of the Web Tailor 8.8.2 installation, the Banner 8.x Self-Service integration parameters for Application Navigator will be seeded into your Database. Confirm that these parameters have the right name and value as noted below:

Parameter Name	Parameter Value	Comments
APP_NAV_INTEGRATION	`/js/banner8ss_integration.js`	Location of the Seamless Navigation client integration JS file

Parameter Name	Parameter Value	Comments
APP_NAV_JQUERY	`/js/jquery-1.6.1.min.js`	Location for JQUERY libraries used for integration with Application Navigator
APP_NAV_MESSAGE_API	`http://<application navigator host>>:<<port>>/applicationNavigator/static/dist/m.js`	Location of the Application Navigator Messaging API

Single Sign-On Support for Banner 8.x Self-Service

Application Navigator supports the ability to call a Banner 8.x Self-Service page using the Single Sign-On (SS0) protocol and URL provided by the BEIS SSO Manager software component. The SSO Manager will receive the page request sent by Application Navigator, authenticate the request and then forward the call to launch the Banner 8.x Self-Service page seamlessly.

To enable the Single Sign-On request, update the property `banner8.SS.url` in the shared `banner_configuration.groovy` file before the Application Navigator WAR file is generated.

```
banner8.SS.url = 'http://<SSO Manager host>:<port>/ssomanager/c/SSB?pkg='
```

For supporting Single Sign-On for a Multi-Entity Processing (MEP) Database, a property can be used that contains a list of the institutional MEP codes available in the database with their respective URL values.

The URL value for each MEP code is the combination of the fully qualified SSO Manager SSB URL along with the Banner 8.x SSB Direct Access URL for the given MEP institution. The example that follows shows the configuration for two MEP institutions - NORTH and SOUTH.

```
mep.banner8.SS.url = [
   NORTH : ‘http://<SSO Manager host>:<port>/ssomanager/c/SSB?pkg=http://
<BANNER8 SSB host>:<port>/SMPL_NORTH/’,
   SOUTH : ‘http://<SSO Manager host>:<port>/ssomanager/c/SSB?pkg=http://
<BANNER8 SSB host>:<port>/SMPL_SOUTH/’
]
```

Note: When using the `mep.banner8.SS.url` property, comment out the old property `banner8.SS.url` as it is not in use.

Enable Seamless Navigation for Banner 9.x Administrative Applications

Banner 9.x applications must be configured to integrate with Application Navigator. Settings in the Banner database must also be updated to indicate that the Banner 9.x module is integrated with Application Navigator. This is accomplished by performing the following steps:

- "Update GUAPAGE controls for each Banner 9.x module" on page 34
- "Rebuild the Banner Menus" on page 35

Update GUAPAGE controls for each Banner 9.x module

This section discusses how to update an application's integration setting and then validate the module URL setting.

Update the Integration setting

To enable Application Navigator for the Banner 9.x administrative application, you must set the application's `gubmodu_integration_value` to Y. This can be done using the GUAPAGE form in Banner, or by executing the appropriate script against the database.

Use one of the following methods for updating the **Integration** indicator setting for each Banner 9.x module that will be integrated with Application Navigator:

- "Change the setting using GUAPAGE" on page 34
- "Change the setting using a script" on page 35

Change the setting using GUAPAGE

To change the setting using the GUAPAGE form, perform the following steps:

1. Log in to Banner with the user ID that has access to the GUAPAGE form.
2. Access the GUAPAGE form.

3. Select the **Integrated** indicator check box for each module that will be integrated with Application Navigator.
4. Save your changes.

Change the setting using a script

You can use the following table to identify the script used to enable Application Navigator for each Banner 9.x module.

Banner 9.x Module	Script used to enable Application Navigator
Student Registration	`update gubmodu SET gubmodu_integration_value = 'Y' where gubmodu_code = 'SF';`
Student Overall	`update gubmodu SET gubmodu_integration_value = 'Y' where gubmodu_code = 'SO';`
General Common	`update gubmodu SET gubmodu_integration_value = 'Y' where gubmodu_code = 'GC';`

Validate the module URL

Verify that the URL setting has been updated for each Banner 9.x application. This information is generally updated when installing the Banner 9.x module.

In the GUAPAGE form, the **URL** field provides the context information associated with the Banner 9.x module. The field must be updated to include the URL for the appropriate deployment of each module.

The value in the URL field must be in the following format:

```
http://<host name>:<port>/<application context name>/
```

Rebuild the Banner Menus

Use one of the following methods to rebuild the menus:

- On the GUAPAGE form change the integrated indicator on any module and the menu will automatically rebuild upon save.

- Login to SQL Plus as `BANINST1` and then execute the following script:

  ```
  sqlplus baninst1/password

  execute gukmenu.p_refresh_horizon_menu();
  commit;
  ```

Enable Seamless Navigation for Banner 9.x Self-Service Applications

Banner 9.x Self Service applications can be configured to integrate with Application Navigator. When configured, a menu item will appear in the Application Navigator menu for each Self Service Application. Once a Self Service application is integrated it will also be included in the search results displayed in Application Navigator.

Configure Banner 9.x Applications

To enable seamless navigation for Banner 9.x self service applications, you must follow the setup steps outlined below.

Web Tailor Menu entries

The Banner 9.x Self Service application URLs must be added to the appropriate menu in Web Tailor (select a menu to show the page). Note that the URL is the unique identifier and the value entered must not contain any empty spaces and should be a valid.

Note: Refer to the Application Navigator Handbook (section 'Integrating with Self Service applications) on how to configure your Banner 9 Self-Service applications within the Web Tailor menus in order to display in the Application Navigator menu.

The same set of URLs will then need to be added into the "seamless.selfServiceApps" configuration property within Application Navigator.

```
seamless.selfServiceApps = [
"http://<Self-Service application host>:<port>/<Banner SSB application>",
"http://<Self-Service application host>:<port>/<Banner SSB application>"
]
```

If the Self-Service application is configured with a MEP Database, then the URL of the Self-Service Application listed must be appended with the MEP code query parameter as shown below:

```
"http://<Self-Service application host>:<port>/<Banner SSB application>
?mepCode={mepCode}"
```

Setting X-Frame-Options

In order to load the Banner 9.x Self-Service application within Application Navigator iframe, the X-Frame-Options must be declared within the application specific configuration file. This enables the application login page to not be exposed to the clickjacking vulnerability when loaded in an iframe.

```
grails.plugin.xframeoptions.urlPattern = '/login/auth'
grails.plugin.xframeoptions.deny = true
```

Navigational MEP support from a Banner 9x to Banner 8x Self Service page

Banner 9.x Application supports the ability to call a MEP context sensitive URL that maps to the appropriate SSB 8x URL. The URL configurations enable the end user SSO access from Banner 9.x Self Service to Banner 8 SSB applications along with preserving the MEP code for that user session.

For example, if we have Banner 9.x Faculty Attendance Tracking App deployed with two MEP institutions (GVU and BANNER):

- `http://myschool.edu:8080/FacultyAttendanceTrackingSsb/ssb/facultyAttendanceTracking?mepCode=BANNER`
- `http://myschool.edu:8080/FacultyAttendanceTrackingSsb/ssb/facultyAttendanceTracking?mepCode=GVU`

An example configuration that would go in the Banner 9.x Faculty Attendance Tracking application configuration groovy file that maps the MEP code in Banner 9x to the target Banner 8x SSB URL is shown below.

```
mep.banner8.SS.url = [
   GVU : ‘http://myschool.edu:8888/ssomanager/c/SSB?pkg=http://
myschool.edu:9020/SMPL_GVU/',
   BANNER : 'http://myschool.edu:8888/ssomanager/c/SSB?pkg=http://
myschool.edu:9020/SMPL_BANNER/'
]
```

Install Application Navigator for CAS SSO

Undeploy the existing application

Before you reinstall Application Navigator, you must undeploy the existing application. The following sections provide the required steps to undeploy the existing Application Navigator installation in Tomcat and WebLogic servers.

Tomcat

You can either use the Tomcat Manager web application to undeploy the existing application or you can shut down Tomcat and manually remove the files.

Undeploy using the Tomcat Manager web application

Use the following procedure to undeploy the application using the Tomcat Manager web application:

1. Access the Tomcat Manager web application at one of the following URLs:

   `http://server:8080/manager`

   or

   `http://server:8080/manager/html`

2. Access the deployment page using a valid user name and password.
3. Under the Commands area, click **Stop** to stop the existing application.
4. In the confirmation dialog box, click **OK**.
5. Under the Commands area, click **Undeploy**.
6. In the confirmation dialog box, click **OK** to undeploy the application.

Undeploy using a manual procedure

The following sections give the steps to manually undeploy the existing application on Unix and Windows operating systems.

Unix

Use the following procedure to manually undeploy the existing application on a Unix operating system:

1. Log in to the server where Tomcat is running, using the same account credentials that were used to start Tomcat.
2. Shut down Tomcat by running the shutdown script:

   ```
   $CATALINA_HOME/bin/shutdown.sh
   ```

3. Remove the current deployment and associated WAR file:

   ```
   cd $CATALINA_HOME
   rm -rf $CATALINA_HOME/webapps/applicationNavigator
   rm -rf $CATALINA_HOME/webapps/applicationNavigator.war
   ```

Windows

Use the following procedure to manually undeploy the existing application on a Windows operating system:

1. Use the command prompt to shut down Tomcat.

   Note: If you installed Tomcat as a service, use the Service Control panel to stop the application. Otherwise, use the shutdown script `%CATALINA_HOME%\bin\shutdown.bat`.

2. Remove the current deployment and associated WAR file:

   ```
   rmdir %CATALINA_HOME%\webapps\applicationNavigator /s/q
   del %CATALINA_HOME%\webapps\applicationNavigator.war /q
   ```

WebLogic

Use the following procedure to stop and undeploy the application:

1. Access the administration server using the following URL:

   ```
   http://server:7001/console
   ```

2. In the Domain Structure frame, click **Deployments**.
3. In the Change Center, click **Lock and Edit**.
4. Select the check box to the left of the application.
5. Click **Stop**.
6. Click **Force Stop Now**.
7. In the Force Stop Application Assistant page, click **Yes**.
8. Select the check box to the left of the application.

9. Click **Delete**.
10. In the Delete Application Assistant page, click **Yes**.
11. In the Change Center frame, click **Activate Changes**.

Customize the WAR file

The name of the release package is `release-applicationNavigator-3.0.zip`. This release package is moved to the `$BANNER_HOME\general\java` subdirectory during the database upgrade. Use the following steps to unzip the release package and customize the WAR file for your institution.

Note: JDK 1.7 must be installed on your system. See the Java dependencies section for more information.

When you locate the release package zip file, copy it to your application server environment. Transfer this file in binary mode using File Transfer Protocol (FTP). To copy the release package, you must have a valid application server account.

Unzip the release package

To unzip the release package into a temporary directory, perform the following steps:

1. Log in to the application server platform.

   Note: You must have a valid application server account to deploy into the application server container (Tomcat or WebLogic).

2. Create a temporary directory. For example:

   ```
   mkdir $HOME/ban9temp
   ```

3. Locate the release package `release-applicationNavigator-3.0.zip.`
4. Transfer this file in binary mode using File Transfer Protocol (FTP) file into the temporary directory. For example:

   ```
   $HOME/ban9temp
   ```

5. Unzip `release-applicationNavigator-3.0.zip` into the temporary directory.

Prepare the installer

To prepare the installer, perform the following steps:

1. Change the directory to the installer directory:

```
cd installer
```

2. Run the `ant` command, which will build the installation tool.

Note: For Unix, make sure the ant file is executable. For example, `chmod +x ant`.

Example:

```
ban9temp $ cd installer
ban9temp/installer $ ./ant
```

The message *Build successful* confirms a successful build.

Install into the product home directory

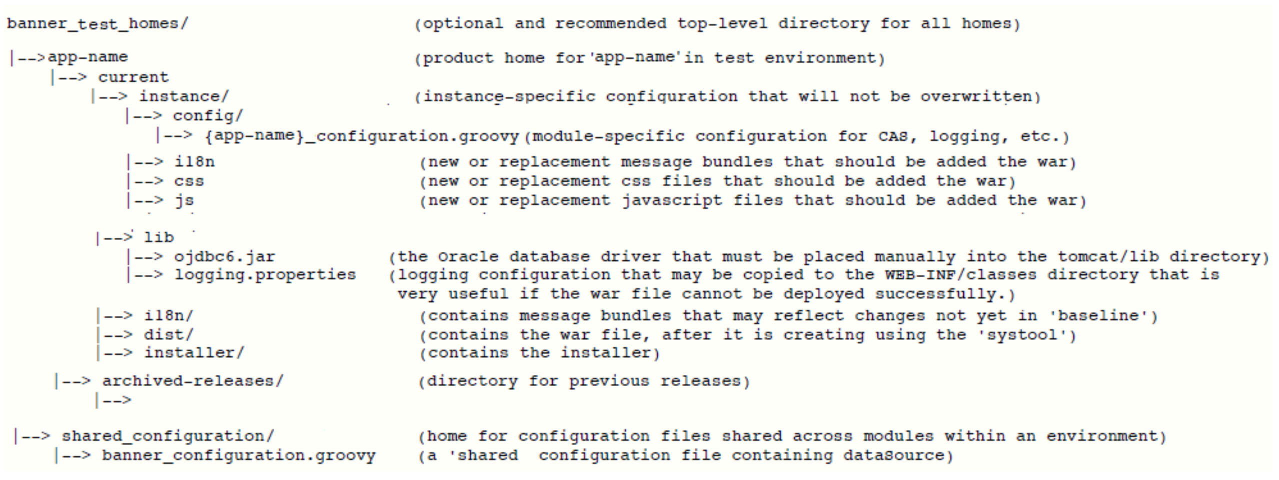
The product home directory supports the configuration and creation of a deployable WAR file. Although Banner 9.x web applications are modular and are installed independently, they share a common configuration. The package provides a common installer that creates consistent product home directory structures for all Banner 9.x applications.

Within a particular environment, you should place the product home directories for Banner 9.x applications in sibling directories. For example, the following directory structure includes four product home directories and a `shared_configuration` directory that support a common test environment.

```
banner_test_homes
|--> Registration 9.2
|--> Catalog 9.3
|--> Schedule 9.3
|--> applicationNavigator
|--> shared_configuration
```

A product home directory is created for each deployment. For example, the home directory that is used for the application within a test environment is different than the home directory that is used for the production environment. When you are supporting different environments for multiple home directories for the same solution, this structure provides the necessary configuration, release level, and custom modification flexibility.

The following directory tree illustrates the product home directory that is created for the test environment:



```
banner_test_homes/                  (optional and recommended top-level directory for all homes)

|-->app-name                        (product home for'app-name'in test environment)
    |--> current
        |--> instance/              (instance-specific configuration that will not be overwritten)
            |--> config/
               |--> {app-name}_configuration.groovy (module-specific configuration for CAS, logging, etc.)
            |--> i18n                (new or replacement message bundles that should be added the war)
            |--> css                 (new or replacement css files that should be added the war)
            |--> js                  (new or replacement javascript files that should be added the war)

        |--> lib
            |--> ojdbc6.jar        (the Oracle database driver that must be placed manually into the tomcat/lib directory)
            |--> logging.properties  (logging configuration that may be copied to the WEB-INF/classes directory that is
                                    very useful if the war file cannot be deployed successfully.)
        |--> i18n/                   (contains message bundles that may reflect changes not yet in 'baseline')
        |--> dist/                   (contains the war file, after it is creating using the 'systool')
        |--> installer/              (contains the installer)

    |--> archived-releases/          (directory for previous releases)
        |-->

|--> shared_configuration/           (home for configuration files shared across modules within an environment)
    |--> banner_configuration.groovy  (a 'shared  configuration file containing dataSource)
```

In addition to the application's product home directory, a separate `shared_configuration` home directory contains cross-application configuration for the test environment. This directory holds the `banner_configuration.groovy` file, which contains the shared JNDI datasource configuration.

To install the installer into the product home directory, perform the following steps:

1. Ensure that the installer is prepared using `ant`.
2. Use the installer to install the release file into the product home directory.

   Note: Your current working directory must be in the installer directory (`ban9temp/installer`) before executing the following commands.

   On Unix:
   ```
   $ bin/install home
   ```

   On Windows:
   ```
   > bin\install home
   ```

3. When prompted, enter the full path of the application home directory. The application will be installed within the current subdirectory within this home directory and the previous release will be archived.

   On Unix:
   ```
   []: Current_home_directory/banner_test_homes/
   applicationNavigator
   ```

   On Windows:
   ```
   []: c:\banner_test_homes\applicationNavigator
   ```

4. Enter the full path of the `shared_configuration` home directory. Banner 9.x applications that refer to this home directory share this configuration file.

   On Unix:
   ```
   []: Current_home_directory/banner_test_homes/
   shared_configuration
   ```

   On Windows:
   ```
   []: c:\banner_test_homes\shared_configuration
   ```

Note: If an identified home directory or the `shared_configuration` home directory does not exist, the installer creates it. The name of a product home directory is not restricted. You can name it when prompted by the installer.

Configure shared settings

The `shared_configuration` home directory contains a cross-application configuration file called `banner_configuration.groovy`. You can change settings in this file.

JNDI datasource

You can configure both Banner Administrative and Self-Service datasources. You can optionally change the datasource name in the configuration file to point to the JNDI datasource that is configured in your application server.

For example, `jndiName ="jdbc/bannerDataSource"` or `"jdbc/bannerSsbDataSource"` are the default configurations. You can change this to match the JNDI datasource names in your environment.

Configure application-specific settings

The `applicationNavigator\current\instance\config` directory contains the `applicationNavigator_configuration.groovy` file. This application-specific configuration file contains settings that you can customize for your specific environment. This directory also contains an `instance.properties` file that references the shared configuration location.

JMX MBean name

The name that is used to register MBeans must be unique for each application that is deployed into the JVM. This configuration should be updated for each instance of each application to ensure uniqueness.

```
jmx {
    exported {
        log4j = "applicationNavigator-log4j"
    } }
```

Location of the logging file

Log4j is the common logging framework used with applications that run on the Java Virtual Machine. For more information, refer to the log4j documentation.

The configuration file includes documentation on various elements that can be modified depending on your environment.

The following is an example of how to override the location where the log file is saved.

```
def string loggingFileDir = "System.properties['logFileDir'] ?
"${System.properties['logFileDir']}" : "target/logs"
def string logAppName = "applicationNavigator"
def string loggingFileName = "${loggingFileDir}/${logAppName}.log".toString()
```

The following is an example of how to override the log file directory properties:

```
export JAVA_OPTS = "-DlogFileDir=/PRODUCT_HOME /"
```

The output logging file location is relative to the application server to which you are deploying.

Logging level

The root logging level is pre-configured to the `ERROR` level. Multiple class or package level configurations, by default, are set to a status of "off." You can set a different logging level for any package or class. However, the running application must be restarted.

For example:

```
case 'production':
root {
error 'appLog' //change the log level here with the
appropriate log level value.
additivity = true
}
```

Note: Changing the logging level to `DEBUG` or `INFO` produces very large log files.

Changes to the `applicationNavigator_configuration.groovy` file require a restart of the application before those changes take effect.

Alternatively, you can use JMX to modify logging levels for any specified package or class, or even at the root level. When using JMX, the logging level changes only affect the running application. When you restart the application, changes that you made using JMX are lost.

For more information on JMX configuration, see “Configure Java Management Extensions” on page 59.

Institutional Home Page Redirection Support

With previous versions of the application, users did not have the option to go back to a home page if they encountered any SSO access issues. This configuration allows Application Navigator to provide an institutional home page to which users can navigate back to if they encounter access issues specific to insufficient privileges when accessing the application.

```
        /**********************************************************************
        *Home Page URL configuration for CAS / SAML Single-Sign On      *
         **********************************************************************/
// Can be institutional home page ex: http://myportal/main_page.html
grails.plugin.springsecurity.homePageUrl='http://APPLICATION_NAVIGATOR_HOST:PORT/
applicationNavigator'
```

Seamless plugin configurations

The following properties describe the configurations required for the seamless navigation plugin. Each property is described in the following table.

Property	Description
`seamless.interceptPattern`	Pre-populated to pull in the CAS server URL
`seamless.menuEndpoints`	Array of the menu endpoint URLs that must be wrapped in quotes and comma-delimited
`seamless.logLevel`	Log level for messages, includes *error* and *debug*
`seamless.brandTitle`	Name of your institution that will be displayed as the title in the navigation bar of Application Navigator. The title is an anchored link that users can click at any time to navigate back to the landing page. Configure the brand title with a default institution value or based on the MEP institution code configured in the Banner database. The sample default value provided is “Ellucian University”
`seamless.ajaxTimeout`	AJAX request timeout in milliseconds
`seamless.messageResponse Timeout`	Timeout for Application Navigator waiting on responses from embedded applications in milliseconds
`seamless.exposeMenu`	Boolean logic that determines whether to enable the example menu service
`seamless.sessionTimeout`	Time in minutes when the session should timeout. Default is 30 minutes, and a value of -1 will keep the session alive indefinitely.
`seamless.sessionTimeout Notification`	Time in minutes when the notification prompt will be displayed to the user prior to session timeout. Default is 5 minutes.

Property	Description
`seamless.excludeObjectsFrom` `Search`	This list includes objects to be excluded

Example:

```
seamless.interceptPattern =
"${grails.plugin.springsecurity.cas.serverUrlPrefix}.*"

seamless.menuEndpoints = [

   "http://<application navigator host>:<port>/applicationNavigator/
commonMenu",

   "http://<application navigator host>:<port>/applicationNavigator/
commonSelfServiceMenu"

]

seamless.logLevel="off"

seamless.brandTitle=["Default": "Ellucian University"]

seamless.ajaxTimeout=30000

seamless.messageResponseTimeout=2000

seamless.exposeMenu=true

seamless.sessionTimeout = 30

seamless.sessionTimeoutNotification = 5

seamless.excludeObjectsFromSearch = [

"GUAGMNU","GUAINIT","GUQSETI","FOQMENU","SOQMENU","TOQMENU","AOQMEMU",
"GOQMENU","ROQMENU","NOQMENU","POQMENU","FACICON","FAQINVP","FAQMINV",
"FAQVINV","FGQACTH","FGQAGYH","FGQDOCB","FGQDOCN","FGQDOCP","FGQFNDE",
"FGQFNDH","FGQLOCH","FGQORGH","FGQPRGH","FOQADDR","FOQDCSR","FOQENCB",
"FOQFACT","FOQINVA","FOQJVCD","FOQPACT","FOQRACT","FOQSDLF","FOQSDLV",
"FPCRCVP","FPQBLAP","FPQCHAP","FRCBSEL","FSCISSR","FSCSTKL","FTQATTS",
"FXQDOCN","FXQDOCP","**SSB_MASKING","TSQCONT","TSQEXPT","TOQCALC",
"GPBADMN","SFQESTS","SFQPREQ","SFQRQST","SFQRSTS","SFQSECM","SFQSECT",
"SHQDEGR","SHQQPNM","SHQSECT","SHQSUBJ","SHQTERM","SHQTRAM","SLQBCAT",
"SLQEVNT","SLQMEET","SLQROOM","SMQSACR","SMQSGCR","SMQSGDF","SMQSPDF",
"SOQCSCP","SOQCTRM","SOQHOLD","RPQLELG","RPQCOMP","ROQADDR"

]
```

Display name support

You can configure the header in Application Navigator to display a person's preferred name. To enable this functionality, edit the following configuration settings.

```
productName='Banner General' //Name of the product
```

The name in the GURNHIR table and the name mentioned in this configuration should match.

```
banner.applicationName='Application Navigator' //Application
Name
```

The name in the GURNHIR table configured and name mentioned in this configuration should match.

You need to have General 8.8.9 installed to use this functionality. Refer to the values in the following table to use the Name Display (GUANDSP) form and enter information manually, if required.

Product	Application
Banner General	Application Navigator

MEP support

Set the value below to ***`true`*** in a MEP database environment.

```
mepEnabled = false
grails.plugin.springsecurity.logout.mepErrorLogoutUrl = '/logout/
customLogout'
```

Self-Service support

- Set 'ssbEnabled' to true for instances that expose Self-Service Banner endpoints. If this is set to false, or if this configuration item is missing, the instance will only support Administrative applications and not Self-Service applications in the unified menu. If this is enabled, Application Navigator will integrate with Banner Self-Service applications using the SSB datasource.

  ```
  ssbEnabled = true
  ```
- The ssbOracleUsersProxied setting is set to false for the Application Navigator deployment by default. Only set 'ssbOracleUsersProxied = true' to ensure that database connections are proxied when the Self-Service user has an Oracle account and there is a requirement to set fine-grained access control (FGAC) for Application Navigator menus. This setting in Application Navigator has no impact on integrated Banner 9 Self-Service applications. The integrated Banner 9 Self-Service applications can be configured separately to allow FGAC on the application specific SSB pages.

  ```
  ssbOracleUsersProxied = false
  ```
- Add commonSelfServiceMenu endpoint to the seamless menu endpoints list if Self-Service menus are to be loaded.

```
   seamless.menuEndpoints = [
        "http://<application navigator host>:<port>/applicationNavigator/
commonMenu",
        "http://<application navigator host>:<port>/applicationNavigator/
commonSelfServiceMenu"
   ]
```

- List the URL entries of Banner Self-Service applications integrating with Application Navigator.

```
   seamless.selfServiceApps = [
   "http://<Self-Service application host>:<port>/<Banner SSB
application>",
   "http://<Self-Service application host>:<port>/<Banner SSB
Application>"
   ]
```

If Application Navigator is configured with a MEP Database, then the URLs of the Self Service Applications must be appended with the MEP code query parameter as shown below:

```
"http://<Self-Service application host>:<port>/<Banner SSB application>
?mepCode={mepCode}"
```

Note: Banner 9 SS application URLs must also exist in the Web Tailor Menu table and should be an exact match with the entries noted in the configuration property.

X-Frame-Options settings

Make sure the values below are set, so that Application Navigator will not be exposed to clickjacking vulnerability when loaded in an iframe.

```
grails.plugin.xframeoptions.urlPattern = '/login/auth'
grails.plugin.xframeoptions.deny = true
```

Spring Security Port Mapper Configuration

The Spring Security Port Forwarding Configuration entries support menu service callbacks based on port forwarding requests. The spring security port mapper configuration is utilized to override the pre-configured port set during application deployment and ensures that the callback URLs are redirected to the forwarded port successfully at run-time.

```
grails.plugin.springsecurity.portMapper.httpPort = <port number>
grails.plugin.springsecurity.portMapper.httpsPort = <SSL port number>
```

Google Analytics

Application Navigator uses Google Analytics to capture usage details of specific pages, including the default authentication login or logout page and the default authentication error pages. All Banner Self-Service 9 applications that are integrated with Application Navigator and configured with Google Analytics will be tracked. This feature is delivered enabled. To enable or disable Google Analytics, use this configuration setting.

```
banner.analytics.trackerId=[institution’s google analytics
tracker ID - default blank]
banner.analytics.allowEllucianTracker=[true|false - default
true]
```

Example:

```
banner.analytics.trackerId = ‘UA-83915850-1’
banner.analytics.allowEllucianTracker = false
```

Use cases:

- If a configuration is not in the configuration file, by default the Ellucian tracking ID will be enabled and Ellucian will track analytics.

- If allowEllucianTracker=true, Ellucian will track the analytics data.
- If allowEllucianTracker=false, the tracking script will not be added in the gsp page.
- If there is only the clientTracker ID in the configuration, then both Ellucian and the client will be tracking the Google Analytics data.
- If allowEllucianTracker=true and the client tracker ID is in the configuration, then both the client and Ellucian will be tracking the analytics data.
- If allowEllucianTracker=false and the client tracker ID is in the configuration, then analytics will be tracked by the client, not Ellucian.

Setup CAS SSO Configuration

The `applicationNavigator\current\instance\config` directory contains the `applicationNavigator_configuration.groovy` file. This application specific configuration file contains settings that you can customize for your specific environment.

Authentication Provider Name

The name identifying the application authentication mechanism. Values are **cas** or **saml**. Specify cas to indicate the application will use CAS SSO protocol for authentication as shown in the following example:

```
/*******************************************************
 *     BANNER AUTHENTICATION PROVIDER CONFIGURATION    *
 *                                                     *
 *******************************************************/
     //
     banner {
         sso {
             authenticationProvider           = 'cas' //  Valid values are:
     'saml' and 'cas' for SSO. 'default' value to be used only when creating
     the release zip file.
             authenticationAssertionAttribute = 'UDC_IDENTIFIER'
             if(authenticationProvider != 'default') {
               grails.plugin.springsecurity.failureHandler.defaultFailureUrl
     = '/login/error'
             }
         }
```

CAS SSO Configuration

Shown below is a sample of the configuration you can enable for SSO between Application Navigator and an Identity Management System that supports CAS SSO protocol. Make sure to uncomment this section when CAS SSO is enabled.

```
     /
     ************************************************************************
```

```
********
 *
*
 *                          CAS SSO Configuration
*
 *
*

************************************************************************
********/
// Set active = true when Application Navigator is configured for CAS SSO

grails {
    plugin {
        springsecurity {
            cas {
                active           = true
                serviceUrl       = 'http://
APPLICATION_NAVIGATOR_HOST:PORT/applicationNavigator/
j_spring_cas_security_check'
                serverName       = 'http://
APPLICATION_NAVIGATOR_HOST:PORT'
                proxyCallbackUrl = 'http://
APPLICATION_NAVIGATOR_HOST:PORT/applicationNavigator/secure/receptor'
                loginUri         = '/login'
                sendRenew        = false
                proxyReceptorUrl = '/secure/receptor'
                useSingleSignout = true
                key = 'grails-spring-security-cas'
                artifactParameter = 'SAMLart'
                serviceParameter = 'TARGET'
                filterProcessesUrl = '/j_spring_cas_security_check'
                serverUrlEncoding = 'UTF-8'
                if (useSingleSignout){

grails.plugin.springsecurity.useSessionFixationPrevention = false
                }
            }
        }
    }
}

// Cannot be declared bean style because the value is not available for
reference in seamless.interceptPattern
grails.plugin.springsecurity.cas.serverUrlPrefix = "https://CAS_HOST:PORT/cas"
```

Note: Depending on your needs, you can customize the `serverUrlPrefix`, `serviceUrl`, and `serverName` entries

Logout URL

You can specify where a user should be directed after logging out of the application by updating the `applicationNavigator_configuration.groovy` file. There are two ways the application can handle logouts:

- Logouts can display the CAS logout page with a redirect URL.
- Logouts can automatically go to a redirect URL (without displaying the CAS logout page).

The redirect URL can be different for each Banner application, depending on where you wish to send the user. If the redirect URL is the same for all Banner applications, it can be defined in the global `banner_configuration.groovy` file.

To display the CAS logout page with a redirect URL

With this method of handling logouts, users see the CAS logout page when they log out of the application. The CAS logout page displays a URL that users must click to continue.

Note: Application Navigator requires CAS as its primary authentication source before the WAR file is deployed.

Use the `logout.afterLogoutUrl` setting to configure the logout URL. This setting is defined as follows:

```
grails.plugin.springsecurity.logout.afterLogoutUrl=''https:/
/<CAS host>:<port>/cas/logout?url=http://<application
navigator host>:<port>/'
```

To go directly to a redirect URL

With this method of handling logouts, users automatically go to a redirect URL. Configure logout URL information as follows:

1. Configure logout information, replacing "url" with "service":

   ```
   grails.plugin.springsecurity.logout.afterLogoutUrl = 'https://<CAS
   host>:<port>/cas/logout?service=http://myportal/main_page.html'
   ```

   - or -

   ```
   grails {
   plugin {
   springsecurity {
   logout {
   afterLogoutUrl = 'https://<CAS host>:<port>/<cas>/logout?url=http://myportal/
   main_page.html'
   ```

   Note: The logout URL can be different for the application, depending on where you wish to direct the user. If the logout URL is same for all the Banner applications, it must be defined in the global `banner_configuration.groovy` file.

2. Set the property `followServiceRedirects` to `true` on the LogoutController that is defined in `cas-servlet.xml`:

   ```
   <bean id="logoutController" class="org.jasig.cas.web.LogoutController"
   p:centralAuthenticationService-ref="centralAuthenticationService"
   ```

```
p:logoutView="casLogoutView"
p:warnCookieGenerator-ref="warnCookieGenerator"
p:ticketGrantingTicketCookieGenerator-ref="ticketGrantingTicket
CookieGenerator"
p:followServiceRedirects="true" />
```

Customize the landing page background image

To customize and replace the landing page background image with an institutional image of your choice, you must perform the following steps before building and deploying the WAR file to the Application Server.

1. Create two CSS files in the deployment staging directory of Application Navigator (example: `~/applicationNavigator/current/instance/css`).
   - `applicationNavigator-custom.css` for LTR support
   - `applicationNavigator-custom-rtl.css` file for RTL support
2. Modify each of these two files by adding the custom CSS styles you wish to change as shown below. For overriding the landing page background image, use the following CSS class and add your institutional background image to the media query style that supports the specific responsive design orientation:

```
/** Custom CSS file which takes precedence over the landing page CSS styles **/
.landing-content {
margin-left: 0px;
padding-left: 0px;
background-repeat: no-repeat;
background-position:center;
background-size: cover;
background-color: #4F585F;
}
@media (orientation:landscape) {
.landing-content {
background-image: url("/css/images/backgrounds/campus-landscape.jpg");
}
}
@media (orientation:portrait) {
.landing-content {
background-image: url("/css/images/backgrounds/campus-portrait.jpg");
}
}
@media (orientation:portrait) and (max-width:320px) {
.landing-content {
background-image: url("/css/images/backgrounds/campus-sml.jpg");
}
}
```

3. The above images and styles shown support responsive designs for landscape, portrait, and mobile views. Make sure your institutional background images you

choose maintain the correct aspect ratio for displaying the custom images in landscape and portrait mode along with supporting different device screen resolutions.

- `campus-landscape.jpg` supports widescreen desktop devices (supported resolution size 1920 * 1080)
- `campus-portrait.jpg` supports laptops and tablet devices (supported resolution size 768 * 1024)
- `campus-sml.jpg` supports mobile devices (supported resolution size 320 * 568)

4. Confirm the custom CSS files and images are in the deployment staging directory of Application Navigator. The directory and file structure should be similar as shown below:

```
applicationNavigator
|-----current
    |-----instance
        |-----css
            |-----applicationNavigator-custom.css
            |-----applicationNavigator-custom-rtl.css
            |-----images
                |-----backgrounds
                    |-----campus-landscape.jpg
                    |-----campus-portrait.jpg
                    |-----campus-sml.jpg
```

Note: The image name can be any custom image name. However, css file names must be the same as specified (`applicationNavigator-custom.css` and `applicationNavigator-custom-rtl.css`) and the class name to override the background image of landing page also must be the same "landing-content."

5. Continue with the next steps of regenerating and deploying the Application Navigator WAR file to your Application Server with the customized CSS files. Verify the landing page background image has changed and meets your institutional needs.

Regenerate the WAR file

Once the shared and application-specific configurations are complete, the application WAR file can be regenerated to include your customizations and application-specific settings. The WAR file can then be deployed into your specific application server.

The systool is used to create the WAR file. To set up the systool and to create the WAR file, perform the following steps:

1. Change your current working directory to the product home directory:

   `PRODUCT_HOME/current/installer`

2. Run the ant command, which will build the systool module.

   Note: For Unix, make sure the ant file is executable. For example, `chmod +x ant`.

   Example:

   ```
   $ cd PRODUCT_HOME/current/installer
   PRODUCT_HOME/current/installer $ ./ant
   ```

3. Use the systool module to create the WAR file.

   Your current working directory must be in the `PRODUCT_HOME/current/installer` directory before you execute the following command.

   On Unix:
   ```
   $ bin/systool war
   ```

   On Windows:
   ```
   > bin\systool war
   ```

The WAR file is created in the `PRODUCT_HOME/current/dist` directory.

You can use external configuration files by setting appropriate system properties, although the configuration files are included in the WAR file to make the WAR file self-sufficient. For information on external configuration, see "Configure the Tomcat server" on page 56 or "Create a WebLogic server" on page 62.

Configure and deploy the WAR file to a web application server

The following sections provides information on configuring the web application and deploying the WAR file to a web application server:

- "Tomcat" on page 55
- "WebLogic" on page 50

Tomcat

The following sections provide information on configuring the web application and deploying the WAR file to the Tomcat server.

Note: If you choose to install the application on a Tomcat server, you do not need to install it on WebLogic.

Note: Supported Tomcat version is required. To download and install the Tomcat server, see http://tomcat.apache.org.

Configure the Tomcat server

Use the following steps to configure the Tomcat server:

1. Locate the Oracle JDBC jar files (`ojdbc6.jar and xdb6.jar`) in the `$PRODUCT_HOME\current\lib` directory.

   Note: Later in the Tomcat configuration process, you will copy the Oracle JDBC jar files into the `\lib` folder under the Tomcat installation directory.

   The account that runs the Tomcat application server must configure environment settings to support the application.

2. On Linux, ensure `CATALINA_HOME` is defined to reference your Tomcat software installation location. For example, `CATALINA_HOME=/opt/apache-tomcat-xx` where xx indicates the point version of Tomcat you installed.

   Warning! *Do not perform this step on the Windows platform.*

3. Define `CATALINA_OPTS` to configure JVM settings. The following settings are recommended:

   ```
   CATALINA_OPTS=-server -Xms2048m -Xmx4g
   -XX:MaxPermSize=512m
   ```

   Note: If you are deploying multiple Banner 9.x applications to the same Tomcat server, increase the max heap (`-Xmx`) by `2g` and `-XX:MaxPermSize` by `128m`. You should deploy Banner 9.x administrative applications to one Tomcat server instance and Banner 9.x self-service applications to a separate Tomcat server instance.

   You can define this variable in the account's profile startup script, or you can add this definition in `$CATALINA_HOME/bin/catalina.sh` for Linux or `catalina.bat` for Windows.

4. (Optional) If you install Tomcat as a Windows service, specify the JVM arguments as follows:

   4.1. Select **Configure Tomcat** application from the Windows **Start** menu.

   4.2. Select the **Java** tab.

   4.3. In the **Java Options** field, add the following:

   ```
   -XX:MaxPermSize=384m
   ```

   4.4. Set the initial memory pool = 2048.

4.5. Set the maximum memory pool = 4096.

4.6. Save the settings.

4.7. Restart the Tomcat Windows service.

5. (Optional) To set up the Tomcat server to enable remote JMX connections, perform the steps in the "Configure Java Management Extensions" section. This is useful for debugging and logging.

6. Define the JNDI datasource resource names for the application as follows:

6.1. Edit `$CATALINA_HOME/conf/context.xml`.

6.2. Uncomment `<Manager pathname="" />` to disable Tomcat session persistence. For example, change the following:

```
<!-- Uncomment this to disable session persistence
across Tomcat restarts -->

<!--
<Manager pathname="" />
-->
```

to:

```
<!-- Uncomment this to disable session persistence
across Tomcat restarts -->

<Manager pathname="" />
```

6.3. Add the following ResourceLink definitions inside the `<Context>` element:

```
<ResourceLink global="jdbc/bannerDataSource"
               name="jdbc/bannerDataSource"
               type="javax.sql.DataSource"/>

<ResourceLink global="jdbc/bannerSsbDataSource"
               name="jdbc/bannerSsbDataSource"
               type="javax.sql.DataSource"/>
```

6.4. Save your changes in `context.xml`.

6.5. Edit `$CATALINA_HOME/conf/server.xml` to configure the database JNDI resource name and connection pool configuration.

6.6. Add the following Resource definitions inside the `<GlobalNamingResources>` element:

For Tomcat 7:

```
<Resource name="jdbc/bannerDataSource" auth="Container"
   type="javax.sql.DataSource"
   driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@//hostname:port/service_name"
   username="banproxy" password="the_banproxy_password"
   initialSize="5" maxActive="100" maxIdle="-1" maxWait="30000"
   validationQuery="select 1 from dual"
   testOnBorrow="true"/>


<Resource name="jdbc/bannerSsbDataSource" auth="Container"
```

```
  type="javax.sql.DataSource"
  driverClassName="oracle.jdbc.OracleDriver"
  url="jdbc:oracle:thin:@//hostname:port/service_name"
  username="ban_ss_user" password="ban_ss_user_pasword"
  initialSize="5" maxActive="100" maxIdle="-1"
  maxWait="30000"
  validationQuery="select 1 from dual"
  testOnBorrow="true"/>
```

For Tomcat 8:

```
<Resource name="jdbc/bannerDataSource" auth="Container"
   type="javax.sql.DataSource"
   driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@//hostname:port/service_name"
   username="banproxy" password="the_banproxy_password"
   initialSize="5" maxTotal="100" maxIdle="-1"
   maxWaitMillis="30000"
   validationQuery="select 1 from dual"
   accessToUnderlyingConnectionAllowed="true"
   testOnBorrow="true"/>

<Resource name="jdbc/bannerSsbDataSource" auth="Container"
   type="javax.sql.DataSource"
   driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@//hostname:port/service_name"
   username="ban_ss_user" password="ban_ss_user_pasword"
   initialSize="5" maxTotal="100" maxIdle="-1"
   maxWaitMillis="30000"
   validationQuery="select 1 from dual"
   accessToUnderlyingConnectionAllowed="true"
   testOnBorrow="true"/>
```

For example, if your database server name is `myserver.university.edu` and the Oracle TNS Listener is accepting connections on port 1521 and your database service name is SEED, then the URL is `jdbc:oracle:thin:@//myserver.university.edu:1521/SEED.`

6.7. Save your changes in `server.xml`.

6.8. Copy the Oracle JDBC jar files (`ojdbc6.jar and xdb6.jar`) from the `$PRODUCT_HOME/current/lib` directory to the `$CATALINA_HOME/lib` directory.

6.9. Validate the configuration of the Tomcat server by starting the application server. To accomplish this, perform the following steps:

– Run `$CATALINA_HOME/bin/startup.`

For Linux:

```
cd $CATALINA_HOME
$ bin/startup.sh
```

For Windows:

```
cd %CATALINA_HOME%
> bin\startup.bat
```

– Browse http://servername:<port>.

To override the configuration that was added into the WAR file, you must set system properties to point to external configuration files. For example, to point to a configuration file residing in the PRODUCT_HOME directory, export JAVA_OPTS=

```
"-DBANNER_APP_CONFIG=/PRODUCT_HOME/shared_configuration/
banner_configuration.groovy
-DAPPLICATION_NAVIGATOR_CONFIG=/PRODUCT_HOME/
applicationNavigator/current/instance/config/
applicationNavigator_configuration.groovy".
```

Note: When externalizing the configuration files, there are certain filters that get injected into the web.xml file as part of the Single Sign-On and X-Frame HTTP Header settings for the WAR file. These values cannot be changed at runtime by referencing external configuration files. The WAR file will have to be regenerated in order for the new Single Sign-On and X-Frame HTTP Header settings to take effect. Any other configuration outside these two sets of filters can be overridden in the external configuration file at runtime.

Configure Java Management Extensions

This is an optional step that is needed only if you want to monitor or debug the application. Java Management Extensions (JMX) is a Java technology that supplies tools for managing and monitoring applications, system objects, devices, and service oriented networks.

Enabling JMX connections allows you to remotely monitor and debug the application server. To enable Java Management Extensions, perform the following steps:

1. Add the following options to the `catalina.sh` or `catalina.bat` file and then restart the Tomcat server:

   ```
   set CATALINA_OPTS=-Dcom.sun.management.jmxremote

   -Dcom.sun.management.jmxremote.port=8999

   -Dcom.sun.management.jmxremote.ssl=false

   -Dcom.sun.management.jmxremote.authenticate=false

   -Djava.rmi.server.hostname=your.hostname.com
   ```

2. Change the `java.rmi.server.hostname` value to the hostname or IP address of the machine where Tomcat is installed. For example:

   ```
   -Djava.rmi.server.hostname=prod.appserver1.com
   ```

- or -

`-Djava.rmi.server.hostname=149.24.3.178`

3. JMX does not define a default port number to use. If necessary, change `com.sun.management.jmxremote.port=8999`.

   Note: It is recommended that you connect remotely to the Tomcat server using JMX.

   Warning! *Ensure that the* `jmxremote.authenticate` *parameter is not set to* `false` *in a production environment. If it is set to* `false`*, it does not require connections to be authenticated and will create a security threat in a production environment. For more information on Tomcat Remote JMX documentation, see* *http://tomcat.apache.org/tomcat-x.x-doc/monitoring.html#Enabling_JMX_Remote* *(Where x.x is the base version of the Tomcat installed.).*

Deploy the WAR file to the Tomcat server

The systool that is used to create the WAR file can also be used to deploy the WAR file to a Tomcat container. You should deploy 9.x administrative applications and 9.x self-service applications to separate Tomcat servers to increase performance.

Note: The systool does not provide the capability to undeploy or redeploy an application. If you are redeploying the application, you must use the Tomcat Manager web application to undeploy the existing application.

The target supports deploying the `dist/WAR` file using the Tomcat Manager web application. Because environments vary significantly with respect to user privileges, clustering approach, web container version, operating system, and more, the target may or may not be suitable for your use.

Note: You can also deploy the WAR file to the Tomcat server by copying the WAR file to the Tomcat `webapps/` directory.

To use the target, you must provide the following information:

URL	This is the URL of the manager application in the Tomcat server. For example: `http://localhost:8080/manager`
Username	This Tomcat server username must have privileges to deploy WAR files.
Password	This is the password of the Tomcat server user.

Username/password combinations are configured in your Tomcat user database `<TOMCAT_HOME>\conf\tomcat-users.xml.` For Tomcat, you must configure at least one username/password combination with the manager role. For example:

```
<user username="tomcat" password="tomcat" (your password)
roles="manager-gui, manager"/>
```

Note: The roles in Tomcat server changed between point releases. Refer to the Tomcat documentation specific to your release for information on enabling access to provide the appropriate role to a user account for deployment.

To deploy the WAR file to the Tomcat server, perform the following steps:

1. Navigate to the `PRODUCT_HOME\current\installer` directory.
2. Enter one of the following commands:

   On Unix:
   ```
   $ bin/systool deploy-tomcat
   ```

   On Windows:
   ```
   > bin\systool deploy-tomcat
   ```
3. Enter the following URL for the Tomcat Manager:

   ```
   []: http://localhost:8080/manager
   ```

   This URL will be accessed to deploy the WAR file into the container.
4. Enter a valid Tomcat username to deploy the WAR file. For example:

   ```
   []: tomcat
   ```

   Note: This user must have the `manager-gui` role.
5. Enter the Tomcat password for the user:

   ```
   []: password
   ```

   Note: This password will not be persisted.
6. Access the web application:

   ```
   http://servername:<port>/applicationNavigator
   ```

WebLogic

The following sections provide information on configuring the web application and deploying the WAR file to the WebLogic server:

Note: If you choose to install the application on a WebLogic server, you do not need to install it on Tomcat.

Verify WebLogic prerequisites

Before configuring your WebLogic server, ensure that the following prerequisites are met:

- WebLogic must be installed. If it is not, download and install WebLogic from the Oracle web site.
- For minimum requirements on OFM and WebLogic support, refer to the Ellucian Oracle Support Calendar. The calendar is available in the Interactive Banner Compatibility Guide, which can be accessed from the Ellucian Download Center.
- Both the WebLogic node manager and the administration server must be started. The administration server can be accessed using the following URL:

```
http://server:7001/console
```

Create a WebLogic machine

Note: If you previously created a WebLogic machine definition, you can skip this section.

To create a WebLogic machine, perform the following steps:

1. In the Change Center frame, click **Lock & Edit**.
2. In the Domain Structure frame, click (**+**) to expand and view the list of environments.
3. Click the **Machines** link.
4. Click **New**.
5. Enter a machine name and click **Next**.
6. Accept the defaults and click **Finish**.
7. In the Change Center frame, click **Activate Changes**.

Create a WebLogic server

Note: If you previously created a WebLogic server, you can skip this section.

Note: If you previously created a WebLogic server for the application, you can use the same server.

To create a WebLogic server, perform the following steps:

1. In the Change Center frame, click **Lock & Edit**.
2. In the Domain Structure frame, click (**+**) to expand and view the list of environments.
3. Click the **Servers** link.

4. Click **New**.
5. Enter a server name and server listen port. For example, you can have server name as `Banner9` and server listen port as `8180`.
6. Click **Finish**.
7. Click the newly created server link.
8. Under the **General** tab, assign the machine to this server.
9. Click **Save**.
10. Select the **Server Start** tab.
11. Add the following to the **Arguments** text area:

    If you are using Sun JVM, use the following parameters:

    ```
    -server -Xms2048m -Xmx4g -XX:MaxPermSize=512m
    ```

    Note: If you are deploying multiple Banner 9.x applications to the same WebLogic server, increase the max heap (`-Xmx`) by `2g` and `-XX:MaxPermSize` by `128m`. You should deploy Banner 9.x administrative applications to one WebLogic server instance and Banner 9.x self-service applications to a separate WebLogic server instance.

    To override the configuration that was added into the WAR file, you can set system properties to point to external configuration files. Append the following to the arguments text area:

    ```
    -DBANNER_APP_CONFIG=<full file path to
    banner_configuration.groovy>
    -DAPPLICATION_NAVIGATOR_CONFIG=<full file path to
    applicationNavigator_configuration.groovy>
    ```

    Note: When externalizing the configuration files, there are certain filters that get injected into the web.xml file as part of the Single Sign-On and X-Frame HTTP Header settings for the WAR file. These values cannot be changed at runtime by referencing external configuration files. The WAR file will have to be regenerated in order for the new Single Sign-On and X-Frame HTTP Header settings to take effect. Any other configuration outside these two sets of filters can be overridden in the external configuration file at runtime.

12. Click **Save**.
13. In the Change Center frame, click **Activate Changes**.
14. In the Domain Structure frame, click the **Servers** link.
15. Select the **Control** tab.
16. Select the check box next to your new server definition.
17. Click **Start**.

Update Oracle JDBC JAR files on the WebLogic server

1. Copy the Oracle JAR file (xdb6.jar) from the `$PRODUCT_HOME/ current/lib` directory to the `$MIDDLEWARE_HOME/modules` directory.
   - $PRODUCT_HOME is where the Application Navigator 3.0 release zip file is unpacked and installed.
   - $MIDDLEWARE_HOME is the location where Oracle WebLogic is installed.
2. For Linux/Unix servers, edit the setDomainEnv.sh file under the `$MIDDLEWARE_HOME/user_projects/domains/<CUSTOM_DOMAIN>/ bin` folder and add these two lines after the ADD EXTENSIONS comment as shown by the example below:

   ```
   #ADD EXTENSIONS TO CLASSPATH

   export MIDDLEWARE_HOME="/u01/app/oracle/Middleware"

   export WLS_MODULES="${MIDDLEWARE_HOME}/modules"

   export EXT_PRE_CLASSPATH="${WLS_MODULES}/xdb6.jar"
   ```

   Note: If you plan to "copy and paste" the configuration settings into the "setDomainEnv.sh" file, make sure there is no typo or special characters that get carried over (especially with double quotes on the variable declarations). If you see "Class NotFoundException" in your logs, chances are there was a typo when you edited the "setDomainEnv.sh" file and the "xdb6.jar" file cannot be found during Application startup.

3. For MS Windows servers, edit the setDomainEnv.cmd under the `$MIDDLEWARE_HOME/user_projects/domains/<CUSTOM_DOMAIN>/ bin` folder and add these two lines after the ADD EXTENSIONS comment as shown by the example below:

   ```
   @REM ADD EXTENSIONS TO CLASSPATH

   set MIDDLEWARE_HOME="D:\Oracle\Middleware"

   set WLS_MODULES="%MIDDLEWARE_HOME%\modules" set

   EXT_PRE_CLASSPATH="%WLS_MODULES%\xdb6.jar"
   ```

   Note: If you plan to "copy and paste" the configuration settings into the "setDomainEnv.cmd" file, make sure there is no typo or special characters that get carried over (especially with double quotes on the variable declarations). If you see "Class NotFoundException" in your logs, chances are there was a typo when you edited the "setDomainEnv.cmd" file and the "xdb6.jar" file cannot be found during Application startup.

4. Restart the WebLogic Managed Server.

Create an administrative datasource and connection pool

Note: If you previously created an administrative datasource, you can skip this section.

To create an administrative datasource and connection pool, perform the following steps:

1. In the Change Center frame, click **Lock & Edit**.
2. In the Domain Structure frame, click (**+**) to expand Services and then select **Data Sources**.
3. Click **New**.
4. Select **Generic DataSource**.
5. Specify a name (for example, `Banner9DS`).
6. Specify the JNDI name (for example, `jdbc/bannerDataSource)`.
7. Specify `Oracle` for Database Type and then click **Next**.
8. Select **Oracle Driver (Thin) for Service Connections** and then click **Next**.
9. Clear the **Supports Global Transactions** check box and then click **Next**.
10. Enter the database name, host name, port, user name, password, and password confirmation, and then click **Next**. For example:

Database name:	`BAN9`
Host name:	`yourhostname.yourdomain.com`
Port:	`1521`
UserName:	`banproxy`
Password:	`your_password`

11. Click **Test Configuration**.
12. Click **Next** for the connection test to be successful.
13. Select the server that you previously created to allow the datasource to be deployed and used by this server.
14. Click **Finish**.
15. Select the datasource link that you created.
16. Select the **Connection Pool** tab.

    16.1. Set the Initial Capacity parameter to specify the minimum number of database connections to create when the server starts up. For example:

    ```
    Initial Capacity = 5
    ```

16.2. Set the Maximum Capacity parameter to specify the maximum number of database connections that can be created. For example:

```
Maximum Capacity = 100
```

17. Change `Statement Cache Type = Fixed`.
18. Change `Statement Cache Size = 0`.
19. Click **Save**.
20. In the Change Center frame, click **Activate Changes**.

Create a self-service datasource and connection pool

Note: If you previously created a self-service datasource, you can skip this section.

To create a self-service datasource and connection pool, perform the following steps:

1. In the Change Center frame, click **Lock & Edit**.
2. In the Domain Structure frame, click (+) to expand Services and then select Data Sources.
3. Click New.
4. Select Generic DataSource.
5. Specify a name (for example, Banner9SsbDS).
6. Specify the JNDI name (for example, jdbc/bannerSsbDataSource).
7. Specify Oracle for Database Type and then click Next.
8. Select Oracle Driver (Thin) for Service Connections and then click Next.
9. On the Transaction Options page, clear the Supports Global Transactions check box and click Next.
10. Enter the database name, host name, port, user name, password, and password confirmation, and click Next.
11. Click Test Configuration.
12. Click Next for the connection test to be successful.
13. Select the server that you previously created to allow the datasource to be deployed and used by this server.
14. Click Finish.
15. Select the datasource link that you created.
16. Select the Connection Pool tab.

16.1. Set the Initial Capacity parameter to specify the minimum number of database connections to create when the server starts up. For example:
Initial Capacity = 5

16.2. Set the Maximum Capacity parameter to specify the maximum number of database connections that can be created. For example:
Maximum Capacity = 100

17. Change Statement Cache Type = LRU.
18. Change Statement Cache Size = 20.
19. Click Save.
20. In the Change Center frame, click Activate Changes.

Deploy and start the application in the WebLogic server

To deploy and start the web application in the WebLogic server, perform the following steps:

1. Change the name of the WAR file to remove the version number. For example, change:

   ```
   applicationNavigator/current/dist/applicationNavigator-
   3.0.war
   ```

   to

   ```
   applicationNavigator/current/dist/
   applicationNavigator.war
   ```

2. Access the administration server at the following URL:

   ```
   http://server:7001/console
   ```

3. In the Domain Structure frame, select the **Deployments** link.
4. In the Change Center frame, select **Lock and Edit**.
5. Click **Install**.
6. Select the WAR file to be deployed and then click **Next**. The file is located in the following directory:

   ```
   applicationNavigator/current/dist
   ```

7. Select **Install this deployment** as an application and then click **Next**.
8. Select the target server on which to deploy this application (for example, `Banner9`) and then click **Next**.
9. Click **Finish**.
10. In the Change Center frame, click **Activate Changes**.
11. Select the deployed application and then click **Start**.
12. Select **Servicing all request**.

13. Access the application using the following URL format:

    ```
    http://<servername>:<port>/<web application>
    ```

    For example:

    ```
    http://localhost:8180/applicationNavigator
    ```

14. Log in to the application using a valid username and password.

Install Application Navigator for SAML 2.0 SSO

Undeploy the existing application

Before you reinstall Application Navigator, you must undeploy the existing application. The following sections provide the required steps to undeploy the existing Application Navigator installation in Tomcat and WebLogic servers.

Tomcat

You can either use the Tomcat Manager web application to undeploy the existing application or you can shut down Tomcat and manually remove the files.

Undeploy using the Tomcat Manager web application

Use the following procedure to undeploy the application using the Tomcat Manager web application:

1. Access the Tomcat Manager web application at one of the following URLs:

   `http://server:8080/manager`

   - or -

   `http://server:8080/manager/html`
2. Access the deployment page using a valid user name and password.
3. Under the Commands area, click **Stop** to stop the existing application.
4. In the confirmation dialog box, click **OK**.
5. Under the Commands area, click **Undeploy**.
6. In the confirmation dialog box, click **OK** to undeploy the application.

Undeploy using a manual procedure

The following sections give the steps to manually undeploy the existing application on Unix and Windows operating systems.

Unix

Use the following procedure to manually undeploy the existing application on a Unix operating system:

1. Log in to the server where Tomcat is running, using the same account credentials that were used to start Tomcat.
2. Shut down Tomcat by running the shutdown script:

   ```
   $CATALINA_HOME/bin/shutdown.sh
   ```

3. Remove the current deployment and associated WAR file:

   ```
   cd $CATALINA_HOME
   rm -rf $CATALINA_HOME/webapps/applicationNavigator
   rm -rf $CATALINA_HOME/webapps/applicationNavigator.war
   ```

Windows

Use the following procedure to manually undeploy the existing application on a Windows operating system:

1. Use the command prompt to shut down Tomcat.

   Note: If you installed Tomcat as a service, use the Service Control panel to stop the application. Otherwise, use the shutdown script `%CATALINA_HOME%\bin\shutdown.bat`.

2. Remove the current deployment and associated WAR file:

   ```
   rmdir %CATALINA_HOME%\webapps\applicationNavigator /s/q
   del %CATALINA_HOME%\webapps\applicationNavigator.war /q
   ```

WebLogic

Use the following procedure to stop and undeploy the application:

1. Access the administration server using the following URL:

   ```
   http://server:7001/console
   ```

2. In the Domain Structure frame, click **Deployments**.
3. In the Change Center, click **Lock and Edit**.
4. Select the check box to the left of the application.
5. Click **Stop**.
6. Click **Force Stop Now**.
7. In the Force Stop Application Assistant page, click **Yes**.
8. Select the check box to the left of the application.

9. Click **Delete**.
10. In the Delete Application Assistant page, click **Yes**.
11. In the Change Center frame, click **Activate Changes**.

Customize the WAR file

The name of the release package is `release-applicationNavigator-3.0.zip`. This release package is moved to the `$BANNER_HOME\general\java` subdirectory during the database upgrade. Use the following steps to unzip the release package and customize the WAR file for your institution.

Note: JDK 1.7 must be installed on your system. See the Java dependencies section for more information.

When you locate the release package zip file, copy it to your application server environment. Transfer this file in binary mode using File Transfer Protocol (FTP). To copy the release package, you must have a valid application server account.

Unzip the release package

To unzip the release package into a temporary directory, perform the following steps:

1. Log in to the application server platform.

   Note: You must have a valid application server account to deploy into the application server container (Tomcat or WebLogic).

2. Create a temporary directory. For example:

   ```
   mkdir $HOME/ban9temp
   ```

3. Locate the release package `release-applicationNavigator-3.0.zip.`
4. Transfer this file in binary mode using File Transfer Protocol (FTP) file into the temporary directory. For example:

   ```
   $HOME/ban9temp
   ```

5. Unzip `release-applicationNavigator-3.0.zip` into the temporary directory.

Prepare the installer

To prepare the installer, perform the following steps:

1. Change the directory to the installer directory:

```
cd installer
```

2. Run the `ant` command, which will build the installation tool.

   Note: For Unix, make sure the ant file is executable. For example, `chmod +x ant`.

Example:

```
ban9temp $ cd installer
ban9temp/installer $ ./ant
```

The message *Build successful* confirms a successful build.

Install into the product home directory

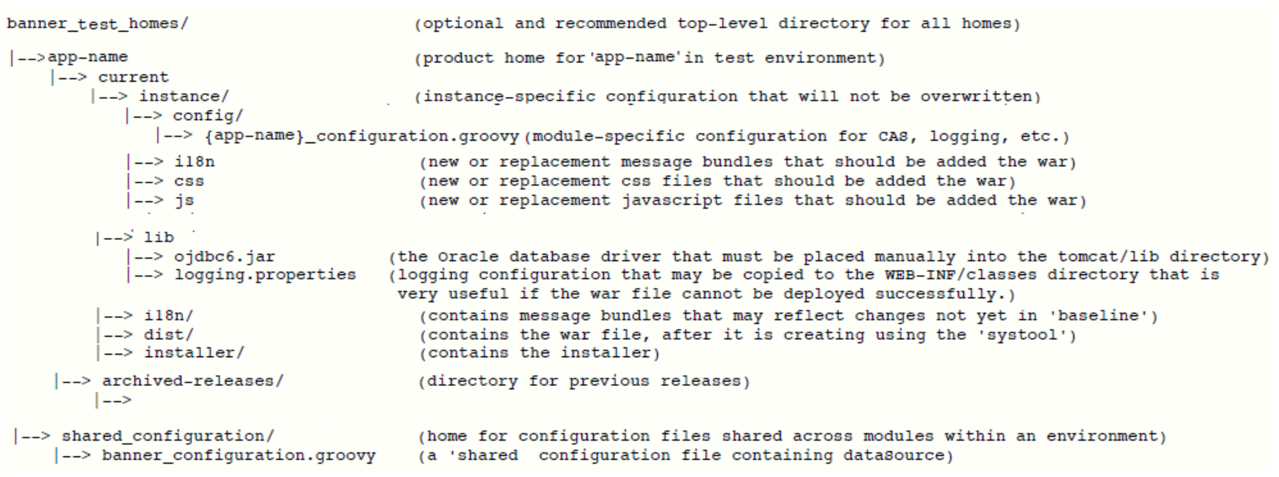
The product home directory supports the configuration and creation of a deployable WAR file. Although Banner 9.x web applications are modular and are installed independently, they share a common configuration. The package provides a common installer that creates consistent product home directory structures for all Banner 9.x applications.

Within a particular environment, you should place the product home directories for Banner 9.x applications in sibling directories. For example, the following directory structure includes four product home directories and a `shared_configuration` directory that support a common test environment.

```
banner_test_homes
|--> Registration 9.2
|--> Catalog 9.3
|--> Schedule 9.3
|--> applicationNavigator
|--> shared_configuration
```

A product home directory is created for each deployment. For example, the home directory that is used for the application within a test environment is different than the home directory that is used for the production environment. When you are supporting different environments for multiple home directories for the same solution, this structure provides the necessary configuration, release level, and custom modification flexibility.

The following directory tree illustrates the product home directory that is created for the test environment:



```
banner_test_homes/                    (optional and recommended top-level directory for all homes)

|-->app-name                          (product home for'app-name'in test environment)
    |--> current
        |--> instance/                (instance-specific configuration that will not be overwritten)
            |--> config/
               |--> {app-name}_configuration.groovy (module-specific configuration for CAS, logging, etc.)
            |--> i18n                  (new or replacement message bundles that should be added the war)
            |--> css                   (new or replacement css files that should be added the war)
            |--> js                    (new or replacement javascript files that should be added the war)

         |--> lib
            |--> ojdbc6.jar         (the Oracle database driver that must be placed manually into the tomcat/lib directory)
            |--> logging.properties  (logging configuration that may be copied to the WEB-INF/classes directory that is
                                      very useful if the war file cannot be deployed successfully.)
         |--> i18n/                    (contains message bundles that may reflect changes not yet in 'baseline')
         |--> dist/                    (contains the war file, after it is creating using the 'systool')
         |--> installer/               (contains the installer)

    |--> archived-releases/            (directory for previous releases)
         |-->

 |--> shared_configuration/            (home for configuration files shared across modules within an environment)
    |--> banner_configuration.groovy   (a 'shared  configuration file containing dataSource)
```

In addition to the application's product home directory, a separate `shared_configuration` home directory contains cross-application configuration for the test environment. This directory holds the `banner_configuration.groovy` file, which contains the shared JNDI datasource configuration.

To install the installer into the product home directory, perform the following steps:

1. Ensure that the installer is prepared using `ant`.
2. Use the installer to install the release file into the product home directory.

   Note: Your current working directory must be in the installer directory (`ban9temp/installer`) before executing the following commands.

   On Unix:
   ```
   $ bin/install home
   ```

   On Windows:
   ```
   > bin\install home
   ```

3. When prompted, enter the full path of the application home directory. The application will be installed within the current subdirectory within this home directory and the previous release will be archived.

   On Unix:
   ```
   []: Current_home_directory/banner_test_homes/
   applicationNavigator
   ```

   On Windows:
   ```
   []: c:\banner_test_homes\applicationNavigator
   ```

4. Enter the full path of the `shared_configuration` home directory. Banner 9.x applications that refer to this home directory share this configuration file.

   On Unix:
   ```
   []: Current_home_directory/banner_test_homes/
   shared_configuration
   ```

   On Windows:
   ```
   []: c:\banner_test_homes\shared_configuration
   ```

Note: If an identified home directory or the `shared_configuration` home directory does not exist, the installer creates it. The name of a product home directory is not restricted. You can name it when prompted by the installer.

Configure shared settings

The `shared_configuration` home directory contains a cross-application configuration file called `banner_configuration.groovy`. You can change settings in this file.

JNDI datasource

You can configure both Banner Administrative and Self-Service datasources. You can optionally change the datasource name in the configuration file to point to the JNDI datasource that is configured in your application server. For example, `jndiName = "jdbc/bannerDataSource"` or `"jdbc/bannerSsbDataSource"` are the default configurations. You can change this to match the JNDI datasource names in your environment.

Generate SAML 2.0 metadata files

This section discusses the creation and handling of metadata files. The following files must be created:

- keystore file
- service provider file
- identity service provider file

An IDP Certificate entry must be added in the newly created keystore file. Then the keystore, service provider, and identity service provider files must be added to the WAR file creation location.

Create a keystore (*.jks) file

1. Create a keystore file (`*.jks`) with the name used in step 2.

   See "Create a keystore (*.jks) file" on page 106 for more information.

2. Place this file in the location specified in the following key value: "`grails.plugin.springsecurity.saml.keyManager.storeFile = 'classpath:security/appnavkeystore.jks'`"

Create a service provider file

1. Create a file `banner-<short-appName>-sp.xml` at the folder path mentioned in “SAML 2.0 SSO Configuration” on page 85.
2. Edit the `banner-<short-appName>-sp.xml` file for the service provider configuration which will configure the Authentication end point and Logout endpoint.

   Replace the parameters below with the configured values for Application Navigator.

   - `<HOSTNAME>`: Application host name
   - `<PORT>`: Deployed Application port number
   - `<ALIAS_NAME>`: Service provider ID set in the Ellucian Ethos Identity service provider setup
   - `<EXTRACTED_DATA>`: Extract a X509 Certificate key from the keystore for `banner-<short-appName>-sp.xml` (See “Extract a X509 Certificate Key” on page 106 for more information.)

     Place the extracted value in the `banner-<short-appName>-sp.xml` file:

     `banner-<short-appName>-sp.xml`

   Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
ID="<ALIAS_NAME>" entityID="<ALIAS_NAME>">
<md:SPSSODescriptor AuthnRequestsSigned="false" WantAssertionsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>
<EXTRACTED_DATA>
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>
<EXTRACTED_DATA>
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/saml/SingleLogout/alias/
<ALIAS_NAME>"/>
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/saml/SingleLogout/
alias/<ALIAS_NAME>"/>
```

```
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress</
md:NameIDFormat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:transient</
md:NameIDFormat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-format:persistent</
md:NameIDFormat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</
md:NameIDFormat>

<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName</
md:NameIDFormat>

<md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
POST" Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/saml/SSO/alias/
<ALIAS_NAME>" index="0" isDefault="true"/>

<md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:profiles:holder-
of-key:SSO:browser" Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/saml/
SSO/alias/<ALIAS_NAME>"
hoksso:ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Artifact"
index="1" xmlns:hoksso="urn:oasis:names:tc:SAML:2.0:profiles:holder-of-
key:SSO:browser"/>

<md:AssertionConsumerService Binding="urn:oasis:names:tc:SAML:2.0:profiles:holder-
of-key:SSO:browser" Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/saml/
SSO/alias/<ALIAS_NAME>"
hoksso:ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST" index="2"
xmlns:hoksso="urn:oasis:names:tc:SAML:2.0:profiles:holder-of-key:SSO:browser"/>

</md:SPSSODescriptor>

</md:EntityDescriptor>
```

Create an identity service provider file

1. Create a file `banner-<short-appName>-idp.xml` at the folder path mentioned in “SAML 2.0 SSO Configuration” on page 85.
2. Edit the `banner-<short-appName>-idp.xml` file for the identity provider configuration.

   The file contains the identity provider information configured in the Ellucian Ethos Identity server over which Application Navigator sends the SAMLrequest and receives the SAML response.

   Replace below parameters with the configured values for Application Navigator.

   - `<HOSTNAME>`: the Ellucian Ethos Identity server host name
   - `<PORT>`: Deployed the Ellucian Ethos Identity server port number
   - `<EXTRACTED_DATA>`: Extract X509 certificate Data for `banner-<short-appName>-idp.xml` (See “Extract X509 certificate data” on page 108 for more information.)

     Place the extracted value in the `banner-<short-appName>-idp.xml` file.

     `banner-<short-appName>-idp.xml`

   Example:

```
<?xml version="1.0"?>

<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
entityID="https://<HOSTNAME>:<PORT>/samlsso" cacheDuration="PT1440M">

<md:IDPSSODescriptor
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
```

```
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>
<EXTRACTED_DATA>
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>
<EXTRACTED_DATA>
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:SingleLogoutService Location="https://<HOSTNAME>:<PORT>/samlsso"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"/>
<md:SingleLogoutService Location="https://<HOSTNAME>:<PORT>/samlsso"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"/>
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified</
md:NameIDFormat>
<md:SingleSignOnService Location="https://<HOSTNAME>:<PORT>/samlsso"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"/>
<md:SingleSignOnService Location="https://<HOSTNAME>:<PORT>/samlsso"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"/>
</md:IDPSSODescriptor>
<md:ContactPerson contactType="administrative"/>
</md:EntityDescriptor>
```

Add an IDP Certificate entry in the newly created keystore file

Add the IDP certificate entry in the newly created `.jks` file. See "Add IDP certificate entry to the .jks file" on page 110 for more information.

Add keystore, service provider, and identity service provider files to WAR file creation location

Place the three files created in this section into the `applicationNavigator\current\instance\config` directory.

After adding the files, the directory should contain the following:

- `appnavkeystore.jks` (The newly created `.jks` file)
- `banner-<short-appName>-idp.xml`

- `banner-<short-appName>-sp.xml`
- `applicationNavigator_configuration.groovy`

Configure application-specific settings

The `applicationNavigator\current\instance\config` directory contains the `applicationNavigator_configuration.groovy` file. This application-specific configuration file contains settings that you can customize for your specific environment. This directory also contains an `instance.properties` file that references the shared configuration location.

JMX MBean name

The name that is used to register MBeans must be unique for each application that is deployed into the JVM. This configuration should be updated for each instance of each application to ensure uniqueness.

```
jmx {
    exported {
        log4j = "applicationNavigator-log4j"
    } }
```

Location of the logging file

Log4j is the common logging framework used with applications that run on the Java Virtual Machine. For more information, refer to the log4j documentation.

The configuration file includes documentation on various elements that can be modified depending on your environment.

The following is an example of how to override the location where the log file is saved.

```
def string loggingFileDir = "System.properties['logFileDir'] ?
"${System.properties['logFileDir']}" : "target/logs"
def string logAppName = "applicationNavigator"
def string loggingFileName = "${loggingFileDir}/
${logAppName}.log".toString()
```

The following is an example of how to override the log file directory properties:

```
export JAVA_OPTS = "-DlogFileDir=/PRODUCT_HOME /"
```

The output logging file location is relative to the application server to which you are deploying.

Logging level

The root logging level is pre-configured to the ERROR level. Multiple class or package level configurations, by default, are set to a status of "off." You can set a different logging level for any package or class. However, the running application must be restarted.

For example:

```
case 'production':
root {
error 'appLog' //change the log level here with the
appropriate log level value.
additivity = true
}
```

Note: Changing the logging level to DEBUG or INFO produces very large log files.

Changes to the `applicationNavigator_configuration.groovy` file require a restart of the application before those changes take effect.

Alternatively, you can use JMX to modify logging levels for any specified package or class, or even at the root level. When using JMX, the logging level changes only affect the running application. When you restart the application, changes that you made using JMX are lost.

For more information on JMX configuration, see “Configure Java Management Extensions” on page 59.

Institutional Home Page Redirection Support

With previous versions of the Application, users did not have the option to go back to a home page if they encountered any SSO access issues. This configuration allows Application Navigator to provide an institutional home page where users can navigate back to, if they encounter access issues specific to insufficient privileges when accessing the application.

```
/*******************************************************************
* Home Page URL configuration for CAS / SAML Single-Sign On *
*******************************************************************/
// Can be institutional home page ex: http://myportal/main_page.html
grails.plugin.springsecurity.homePageUrl='http://APPLICATION_NAVIGATOR_HOST:PORT/
applicationNavigator'
```

Seamless plugin configurations

The following properties describe the configurations required for the seamless navigation plugin. Each property is described in the following table.

Property	Description
`seamless.interceptPattern`	Pre-populated to pull in the CAS server URL
`seamless.menuEndpoints`	Array of the menu endpoint URLs that must be wrapped in quotes and comma-delimited
`seamless.logLevel`	Log level for messages, includes *error* and *debug*
`seamless.brandTitle`	Name of your institution that will be displayed as the title in the navigation bar of Application Navigator. The title is an anchored link that users can click at any time to navigate back to the landing page. Configure the brand title with a default institution value or based on the MEP institution code configured in the Banner database. The sample default value provided is “Ellucian University”
`seamless.ajaxTimeout`	AJAX request timeout in milliseconds
`seamless.messageResponse Timeout`	Timeout for Application Navigator waiting on responses from embedded applications in milliseconds
`seamless.exposeMenu`	Boolean logic that determines whether to enable the example menu service
`seamless.sessionTimeout`	Time in minutes when the session should timeout. Default is 30 minutes, and a value of -1 will keep the session alive indefinitely.
`seamless.sessionTimeout Notification`	Time in minutes when the notification prompt will be displayed to the user prior to session timeout. Default is 5 minutes.
`seamless.excludeObjectsFrom Search`	This list includes objects to be excluded from search.

Example:

```
seamless.interceptPattern =
"${grails.plugin.springsecurity.cas.serverUrlPrefix}.*"
     seamless.menuEndpoints = [
        "http://APPLICATION_NAVIGATOR_HOST:PORT/applicationNavigator/
     commonMenu",
        "http://APPLICATION_NAVIGATOR_HOST:PORT/applicationNavigator/
     commonSelfServiceMenu"

]
seamless.logLevel="off"
seamless.brandTitle=["Default": "Ellucian University"]
seamless.ajaxTimeout=30000
seamless.messageResponseTimeout=2000
seamless.exposeMenu=true
seamless.sessionTimeout = 30
seamless.sessionTimeoutNotification = 5
seamless.excludeObjectsFromSearch = [
"GUAGMNU","GUAINIT","GUQSETI","FOQMENU","SOQMENU","TOQMENU","AOQMEMU",
"GOQMENU","ROQMENU","NOQMENU","POQMENU","FACICON","FAQINVP","FAQMINV",
"FAQVINV","FGQACTH","FGQAGYH","FGQDOCB","FGQDOCN","FGQDOCP","FGQFNDE",
"FGQFNDH","FGQLOCH","FGQORGH","FGQPRGH","FOQADDR","FOQDCSR","FOQENCB",
"FOQFACT","FOQINVA","FOQJVCD","FOQPACT","FOQRACT","FOQSDLF","FOQSDLV",
"FPCRCVP","FPQBLAP","FPQCHAP","FRCBSEL","FSCISSR","FSCSTKL","FTQATTS",
"FXQDOCN","FXQDOCP","**SSB_MASKING","TSQCONT","TSQEXPT","TOQCALC",
"GPBADMN","SFQESTS","SFQPREQ","SFQRQST","SFQRSTS","SFQSECM","SFQSECT",
"SHQDEGR","SHQQPNM","SHQSECT","SHQSUBJ","SHQTERM","SHQTRAM","SLQBCAT",
"SLQEVNT","SLQMEET","SLQROOM","SMQSACR","SMQSGCR","SMQSGDF","SMQSPDF",
"SOQCSCP","SOQCTRM","SOQHOLD","RPQLELG","RPQCOMP","ROQADDR"
]
```

Display name support

You can configure the header in Application Navigator to display a person's preferred name. To enable this functionality, edit the following configuration settings.

```
productName='Banner General' //Name of the product
```

The name in the GURNHIR table and the name mentioned in this configuration should match.

```
banner.applicationName='Application Navigator' //Application
Name
```

The name in the GURNHIR table configured and name mentioned in this configuration should match.

You need to have General 8.8.5 installed to use this functionality. Ensure the seed data script delivered as part of the Application Navigator release package is executed. If the seed data is not available, refer to the values in the following table to use the Name Display (GUANDSP) form and enter the information manually.

Product	Application
Banner General	Application Navigator

MEP support

Set the value below to ***`true`*** in a MEP database environment.

```
mepEnabled = false
grails.plugin.springsecurity.logout.mepErrorLogoutUrl = '/logout/
customLogout'
```

Self-Service support

- Set 'ssbEnabled' to true for instances that expose Self-Service Banner endpoints. If this is set to false, or if this configuration item is missing, the instance will only support Administrative applications and not Self-Service applications in the unified menu. If this is enabled, Application Navigator will integrate with Banner Self-Service applications using the SSB datasource.

```
ssbEnabled = true
```

- The ssbOracleUsersProxied setting is set to false for Application Navigator deployment by default. Only set 'ssbOracleUsersProxied = true' to ensure that database connections are proxied when the Self-Service user has an Oracle account and there is a requirement to set fine-grained access control (FGAC) for Application Navigator menus. This setting in Application Navigator has no impact on integrated Banner 9 Self-Service applications. The integrated Banner 9 Self-Service applications can be configured separately to allow FGAC on the application specific SSB pages.

```
ssbOracleUsersProxied = false
```

- Add commonSelfServiceMenu endpoint to the seamless menu endpoints list if Self-Service menus are to be loaded.

```
  seamless.menuEndpoints = [
        "http://<application navigator host>:<port>/applicationNavigator/
commonMenu",
        "http://<application navigator host>:<port>/applicationNavigator/
commonSelfServiceMenu"
]
```

- List the URL entries of Banner Self-Service applications integrating with Application Navigator.

```
  seamless.selfServiceApps = [
  "http://<Self-Service application host>:<port>/<Banner SSB
application>",
  "http://<Self-Service application host>:<port>/<Banner SSB
application>"
      ]
```

If Application Navigator is configured with a MEP Database, then the URLs of the Self Service Applications must be appended with the MEP code query parameter as shown below:

```
  "http://<Self-Service application host>:<port>/<Banner SSB
application>
  ?mepCode={mepCode}"
```

Note: Banner 9 Self-Service application URLs must also exist in the Web Tailor Menu table and should be an exact match with the entries noted in the configuration property.

X-Frame-Options settings

Make sure the values below are set, so that Application Navigator will not be exposed to clickjacking vulnerability when loaded in an iframe.

```
grails.plugin.xframeoptions.urlPattern = '/login/auth'
grails.plugin.xframeoptions.deny = true
```

Spring Security Port Mapper Configuration

The Spring Security Port Forwarding Configuration entries support menu service callbacks based on port forwarding requests. The spring security port mapper configuration is utilized to override the pre-configured port set during application deployment and ensures that the callback URLs are redirected to the forwarded port successfully at run-time.

```
grails.plugin.springsecurity.portMapper.httpPort = <port number>
grails.plugin.springsecurity.portMapper.httpsPort = <SSL port number>
```

Google Analytics

Application Navigator uses Google Analytics to capture usage details of specific pages, including the default authentication login or logout page and the default authentication error pages. All Banner Self-Service 9 applications that are integrated with Application Navigator and configured with Google Analytics will be tracked. This feature is delivered enabled. To enable or disable Google Analytics, use this configuration setting.

```
banner.analytics.trackerId=[institution’s google analytics
tracker ID - default blank]
banner.analytics.allowEllucianTracker=[true|false - default
true]
```

Example:

```
banner.analytics.trackerId = ‘UA-83915850-1’
banner.analytics.allowEllucianTracker = false
```

Use cases:

- If a configuration is not in the configuration file, by default the Ellucian tracking ID will be enabled and Ellucian will track analytics.
- If allowEllucianTracker=true, Ellucian will track the analytics data.
- If allowEllucianTracker=false, the tracking script will not be added in the gsp page.
- If there is only the clientTracker ID in the configuration, then both Ellucian and the client will be tracking the Google Analytics data.
- If allowEllucianTracker=true and the client tracker ID is in the configuration, then both the client and Ellucian will be tracking the analytics data.
- If allowEllucianTracker=false and the client tracker ID is in the configuration, then analytics will be tracked by the client, not Ellucian.

Set up SAML 2.0 SSO Configuration

The `applicationNavigator\current\instance\config` directory contains the `applicationNavigator_configuration.groovy` file. This application specific configuration file contains settings that you can customize for your specific environment.

Authentication Provider Name

The name identifying the application authentication mechanism. Values are **cas** or **saml**. Specify saml to indicate the application will use SAML SSO protocol for authentication as shown in the following example:

```
/*******************************************************
 *     BANNER AUTHENTICATION PROVIDER CONFIGURATION    *
 *                                                     *
 *******************************************************/
    //
    banner {
        sso {
            authenticationProvider           = 'saml' //  Valid values are:
    'saml' and 'cas' for SSO. 'default' value to be used only when creating
    the release zip file.
            authenticationAssertionAttribute = 'UDC_IDENTIFIER'
            if(authenticationProvider != 'default') {
              grails.plugin.springsecurity.failureHandler.defaultFailureUrl
    = '/login/error'
            }
    }
```

Logout URL

You can specify where a user is directed after logging out of the application by updating the `applicationNavigator_configuration.groovy` file. There are three ways the application can handle logouts:

- Logouts can display the CAS logout page with a redirect URL.
- Logouts can automatically go to a redirect URL (without displaying the CAS logout page).
- Logouts can take the user to a custom logout page with a redirect URL to Home Page.

In saml mode, logout directs the user to a custom logout page.

SAML 2.0 SSO Configuration

Shown below is a sample of the configuration you can enable for SSO between Application Navigator and an Identity Management System that support SAML 2.0 SSO protocol. Make sure to uncomment this section when SAML 2.0 SSO is enabled.

The following properties describe the configurations required for the application to work in SAML 2.0 SSO protocol. Each property is described in the following table:

Property	Description
`grails.plugin.springsecurity.saml.active`	Default value is set to false. Set active = true if Application Navigator is configured with SAML2 SSO.
`banner.sso.authentication.saml.localLogout`	Default value is set to false, indicating that the application will participate in Global logout. An application participating in global logout will notify the Identity Server about logout within the application. If this is set to true, indicating local logout, the application will not notify the Identity Server to log the user out from all applications.
`grails.plugin.springsecurity.auth.loginFormUrl`	Pre-populated to provide the Login URL.
`grails.plugin.springsecurity.saml.afterLogoutUrl`	Pre-populated to provide the Logout URL.
`grails.plugin.springsecurity.saml.keyManager.defaultKey`	Key name used at the time of key-store creation ("Create a keystore (*.jks) file" on page 106). Example: `appnavkeystore.jks`
`grails.plugin.springsecurity.saml.keyManager.storeFile`	Location of the keystore file. This could be a classpath (for example, `classpath:securityappnavkeystore.jks`) or it could be an absolute location on the machine (for example, `file:c://temp/appnavkeystore.jks` or `u02/appnavkeystore.jks`).
`grails.plugin.springsecurity.saml.keyManager.storePass`	Password used to decrypt the keys in the keystore created above.
`grails.plugin.springsecurity.saml.keyManager.passwords`	Key value pair to validate the key. Contains the alias key name used at the time of key-store creation and password used to decrypt the key.

Property	Description
`grails.plugin.springsecurity.saml.metadata.sp.file`	Location of the service provider metadata file. This could be a classpath location, (for example, `security/sp.xml`) or it could be a absolute location on the machine (for example, `C://temp/banner-appnav-sp file:/home/u02/banner-sp.xml`).
`grails.plugin.springsecurity.saml.metadata.providers`	Key value pair mapping to validate the identity provider configured in `banner-<short-appName>-idp.xml`. Example: `adfs : 'security/banner-appnav-idp.xml'`. Possible keys are adfs, Okta, Shibb, etc.
`grails.plugin.springsecurity.saml.metadata.defaultIdp`	Provide the default IDP to be used from the IDP providers set. This is the same value specified above for the key.
`grails.plugin.springsecurity.saml.metadata.sp.defaults`	• local: Pre-populated to indicate value to be picked up. • alias: An alias name that is unique to this application (for example, `banner-<application-shortname>-sp`). • securityProfile: Pre-populated value. • signingKey: A key used to sign the messages that is unique to this application (for example, `banner-<application-shortname>-sp`). • encryptionKey: A key to to encrypt the message that is unique to this application, (for example, `banner-<application-shortname>-sp`) • tlsKey: A tls key that is unique to this application (for example, `banner-<application-shortname>-sp`). • requireArtifactResolveSigned: Pre-Populated to set to false indicating artifact to be signed or not. • requireLogoutRequestSigned: Pre-Populated to set to false indicating logout request to be signed or not. • requireLogoutResponseSigned: Pre-Populated to set to false indicating logout response to be signed or not.

```
/*******************************************************************************
 *                                                                             *
 *                         SAML2 SSO Configuration                             *
 *                                                                             *
 *******************************************************************************/
// Set active = true when Application Navigator is configured for SAML2 SSO
```

```
grails.plugin.springsecurity.saml.active = true
grails.plugin.springsecurity.saml.afterLogoutUrl ='/logout/customLogout'

banner.sso.authentication.saml.localLogout='false' // Setting localLogout to false, allows
the application to send a single or global logout request to the Identity Service Provider

grails.plugin.springsecurity.saml.keyManager.storeFile = 'classpath:security/
samlkeystore.jks'     // for unix based 'file:/home/u02/samlkeystore.jks'
grails.plugin.springsecurity.saml.keyManager.storePass = 'changeit'
grails.plugin.springsecurity.saml.keyManager.passwords = [ 'banner-appnav-sp': 'changeit' ]
// banner-appnav-sp is the value set in EIS Service provider setup
grails.plugin.springsecurity.saml.keyManager.defaultKey = 'banner-appnav-sp'
// banner-appnav-sp is the value set in EIS Service provider setup

grails.plugin.springsecurity.saml.metadata.sp.file = 'security/banner-appnav-sp.xml'
// for unix based '/home/u02/banner-appnav-sp.xml'
grails.plugin.springsecurity.saml.metadata.providers = [adfs: 'security//banner-appnav-
idp.xml']
// for unix based '/home/u02/banner-appnav-idp.xml'
grails.plugin.springsecurity.saml.metadata.defaultIdp = 'adfs'
grails.plugin.springsecurity.saml.metadata.sp.defaults = [
        local: true,
       alias: 'banner-appnav-sp',
// banner-appnav-sp is the value set in EIS Service provider setup
        securityProfile: 'metaiop',
       signingKey: 'banner-appnav-sp',
// banner-appnav-sp is the value set in EIS Service provider setup
       encryptionKey: 'banner-appnav-sp',
// banner-appnav-sp is the value set in EIS Service provider setup
       tlsKey: 'banner-appnav-sp',
// banner-appnav-sp is the value set in EIS Service provider setup
        requireArtifactResolveSigned: false,
        requireLogoutRequestSigned: false,
        requireLogoutResponseSigned: false
]
```

Customize the landing page background image

To customize and replace the landing page background image with an institutional image of your choice, you must perform the following steps before building and deploying the WAR file to the Application Server:

1. Create two CSS files in the deployment staging directory of Application Navigator (example: ~/applicationNavigator/current/instance/css).
   - `applicationNavigator-custom.css` for LTR support
   - `applicationNavigator-custom-rtl.css` file for RTL support
2. Modify each of these two files by adding the custom CSS styles you wish to change as shown below. For overriding the landing page background image, use the following CSS class and add your institutional background image to the media query style that supports the specific responsive design orientation.

```
/** Custom CSS file which takes precedence over the landing page CSS styles **/
```

```
.landing-content {
    margin-left: 0px;
    padding-left: 0px;
    background-repeat: no-repeat;
    background-position:center;
    background-size: cover;
    background-color: #4F585F;
}

@media (orientation:landscape) {
.landing-content {
background-image: url("/css/images/backgrounds/campus-landscape.jpg");
}
}

@media (orientation:portrait) {
.landing-content {
background-image: url("/css/images/backgrounds/campus-portrait.jpg");
}
}
@media (orientation:portrait) and (max-width:320px) {
.landing-content {
background-image: url("/css/images/backgrounds/campus-sml.jpg");
}
}
```

3. The above images and styles shown support responsive designs for landscape, portrait and mobile views. Make sure your institutional background images you choose maintain the correct aspect ratio for displaying the custom images in landscape and portrait mode along with supporting different device screen resolutions.

   - `campus-landscape.jpg` supports widescreen desktop devices (supported resolution size 1920 * 1080)
   - `campus-portrait.jpg` supports laptops and tablet devices (supported resolution size 768 * 1024)
   - `campus-sml.jpg` supports mobile devices (supported resolution size 320 * 568)

4. Confirm the custom CSS files and images are in the deployment staging directory of Application Navigator. The directory and file structure should be similar as shown below:

```
applicationNavigator
|-----current
    |-----instance
        |-----css
            |-----applicationNavigator-custom.css
            |-----applicationNavigator-custom-rtl.css
            |-----images
                |-----backgrounds
                    |-----campus-landscape.jpg
```

```
|-----campus-portrait.jpg
|-----campus-sml.jpg
```

Note: The image name can be any custom image name. However, css file names must be the same as specified (`applicationNavigator-custom.css` and `applicationNavigator-custom-rtl.css`), and the class name used to override the background image of landing page also must be the same landing-content.

5. Continue with the next steps of regenerating and deploying the Application Navigator WAR file to your Application Server with the customized CSS files. Verify the landing page back-ground image has changed and meets your institutional needs.

Regenerate the WAR file

This section describes how you can regenerate the application WAR file to include your customizations and application-specific settings. You can then deploy the WAR file into your application server.

Use the systool to create the WAR file. Complete the following steps to set up the systool to create the WAR file:

1. Change your current working directory to the product home directory:

   ```
   PRODUCT_HOME/current/installer
   ```

2. Run the ant command, which builds the systool module.

   For Unix, make sure the ant file is executable, for example, `chmod +x ant.`

   ```
   $ cd PRODUCT_HOME/current/installer
   ```

   ```
   PRODUCT_HOME/current/installer $ ./ant
   ```

3. Create the WAR file using the systool module.

   Your current working directory must be in the `PRODUCT_HOME/current/installer` directory before you execute one of the following commands based on your platform:

   On Unix:

   ```
   $ bin/systool war
   ```

   On Windows:

   ```
   > bin\systool war
   ```

   The WAR file is created in the `PRODUCT_HOME/current/dist directory`.

You can use external configuration files by setting appropriate system properties, although the configuration files are included in the WAR file to make the WAR file self-sufficient. For information on external configuration, see “Configure the Tomcat server” on page 90 or

"Create a WebLogic server" on page 97.

Configure and deploy the WAR file to a web application server

The following sections provides information on configuring the web application and deploying the WAR file to a web application server:

- "Tomcat" on page 90
- "WebLogic" on page 96

Tomcat

The following sections provide information on configuring the web application and deploying the WAR file to the Tomcat server.

Note: If you choose to install the application on a Tomcat server, you do not need to install it on WebLogic.

Note: Supported Tomcat version is required. To download and install the Tomcat server, see http://tomcat.apache.org.

Configure the Tomcat server

Use the following steps to configure the Tomcat server:

1. Locate the Oracle JDBC jar files (`ojdbc6.jar and xdb6.jar`) in the `PRODUCT_HOME\current\lib` directory.

   Note: Later in the Tomcat configuration process, you will copy the Oracle JDBC jar files into the `\lib` folder under the Tomcat installation directory.

   The account that runs the Tomcat application server must configure environment settings to support the application.

2. On Linux, ensure `CATALINA_HOME` is defined to reference your Tomcat software installation location. For example, `CATALINA_HOME=/opt/apache-tomcat-xx` where xx indicates the point version of Tomcat you installed.

   Warning! *Do not perform this step on the Windows platform.*

3. Define `CATALINA_OPTS` to configure JVM settings. The following settings are recommended:

   ```
   CATALINA_OPTS=-server -Xms2048m -Xmx4g
   -XX:MaxPermSize=512m
   ```

   Note: If you are deploying multiple Banner 9.x applications to the same Tomcat server, increase the max heap (`-Xmx`) by `2g` and `-XX:MaxPermSize` by `128m`. You should deploy Banner 9.x administrative applications to one Tomcat server instance and Banner 9.x self-service applications to a separate Tomcat server instance.

   You can define this variable in the account's profile startup script, or you can add this definition in `$CATALINA_HOME/bin/catalina.sh` for Linux or `catalina.bat` for Windows.

4. (Optional) If you install Tomcat as a Windows service, specify the JVM arguments as follows:

   4.1. Select **Configure Tomcat** application from the Windows **Start** menu.

   4.2. Select the **Java** tab.

   4.3. In the **Java Options** field, add the following:

   ```
   -XX:MaxPermSize=384m
   ```

   4.4. Set the initial memory pool = 2048.

   4.5. Set the maximum memory pool = 4096.

   4.6. Save the settings.

   4.7. Restart the Tomcat Windows service.

5. (Optional) To set up the Tomcat server to enable remote JMX connections, perform the steps in the "Configure Java Management Extensions" section. This is useful for debugging and logging.

6. Define the JNDI datasource resource names for the application as follows:

   6.1. Edit `$CATALINA_HOME/conf/context.xml`.

   6.2. Uncomment `<Manager pathname="" />` to disable Tomcat session persistence. For example, change the following:

   ```
   <!-- Uncomment this to disable session persistence
   across Tomcat restarts -->

   <!--
   <Manager pathname="" />
   -->
   ```

   to:

   ```
   <!-- Uncomment this to disable session persistence
   across Tomcat restarts -->

   <Manager pathname="" />
   ```

6.3. Add the following ResourceLink definitions inside the `<Context>` element:

```
<ResourceLink global="jdbc/bannerDataSource"
                name="jdbc/bannerDataSource"
                type="javax.sql.DataSource"/>
<ResourceLink global="jdbc/bannerSsbDataSource"
                name="jdbc/bannerSsbDataSource"
                type="javax.sql.DataSource"/>
```

6.4. Save your changes in `context.xml`.

6.5. Edit `$CATALINA_HOME/conf/server.xml` to configure the database JNDI resource name and connection pool configuration.

6.6. Add the following Resource definitions inside the `<GlobalNamingResources>` element:

For Tomcat 7:

```
<Resource name="jdbc/bannerDataSource" auth="Container"
   type="javax.sql.DataSource"
   driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@//hostname:port/service_name"
   username="banproxy" password="the_banproxy_password"
   initialSize="5" maxActive="100" maxIdle="-1" maxWait="30000"
   validationQuery="select 1 from dual"
   testOnBorrow="true"/>

<Resource name="jdbc/bannerSsbDataSource" auth="Container"
   type="javax.sql.DataSource"
   driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@//hostname:port/service_name"
   username="ban_ss_user" password="ban_ss_user_pasword"
   initialSize="5" maxActive="100" maxIdle="-1" maxWait="30000"
   validationQuery="select 1 from dual"
   testOnBorrow="true"/>
```

For Tomcat 8:

```
<Resource name="jdbc/bannerDataSource" auth="Container"
   type="javax.sql.DataSource"
   driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@//hostname:port/service_name"
   username="banproxy" password="the_banproxy_password"
   initialSize="5" maxTotal="100" maxIdle="-1"
   maxWaitMillis="30000"
   validationQuery="select 1 from dual"
   accessToUnderlyingConnectionAllowed="true"
   testOnBorrow="true"/>

<Resource name="jdbc/bannerSsbDataSource" auth="Container"
   type="javax.sql.DataSource"
```

```
driverClassName="oracle.jdbc.OracleDriver"
url="jdbc:oracle:thin:@//hostname:port/service_name"
username="ban_ss_user" password="ban_ss_user_pasword"
initialSize="5" maxTotal="100" maxIdle="-1"
maxWaitMillis="30000"
validationQuery="select 1 from dual"
accessToUnderlyingConnectionAllowed="true"
testOnBorrow="true"/>
```

For example, if your database server name is `myserver.university.edu` and the Oracle TNS Listener is accepting connections on port 1521 and your database service name is SEED, then the URL is `jdbc:oracle:thin:@//myserver.university.edu:1521/SEED`.

6.7. Save your changes in `server.xml`.

6.8. Copy the Oracle JDBC jar files (`ojdbc6.jar and xdb6.jar`) from the `PRODUCT_HOME/current/lib` directory to the `$CATALINA_HOME/lib` directory.

6.9. Validate the configuration of the Tomcat server by starting the application server. To accomplish this, perform the following steps:

– Run `$CATALINA_HOME/bin/startup.`

For Linux:

```
cd $CATALINA_HOME
$ bin/startup.sh
```

For Windows:

```
cd %CATALINA_HOME%
> bin\startup.bat
```

– Browse http://servername:<port>.

To override the configuration that was added into the WAR file, you must set system properties to point to external configuration files. For example, to point to a configuration file residing in the `PRODUCT_HOME` directory, export `JAVA_OPTS=`

```
"-DBANNER_APP_CONFIG=/PRODUCT_HOME/shared_configuration/
banner_configuration.groovy
-DAPPLICATION_NAVIGATOR_CONFIG=/PRODUCT_HOME/
applicationNavigator/current/instance/config/
applicationNavigator_configuration.groovy".
```

Note: When externalizing the configuration files, there are certain filters that get injected into the web.xml file as part of the Single Sign-On and X-Frame HTTP Header settings for the WAR file. These values cannot be changed at runtime by referencing external configuration files. The WAR file will have to be regenerated in order for the new Single Sign-On and X-Frame HTTP Header settings to take effect. Any other configuration outside these two sets of filters can be overridden in the external configuration file at runtime.

Configure Java Management Extensions

This is an optional step that is needed only if you want to monitor or debug the application. Java Management Extensions (JMX) is a Java technology that supplies tools for managing and monitoring applications, system objects, devices, and service oriented networks.

Enabling JMX connections allows you to remotely monitor and debug the application server. To enable Java Management Extensions, perform the following steps:

1. Add the following options to the `catalina.sh` or `catalina.bat` file and then restart the Tomcat server:

   ```
   set CATALINA_OPTS=-Dcom.sun.management.jmxremote

   -Dcom.sun.management.jmxremote.port=8999

   -Dcom.sun.management.jmxremote.ssl=false

   -Dcom.sun.management.jmxremote.authenticate=false

   -Djava.rmi.server.hostname=your.hostname.com
   ```

2. Change the `java.rmi.server.hostname` value to the hostname or IP address of the machine where Tomcat is installed. For example:

   ```
   -Djava.rmi.server.hostname=prod.appserver1.com
   ```

   - or -

   ```
   -Djava.rmi.server.hostname=149.24.3.178
   ```

3. JMX does not define a default port number to use. If necessary, change `com.sun.management.jmxremote.port=8999`.

   Note: It is recommended that you connect remotely to the Tomcat server using JMX.

   Warning! *Ensure that the* `jmxremote.authenticate` *parameter is not set to* `false` *in a production environment. If it is set to* `false`*, it does not require connections to be authenticated and will create a security threat in a production environment. For more information on Tomcat Remote JMX documentation, see http://tomcat.apache.org/tomcat-x.x-doc/monitoring.html#Enabling_JMX_Remote (Where x.x is the base version of the Tomcat installed.).*

Deploy the WAR file to the Tomcat server

The systool that is used to create the WAR file can also be used to deploy the WAR file to a Tomcat container. You should deploy 9.x administrative applications and 9.x self-service applications to separate Tomcat servers to increase performance.

Note: The systool does not provide the capability to undeploy or redeploy an application. If you are redeploying the application, you must use the Tomcat Manager web application to undeploy the existing application.

The target supports deploying the `dist/WAR` file using the Tomcat Manager web application. Because environments vary significantly with respect to user privileges, clustering approach, web container version, operating system, and more, the target may or may not be suitable for your use.

Note: You can also deploy the WAR file to the Tomcat server by copying the WAR file to the Tomcat `webapps/` directory.

To use the target, you must provide the following information:

URL	This is the URL of the manager application in the Tomcat server. For example: `http://localhost:8080/manager`
Username	This Tomcat server username must have privileges to deploy WAR files.
Password	This is the password of the Tomcat server user.

Username/password combinations are configured in your Tomcat user database `<TOMCAT_HOME>\conf\tomcat-users.xml.` For Tomcat, you must configure at least one username/password combination with the manager role. For example:

```
<user username="tomcat" password="tomcat" (your password)
roles="manager-gui, manager"/>
```

Note: The roles in Tomcat server changed between point releases. Refer to the Tomcat documentation specific to your release for information on enabling access to provide the appropriate role to a user account for deployment.

To deploy the WAR file to the Tomcat server, perform the following steps:

1. Navigate to the `PRODUCT_HOME\current\installer` directory.
2. Enter one of the following commands:

   On Unix:
   ```
   $ bin/systool deploy-tomcat
   ```
   On Windows:
   ```
   > bin\systool deploy-tomcat
   ```
3. Enter the following URL for the Tomcat Manager:

   ```
   []: http://localhost:8080/manager
   ```

This URL will be accessed to deploy the WAR file into the container.

4. Enter a valid Tomcat username to deploy the WAR file. For example:

```
[]: tomcat
```

Note: This user must have the `manager-gui` role.

5. Enter the Tomcat password for the user:

```
[]: password
```

Note: This password will not be persisted.

6. Access the web application:

```
http://<servername>:<port>/applicationNavigator
```

WebLogic

The following sections provide information on configuring the web application and deploying the WAR file to the WebLogic server:

Note: If you choose to install the application on a WebLogic server, you do not need to install it on Tomcat.

Verify WebLogic prerequisites

Before configuring your WebLogic server, ensure that the following prerequisites are met:

- WebLogic must be installed. If it is not, download and install WebLogic from the Oracle web site.
- For minimum requirements on OFM and Weblogic support, refer to the Ellucian Oracle Support Calendar. The calendar is available in the Interactive Banner Compatibility Guide, which can be accessed from the Ellucian Download Center.
- Both the WebLogic node manager and the administration server must be started. The administration server can be accessed using the following URL:

```
http://server:7001/console
```

Create a WebLogic machine

Note: If you previously created a WebLogic machine definition, you can skip this section.

To create a WebLogic machine, perform the following steps:

1. In the Change Center frame, click **Lock & Edit**.
2. In the Domain Structure frame, click (**+**) to expand and view the list of environments.
3. Click the **Machines** link.
4. Click **New**.
5. Enter a machine name and click **Next**.
6. Accept the defaults and click **Finish**.
7. In the Change Center frame, click **Activate Changes**.

Create a WebLogic server

Note: If you previously created a WebLogic server, you can skip this section.

Note: If you previously created a WebLogic server for the application, you can use the same server.

To create a WebLogic server, perform the following steps:

1. In the Change Center frame, click **Lock & Edit**.
2. In the Domain Structure frame, click (**+**) to expand and view the list of environments.
3. Click the **Servers** link.
4. Click **New**.
5. Enter a server name and server listen port. For example, you can have server name as `Banner9` and server listen port as `8180`.
6. Click **Finish**.
7. Click the newly created server link.
8. Under the **General** tab, assign the machine to this server.
9. Click **Save**.
10. Select the **Server Start** tab.
11. Add the following to the **Arguments** text area:

    If you are using Sun JVM, use the following parameters:

    ```
    -server -Xms2048m -Xmx4g -XX:MaxPermSize=512m
    ```

    Note: If you are deploying multiple Banner 9.x applications to the same WebLogic server, increase the max heap (`-Xmx`) by `2g` and `-XX:MaxPermSize` by `128m`. You should deploy Banner 9.x administrative applications to one WebLogic server instance and Banner 9.x self-service applications to a separate WebLogic server instance.

To override the configuration that was added into the WAR file, you can set system properties to point to external configuration files. Append the following to the arguments text area:

```
-DBANNER_APP_CONFIG=<full file path to
banner_configuration.groovy>
-DAPPLICATION_NAVIGATOR_CONFIG=<full file path to
applicationNavigator_configuration.groovy>
```

Note: When externalizing the configuration files, there are certain filters that get injected into the web.xml file as part of the Single Sign-On and X-Frame HTTP Header settings for the WAR file. These values cannot be changed at runtime by referencing external configuration files. The WAR file will have to be regenerated in order for the new Single Sign-On and X-Frame HTTP Header settings to take effect. Any other configuration outside these two sets of filters can be overridden in the external configuration file at runtime.

12. Click **Save**.
13. In the Change Center frame, click **Activate Changes**.
14. In the Domain Structure frame, click the **Servers** link.
15. Select the **Control** tab.
16. Select the check box next to your new server definition.
17. Click **Start**.

Update Oracle JDBC JAR files on the WebLogic server

1. Copy the Oracle JAR files (xdb6.jar) from the `$PRODUCT_HOME/ current/lib` directory to the `$MIDDLEWARE_HOME/modules` directory.
   - $PRODUCT_HOME is where the Application Navigator release zip file is unpacked and installed.
   - $MIDDLEWARE_HOME is the location where Oracle WebLogic is installed.
2. For Linux/Unix servers, edit the setDomainEnv.sh file under the `$MIDDLEWARE_HOME/user_projects/domains/<CUSTOM_DOMAIN>/ bin` folder and add these two lines after the ADD EXTENSIONS comment as shown by the example below:

```
#ADD EXTENSIONS TO CLASSPATH

export MIDDLEWARE_HOME="/u01/app/oracle/Middleware"

export WLS_MODULES="${MIDDLEWARE_HOME}/modules"

export EXT_PRE_CLASSPATH="${WLS_MODULES}/xdb6.jar"
```

Note: If you plan to "copy and paste" the configuration settings into the "setDomainEnv.sh" file, make sure there is no typo or special characters that get carried over (especially with double quotes on the variable declarations). If you see "Class NotFoundException" in your logs, chances are there was a typo when you edited the "setDomainEnv.sh" file and the "xdb6.jar" file cannot be found during Application startup.

3. For MS Windows servers, edit the setDomainEnv.cmd under the `$MIDDLEWARE_HOME/user_projects/domains/<CUSTOM_DOMAIN>/bin` folder and add these two lines after the ADD EXTENSIONS comment as shown by the example below:

```
@REM ADD EXTENSIONS TO CLASSPATH

set MIDDLEWARE_HOME="D:\Oracle\Middleware"

set WLS_MODULES="%MIDDLEWARE_HOME%\modules" set

EXT_PRE_CLASSPATH="%WLS_MODULES%\xdb6.jar"
```

Note: If you plan to "copy and paste" the configuration settings into the "setDomainEnv.cmd" file, make sure there is no typo or special characters that get carried over (especially with double quotes on the variable declarations). If you see "Class NotFoundException" in your logs, chances are there was a typo when you edited the "setDomainEnv.cmd" file and the "xdb6.jar" file cannot be found during Application startup.

4. Restart the WebLogic Managed Server.

Create an administrative datasource and connection pool

Note: If you previously created an administrative datasource, you can skip this section.

To create an administrative datasource and connection pool, perform the following steps:

1. In the Change Center frame, click **Lock & Edit**.
2. In the Domain Structure frame, click (**+**) to expand Services and then select **Data Sources**.
3. Click **New**.
4. Select **Generic DataSource**.
5. Specify a name (for example, `Banner9DS`).
6. Specify the JNDI name (for example, `jdbc/bannerDataSource`).
7. Specify `Oracle` for Database Type and then click **Next**.
8. Select **Oracle Driver (Thin) for Service Connections** and then click **Next**.
9. Clear the **Supports Global Transactions** check box and then click **Next**.
10. Enter the database name, host name, port, user name, password, and password confirmation, and then click **Next.** For example:

Database name:	`BAN9`
Host name:	`yourhostname.yourdomain.com`
Port:	`1521`
UserName:	`banproxy`
Password:	`your_password`

11. Click **Test Configuration**.
12. Click **Next** for the connection test to be successful.
13. Select the server that you previously created to allow the datasource to be deployed and used by this server.
14. Click **Finish**.
15. Select the datasource link that you created.
16. Select the **Connection Pool** tab.
    - **16.1.** Set the Initial Capacity parameter to specify the minimum number of database connections to create when the server starts up. For example:

      ```
      Initial Capacity = 5
      ```
    - **16.2.** Set the Maximum Capacity parameter to specify the maximum number of database connections that can be created. For example:

      ```
      Maximum Capacity = 100
      ```
17. Change `Statement Cache Type = Fixed`.
18. Change `Statement Cache Size = 0`.
19. Click **Save**.
20. In the Change Center frame, click **Activate Changes**.

Create a self-service datasource and connection pool

Note: If you previously created a self-service datasource, you can skip this section.

To create a self-service datasource and connection pool, perform the following steps:

1. In the Change Center frame, click Lock & Edit.
2. In the Domain Structure frame, click (+) to expand Services and then select Data Sources.
3. Click New.
4. Select Generic DataSource.

5. Specify a name (for example, Banner9SsbDS).
6. Specify the JNDI name (for example, jdbc/bannerSsbDataSource).
7. Specify Oracle for Database Type and then click Next.
8. Select Oracle Driver (Thin) for Service Connections and then click Next.
9. On the Transaction Options page, clear the Supports Global Transactions check box and then click Next.
10. Enter the database name, host name, port, user name, password, and password confirmation, and then click Next.
11. Click Test Configuration.
12. Click Next for the connection test to be successful.
13. Select the server that you previously created to allow the datasource to be deployed and used by this server.
14. Click Finish.
15. Select the datasource link that you created.
16. Select the Connection Pool tab.
    - **16.1.** Set the Initial Capacity parameter to specify the minimum number of database connections to create when the server starts up. For example:
      Initial Capacity = 5
    - **16.2.** Set the Maximum Capacity parameter to specify the maximum number of database connections that can be created. For example:
      Maximum Capacity = 100
17. Change Statement Cache Type = LRU.
18. Change Statement Cache Size = 20.
19. Click Save.
20. In the Change Center frame, click Activate Changes.

Deploy and start the application in the WebLogic server

To deploy and start the web application in the WebLogic server, perform the following steps:

1. Change the name of the WAR file to remove the version number. For example, change:

   ```
   applicationNavigator/current/dist/applicationNavigator-
   3.0.war
   ```

   to

   ```
   applicationNavigator/current/dist/
   applicationNavigator.war
   ```

2. Access the administration server at the following URL:

   ```
   http://server:7001/console
   ```

3. In the Domain Structure frame, select the **Deployments** link.
4. In the Change Center frame, select **Lock and Edit**.
5. Click **Install**.
6. Select the WAR file to be deployed and then click **Next**. The file is located in the following directory:

   ```
   applicationNavigator/current/dist
   ```
7. Select **Install this deployment** as an application and then click **Next**.
8. Select the target server on which to deploy this application (for example, `Banner9`) and then click **Next**.
9. Click **Finish**.
10. In the Change Center frame, click **Activate Changes**.
11. Select the deployed application and then click **Start**.
12. Select **Servicing all request**.
13. Access the application using the following URL format:

    ```
    http://<servername>:<port>/<web application>
    ```

    For example:

    ```
    http://localhost:8180/applicationNavigator
    ```
14. Log in to the application using a valid username and password.

Add service provider in the Ellucian Ethos Identity Server

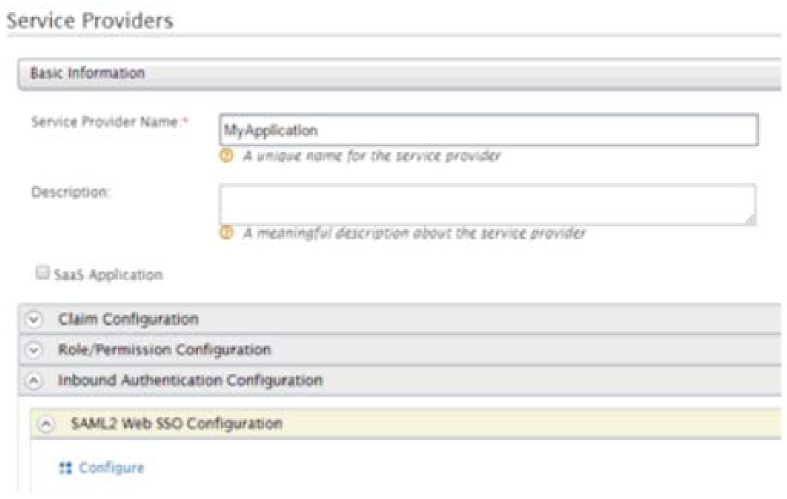
To add a service provider, complete the following steps:

1. Log in to the Management Console and click the **Main** tab.
2. Under Service Providers, click **Add**. The **Add Service Provider** screen appears.
3. Enter a service provider name in the **Service Provider Name** field. You can also add an optional description in the **Description** field.
4. Click **Register** to add the service provider. The **Service Providers** screen appears.

Add SAML settings

To add SAML setting for the integrating application, complete these steps:

1. On the Service Providers screen, expand the Inbound Authentication Configuration panel, then expand the SAML2 Web SSO Configuration panel, as shown in the following example:


Service Providers

Basic Information

Service Provider Name:* MyApplication

A unique name for the service provider

Description:

A meaningful description about the service provider

SaaS Application

Claim Configuration

Role/Permission Configuration

Inbound Authentication Configuration

SAML2 Web SSO Configuration

Configure

2. Click the **Configure** link.
3. Enter the Issuer value that is configured in the service provider application. This value is validated against the SAML Authentication Request issued by the service provider.

   Note: Make sure to provide the same value that is configured as the "alias" in the configuration property `'grails.plugin.springsecurity.saml.metadata.sp.defaults'` in the `applicationNavigator_configuration.groovy` file

4. Enter a valid Assertion Consumer URL where the browser redirects the SAML Response after authentication.
5. Enter a valid NameID format supported by Ellucjan Ethos Identity. The following values can be used.
   - `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
   - `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`
   - `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`
   - `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`
   - `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`
   - `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName`
   - `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`
   - `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`
6. Select **Use fully qualified username** in the NameID if the user store domain and user ID must be preserved in the SAML 2.0 Response. In most cases, this can be left unselected.

7. Select **Enable Response Signing** to sign the SAML 2.0 Responses returned after the authentication process.
8. Select **Enable Assertion Signing** to sign the SAML 2.0 Assertions returned after the authentication.
9. Select **Enable Signature Validation** in Authentication Requests and Logout Requests if the identity provider must validate the signature of the SAML 2.0 Authentication and Logout Requests that are sent by the service provider.
10. Select **Assertion Encryption** to encrypt the assertions.
11. Select **Certificate Alias** for the service provider's public certificate.

    This certificate is used to validate the signature of SAML 2.0 Requests and is used to generate encryption.
12. Select **Enable Single Logout** so that all sessions across all authenticated service providers are terminated once the user signs out from one server.

    If the service provider supports a different URL than the Assertion Consumer URL for logout, enter a Custom Logout URL for logging out. This should match the URL set up in the `.xml` file property "SingleLogoutService" set in."Add an IDP Certificate entry in the newly created keystore file" on page 77.
13. Select **Enable Attribute Profile** to add a basic attribute profile where the identity provider can include the user's attributes in the SAML Assertions as part of the attribute statement.
14. Select **Include Attributes** in the Response Always if the identity provider should always include the attribute values related to the selected claims in the SAML attribute statement.

    This is required so that UDC_IDENTIFIER configured in claims are sent across.
15. Select **Enable Audience Restriction** to restrict the audience. Add audience members using the Audience text box and click Add Audience.
16. Select **Enable IdP Initiated SSO** and the service provider is not required to send the SAML 2.0 Request.
17. Click **Register** to save settings and return to the Service Provider Configuration page.
18. Click **Update** to save all settings.

Modify Identity Provider Issuer

This step provides instructions on how to add a resident identity provider as per the `idp-local.xml` configuration.

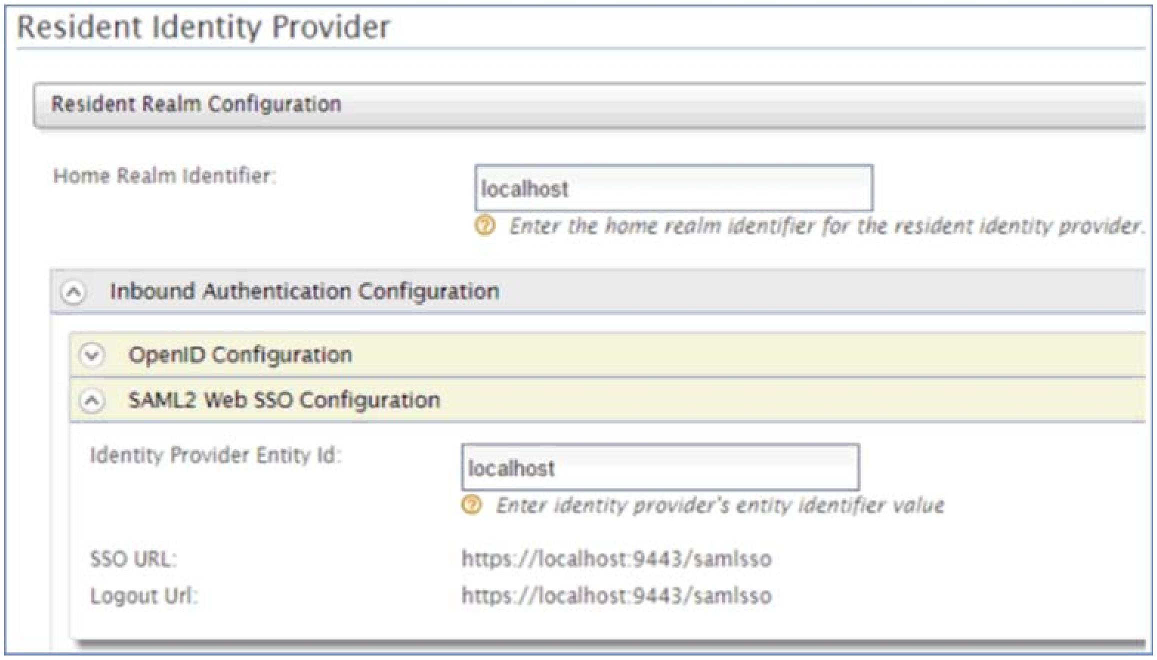
Ellucian Ethois Identity can mediate authentication requests between service providers and identity providers. At the same time, the identity server can act as a service provider and an identity provider. When acting as an identity provider, it is known as the resident identity provider. This converts the identity server into a federated hub.

The resident identity provider configuration is relevant for you if you are a service provider and want to send an authentication request or a provisioning request to the identity server (for example, via SAML, OpenID, OpenID Connect, SCIM, and WS-Trust).

Resident identity provider configuration is a one-time configuration for a given tenant. It shows you the identity server's metadata, like the endpoints. In addition, you can secure the WS-Trust endpoint with a security policy.

You must change the Identity Provider Entity Id to the expected URL of the Issuer statement in SAML 2.0 Responses. Complete the following steps to change the ID.

1. Log in to the Management Console and click the **Main** tab.
2. In the Main menu under the Identity section, click **List** under Identity Providers.
3. Enter a service provider name in the **Service Provider Name** field. You can also add an optional description in the **Description** field.
4. Click **List** under Identity Providers.
5. Click **Resident Identity Provider**.
6. Expand the Inbound Authentication Configuration panel, then expand the SAML2 Web SSO Configuration panel, as shown in the following example:


Resident Identity Provider
Resident Realm Configuration
Inbound Authentication Configuration
OpenID Configuration
SAML2 Web SSO Configuration

Home Realm Identifier: localhost
Enter the home realm identifier for the resident identity provider.

Identity Provider Entity Id: localhost
Enter identity provider's entity identifier value

SSO URL: https://localhost:9443/samlsso
Logout Url: https://localhost:9443/samlsso

7. Enter a valid identity provider name or URL to be used by all service providers.
8. Click **Update** to save the settings.

SAML 2.0 Configuration Sub-tasks

Subtasks referred to in the Install Application Navigator for SAML 2.0 SSO chapter are discussed in this section.

Create a keystore (*.jks) file

During SAML configuration, you must create a keystore file (`*.jks`). Follow these steps to create the file:

1. Open your operating system's command console and navigate to the directory where `keytool.exe` is located. This is usually where the JRE is located (for example, `c:\Program Files\Java\jre7\bin` on Windows machines).
2. Run the command below (where validity is the number of days before the certificate expires).
    - **2.1.** Fill in the prompts for your organization information.
    - **2.2.** When asked for your first and last name, enter the domain name of the server that users will be entering to connect to your application.

```
C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -genkey -keyalg RSA -alias mykey
-keystore appnavkeystore.jks -storepass password -validity 360 -keysize 2048
What is your first and last name?
  [Unknown]:  John Doe
What is the name of your organizational unit?
  [Unknown]:  Ellucian
What is the name of your organization?
  [Unknown]:  Ellucian
What is the name of your City or Locality?
  [Unknown]:  Malvern
What is the name of your State or Province?
  [Unknown]:  PA
What is the two-letter country code for this unit?
  [Unknown]:  US
Is CN=John Doe, OU=Ellucian, O=Ellucian, L=Malvern, ST=PA, C=US correct?
  [no]:  yes
Enter key password for <mykey>
(RETURN if same as keystore password):changeit
C:\Program Files\Java\jdk1.7.0_67\jre\bin>
```

This creates an `appnavkeystore.jks` file containing a private key and your self-signed certificate.

Extract a X509 Certificate Key

During SAML configuration, you must extract a X509 certificate key from the keystore for the `banner-<Application_Name>-sp.xml` file. Follow these steps to extract one.

1. With the `appnavkeystore.jks` file you created, execute the command below to check which certificates are in a Java keystore:

   See "Create a keystore (*.jks) file" on page 106 for more information.

```
C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -list -v -keystore
appnavkeystore.jks
Enter keystore password:
Keystore type: JKS
Keystore provider: SUN
Your keystore contains 1 entry
Alias name: mykey
Creation date: Apr 6, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=John Doe, OU=Ellucian, O=Ellucian, L=Malvern, ST=PA, C=US
Issuer: CN=John Doe, OU=Ellucian, O=Ellucian, L=Malvern, ST=PA, C=US
Serial number: 78548b7
Valid from: Mon Apr 06 12:52:39 IST 2015 until: Thu Mar 31 12:52:39 IST 2016
Certificate fingerprints:
MD5:  5D:55:F4:18:3D:CF:AE:5A:27:B8:85:68:42:47:CA:76
SHA1: CD:09:04:F5:01:60:14:CC:DF:48:07:4A:93:99:17:BF:10:83:F3:55
SHA256:
79:5A:7F:0C:A4:B1:0E:30:9C:B0:DD:87:2C:CA:19:A1:0E:89:29:2F:95:A1:35:E9:EC:A2:AA:B
9:F6:2D:BE:35
Signature algorithm name: SHA256withRSA
Version: 3
Extensions:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: B2 AC D7 09 01 15 22 A3   32 08 86 64 E8 25 5A 15  ......".2..d.%Z.
0010: CB A0 C6 D9                                        ....
]
]
*******************************************
*******************************************
```

   This command shows all the available keystores in the `.jks` file.

   Note: To get information about the specific certificate, execute this command:

   ```
   keytool -list -v -keystore appnavkeystore.jks -
   alias mykey
   ```

2. Execute the following command to export a certificate from a keystore:

   ```
   C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -export
   -alias mykey -file mykey.crt -keystore appnavkeystore.jks
   Enter keystore password: password
   ```

```
Certificate stored in file <mykey.crt>
```

3. Execute the following command to get the X509 certificate:

```
C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -printcert -rfc -file mykey.crt
```

```
-----BEGIN CERTIFICATE-----
MIIDfTCCAmWgAwIBAgIEB4VItzANBgkqhkiG9w0BAQsFADBvMQswCQYDVQQGEwJJTjELMAkGA1UE
CBMCSU4xEjAQBgNVBAcTCUJhbmdhbG9yZTERMA8GA1UEChMIRWxsdWNpYW4xETAPBgNVBAsTCEVs
bHVjaWFuMRkwFwYDVQQDExBTcGhvb3J0aSBBY2hhcnlhMB4XDTE1MDQwNjA3MjIzOVoXDTE2MDMz
MTA3MjIzOVowbzELMAkGA1UEBhMCSU4xCzAJBgNVBAgTAklOMRIwEAYDVQQHEwlCYW5nYWxvcmUx
ETAPBgNVBAoTCEVsbHVjaWFuMREwDwYDVQQLEwhFbGx1Y2lhbjEZMBcGA1UEAxMQU3Bob29ydGkg
QWNoYXJ5YTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAN2cS+2OX37HioFzwOWLm/S0
F+zt6ldtLfmHc16V9iqkZChkMiXpKgXVPqGLFjBrhwfsWtuMfRy2NYf3forEDFTaV4/fLXRo+Npd
xTfqWhuZTafDJEyKQc57KY8G3feg1CSjfKkCk1LF+zbGClHQ0bg1dwUjJlp7eKjWM0rbsKMd5pZ7
0tGAPcYsi6MtGvJupaVhy3jNTDg+kh4/D92y/mTaLlCR4QQr1qlU9+H+it3m9jiDrZ7svrdBlsDN
1BVcXDooqUGTuc10IBxYEsb7hFucSFpdJnGJvbg35ll9K9F5S8lEmiQmeOUQ1gQe2Ow01kFl56Qz
4evM0xgeskNid9sCAwEAAaMhMB8wHQYDVR0OBBYEFLKs1wkBFSKjMgiGZOglWhXLoMbZMA0GCSqG
SIb3DQEBCwUAA4IBAQANJbYRTcMwhrETz+mo+n1okrXIs118AIm7s1yJJd1nyJuaKrn7DcPPLzy/
RjHGKP02uLiupgqar+UUaPqSZjJXSzktLLyq7H6DRrW0Jp2rw48a+Kou+XOvQ8ZWR9ZXIa1XoAoD
PaSSE2omcVOVGZmQKUYardVeSvQth3IVMW9w9Jl+DuavXavVjIx5IN6RRhXGfaJjQLKFzIDqZNAp
OcxMEKXHOUqj0ksTRARLpKWSPu7gFOWO/6qapNp5l8rlPjnVxDhqHqCKC3E40VI5n+C+KJHZQqab
Tfhd6erqEy7S1Cazr655Yq22Jm6L7IXsXgpRwmZnoietLsrFIRyPe1DY
-----END CERTIFICATE-----
```

Extract X509 certificate data

During SAML configuration, you must extract X509 certificate data for banner-<Application_Name>-sp.xml. Follow these steps to extract data:

1. Go to the deployed WSO2 server / Ellucian Ethos Identity server. For example, `C:\>cd C:\work\eis\`.
2. Navigate to the server certificate location. By default, it is located at: `$EIS_HOME\repository\resources\security`
3. Execute the following command, where `saml-idp.cer` is the certificate file in the server:

   ```
   C:\work\eis\repository\resources\security>keytool -
   printcert -rfc -file saml-idp.cer
   ```

   This would return a value similar to the following:

```
-----BEGIN CERTIFICATE-----
MIICdDCCAd2gAwIBAgIEEsIIcDANBgkqhkiG9w0BAQsFADBtMRAwDgYDVQQGEwdVbmtub3duMRAw
DgYDVQQIEwdVbmtub3duMRAwDgYDVQQHEwdVbmtub3duMRAwDgYDVQQKEwdVbmtub3duMRAwDgYD
VQQLEwdVbmtub3duMREwDwYDVQQDEwhXU08yIElEUDAeFw0xNDA4MTgxMzA3NTFaFw0xNTA4MTgx
MzA3NTFaMG0xEDAOBgNVBAYTB1Vua25vd24xEDAOBgNVBAgTB1Vua25vd24xEDAOBgNVBAcTB1Vu
a25vd24xEDAOBgNVBAoTB1Vua25vd24xEDAOBgNVBAsTB1Vua25vd24xETAPBgNVBAMTCFdTTzIg
SURQMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC44MMcoAPb+aR8l5gtTsSb+SzslHFESmKL
wO1+SAY6p2iiO+G9qR/511ufCzKWrMdMMpoCJLC0myDwoUuGvk0dycTpm5NwUX6CnqDtYhtGkYg8
JT+LtG67k6yjXNa9wrE6VBJynDDPnlL8gLUl9ZCIFrmevJ75rOCaLsoFsghHPwIDAQABoyEwHzAd
BgNVHQ4EFgQULyCNnvW9Ngy5zM7Waf205mR11ZAwDQYJKoZIhvcNAQELBQADgYEApWlSy2GUSaHM
```

```
Kkc8XZmdQ0//SId8DKRKaFaZW388K4dJGTSWUnzq4iCWFrAN9O4D1DBnNE+dCDEmV8HvmyQBedsG
JnAre0VisqKz9CjIELcGUaEABKwkOgLe1YyqV29vS4Y3PuTxAhbkyphFb5PxjHDHH/WLQ8pOTbsC
vX4wO04=
-----END CERTIFICATE-----
```

3.1. If no certificate file is found, navigate to the `.jks` file. The `.jks` file can be found via `carbon.xml`. By default, it is located at:

`$EIS_HOME\repository\conf\carbon.xml`

3.2. Locate the KeyStore file location in the `carbon.xml` file, as shown in the following example:

```
<KeyStore>
    <!-- Keystore file location-->
    <Location>${carbon.home}/repository/resources/security/cacerts</Location>
    <!-- Keystore type (JKS/PKCS12 etc.)-->
    <Type>JKS</Type>
    <!-- Keystore password-->
    <Password>changeit</Password>
    <!-- Private Key alias-->
    <KeyAlias>mykey</KeyAlias>
    <!-- Private Key password-->
    <KeyPassword>changeit</KeyPassword>
</KeyStore>
```

3.3. Create the `saml-idp.cer` file by executing the following command:

```
keytool -export -keystore cacerts -alias mykey -file
~/saml-idp.cer
```

4. Go to the keystore location and execute the following command.

Note: The `password` and `alias` referenced in this example are also contained in the `carbon.xml` file accessed earlier in this task.

```
C:\work\eis\repository\resources\security>keytool -export -keystore cacerts -rfc -
alias mykey
Enter keystore password:
-----BEGIN CERTIFICATE-----
MIICdDCCAd2gAwIBAgIEEsIIcDANBgkqhkiG9w0BAQsFADBtMRAwDgYDVQQGEwdVbmtub3duMRAw
DgYDVQQIEwdVbmtub3duMRAwDgYDVQQHEwdVbmtub3duMRAwDgYDVQQKEwdVbmtub3duMRAwDgYD
VQQLEwdVbmtub3duMREwDwYDVQQDEwhXU08yIElEUDAeFw0xNDA4MTgxMzA3NTFaFw0xNTA4MTgx
MzA3NTFaMG0xEDAOBgNVBAYTB1Vua25vd24xEDAOBgNVBAgTB1Vua25vd24xEDAOBgNVBAcTB1Vu
a25vd24xEDAOBgNVBAoTB1Vua25vd24xEDAOBgNVBAsTB1Vua25vd24xETAPBgNVBAMTCFdTTzIg
SURQMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC44MMcoAPb+aR8l5gtTsSb+SzslHFESmKL
wO1+SAY6p2iiO+G9qR/511ufCzKWrMdMMpoCJLC0myDwoUuGvk0dycTpm5NwUX6CnqDtYhtGkYg8
JT+LtG67k6yjXNa9wrE6VBJynDDPnlL8gLUl9ZCIFrmevJ75rOCaLsoFsghHPwIDAQABoyEwHzAd
BgNVHQ4EFgQULyCNnvW9Ngy5zM7Waf205mR11ZAwDQYJKoZIhvcNAQELBQADgYEApWlSy2GUSaHM
Kkc8XZmdQ0//SId8DKRKaFaZW388K4dJGTSWUnzq4iCWFrAN9O4D1DBnNE+dCDEmV8HvmyQBedsG
JnAre0VisqKz9CjIELcGUaEABKwkOgLe1YyqV29vS4Y3PuTxAhbkyphFb5PxjHDHH/WLQ8pOTbsC
vX4wO04=
-----END CERTIFICATE-----
```

Add IDP certificate entry to the .jks file

During SAML configuration, you must add the IDP certificate entry to the new `.jks` file you created as part of this sub-task "Create a keystore (*.jks) file" on page 106. Follow these steps to add the IDP certificate entry to the `.jks` file you created for that sub-task:

1. Navigate to where the server certificate exists. By default, it is located at:

   `$EIS_HOME\repository\resources\security`

2. Extract X509 certificate Data for `Idp.xml`.

3. Copy `saml-idp.cer` to the `.jks` file by executing the following command:

```
C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -import -trustcacerts -alias
mykey -file saml-idp.cer -keystore appnavkeystore.jks
Enter keystore password: password
Owner: CN=WSO2 IDP, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Issuer: CN=WSO2 IDP, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Serial number: 12c20870
Valid from: Mon Aug 18 18:37:51 IST 2014 until: Tue Aug 18 18:37:51 IST 2015
Certificate fingerprints:
MD5:  8B:65:A6:A0:0F:F3:EA:B6:2A:32:37:7A:21:B5:CF:B6
SHA1: C5:73:6C:FD:63:15:45:C4:74:CF:E2:9D:DE:18:9A:4B:F9:6C:9C:5C
SHA256: 33:47:F1:95:1E:E7:DD:8B:F9:5C:17:A1:88:82:3E:0D:8B:B9:5C:9E:22:10:0B:57:
8F:51:62:9E:FF:1B:38
Signature algorithm name: SHA256withRSA
Version: 3
Extensions:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 2F 20 8D 9E F5 BD 36 0C   B9 CC CE D6 69 FD B4 E6  / ....6.....i...
0010: 64 75 D5 90 du..
]
]
Trust this certificate? [no]:  yes
Certificate was added to keystore
```

4. To verify the certificate was added, execute the following commands:

```
C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -list -v -keystore
appnavkeystore.jks Enter keystore password: password
Keystore type: JKS
Keystore provider: SUN
Your keystore contains 2 entries
Alias name: mykey
Creation date: Apr 6, 2015
Entry type: trustedCertEntry
Owner: CN=WSO2 IDP, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Issuer: CN=WSO2 IDP, OU=Unknown, O=Unknown, L=Unknown, ST=Unknown, C=Unknown
Serial number: 12c20870
Valid from: Mon Aug 18 18:37:51 IST 2014 until: Tue Aug 18 18:37:51 IST 2015
Certificate fingerprints:
```

```
MD5:  8B:65:A6:A0:0F:F3:EA:B6:2A:32:37:7A:21:B5:CF:B6
SHA1: C5:73:6C:FD:63:15:45:C4:74:CF:E2:9D:DE:18:9A:4B:F9:6C:9C:5C
SHA256: 33:47:F1:95:1E:E7:DD:8B:F9:5C:17:A1:88:82:3E:0D:8B:B9:5C:9E:22:10:0B:57:
8F:51:62:9E:FF:1B:38
Signature algorithm name: SHA256withRSA
Version: 3
Extensions:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 2F 20 8D 9E F5 BD 36 0C   B9 CC CE D6 69 FD B4 E6  / ....6.....i...
0010: 64 75 D5 90 du..
]
]
*******************************************
*******************************************


Alias name: mykey
Creation date: Apr 6, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=John Doe, OU=Ellucian, O=Ellucian, L=Malvern, ST=PA, C=US
Issuer: CN=John Doe, OU=Ellucian, O=Ellucian, L=Malvern, ST=PA, C=US
Serial number: 126891cb
Valid from: Mon Apr 06 16:06:54 IST 2015 until: Thu Mar 31 16:06:54 IST 2016
Certificate fingerprints:
MD5:  D7:A6:90:A0:7D:19:DF:7E:D9:FF:01:5B:18:1D:FE:71
SHA1: 46:24:3E:A0:1E:65:76:21:D2:93:0F:29:76:60:17:40:07:0C:72:58
SHA256: ED:1B:C7:B5:07:49:80:4A:91:93:87:A1:15:9A:20:23:A7:BB:8B:99:89:02:47:
5F:5C:6E:42:47:AA:68:55
Signature algorithm name: SHA256withRSA
Version: 3
Extensions:
#1: ObjectId: 2.5.29.14 Criticality=false
SubjectKeyIdentifier [
KeyIdentifier [
0000: 6F 8C FC 5C BA F1 11 FF   E2 58 C8 4F 2F 60 DA 2B  o..\.....X.O/`.+
0010: A2 EA D8 78                                        ...x
]
]
*******************************************
*******************************************
```

Appendix - CAS Installation Checklist

The following checklist is provided to help you ensure that you have completed all required steps to install Application Navigator in CAS authentication mode.

Description	Reference
Implement CAS SSO	Refer to the *CAS Single Sign On Handbook*
Upgrade the Database	"Upgrade the Database" on page 19
• Update `login.sql`	"Update login.sql" on page 19
• Verify that the required products are applied	"Verify that the required products are applied" on page 19
• Verify the banproxy database account	"Verify the banproxy database account" on page 20
• Migrate staged files to the permanent directories	"Migrate staged files to the permanent directories" on page 20
• Update the version number	"Update the version number" on page 22
Enable seamless navigation for Banner 8.x Forms	"Enable Seamless Navigation for Banner 8.x Forms" on page 23
• Run the Seamless Navigation Enhancement Utility	"Run the Seamless Navigation Enhancement Utility" on page 23
• Update the HTML or JavaScript files - Change `base.htm` - Change `js.html` - Change `forms_base_ie.js` - Change `banner8_integration.js`	"Update the Oracle WebUtil HTML files" on page 23
• Configure the Forms Server - Update Forms Web Configuration - Configure JVM Pooling	"Configure the Forms Server" on page 25
• Configure INB Accelerator Keys for unified menu shortcuts	"Configure INB Accelerator Keys for unified menu shortcuts" on page 29
• Update GUAUPRF settings for the unified menu	"Update GUAUPRF settings for the unified menu" on page 30
Enable seamless navigation for Banner 9.x administrative applications	"Enable Seamless Navigation for Banner 9.x Administrative Applications" on page 34
• Configure the Banner 9.x applications	"Update GUAPAGE controls for each Banner 9.x module" on page 34

Description	Reference
• Update GUAPAGE controls for each Banner 9.x module	“Update GUAPAGE controls for each Banner 9.x module” on page 34
• Rebuild the Banner menus	“Rebuild the Banner Menus” on page 35
Enable Seamless Navigation for Banner 9.x Self-Service Applications	“Enable Seamless Navigation for Banner 9.x Self-Service Applications” on page 37
• Configure Banner 9.x Applications	“Configure Banner 9.x Applications” on page 37
Install Application Navigator for CAS	“Install Application Navigator for CAS SSO” on page 39
• Undeploy the existing application	“Undeploy the existing application” on page 39
• Customize the WAR file	“Customize the WAR file” on page 41
• Regenerate the WAR file	“Regenerate the WAR file” on page 54

Appendix - SAML 2.0 Installation Checklist

The following checklist is provided to help you ensure that you have completed all required steps to install Application Navigator in SAML 2.0 authentication mode.

Description	Reference
Upgrade the Database	"Upgrade the Database" on page 19
• Update `login.sql`	"Update login.sql" on page 19
• Verify that the required products are applied	"Verify that the required products are applied" on page 19
• Verify the banproxy database account	"Verify the banproxy database account" on page 20
• Migrate staged files to the permanent directories	"Migrate staged files to the permanent directories" on page 20
• Update the version number	"Update the version number" on page 22
Enable seamless navigation for Banner 8.x Forms	"Enable Seamless Navigation for Banner 8.x Forms" on page 23
• Run the Seamless Navigation Enhancement Utility	"Run the Seamless Navigation Enhancement Utility" on page 23
• Update the HTML or JavaScript files - Change `base.htm` - Change `js.html` - Change `forms_base_ie.js` - Change `banner8_integration.js`	"Update the Oracle WebUtil HTML files" on page 23
• Configure the Forms Server - Update Forms Web Configuration - Configure JVM Pooling	"Configure the Forms Server" on page 25
• Configure INB Accelerator Keys for unified menu shortcuts	"Configure INB Accelerator Keys for unified menu shortcuts" on page 29
• Update GUAUPRF settings for the unified menu	"Update GUAUPRF settings for the unified menu" on page 30
Enable seamless navigation for Banner 9.x administrative applications	"Enable Seamless Navigation for Banner 9.x Administrative Applications" on page 34
• Configure the Banner 9.x applications	"Update GUAPAGE controls for each Banner 9.x module" on page 34

Description	Reference
• Update GUAPAGE controls for each Banner 9.x module	“Update GUAPAGE controls for each Banner 9.x module” on page 34
• Rebuild the Banner menus	“Rebuild the Banner Menus” on page 35
Enable Seamless Navigation for Banner 9.x Self-Service Applications	“Enable Seamless Navigation for Banner 9.x Self-Service Applications” on page 37
• Configure Banner 9.x Applications	“Configure Banner 9.x Applications” on page 37
Install Application Navigator for SAML 2.0	“Install Application Navigator for SAML 2.0 SSO” on page 69
• Undeploy the existing application	“Undeploy the existing application” on page 69
• Customize the WAR file	“Customize the WAR file” on page 71
• Create a service provider file	“Add an IDP Certificate entry in the newly created keystore file” on page 77
• Create an identity service provider file	“Create an identity service provider file” on page 76
• Regenerate the WAR file	“Regenerate the WAR file” on page 89
• Create a keystore (`*.jks`) file	“Create a keystore (*.jks) file” on page 106
• Extract a X509 Certificate Key	“Extract a X509 Certificate Key” on page 106
• Extract X509 certificate data	“Extract X509 certificate data” on page 108
• Add the IDP Certificate entry to newly created keystore (`*jks`) file	“Add IDP certificate entry to the .jks file” on page 110