ellucian

Banner General
Event Management Self-Service
Installation Guide

Release 9.4
May 2016

Without limitation: Ellucian®, Banner®, Colleague®, and Luminis® are trademarks of the Ellucian group of companies that are registered in the U.S. and certain other countries; and Ellucian Advance™, Ellucian Course Signals™, Ellucian Degree Works™, Ellucian PowerCampus™, Ellucian Recruiter™, Ellucian SmartCall™, are also trademarks of the Ellucian group of companies. Other names may be trademarks of their respective owners.

© 2013, 2016 Ellucian.

Contains confidential and proprietary information of Ellucian and its subsidiaries. Use of these materials is limited to Ellucian licensees, and is subject to the terms and conditions of one or more written license agreements between Ellucian and the licensee in question.

In preparing and providing this publication, Ellucian is not rendering legal, accounting, or other similar professional services. Ellucian makes no claims that an institution's use of this publication or the software for which it is provided will guarantee compliance with applicable federal or state laws, rules, or regulations. Each organization should seek legal, accounting, and other similar professional services from competent providers of the organization's own choosing.

Ellucian
4375 Fair Lakes Court
Fairfax, VA 22033
United States of America

Revision History

Publication Date	Summary
May 2016	New version that supports Banner General Event Management 9.4 software.

Contents

Introduction. 7

Known installation issues 7

Hardware and software requirements 7

Hardware requirements 7
- CPU and memory 7
- Screen resolution 7
- Tablets 8
- Mobiles 8

Software requirements 8
- Oracle Database 8
- Application server 8
- Middle Tier (application server) platforms 8
- Ellucian software 9
- Release tagging. 10
- Single sign on (SSO) support 10

Supported browsers 10
- Chrome Java support on Mac OS X 11
- Chrome Compatibility Mode 11
- Internet Explorer 9 Compatibility View. 11

Java dependencies. 11

Deployment of multiple web applications 12

Navigation among 9.x applications 13

With CAS 13

Without CAS 13

F5 load balancer configuration 14

Upgrade the Database 15

Perform the Banner DB Upgrade steps 15

Update login.sql 15

Verify that the required products are applied 16

Verify the banproxy database account 16

Verify the ban_ss_user database account 16

Verify Oracle user accounts to connect through banproxy 16

Set up access for application users with an administrative account 17

Migrate staged files to the permanent directories 17

Unix 17

Windows 18

Update the version numbers 19

Install Event Management Self-Service Application 20

Undeploy the existing application 20

Tomcat 20

Undeploy using Tomcat Manager web application 20

Undeploy using a manual procedure 21

WebLogic 22

Customize the WAR file 22

Prepare the installer 23

Install into the product home directory 23

Configure shared settings 25

JNDI datasource 26

banner8.SS.url 26

Link to Self-Service Banner 8.x 26

Session timeouts 28

Configure application-specific settings 29

Configure the WebTailor to display Event listing page 29

Enable non-SPRIDEN user access 29

Configuration for Atom Feed absolute links 29

Enable Guest Log In link 30

Configure footerFadeAwayTime 31

Configure Extensibility in Events Self Service application 31

Configure loginEndpoint 32

Configure Forgot pin 32

JMX MBean names 32

Logging 33

Institutional Home Page Redirection Support 34

ssbOracleUsersProxied setting 34

Logout URL 35

Password reset 37
Redirect pages in a MEP environment 38
Self-service end point 38

Regenerate the WAR file 38

Configure and deploy the WAR file to a web application server 39

Tomcat 39
Configure the Tomcat server 39
Configure/enable CAS in the product groovy file 43
Configure Java Management Extensions 43
Deploy the WAR file to the Tomcat server 44

WebLogic 46
WebLogic prerequisites 46
Create a WebLogic machine 46
Create a WebLogic server 47
Update Oracle JDBC JAR files on the WebLogic server 48
Create an administrative datasource and connection pool 49
Create a self-service datasource and connection pool 50
Configure server communication 51
Deploy and start the application in the WebLogic server 52

Configure the application 53

Name format 53

Phone format 53

Date format 53
default.date.format 54
js.datepicker.dateFormat 55

Time format 55

Multiple calendars 56

CSS customization 56

Institution name 57

Custom JavaScript 57

Install Banner General Event Management SSB for SAML 2.0 SSO . . . 58

Generate SAML 2.0 metadata files 58

Create a keystore (*.jks) file 58

Create a service provider file 59

Create an identity service provider file 60

Add an IDP Certificate entry in the newly created keystore file 61

Add keystore, service provider, and identity service provider files to WAR file creation location 61

Set up SAML 2.0 SSO Configuration 62

Authentication Provider Name 62

Logout URL 63

SAML 2.0 SSO Configuration 63

Add service provider in EIS Server 66

Add SAML settings 66

Modify Identity Provider Issuer 68

SAML 2.0 Configuration Sub-tasks 70

Create a keystore (*.jks) file 70

Extract a X509 Certificate Key 70

Extract X509 certificate data 72

Add IDP certificate entry to the .jks file 74

Introduction

This installation guide details the steps required to install the Banner General Event Management Self-Service component of Banner General Event Management 9.4.

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

Known installation issues

Before you install Banner General Event Management 9.4, refer to the following articles on the Ellucian Support Center (http://www.ellucian.com/Solutions/Ellucian-Client-Support/) for any issues that were reported after the release was posted:

Article 000036091 Banner 9.4 General Event Management Upgrade Issues - general-events-90400u.

Hardware and software requirements

This section documents the hardware and software requirements for Banner General Event Management 9.4.

Hardware requirements

The application has the following hardware requirements:

CPU and memory

Recommended:	Quad core CPU with 4 to 8 GB of memory for the application server
Minimum:	Dual core CPU with 2 GB of memory for the application server

Screen resolution

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

Tablets

The following tablets are supported in the application:

- iPad, iPad Mini, iOS 9.x
- Android OS 5.x
- Windows surface 8.1

Mobiles

- Samsung Glaxy S5 (Android 4.x), Iphone 6(9.x)

Software requirements

The application has the following software requirements:

Oracle Database

The application supports Oracle Database 11.1.0.7 and 11.2.0.2:

- Minimum version for 11gR1: 11.1.0.7
- Minimum version for 11gR2: 11.2.0.2

Application server

The application was tested on the following application servers:

- Oracle Fusion Middleware 11gR2 using WebLogic 10.3.3, 10.3.4, 10.3.5, and 10.3, and 12.3.1.
- Apache Tomcat 7 or 8

Middle Tier (application server) platforms

The application supports the following application server and operating system combinations:

Tomcat (64 bit)	WebLogic (64 bit)
Red Hat Linux 6	Red Hat Linux 5.3
Windows Server 2008	Windows Server 2008
Solaris 10	Solaris 10

Tomcat (64 bit)	WebLogic (64 bit)
AIX 6.1 (JDK 1.7 SR10 or later)	AIX 6.1 (JDK 1.6.0 SR10 or later)
HP-UX	HP-UX 11iV3 (11.31)

Note: Banner 9.x applications are 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 10.3.3:
http://docs.oracle.com/cd/E14571_01/web.1111/e13720/using_toplink.htm#i1221315

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#autold2

WebLogic 12.1.3:
http://docs.oracle.com/middleware/1213/wls/EJBAD/using_toplink.htm#EJBAD1288

For WebLogic server 12.1.x environments, JPA 2.1 support must be enabled. WebLogic server does not enable JPA by default. To enable JPA, use the steps in the appropriate Oracle documentation.

Ellucian software

Depending on the products that are licensed at your institution, the following product upgrades must be applied:

- Banner DB Upgrade 9.4
- Banner General 8.8.1 AND 8.8.3
- Banner Student 8.9.2
- Banner Web Tailor 8.8
- Banner Web General 8.7.1

Developer Requirements

Developers who extract the source code from GIT will need to use the following versions of software when making extensions.

- Grails 2.5.0
- JDK 1.7
- Tomcat 7
- Oracle 11.2.0.4 or higher

Release tagging

Use the following tag for application source code in the GIT repository.

Application	Tag
banner_general_events_ssb_app	rel-events-ssb-9.4

Single sign on (SSO) support

Banner 9.x applications natively support SSO. Central Authentication Service (CAS) is the central access manager for SSO.

Seamless navigation between Banner 8.x and Banner 9.x administrative applications can be accomplished by using Application Navigator. The integration between Application Navigator and Banner applications (8.x and 9.x) requires CAS.

Note: SSO for Banner 8.x forms also requires the SSO Manager, a component of Banner Enterprise Identity Services (BEIS).

Refer to the *CAS Single Sign On Handbook*, available on the Ellucian Support Center, for details on configuring Banner for CAS.

Supported browsers

The following browsers are compatible with the application:

- Chrome 49.x
- Firefox 45.x
- Internet Explorer 9 (Windows 7), 10, and 11
- Safari 6 and 7

For more information about supported browsers, refer to the *Interactive Banner Compatibility Guide* on the Ellucian Download Center.

Chrome Java support on Mac OS X

Chrome does not support Java 7 on Mac OS X. Java 7 runs only on 64-bit browsers, and Chrome is a 32-bit browser.

If you download Java 7, you cannot run Java content in Chrome on Mac OS X. You must use a 64-bit browser (such as Safari or Firefox) to run Java content within a browser. Additionally, installing Java 7 disables the ability to use Apple Java 6 on your system.

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 your 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**.

Internet Explorer 9 Compatibility View

When using Internet Explorer 9, users must disable Compatibility View. Users must be in Internet Explorer Standard Mode. If not, users might receive the following message:

You are viewing this webpage in Compatibility View. Please turn off Compatibility View in your browser (Tools menu) for optimal viewing experience.

To disable Compatibility View in Internet Explorer 9, perform the following steps:

1. Select **Tools** > **Compatibility View Settings**.
2. Clear the **Display intranet sites in Compatibility View** and **Display all websites in Compatibility View** check boxes.
3. Click **Close**.

You can also deactivate the **Compatibility View** item in the **Tools** menu.

Java dependencies

The Java 7 JDK - Java 1.7.x (64-bit version) and Java 8 components must be installed on the application server before you install the application:

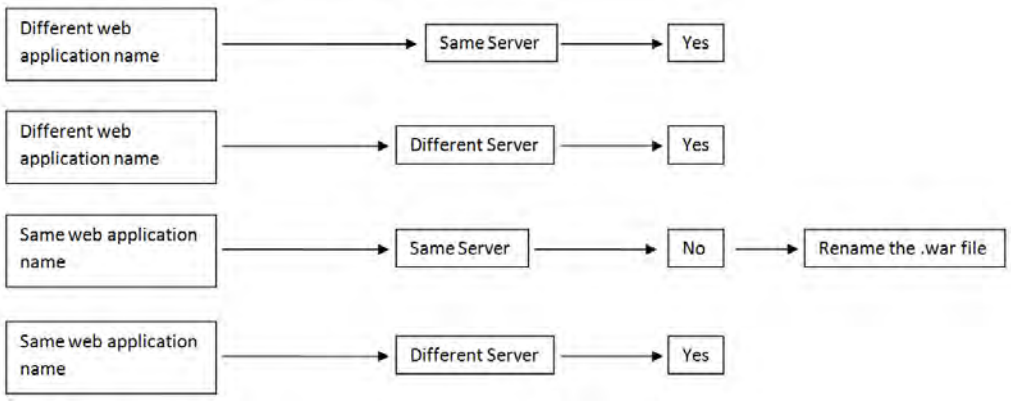
The JDK bin directory must be defined in the PATH system property.

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

Note: Java 1.7.x (64-bit version) must be installed on the application server before you install the application. The same version of Java must be used to customize and deploy the WAR file.
The JDK bin directory must be defined in the PATH system property.?Note: Development for Banner 9.x is currently supported on Java 7 only.
Java 7 includes security restrictions for Rich Internet Applications. Refer to Article 000030656 on the Ellucian Support Center for details on Java 7 security restrictions with Liveconnect calls to Oracle Forms Applet.

Deployment of multiple web applications

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



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

You can navigate among multiple Banner 9.x applications with or without CAS. 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_number>/<application_context_name>/
```

With CAS

If CAS 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. If CAS SSO is enabled when a user logs out of an application, the user is logged out of all applications that are open.

Use the following procedure to navigate among 9.x applications using CAS:

1. Log in to CAS SSO using a valid CAS SSO user name and password.
2. Access any Banner 9.x module (for example, Banner Student Course Catalog).
3. Navigate to another Banner 9.x module (for example, Banner Student Class Schedule). You can navigate among 9.x modules without prompting for a second login.

Refer to the *CAS Single Sign On Handbook*, available on the Ellucian Support Center, for details on configuring Banner for CAS.

Without CAS

If CAS SSO is not enabled, a user must authenticate before accessing each Banner 9.x application. If CAS 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).

Upgrade the Database

The following steps are used to upgrade the database:

- "Perform the Banner DB Upgrade steps" on page 15
- "Update login.sql" on page 15
- "Verify that the required products are applied" on page 16
- "Verify the banproxy database account" on page 16
- "Verify the ban_ss_user database account" on page 16
- "Verify Oracle user accounts to connect through banproxy" on page 16
- "Set up access for application users with an administrative account" on page 17
- "Migrate staged files to the permanent directories" on page 17
- "Update the version numbers" on page 19

Perform the Banner DB Upgrade steps

Some database upgrade steps are common to all Banner 9.x applications. These common database upgrade steps must be performed before you upgrade the database for the Banner General Event Management application.

Refer to the instructions in the *Banner DB Upgrade, Upgrade Guide* (`Banner_db_upgrade_9.4_Upgrade_Guide.pdf`) for the common database upgrade steps. The *Banner DB Upgrade, Upgrade Guide* is delivered in the `banner-db-upgrade-90400d.trz` file.

Update login.sql

You must edit `login.sql` 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 sure to do this 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 provides a method 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.

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 the ban_ss_user database account

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

In order for Self-Service Banner to function properly when PII is enabled, both BANPROXY and BAN_SS_USER must be exempted from PII processing via the setting on the GOAFPUD form.

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**.

Set up access for application users with an administrative account

A new security object named SELFSERVICE is created during the installation of the self-service application. Application users who have an administrative account associated with their login on the Enterprise Access Controls (GOAEACC) page must be assigned this new object with BAN_DEFAULT_M privilege.

Note: The SELFSERVICE object was also added to the BAN_GENERAL_C class. As an alternative, you may associate your administrative users with this class.

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 may, 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 numbers

To insert the release version numbers into the Application Version History Table, perform the following steps:

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

```
sqlplus general/password
start versionupdate
```

2. Review the versionupdate listing.

Install Event Management Self-Service Application

The following sections detail the installation steps for the Banner General Event Management 9.4 application:

- "Undeploy the existing application" on page 20
- "Customize the WAR file" on page 22
- "Regenerate the WAR file" on page 38
- "Configure and deploy the WAR file to a web application server" on page 39
- "Configure the application" on page 53

Undeploy the existing application

Before you install Banner General Event Management 9.4, you must undeploy any previous 9.x versions of Banner General Event Management.

Note: If no previous 9.x versions of Banner General Event Management are installed, skip this section.

The following sections give the steps that are required to undeploy the existing Banner 9.x applications in Tomcat and WebLogic servers.

Tomcat

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

Undeploy using 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.

Note: Make sure you follow the preceding procedure to undeploy existing Banner 9.x online help files in Tomcat server.

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/banner9application
   rm -rf $CATALINA_HOME/webapps/banner9application.war
   ```

Note: Make sure you follow the preceding procedure to undeploy existing Banner 9.x online help files.

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\banner9application /s/q
del %CATALINA_HOME%\webapps\banner9application.war /q
```

Note: Make sure you follow the preceding procedure to undeploy existing Banner 9.x online help files.

WebLogic

Use the following procedure to stop and undeploy the Banner 9.x 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 Banner 9.x 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 Banner 9.x application.
9. Click **Delete**.
10. In the Delete Application Assistant page, click **Yes**.
11. In the Change Center frame, click **Activate Changes**.

Note: Make sure you follow the preceding procedure to undeploy existing Banner 9.x online help files in WebLogic server.

Customize the WAR file

The release package is contained in the `BANNER_HOME\general\java` subdirectory. The name of the release package is `release-SelfServiceBannerGeneralEventManagement-9.4.zip`. This file is moved to this location during the database upgrade.

Note: Java 7 or 8 must be installed on your system. See the Java dependencies section for more information.

To copy 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-SelfServiceBannerGeneralEventManagement-9.4.zip.`

4. Transfer this file in binary mode using File Transfer Protocol (FTP) file into the temporary directory. For example:

   ```
   $HOME/ban9temp
   ```

5. Unzip the release package `release-SelfServiceBannerGeneralEventManagement-9.4.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 is used to support the configuration and creation of a deployable WAR file. Banner 9.x web applications are modular and are installed independently from each other, although they share a common configuration. The package provides a common structure for releases and a common installer. The product home directory structures that the installer creates are consistent across all Banner 9.x modules.

You should place the product homes for Banner 9.x applications, within a particular environment, in sibling directories. For example, the following directory structure illustrates four product homes and a `shared_configuration` directory that support a common test environment.

```
banner_test_homes
|--> Catalog 9.2
|--> Schedule 9.2
|--> Event_Management_Admin
|--> Event_Management_ss
|--> shared_configuration
```

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

The following directory tree illustrates the product home directory that is created using the steps below 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, there is a separate `shared_configuration` home directory that contains cross-module configuration for the test environment. This directory holds a single `banner_configuration.groovy` file that contains the shared JNDI datasource configuration.

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

1. After the installer is prepared using `ant`, use the installer to install the release file into a 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
   ```

2. When prompted, enter the full path of the application home.

   The application will be installed within the current subdirectory within this home and the previous release will be archived.

   On Unix:

   ```
   []: Current_home_directory/banner_test_homes/PRODUCT_HOME
   ```

   On Windows:

   ```
   []: c:\banner_test_homes\PRODUCT_HOME
   ```

3. Enter the full path of the `shared_configuration` home directory. Banner 9.x applications that are configured to refer to this home 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 or the directory for shared configuration does not exist, the installer creates it. The name of a product home is not restricted. You can name it when prompted by the installer.

Configure shared settings

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

JNDI datasource

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/bannerSsbDataSource"` is the default configuration. You can change this to match the JNDI datasource name in your environment.

For more information on JNDI DataSource configuration, see "Configure the Tomcat server" on page 39 or "WebLogic" on page 46.

banner8.SS.url

To display existing Self-Service Banner 8.x menus and breadcrumbs in self-service applications, the following configuration must be updated with the URL to your existing Self-Service Banner 8.x application:

```
//replace with the URL pointing to a Self-Service Banner 8.x instance
banner8.SS.url = '<scheme>://<server hosting Self-Service Banner 8.x>:<port>/<context root>/'
```

For example:

`banner8.SS.url = 'http://localhost:8002/ssb8x/',` this is the URL for the existing Self-Service Banner 8x application. This configuration is applicable to all the Banner 9.x Self-Service applications. We recommend that you add the `banner8.SS.url` field in the `banner_configuration.groovy` so that all the Self-Service applications will use this configuration setting to display the Self-Service menus. If you want to override this configuration setting for a specific Self-Service application, you must add the `banner8.SS.url` field in the Self-Service application specific groovy file (`SelfServiceBannerGeneralEventManagement_configuration.groovy`) with the overridden value. If you do not want to override the configuration settings in banner_configuration.groovy file, you must delete the `banner8.SS.url` field from the Self-Service application specific groovy file (`SelfServiceBannerGeneralEventManagement_configuration.groovy`).

Note: If the `banner8.SS.url` field is not in the `banner_configuration.groovy` file, you must add it to the file before you continue with the installation.

Link to Self-Service Banner 8.x

Self-service application menus can be used to access your Banner 8.x applications. Jasig Central Authentication Service (CAS) (www.apereo.org/cas) is required as an external authentication provider.

```
i18NFieldDisplayEnabled = false in SelfServiceBannerGeneralEventManagement_configuration
```

Atom feed

```
facebookLikeEnabled = true is added in
SelfServiceBannerGeneralEventManagement_configuration.groovy
```

Footer fade time

The following are the additional parameters for Preferred Names

```
displayPreferredName = true
```

The preferred names' default values are:

```
banner.applicationName = "Event Self-service"

productName = "General"

banner.pageName = "Event Details"

banner.sectionName = "My Events"
```

The following components must be configured to authenticate using CAS:

- The Banner 9.x applications must be configured to authenticate using CAS.
- The SSO Manager must be deployed and configured to enable Self-Service Banner 8.x authentication using CAS. The SSO Manager is a component of Banner Enterprise Identity Services (BEIS).

To allow linking from Banner 9.x self-service applications to Self-Service Banner 8.x, add the following URL in the `banner_configuration.groovy` file:

```
banner8.SS.url='http://beissmpl.university.com:7777/
ssomanager/c/SSB?pkg='
```

The `banner8.SS.url` references the SSO Manager URL (`'http://beissmpl.university.com:7777/ssomanager/c/SSB'`) and appends the `'?pkg='` suffix to support deep linking to a specific Self-Service Banner 8.x page.

The following is an example of a Banner 8.x SSO URL through the SSO Manager:

```
http://beissmpl.greatvalleyu.com:7777/ssomanager/c/
SSB?pkg=bwgkogad.P_SelectAtypView
```

Note: For SPRIDEN users, if you do not reference `banner8.SS.url` with a valid URL, the menus and breadcrumbs are not displayed in the self-service application.

Note: Without CAS and the SSO Manager, navigation to Banner 8.x is not seamless. A user must log in to Self-Service Banner 8.x using a valid Self-Service Banner 8.x user name and password. Navigation terminates at the Self-Service Banner 8.x main menu page. To log in without the SSO Manager, see "banner8.SS.url" on page 26.

Session timeouts

The following timeouts are used in the self-service application:

- "banner.transactionTimeout" on page 28
- "AJAX timeout" on page 28
- "defaultWebSessionTimeout" on page 28

banner.transactionTimeout

The `banner.transactionTimeout` property is used to prevent excessive delays due to long database transactions. To use this timeout, you must edit the `banner_configuration.groovy` file and set the `banner.transactionTimeout` property to `300` seconds.

Ensure that either the configuration file is deployed with the application, or the application is using the configuration file where it is currently located.

If a database transaction takes longer than `banner.transactionTimeout` seconds, it is aborted and any change is rolled back.

Note: In some cases, the web user interface might ignore the database timeout/error notification and not remove the loading spinner. If this occurs, you must refresh the page to continue using the application.

AJAX timeout

The AJAX timeout terminates HTTP requests that exceed the specified time limit.

The self-service application sets the timeout value based on the `banner.transactionTimeout` plus an increment to allow for communication and processing of the request.

You do not need to override the AJAX timeout, but the timeout value can be overridden in JavaScript by calling `$.ajaxSetup( { timeout: timeoutValue } );`

Note: The `timeoutValue` must be in milliseconds.

Note: Although the web user interface continues after an AJAX timeout, the server might continue processing the request until it completes or reaches a database `transactionTimeout`.

defaultWebSessionTimeout

The `defaultWebSessionTimeout` property is used to terminate user sessions that are left idle or abandoned without logging out. The `defaultWebSessionTimeout` can be configured in the `banner_configuration.groovy file`. The

`defaultWebSessionTimeout` configuration property can be used to set the overall session inactivity time.

Note: If `defaultWebSessionTimeout` is not specified, a default value of *1500* seconds is used.

Role-based web session timeouts are also configurable in Banner Web Tailor. In Banner WebTailor, `TWTVROLE` is the logged in user role and `TWTVROLE_TIME_OUT` is the timeout for users with that role.

Note: An individual's session timeout is the longer of the `defaultWebSessionTimeout` and the role-based timeout from `TWTVROLE`.

Configure application-specific settings

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

Configure the WebTailor to display Event listing page

In Event Management Self-Service, if you want the Event listing page to be displayed in WebTailor, you must modify the default base URL to your institution's local URL. For example, http://<host>:<port>/SelfServiceBannerGeneralEventManagement/ssb/events. The delivered link can be found as a menu item on **P_MainMnu**. For information on how to modify the default base URL to institution's local URL, refer to *Banner Web Tailor User Guide*.

Enable non-SPRIDEN user access

To enable the non-SPRIDEN users access to the application make sure you have the following setting:

`guestAuthenticationEnabled=true`.

Configuration for Atom Feed absolute links

Every Atom entry requires an absolute hyperlink that points to its article. For more information on Atom Feed, refer to the Banner General Event Management Handbook.

Make sure you link the `banner.eventsFeed.base.url` property to Banner General Event Management Self-Service application URL.

For example, banner.eventsFeed.base.url='<scheme>://<hostname or ip address>:<port>/<context root>' where `banner.eventsFeed.base.url` is the property that points to the base url of the Banner General Event Management Self-Service application, which will be used to create the absolute link to the event details from the feed entry.

For example:

If the Banner General Event Management Self-Service application is deployed in the `BANNER9_HOST` with port 8080 and content root is SelfServiceBannerGeneralEventManagement then the setting will be `banner.eventsFeed.base.url`='http://BANNER9_HOST:8080/SelfServiceBannerGeneralEventManagement'.

For example, for Event A0001 the following absolute URL gets generated from the feed controller:

http://BANNER9_HOST:8080/SelfServiceBannerGeneralEventManagement/ssb/events/details/A0001

Enable Guest Log In link

To enable the guest log in on the Event Management Self-Service application, you need to perform the following configuration in application specific configuration file:

`guestAuthenticationEnabled` = `true`

`banner.sso.authenticationProvider` can be set to anything apart from `default`

As a Guest user, when you try to log out of the Event Management Self-Service application, you will be redirected to the CAS log out page. For more information on redirecting to the CAS log out page, refer to "Logout URL" on page 35.

Note: In CAS mode, uncomment the below provider list. This entry will update the provider list if guestAuthenticationEnabled is true else comment out.

```
//grails.plugin.springsecurity.providerNames =
['casBannerAuthenticationProvider','selfServiceBa
nnerAuthenticationProvider']
```

In SAML mode, uncomment the below provider list. This entry will update the provider list if guestAuthenticationEnabled is true else comment out.

```
//grails.plugin.springsecurity.providerNames =
['samlAuthenticationProvider','selfServiceBannerA
uthenticationProvider']
```

Configure footerFadeAwayTime

To configure the footer fade away time. This will be defaulted to 2 seconds.

Application's Secured Landing page:

1. If footerFadeAwayTime is configured then user will be able to view the footer when accessing the application that is secured page, once in 24hrs
2. If footerFadeAwayTime is not configured the user will be able to view the footer for 2 seconds when accessing the application that is secured page, once in 24hrs
3. If within 24hs the cache is cleared and the secured page of the application is accessed by the same user then the user can view the footer for the configured time
4. If the user uses a different machine but the same credential then the user will view the footer for the configured time

Application's Unsecured Landing page:

User will be able to see the footer for the configured time or default (time 2 seconds) on the applications unsecured landing pages on every access

Configure Extensibility in Events Self Service application

Following configuration change is needed before using the tool. Each application can have the JSON file directory specified in its own [application]_configuration.groovy file or a default directory can be specified in the banner_configuration.groovy file. For example, in either the application-specific configuration file or the banner_configuration.groovy, include the following:

Example external configuration files

```
webAppExtensibility {
    locations {
       extensions = "C:/BanXE/Extensions/ss_ext/extensions/"
        resources = "C:/BanXE/Extensions/ss_ext/i18n/"
    }
}
```

The "extensions" configuration specifies the directory that will contain the JSON files containing modifications to the application pages. The "resources" configuration specifies the directory that will contain property files for any multilingual text values required for the changes. For example, modifications to field labels and title text.

Security has been configured for all non-production environments to have the WEBTAILORADMIN role as the default role to access the Extension Editor. This role can be changed by an institution if it is required. For the Production environment, no default role is set and no access will be given to any user as a default of the installation. Administrative functions of extensibility, like the ability to edit the extension in the web interface are disabled in the application (when running the application in development or

test mode users with the WTAILORADMIN role have access to the administrative functionality to support unit testing), unless explicitly allowed in the webAppExtensibility adminRoles configuration:

Example external configuration files

```
webAppExtensibility {
    …
    //Comma separated list of roles
    adminRoles = "ROLE_SELFSERVICE-
WTAILORADMIN_BAN_DEFAULT_M"
}
```

Configure loginEndpoint

To sign in to Event Management self-service application, the `loginEndpoint` must be set to application URL *postSignLoadDetails* page in the `SelfServiceBannerGeneralEventManagement_configuration.groovy` file.

For example, `loginEndpoint`='http://BANNER9_HOST:PORT/APP_NAME/ssb/events/postSignLoadDetails'

Configure Forgot pin

To enable the Forgot pin functionality for a SPRIDEN user, set `ssbPassword.reset.enabled=true`

To enable the Forgot pin functionality for a non SPRIDEN user, set `ssbPassword.guest.reset.enabled=true`

To enable the reset pin email notification for non SPRIDEN user, set the following URL to the Self-Service application:

banner.events.resetpassword.guest.url='<scheme>://<hostname or ip address>:<port>'

For example, if the Banner General Event Management Self-Service application is deployed in the BANNER9_HOST with port 8080 then the setting will be banner.events.resetpassword.guest.url='http://BANNER9_HOST:8080'

For more information on Forgot Pin functionality, refer to B*anner General Event Management Handbook*.

JMX MBean names

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

```
jmx {
    exported {
        log4j = "self-service-
banner_general_event_management-log4j"
    } }
```

Logging

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.

Output logging file location

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

```
loggingFileDir = "System.properties['logFileDir'] ?
"${System.properties['logFileDir']}" : "target/logs"
logAppName = "selfservicebannergeneraleventmanagementapp"
loggingFileName = "${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 you are deploying. See “Configure the Tomcat server” on page 39 or “WebLogic” on page 46.

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 `SelfServiceBannerGeneralEventManagement_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 you modify the logging levels using JMX, the logging level changes only affect the running application. If you restart the application, any changes you made using JMX are lost. For more information on JMX configuration, see "Configure Java Management Extensions" on page 43.

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 Event Management application 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
*
**************************************************************
*********/
grails.plugin.springsecurity.homePageUrl='http://
servername:PORT/SelfServiceBannerEventManagement' //
can be institutional home page ex: http://myportal/
main_page.html
```

ssbOracleUsersProxied setting

When connecting to self-service applications, the `ssbOracleUsersProxied` setting specifies whether the Oracle account is used for the connection. This also controls whether Value Based Security (VBS) is enabled in the self-service application. The following values can be used for this setting:

- `ssbOracleUsersProxied = False` - The Oracle account is not used for the connection and VBS is not enabled in the self-service application. If the Oracle account is locked, the user is not prevented from logging in to the self-service application.

- `ssbOracleUsersProxied = True` - The Oracle account is used for the connection and VBS is enabled in the self-service application. If the Oracle account is locked, the user is prevented from logging in to the self-service application.

Logout URL

You can specify where a user should be directed after logging out of the application by updating the `SelfServiceBannerGeneralEventManagement_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.

Configure logout URL information as follows:

```
grails.plugins.springsecurity.logout.afterLogoutUrl=https://
<CAS_HOST>:<PORT>/<cas>/logout?url=http://myportal/
main_page.html
```

The logout URL in the following example instructs the CAS server to redirect the browser to `http://myportal/main_page.html` after performing a CAS single logout.

```
//                  +++ CAS CONFIGURATION +++
**********************************************************

// This entry is required to ensure that 'Sign In' link takes
you to corresponding login i.e. for CAS, any other
authentication system or Default (Banner).

// This entry is a must for Events SSB-CAS specific logins.

loginEndpoint='http://BANNER9_HOST:PORT/APP_NAME/ssb/events/
postSignLoadDetails'


banner {
    sso {
           authenticationProvider = ‘cas' // Valid values
are: 'default', 'cas', ’saml’
        authenticationAssertionAttribute = 'UDC_IDENTIFIER'
            /** CAS or SAML configuration **/
                   if (authenticationProvider != 'default')
```

```
            {
            grails.plugin.springsecurity.failureHandler.defa
            ultFailureUrl = '/login/error'
        }
        }
}
//defines the CAS logout URL and instructs the CAS server to
redirect the browser to http://myportal/main_page.html after
performing a CAS single logout.

//
grails.plugin.springsecurity.logout.afterLogoutUrl='https://
<CAS_HOST>:<PORT>/<cas>/logout?url=http://myportal/
main_page.html'


grails {
 plugin {
  springsecurity {
   cas {
    active = true
    serverUrlPrefix  = 'https://CAS_HOST:PORT/cas'
    serviceUrl = 'http://BANNER9_HOST:PORT/
selfservicebannergeneraleventmanagementapp/
j_spring_cas_security_check'
    serverName = 'http://BANNER9_HOST:PORT'
    proxyCallbackUrl = 'http://BANNER9_HOST:PORT/
selfservicebannergeneraleventmanagementapp/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'
   }
logout {
afterLogoutUrl = 'https://cas-server/logout?url=http://
myportal/main_page.html'
mepErrorLogoutUrl = '/logout/logoutPage'
}
```

```
saml {
active = false
}
}
}
```

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

In an application set only 1 active at a time. That is either
`grails.plugin.springsecurity.cas.active = true`
OR
`grails.plugin.springsecurity.saml.active = true`

Both modes being active is an incorrect state of the application and you will see unexpected behaviors.
And In default or external modes of authentication Provider set both active = false

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 as follows, replacing "url" with "service."

   Example:

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

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="ticketGrantingTicketCookieGenerator"
      p:followServiceRedirects="true" />
   ```

Password reset

You can specify whether users have the ability to reset their passwords by updating the `ssbPassword.reset.enabled` setting. If the value of the setting is `true`, users can reset their passwords. If the value of the setting is `false`, a "disabled" error message is displayed.

Redirect pages in a MEP environment

Two settings in the configuration file must be configured if your environment is enabled for Multi-Entity Processing (MEP). You do not need to configure these settings if your environment is not enabled for MEP.

If a user tries to access a self-service URL that does not include a valid MEP code, an error prompt is displayed. The user responds by clicking the **Logout** or **Return Home** button. Configuration settings determine where each button redirects the user:

- `grails.plugin.springsecurity.logout.afterLogoutUrl` - This setting contains the URL of the institution-specific page where the user is redirected if the **Logout** button is clicked. An example is a portal page:

  `"https://cas-server/logout?url=http://myportal/main_page.html"` (CAS environment)

  `"http://myportal/main_page.html"` (non-CAS environment)

- `grails.plugin.springsecurity.logout.mepErrorLogoutUrl` - This setting contains the URL of the institution-specific page where the user is redirected if the **Return Home** button is clicked.

Each URL can be any page that does not require a MEP-enabled database connection, or any page outside the self-service application.

Self-service end point

The `ssbEnabled` setting is set to `true` for instances that expose self-service end points. Banner General Event Management is a self-service application, so the default value is always `true`.

Regenerate the WAR file

Once the shared and application-specific configurations are complete, the application WAR file can be regenerated to include your customizations, online help, 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. This 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, making the WAR file self-sufficient. For information on external configuration, see "Tomcat" on page 39 or "WebLogic" on page 46.

Configure and deploy the WAR file to a web application server

The following sections provides information on how to configure the web application and deploy the WAR file to a web application server:

- "Tomcat" on page 39
- "WebLogic" on page 46

Tomcat

The following sections provide information on how to configure the web application and deploy the WAR file to the Tomcat server.

Configure the Tomcat server

Use the following procedure to configure the Tomcat server:

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

Note: Tomcat version 6 or 7 is required. To download and install the Tomcat 6 or 7 server, see http://tomcat.apache.org.

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

   Note: Later in the Tomcat configuration process, you will copy the Oracle JDBC jar file 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-6.0.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.

   This variable can be defined 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 name 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 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 file (`ojdbc6.jar`) from the `PRODUCT_HOME/current/lib` directory to the `$CATALINA_HOME/lib` directory.

6.9. Copy xdb6.jar from the StudentApi/current/lib directory to the $CATALINA_HOME/lib directory.

Note: The 6.9 step is not required for Tomcat 8.

6.10. 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
-DSELF_SERVICE_BANNER_GENERAL_EVENT_MANAGEMENT__CONFIG=
/PRODUCT_HOME/SelfServiceBannerGeneralEventManagement/
current/instance/config/
SelfServiceBannerGeneralEventManagement_configuration.groovy
"
```

Configure/enable CAS in the product groovy file

CAS must be configured in the product groovy file in the `current/instance/config/Self-ServiceBannerGeneralEventManagement_configuration.groovy` file before the war file is built with the system tool command.

CAS parameters present in the external configuration file will not be read when the Banner 9 XE application is launched.

Ensure that the CAS is enabled and the external BANNER_CONFIGURATION_GROOVY file has the CAS configuration properties present in the file. For more information, refer to the "To display the CAS logout page with a redirect URL" on page 35 section.

When CAS is enabled in an external configured file, the user will be able to login to the application.

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-6.0-doc/monitoring.html#Enabling_JMX_Remote.*

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. 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.

Note: You should deploy 9.x administrative applications and 9.x self-service applications to separate Tomcat servers to increase performance.

The target provides support for deploying the `dist/WAR` file using the Tomcat Manager web application.

Note: Instead of using the target, the WAR file deployment to the Tomcat server can also be accomplished by copying the WAR file to the Tomcat `webapps/` directory.

Note: 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.

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

- The URL of the manager application in the Tomcat server. For example:

  ```
  http://localhost:8080/manager
  ```

- A Tomcat server username that has privileges to deploy WAR files.

  Note: For Tomcat 6.x, you must configure at least one username/password combination in your Tomcat user database `<TOMCAT_HOME>\conf\tomcat-users.xml`, which includes the manager role. For example:

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

- The password of the Tomcat server user.

  Note: The roles in Tomcat server changed between point releases in version 6.x. 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>/SelfServiceBannerEventManagement
```

WebLogic

The following sections provide information on how to configure the web application and deploy 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.

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.
- The minimum requirements are OFM 11.1.14 using WebLogic 10.3.4.
- 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

To create a WebLogic machine, perform the following steps:

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

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

To create a WebLogic server, perform the following steps:

Note: If you already 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.

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-SS` 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 and then click **Save**.
9. Select the **Server Start** tab.
10. 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 managed server, increase the max heap (`-Xmx`) by `2g` and `-XX:MaxPermSize` by `128m`. You should deploy Banner 9.x administrative applications to one WebLogic managed server instance and Banner 9.x self-service applications to a separate managed server instance.

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

    ```
    -Xms2048m -Xmx4g
    ```

    Note: For JRockit, increase the max heap `-Xmx` by `2G` for each Banner 9.x application that is deployed.

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>
-DSELF_SERVICE_BANNER_GENERAL_EVENT_MANAGEMENT_CONFIG=
<full file path to
SelfServiceBannerGeneralEventManagement_configuration.
groovy>
```

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

Update Oracle JDBC JAR files on the WebLogic server

1. Copy the Oracle JAR files (ojdbc6.jar and xdb6.jar) from the $PRODUCT_HOME/current/lib directory to the $MIDDLEWARE_HOME/modules directory.
   - $PRODUCT_HOME is where the Self-Service 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:${WLS_MODULES}/ojdbc6.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" or "ojdbc.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;%WLS_MODULES%\ojdbc6.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" or "ojdbc.jar" file cannot be found during Application startup.

4. Restart the WebLogic Managed Server.

Create an administrative datasource and connection pool

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. 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**. 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

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, `Banner9DS`).
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**. For example:

Database name:	`BAN9`
Host name:	`yourhostname.yourdomain.com`
Port:	`1521`
UserName:	`ban_ss_user`
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**.

Configure server communication

When using different servers for administrative and self-service applications, 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 **DataSources**.
3. Select the datasource for the administrative application (for example, `Banner9DS`).
4. Select the **Targets** tab.
5. Select the check box for the self-service server (for example, `Banner9-SS`).

6. Click **Save**.
7. 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:

   ```
   PRODUCT_HOME/current/dist/
   SelfServiceBannerGeneralEventManagement-9.4.war
   ```

   to

   ```
   PRODUCT_HOME/current/dist/
   SelfServiceBannerGeneralEventManagement.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. It is located in the following directory:

   ```
   PRODUCT_HOME/current/dist
   ```

7. Click **Next**.
8. Select **Install this deployment** as an application.
9. Click **Next**.
10. Select the target server on which to deploy this application (for example, `Banner9-SS`).
11. Click **Next**.
12. Click **Finish**.
13. In the Change Center frame, click **Activate Changes**.
14. Select the deployed application and then click **Start**.
15. Select **Servicing all request**.
16. Access the application using the following URL format:

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

    For example:

    ```
    http://localhost:8080/SelfServiceBannerEventManagement
    ```

17. Log in to the application using a valid user name and password.

Configure the application

The following aspects of the application can be configured.

Name format

The default name format is First Middle Last. You can change the format to any order for any locale.

Within the `PRODUCT_HOME/current/i18n` folder, find the `message.properties` file for your language. The default for American English is the `message.properties` file.

Open the file with a text editor. Add the following entry for the name format, using the `$firstName`, `$mi`, `$lastName`, and `$surnamePrefix` keys:

```
default.name.format=$surnamePrefix $firstName $mi $lastName
```

Note: The login user name shows the person's name in the format First Middle Last.

Phone format

Within the `PRODUCT_HOME/current/i18n` folder, find the `message.properties` file for your language. The default for American English is the `message.properties` file.

Open the file with a text editor. Add an entry for the `default.personTelephone.format`. Formats include `$phoneArea`, `$phoneNumber`, `$phoneExtension`, `$phoneCountry`, and `$phoneInternational`.

Date format

The date format can be customized by using the following keys in the `messages_<ISO_language_code>_<ISO_country_code>.properties` file:

- `default.date.format`
- `js.datepicker.dateFormat`

default.date.format

This key determines the date format for display and data entry in the user interface. It must match the ICU specification and can be changed based on locale. For more information, refer to the following URL: http://userguide.icu-project.org/formatparse/datetime.

For the `default.date.format` for June 1, 2012, use either of the following variables for the year:

Year format	Interpretation	Comment
yy	12	Two digit year
yyyy	2012	Four digit year

For the `default.date.format` for June 1, 2012, use one of the following variables for the month:

Month format	Interpretation	Comment
M	6	Single digit month (no leading zero)
MM	06	Double digit month
MMM	Jun	Short month name
MMMM	June	Long month name

For the `default.date.format` for June 1, 2012, use one of the following variables for the day:

Day format	Interpretation	Comment
d	1	Single digit day in a month (no leading zero)
dd	01	Double digit day in a month

The `default.date.format` and `js.datepicker.dateFormat` formats use different specifications to represent the date. The ICU format is dd/MM/yyyy and Java Script or Keith Wood format is dd/mm/yyyy. For the dates to be displayed properly, the two formats must match. For example, for 1st June, 2012, the calendar parses 01/06/2012 using `js.datepicker.dateFormat`, and when the dates are saved, the date value on the server side is parsed by groovy using `default.date.format` to display 01/06/2012 in the application.

js.datepicker.dateFormat

This key determines the date format for the interactive date selection control. It must match the jQuery Keith Wood datepicker format specification. For more information on Keith Wood datepicker format, refer to the following URL: http://keith-wood.name/datepick.html#format.

For the `js.datepicker.dateFormat` for June 1, 2012, use either of the following variables for the year:

Year format	Interpretation	Comment
yy	12	Two digit year
yyyy	2012	Four digit year

For the `js.datepicker.dateFormat` for June 1, 2012, use one of the following variables for the month:

Month format	Interpretation	Comment
m	6	Single digit month (no leading zero)
mm	06	Double digit month
M	Jun	Short month name
MM	June	Long month name

For the `js.datepicker.dateFormat` for June 1, 2012, use one of the following variables for the day:

Day format	Interpretation	Comment
d	1	Single digit day in a month (no leading zero)
dd	01	Double digit day in a month

Time format

The `default.time.format` is a simple Java date format key available in the Banner General Event Management application. For more information on the Java date format, refer to http://docs.oracle.com/javase/6/docs/api/java/text/SimpleDateFormat.html.

Multiple calendars

In the application, customization of multiple calendars is implemented for the Arabic language (AR). The file name for Arabic is `messages_ar.properties`.

If you want to display multiple calendars in the application, you must use the following keys:

- `default.calendar`
- `default.calendar1`
- `default.calendar2`

Note: The `default.calendar2` is optional.

In the following example, the keys set your default calendar format as Islamic, the first alternate calendar displayed as Gregorian, and the second alternate calendar as Arabic.

```
default.calendar=islamic

default.calendar1=gregorian

default.calendar2=islamic
```

Depending on the order you want to view the calendars, you can interchange the following values:

- `islamic`
- `gregorian`

CSS customization

To customize the appearance of the self-service application, you can provide custom CSS and image files.

Create a CSS file `PRODUCT_HOME/current/instance/css/bannerSelfService-custom.css` containing the custom CSS directives.

If you want to provide custom images, save them in the `PRODUCT_HOME/current/instance/css/images` directory, and in the CSS, specify their paths as a URL. For example:

```
(./images/filename.png)
```

Note: If this directory structure does not exist, create the directory structure.

Institution name

The default layout includes an institutional branding area to display an institution name or logo.

To customize the system/university name, you must provide a replacement logo image and a custom CSS file to override the default styling for the institutional branding area.

The default layout styles the institutional branding area as follows:

```
.institutionalBranding {
    position:relative;
    float:left;
    left:10px;
    top:11px;
    height:19px;
    width:179px;
   background:url("images/ellucian-university-logo-sm.png")
no-repeat;}
```

You can override the image displayed by creating a CSS file named `PRODUCT_HOME/current/instance/css/bannerSelfService-custom.css` that contains the following:

```
.institutionalBranding {background-image: url("./images/
institutionLogo.png");}
```

Place the logo image in the following directory:

```
PRODUCT_HOME/current/instance/css/images/institutionLogo.png
```

To deploy your updates, you must rebuild and redeploy the WAR file. For more information, see "Regenerate the WAR file" on page 38.

Custom JavaScript

You can optionally add a custom JavaScript file by placing your JavaScript file named `bannerSelfService-custom.js` in the following location:

```
PRODUCT_HOME/current/instance/js/bannerSelfService-custom.js
```

Install Banner General Event Management SSB for SAML 2.0 SSO

The following sections detail the installation steps for the Banner General Event Management Self-Service 9.4 application with the Ellucian Identity Service (EIS) as the primary Identity Management System that supports the SAML 2.0 SSO protocol:

- "Generate SAML 2.0 metadata files" on page 58
- "Set up SAML 2.0 SSO Configuration" on page 62
- "Add service provider in EIS Server" on page 66
- "Modify Identity Provider Issuer" on page 68
- "SAML 2.0 Configuration Sub-tasks" on page 70

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 70 for more information.

2. Place this file in the location specified in the following key value:
   `"grails.plugin.springsecurity.saml.keyManager.storeFile = 'classpath:security/<short-appName>keystore.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 63.
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 Self Service Banner General Event Management.

   - <HOSTNAME>: Application host name
   - <PORT>: Deployed Application port number
   - <ALIAS_NAME>: Service provider ID set in EIS 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 70 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>/ssb/events/saml/
SingleLogout/alias/<ALIAS_NAME>"/>
<md:SingleLogoutService Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Redirect" Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/ssb/events/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>/ssb/
events/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>/ssb/
events/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 63.
2. Edit the `banner-<short-appName>-idp.xml` file for the identity provider configuration.

   The file contains the identity provider information configured in EIS server over which Self Service Banner General Event Managementsends the SAML request and receives the SAML response.

   Replace below parameters with the configured values for Self Service Banner General Event Management.

   - <HOSTNAME>: EIS server host name
   - <PORT>: Deployed EIS server port number
   - <EXTRACTED_DATA>: Extract X509 certificate Data for `banner-<short-appName>-idp.xml` (See "Extract X509 certificate data" on page 72 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 74 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 `SelfServiceBannerGeneralEventManagement\current\instance\config` directory.

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

- `<short-appName>-keystore.jks` (The newly created `.jks` file)
- `banner-<short-appName>-idp.xml`
- `banner-<short-appName>-sp.xml`
- `SelfServiceBannerGeneralEventManagement_configuration.groovy`

Set up SAML 2.0 SSO Configuration

The `SelfServiceBannerGeneralEventManagement\current\instance\config` directory contains the `SelfServiceBannerGeneralEventManagement_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 2.0 SSO protocol authentication as shown in the following example:

```
/**************************************************
*     BANNER AUTHENTICATION PROVIDER CONFIGURATION    *
*                                                    *
**************************************************/


banner {
sso {
authenticationProvider = 'saml' //  Valid values are: 'saml' and 'cas'.
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 `SelfServiceBannerGeneralEventManagement_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 Self-Service Banner General Event Management 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 Self-Service Banner General Event Management 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.

Property	Description
grails.plugin.springsecurity.saml.keyManager.defaultKey	Key name used at the time of key-store creation ("Create a keystore (*.jks) file" on page 70). Example: <short-appName>keystore.jks
grails.plugin.springsecurity.saml.keyManager.storeFile	Location of the keystore file. This could be a classpath (for example, classpath:security<short-appName>key store.jks) or it could be an absolute location on the machine (for example, file:c://temp/<short-appName>keystore.jks or u02/<short-appName>keystore.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.
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-<short-appName>-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-<short-appName>-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.

Property	Description
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 saml.active = true if Self-Service Banner General Event Management is
configured with SAML2 SSO *
*********************************************************/
grails.plugin.springsecurity.saml.active = false
grails.plugin.springsecurity.auth.loginFormUrl = '/saml/login'
grails.plugin.springsecurity.saml.afterLogoutUrl ='/logout/customLogout'


banner.sso.authentication.saml.localLogout='false' // To disable single logout set
this to true.


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


grails.plugin.springsecurity.saml.metadata.sp.file = 'se-curity/banner-<short-
appName>-sp.xml'               // for unix based '/home/u02/banner-<short-appName>-
sp.xml'
```

```
grails.plugin.springsecurity.saml.metadata.providers = [adfs: 'security//banner-<short-appName>-idp.xml']    // for unix based '/home/u02/banner-<short-appName>-idp.xml'

grails.plugin.springsecurity.saml.metadata.defaultIdp = 'adfs'

grails.plugin.springsecurity.saml.metadata.sp.defaults = [

local: true,

alias: 'banner-<short-appName>-sp',
// banner-<short-appName>-sp is the value set in EIS Service provider setup

securityProfile: 'metaiop',

signingKey: 'banner-<short-appName>-sp',
// banner-<short-appName>-sp is the value set in EIS Service provider setup

encryptionKey: 'banner-<short-appName>-sp',
// banner-<short-appName>-sp is the value set in EIS Service provider setup

tlsKey: 'banner-<short-appName>-sp',
// banner-<short-appName>-sp is the value set in EIS Service provider setup

requireArtifactResolveSigned: false,

requireLogoutRequestSigned: false,

requireLogoutResponseSigned: false

]
```

Add service provider in EIS Server

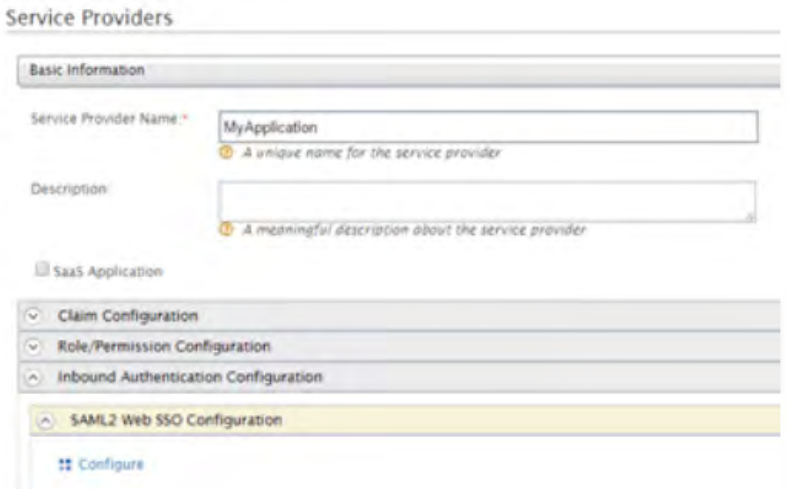
To add a service provider, complete the following steps:

1. Log in to the EIS 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:



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 `SelfServiceBannerGeneralEventManagement_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 EIS. 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."Create a service provider file" on page 59.
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.

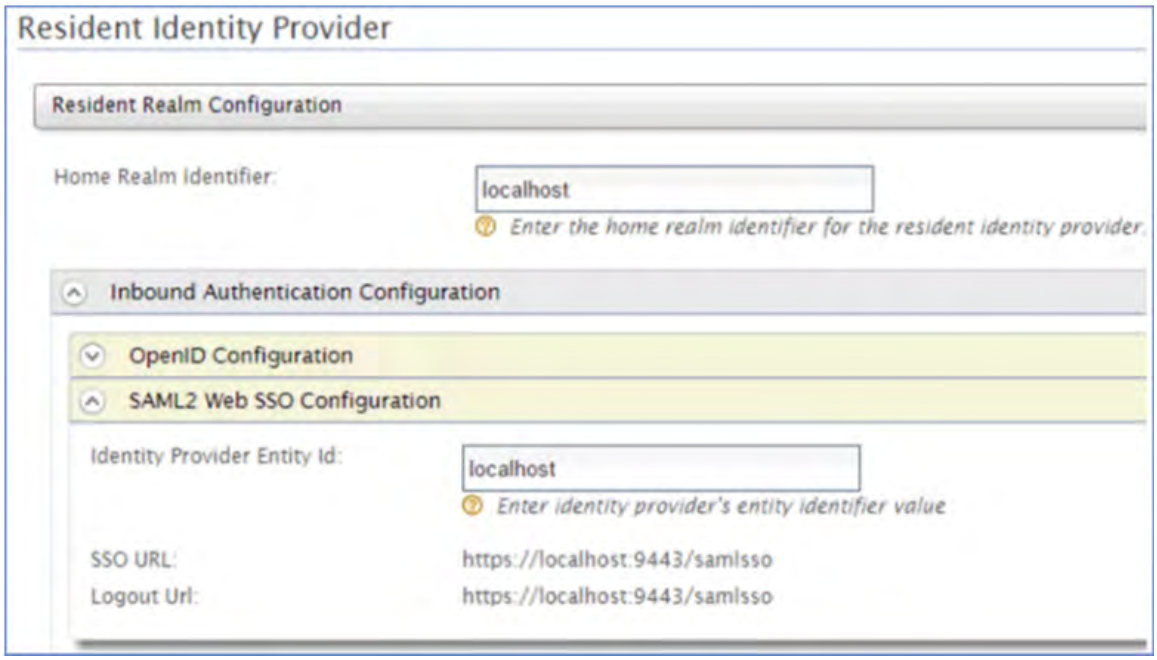
The EIS Identity Server can mediate authentication requests between service providers and identity providers. At the same time, the identity server can act as a service provider and an identity provider. When acting as an identity provider, it is known as the resident identity provider. This converts 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 EIS 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:



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

This section discusses the sub tasks for SAML 2.0 configuration.

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 <short-appName>keystore.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 `<short-appName>keystore.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 `<short-appName>keystore.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 70 for more information.

```
C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -list -v -keystore <short-
appName>keystore.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 <short-
   appName>keystore.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 <short-
   appName>keystore.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+DuavXavjIx5IN6RRhXGfaJjQLKFzIDqZNAp
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 / EIS 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 70. 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 <short-appName>keystore.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
<short-appName>keystore.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
]
]
*******************************************
*******************************************
```