ellucian.

Banner

General Self-Service

Installation Guide

Release 9.2

April 2018

Notices

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

Contents

Before you get started 7
- Use Ellucian Solution Manager to install your product upgrades 7
- Supporting documentation 7
- Hardware requirements 8
- Software requirements 8
  - Oracle database 8
  - Web application servers 8
  - Middle tier (application server) platforms 9
  - Developer requirements 9
  - Ellucian applications 9
  - Single sign-on (SSO) support 10
  - Java dependencies 10
  - Browsers 10
    - Internet Explorer Compatibility view 11
  - F5 load balancer configuration 11
  - Configure Application Navigator 11

Upgrade the database 12
- Update login.sql 12
- Verify that the required products are applied 12
- Verify the banproxy database account 13
- Verify the ban_ss_user database account 13
- Verify Oracle user accounts to connect through banproxy 13
- Set up access for application users with an administrative account 13
- Migrate staged files to the permanent directories 14
  - Unix 14
  - Windows 15
- Update the version numbers 15

Undeploy existing application 16
- Undeploy applications by using the Tomcat Manager web application 16
- Undeploy applications manually on UNIX 16
- Undeploy applications manually on Windows 17
- Undeploy applications by using WebLogic 18

Customize the WAR file 19
- Unzip the release package 19
- Prepare the installer 19
- Install into the product home directory 20
- Configure shared settings 22
  - JNDI datasource 22
  - Link to Self-Service Banner 8.x 22
  - Navigational MEP support from General Self-Service to Banner Self-Service 8.x 23

Session timeouts....24
banner.transactionTimeout.... 24
AJAX timeout....25
defaultWebSessionTimeout....25
Location of photographs....25
Default photograph.... 26
Configure application-specific settings....26
JMX MBean name....26
Location of the logging file....27
Logging level....27
Customize a theme for the application.... 28
Institutional home page redirection support.... 28
Proxied Oracle users....28
Guest authentication....28
Logout URL....29
Provide the CAS logout page with a redirect URL....29
Provide only a redirect URL.... 30
Password reset.... 31
Redirect pages in a MEP environment.... 31
Theme Editor tool.... 32
Google Analytics....34
Action Item Processing configuration....34
Enable Banner Page Builder for Action Item Processing.... 35
Enable Banner Communication Management (BCM) for Action Item Processing.... 36
Enable Quartz Jobs settings for Action Item Processing....37
Configure application-specific settings by modifying messages.properties file....38
Change the name format.... 38
Format name....39
Format addresses....39
Change the phone format.... 40
Change the date format.... 41
Date format keys....41
Display multiple calendars....43
Customize CSS....44
Change the institution logo....44
Customize JavaScript.... 45
Change the disclaimer text....45
Insert a line break in the disclaimer text....46

Regenerate the WAR file.... 47

Configure the web application server and deploy the WAR file.... 48

Configure the Tomcat server....48
Configure Java management extension....51
Deploy the WAR file to the Tomcat server.... 52
WebLogic server.... 53
Verify WebLogic prerequisites.... 53
Set up the cookie path....54
Create a WebLogic machine....55

Configure the WebLogic server....55
Configure weblogic.xml file to make Banner 9.x JSession cookie secure....56
Update Oracle JDBC JAR files on the WebLogic server....57
Create an administrative datasource and connection pool....58
Create a Self-Service datasource and connection pool....59
Configure server communication....60
Deploy and start the application in the WebLogic server....61

Install Banner General Self-Service for SAML 2.0 SSO....62

Generate SAML 2.0 metadata files....62
Create a keystore (*.jks) file....62
Create a service provider file....63
Create an identity service provider file....65
Add an IDP Certificate entry in the newly created keystore file....66
Add keystore, service provider, and identity service provider files to WAR file creation location....66
Configure application-specific settings....67
JMX MBean name....67
Location of the logging file....67
Logging level....68
Institutional home page redirection support....69
Proxied Oracle users....69
Guest authentication....69
Enable SAML....70
Set up SAML SSO configuration....70
Authentication Provider Name....70
Logout URL....70
SAML 2.0 SSO configuration....71
Customize the landing page background image....74
Regenerate the WAR file....76
Add service provider in EIS server....76
Add SAML settings for EIS server....77
Modify identity provider issuer....78
SAML 2.0 Configuration Sub-tasks....79
Extract a X509 Certificate Key....79
Extract X509 certificate data....81
Add an IDP certificate entry in the newly created keystore file....83

Personal Information configuration....86

Personal Information sections....86
Overview section....87

Provide access to the Personal Information application....89

Direct Deposit user and role configuration....90

User setup....90
Banner ID....91
Oracle ID....91

Banner Self-Service PIN.......... 91
Banner Web Tailor role.......... 91
Web Tailor parameters.......... 91

Provide access to the Direct Deposit application.......... 93

Before you get started

Review these prerequisites for hardware, software, and the supporting documentation you can reference.

Use Ellucian Solution Manager to install your product upgrades

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

With Solution Manager, you can:

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

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

Supporting documentation

How you can find documentation associated with the installation process.

- You can obtain documentation for any application you download through Ellucian Solution Manager by going to **Products page** > **Latest Available Releases version number** > **documentation icon.**
- *CAS Single Sign-On Handbook* - found in the Banner General library in the Ellucian Support Center.
- *Ellucian Solution Manager Installation and Configuration Guide* (latest version) - found in the Ellucian Solution Manager documentation library in the Ellucian Support Center.
- *Banner Middle Tier Implementation Guide*

Hardware requirements

These are the hardware requirements Ellucian recommends.

CPU and memory

Recommended: Quad core CPU with 8 GB of memory for the application server.

Screen resolution

Minimum: 1024 x768

Tablets

Ellucian supports the following minimum tablet versions. You can implement on one of these versions or higher.

- iPad, iPad Mini, iOS 6x, and iOS 7.x
- Android OS 4.c
- Microsoft Surface 1.0, RT, and PRO

Mobile devices

The application is supported on certain mobile devices, including iPhone, Nexus, and Samsung.

Software requirements

These are the software requirements Ellucian recommends.

Oracle database

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

Web application servers

The Banner General Self-Service application is supported by WebLogic and Tomcat web servers.

- Oracle WebLogic: 10.3.3, 10.3.4, 10.3.5, 10.3.6, and 12.1
- Apache Tomcat: 7 and 8

Oracle Fusion Middleware (OFM) consists of several software products, including WebLogic Server. WebLogic Server is required for an Oracle Banner 9.x application server environment. No other

OFM products are required. However, if an SSL-enabled Oracle HTTP Server (OHS) port is used, the Oracle Web Tier should also be installed to use the `mod_wl_ohs`.

Middle tier (application server) platforms

The General Self-Service application supports multiple operating systems on both Tomcat and WebLogic servers.

Tomcat (64 bit)	WebLogic (64 bit)
Red Hat Linux 6.0 (minimum version)	Red Hat Linux 6.0 (minimum version)
Windows Server 2008	Windows Server 2008
Solaris 10	Solaris 10
AIX 6.1 (JDK 1.7 SR 10 or later)	AIX 6.1 (JDK 1.7 SR 10 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.

Developer requirements

Users who pull the source code from GIT need to use specific versions of software when making extensions.

- Grails 2.5.0
- JDK 1.7
- Tomcat 8
- Oracle 11.2.0.4 or higher
- Release GIT Tag, `rel-generalselfservice-9.2`

Ellucian applications

These Ellucian applications are required.

Note: Ellucian Solution Manager checks for the latest required software with each installation you perform. If you do not have the required software, ESM will prompt you to install that software.

- Banner Common DB Upgrade 9.14
- Banner General 8.10
- Banner Web General 8.7

- For Action Item Processing (AIP), Banner Communication Management (BCM) must be installed.

Single sign-on (SSO) support

Ellucian recommends that you first establish the single sign-on environment before implementing General Self-Service. Central Authentication Service (CAS) and Security Assertion Markup Language (SAML) 2.0 are the Single Sign-On (SSO) protocols supported by General Self-Service. Both are certified to run with Ethos Identity Server (EIS).

You can download EIS from the Ellucian Support Center. For more information on how to establish a single sign-on environment, refer to the *CAS Single Sign-On Handbook* or *Setting Up Ellucian Identity Service* Both are available for download from the Ellucian Support Center.

Warning! If you do not enable Single Sign-On (SSO), a user must authenticate before accessing each Banner 9.x application. If a user logs out of an application, the user is logged out of that current application, but is still logged in to all other applications that are currently open.

Java dependencies

Development for Banner 9.x is currently supported on Java 7 only, and 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: 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.

Browsers

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

- Internet Explorer 10 or 11
- Google Chrome
- Mozilla Firefox
- Safari 6 or 7
- Microsoft Edge

Internet Explorer Compatibility view

When using Internet Explorer, you must be in Internet Explorer Standard Mode.

Procedure

1. **Tools** > **Compatibility View Settings**
2. Clear the **Display intranet sites the Compatibility View** check box, and select the **Display all websites** check box.

F5 load balancer configuration

The application was tested using an F5 load balancer.

It was configured with the following settings:

```
Load Balancing type = Round Robin
Persistence = Cookie
```

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

Configure Application Navigator

Application Navigator is a software component that facilitates seamless navigation between Banner 8.x and Banner 9.x Administrative applications, and Banner 9.x Self-Service applications.

You can configure Application Navigator so that a menu item will display in the Application Navigator menu for each Self-Service Application. Refer to the *Application Navigator Installation Guide* and *Application Navigator Handbook* for instructions on setting up this configuration. These guides are available in the Ellucian Support Center in the Banner General Documentation Library.

Upgrade the database

General Self-Service requires a minimum of Common Database Upgrade 9.14.

For more information about upgrading the database, see the *Banner DB Upgrade, Upgrade Guide 9.14* (Banner_db_upgrade_9.14_Upgrade_Guide.pdf), which you can download from the Ellucian Support Center in the Banner General Documentation Library.

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.

Procedure

1. Replace the `#UPDATEME#` string with the value of a particular schema owner's password in your environment. Make this update in your environment for each Banner schema owner.
2. Set the value that gets assigned to `splpref`.

   The value can be set to the ORACLE_SID or to a directory name. Your options depend on the operating system.

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

Verify that the required products are applied

You have to check and verify that all prerequisite products are applied to the environment.

Procedure

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 `USR_SS_DEFAULT_M` role to `ban_ss_user`. If this role is revoked, the application will not start successfully.

Verify Oracle user accounts to connect through banproxy

All Internet Native Banner (INB) or Oracle user accounts must connect using the `banproxy` privilege. Verify that Oracle user accounts can connect through `banproxy`.

Procedure

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 selfservice 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 can 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 can, however, use the file `GENMIGR.TXT` as a starting point for writing your own migration scripts.

Unix

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

About this task

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:

Procedure

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.

Procedure

1. Check the value of the `BANENV` environment variable by executing the SET command from the DOS prompt.
   a) 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.`
   b) 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 Web Application (GURWAPP) table, perform the following steps.

Procedure

1. Invoke SQL*Plus and run the following procedure to insert the version number for the self-service application:
   ```
   sqlplus general/password
   ```
   ```
   start versionupdate
   ```
2. Review the `versionupdate` listing.

Undeploy existing application

Before you install Banner General Self-Service 9.2, you must undeploy the previously deployed Banner General Self-Service 9.x application. If you have been using the Personal Information and Direct Deposit applications, then you must also undeploy Personal Information 9.x and Direct Deposit 9.x.

The procedure for undeploying the applications varies depending on whether you are using Tomcat or WebLogic servers.

Undeploy applications by using the Tomcat Manager web application

This task allows you to undeploy any previously deployed versions of Banner General Self-Service 9.x, Banner Personal Information 9.x, and Banner Direct Deposit 9.x applications on your systems from the Tomcat server by using the Tomcat Manager web application.

Procedure

1. Access the Tomcat Manager web application at one of the following URLs:
   - `http://server:8080/manager`
   - `http://server:8080/manager/html`
2. Access the deployment page by using a valid user name and password.
3. If you have been using General Self-Service 9.x, Personal Information 9.x, or Direct Deposit 9.x, then stop previously deployed versions of these applications.
   a) In the **Commands** area, click **Stop** to stop the existing application.
4. In the confirmation dialog box, click **OK**.
5. In the **Commands** area, click **Undeploy**.
6. In the confirmation dialog box, click **OK** to undeploy the application.

Undeploy applications manually on UNIX

This task allows you to shut down Tomcat and manually undeploy any previously deployed versions of Banner General Self-Service 9.x, Banner Personal Information 9.x, and Banner Direct Deposit 9.x on your UNIX systems.

Procedure

1. Log in to the server on which Tomcat is running by using the credentials that were used to start Tomcat.

2. Shut down Tomcat by running the following shutdown script: `$CATALINA_HOME/bin/shutdown.sh`.
3. Remove the current deployment:
   a) If you have been using General Self-Service 9.x, Personal Information 9.x, or Direct Deposit 9.x, then remove the previously deployed versions of these applications.

   ```
   cd $CATALINA_HOME
   rm -rf $CATALINA_HOME/webapps/<General Self Service 9x Application>
   ```

4. Remove the associated WAR file for each of the applications removed in the previous step: `rm -rf $CATALINA_HOME/webapps/`*`war_file_name`*.

Undeploy applications manually on Windows

This task allows you to shut down Tomcat and manually undeploy any previously deployed versions of Banner General Self-Service 9.x, Banner Personal Information 9.x, and Banner Direct Deposit 9.x applications on your Windows systems.

Procedure

1. Shut down Tomcat.

If you have	Then
Installed Tomcat as a service	Use the Service Control panel to stop the application.
Not installed Tomcat as a service	Use the following shutdown script: `%CATALINA_HOME%\bin\shutdown.bat`

2. If you have been using Banner General Self-Service 9.x, Banner Personal Information 9.x, and Banner Direct Deposit 9.x applications, then stop previously deployed versions of these applications.
3. Remove the previously deployed versions of the applications that you have stopped in the previous step.

```
rmdir %CATALINA_HOME%\webapps\<General Self Service 9x Application>
```

4. Remove the associated WAR file for each of the applications removed in the previous step: `del %CATALINA_HOME%\webapps\`*`war_file_name`*.

Undeploy applications by using WebLogic

This task allows you to undeploy any previously deployed versions of Banner General Self-Service 9.x, Banner Personal Information 9.x, and Banner Direct Deposit 9.x on your systems by using WebLogic.

Procedure

1. Access the administration server by 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. If you have used Banner General Self-Service 9.x, Banner Personal Information 9.x, and Banner Direct Deposit 9.x, then select the check box for the versions of the applications that you want to undeploy.
5. Click **Stop**.
6. Click **Force Stop Now**.
7. In the **Force Stop Application Assistant** page, click **Yes**.
8. Select the check box of the application that you have stopped in the previous step.
9. Click **Delete**.
10. In the **Delete Application Assistant** page, click **Yes**.
11. In the **Change Center** frame, click **Activate Changes**.

Customize the WAR file

The release package must be unzipped and the WAR file will be customized for your institution.

The `release-BannerGeneralSsb-9.2.zip` release package must be moved to `$BANNER_HOME/general/java` subdirectory during the database upgrade.

Note: JDK 1.7 must be installed on your system.

Unzip the release package

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

Procedure

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-BannerGeneralSsb-9.2.zip`.
4. Transfer this file in binary mode using File Transfer Protocol (FTP) file into the temporary directory. For example:

   ```
   $HOME/ban9temp
   ```

5. Unzip `release-BannerGeneralSsb-9.2.zip` into the temporary directory.

Prepare the installer

To prepare the installer, perform the following steps.

Procedure

1. Change the directory to the installer directory:

   ```
   cd installer
   ```

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

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

   Example:

   ```
    ban9temp $ cd installer
   ```

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

The message `Build successful` confirms a successful build.

Install into the product home directory

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

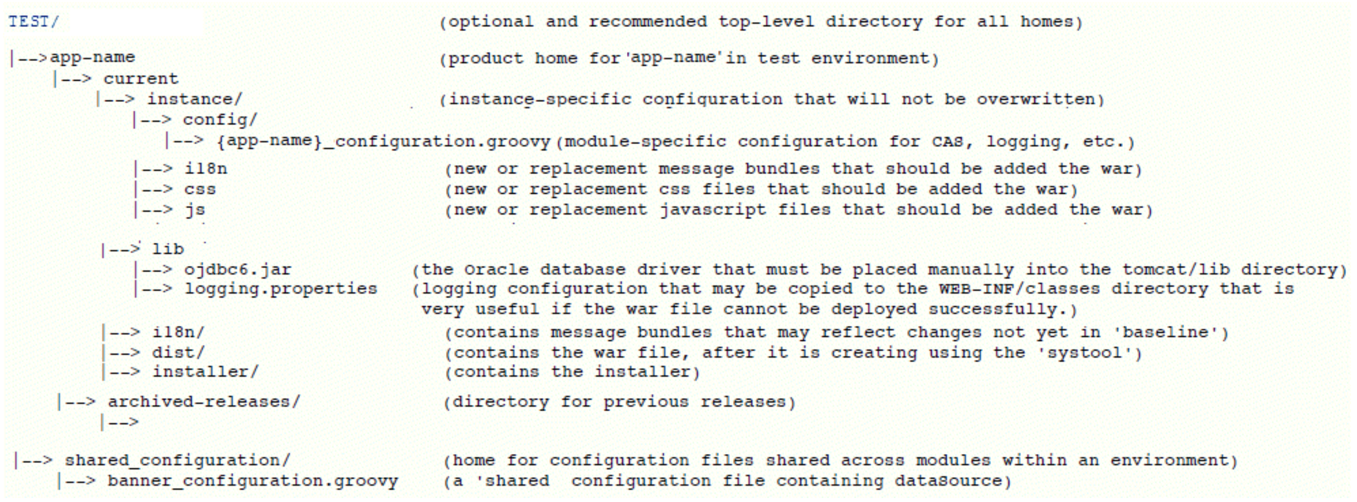
About this task

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

```
TEST/
|--> StudentRegistration/
|--> shared_configuration/
```

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

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



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

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

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

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

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

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

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

Procedure

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

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

   On Unix:

   ```
   $ bin/install home
   ```

   On Windows

   ```
   > bin\install home
   ```
3. When prompted, enter the full path of the application home directory.

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

   On Unix:

   ```
   []: Current_home_directory/banner_test_homes/BannerGeneralSsb
   ```

   On Windows:

   ```
   []: c:\banner_test_homes\BannerGeneralSsb
   ```
4. Enter the full path of the `shared_configuration` home directory.

   Banner 9.x applications that refer to this home directory share this configuration file.

   On Unix:

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

   On Windows:

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

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

Configure shared settings

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

JNDI datasource

You can 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. If you change the jndiName, you must regenerate the WAR file, even if you are using an external configuration.

Link to Self-Service Banner 8.x

To display existing Self-Service Banner 8.x menus and breadcrumbs in General Self-Service, 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/'`

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

To provide single sign on (SSO) to Banner 8.x, Jasig Central Authentication Service (CAS) http://www.apereo.org/cas(http://www.apereo.org/cas) is required as an external authentication provider. The following components must be configured to authenticate using CAS:

- General Self-Service application 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 SSO linking from the General Self-Service application 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` setting 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?
pgk=bwgkogad.P_SelectAtypView
```

Banner Self-Service now supports locale specific redirect to Banner 8x from General Self-Service. The following is an example of a Banner 8 SS Locale URL:

```
banner8.SS.locale.url =[
default : 'http://localhost:8002/DEFAULT/',
en : 'http://localhost:8002/EN/',
en_US : 'http://localhost:8002/enUS/',
en_AU : 'http://localhost:8002/enAU/',
en_GB : 'http://localhost:8002/enGB/',
en_IE : 'http://localhost:8002/enIE/',
en_IN : 'http://localhost:8002/enIN/',
fr : 'http://localhost:8002/FR/',
fr_CA : 'http://localhost:8002/frCA/',
pt : 'http://localhost:8002/PT/',
es : 'http://localhost:8002/ES/',
ar : 'http://localhost:8002/AR/',
]
```

Note: The following are important notes about the redirect from General Self-Service to Banner 8.x.

- If a locale specific URL does not exist, then it will use the default URL defined in the `banner8.SS.locale.url` map.
- For SPRIDEN users, if you do not define the `banner8.SS.url` setting with a valid URL, Banner 8.x menus and breadcrumbs are not displayed in the General Self-Service application.
- Without CAS and the SSO Manager, navigation to Banner 8.x is not seamless. A user must log in to Banner Self-Service 8.x using a valid user name and password. Navigation terminates at the Banner Self-Service 8.x main menu page.

Navigational MEP support from General Self-Service to Banner Self-Service 8.x

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

If your institution employs the use of MEP, key configuration changes and URL contexts must be updated accordingly. The key for the string must be in the following format: `mep.banner8.SS.url` and the key must be configured with the following map list: `mep.banner8.SS.url = ['GVU': 'http://<host_name>:<port_number>/<banner8>/SMPL_GVU', 'BANNER': 'http://<host_name>:<port_number>/<banner8>/SMPL_BANNER' ]`

The following is an example of a Banner 8.x SSO URL through the SSO Manager in a MEP environment: `mep.banner8.SS.url = ['GVU': 'http://e009070.university.com:8888/ssomanager/c/SSB?pkg=http://`

```
e009070.university.com:9020/SMPL_GVU/', 'BANNER': 'http://
e009070.university.com:8888/ssomanager/c/SSB?pkg=http://
e009070.university.com:9020/SMPL_BANNER/' ].
```

Use the following structure to support a locale specific redirect to Banner 8:

```
mep.banner8.SS.locale.url=[
      GVU:
               [
                    “default” : 'sampleUrlDefault ',
                    "ar": 'sampleUrlAR ',
                    "fr": 'sampleUrlfr',
                    "fr_CA": 'sampleUrlfrca'
               ],
      BANNER :
               [
                     “default” : 'sampleUrlDefault',
                     "ar": 'sampleUrlAR',
                     "fr": 'sampleUrlfr ',
                     "fr_CA": 'sampleUrlfrca'
               ]
]
```

Note: If a locale specific URL does not exist, then it will use the default URL defined in the `mep.banner8.SS.locale.url` map.

Session timeouts

When you determine the timeout settings for CAS and Banner 9.x applications, consider the time limits imposed for each.

For example, if the Banner 9.x timeout setting is less than the CAS timeout setting, SSO authentication might still be valid even though the user is exited from the Banner 9.x application. As long as the CAS authentication remains valid, the user can re-enter the Banner 9.x application without re-authenticating.

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

banner.transactionTimeout

The `banner.transactionTimeout` setting is used to prevent excessive delays due to long database transactions. The setting is configured in the `banner_configuration.groovy` file. To use this timeout, set `the banner.transactionTimeout` setting 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, the transaction 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, 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 setting` 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. This setting is used to set the overall session inactivity time. The `defaultWebSessionTimeout` setting can be configured in the `banner_configuration.groovy file`.

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

Role-based web session timeouts are configurable in Banner Web Tailor where `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`.

Location of photographs

The `banner.picturesPath` setting in the `banner_configuration.groovy` file specifies the location of student, faculty, and employee photographs.

In `banner { picturesPath = System.getProperty(‘base.dir’) + ‘/test/images’ }`, `banner.picturesPath` setting is the application base directory appended with `/test/images`.

The `base.dir` is a system property that can be set through `JAVA_OPTS`.

The value of `banner.picturesPath` must be a fully qualified path to the directory that contains the image files. For example, `banner { picturesPath = '/home/banner/test/images' }`.

Default photograph

The `banner.defaultPhoto` setting specifies the default photograph that is displayed if a photograph does not exist for a person.

Configure application-specific settings

Banner XE applications read configuration at startup from a general configuration file (`banner_configuration.groovy`) and from the General Self-Service configuration file (`BannerGeneralSsb_configuration.groovy`), which provides application-specific settings and can override settings in the general configuration file.

About this task

Applications often override general settings for logging and keeping application logs in a file separate from other applications. Some applications also define custom settings. For example, the Banner Student Advisor application allows configuration of roles that may access specific views within the application.

Procedure

1. Copy `BannerGeneralSsb_configuration.example` and change the name to `BannerGeneralSsb_configuration.groovy`.

   The installer creates the `BannerGeneralSsb_configuration.example` file.
2. Place the `BannerGeneralSsb_configuration.groovy` file in the `BannerGeneralSsb\current\instance\config` directory.

   This application-specific configuration file contains settings that you can customize for your specific environment.

JMX MBean name

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

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

In the example, the user needs to update `BannerGeneralSsb-log4j` identifier for each installation of each application. This allows the JMX management to distinguish between different installations of the application.

Location of the logging file

Log4j is the common logging framework used with applications that run on the Java Virtual Machine. You can configure the location at which the log file is saved.

For more information about the Log4j settings, see http://docs.grails.org/2.5.0/guide/conf.html.

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

The following example shows how to configure the location where the log file is saved:

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

The following example shows how to override the log file directory properties:

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

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

Logging level

The root logging level is pre-configured to the `ERROR` level. Multiple class or package level configurations, by default, are set to a status of `off`. You can set a different logging level for any package or class. However, configuration changes for logging take effect when the application is 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 `BannerGeneralSsb_configuration.groovy` file take effect after the application is restarted.

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

Related Links

Configure Java management extension on page 51

Customize a theme for the application

Update the variables banner.theme.url, banner.theme.name, and banner.theme.template to configure the application to use themes. The application is delivered with a template file located within the WAR file at `css/theme/BannerGeneralSsb.scss`. This file needs to be uploaded to the theme server to make full use of themes.

The configuration of themes is documented separately in section “Configure a Banner XE Self Service Application to use Themes” of the *Banner Extensibility Handbook*.

Institutional home page redirection support

The institutional home page redirection support configuration allows General Self-Service to provide an institutional home page to which users can navigate back to if they have access issues specific to insufficient privileges when accessing the General Self-Service application.

```
/****************************************************************
* Home Page link when error happens during authentication. *
****************************************************************/
grails.plugin.springsecurity.homePageUrl='http://URL:PORT/'
```

Proxied Oracle users

When connecting to Self-Service applications, the `ssbOracleUsersProxied` setting specifies whether the Oracle account is used for the database connections. The `ssbOracleUsersProxied` setting also controls whether Value Based Security (VBS) is enabled in the Self-Service application.

The following values can be used for the `ssbOracleUsersProxied` setting:

- `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 can still log in to the Self-Service application.
- `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 cannot log in to the Self-Service application.

Guest authentication

The **guestAuthenticationEnabled** setting determines how authentication should be handled when the application is configured for CAS or SAML sign-on.

When **guestAuthenticationEnabled** is true, the single sign-on authentication will be bypassed for API URLs which allows users to login using Basic authentication with just a username and password. When **guestAuthenticationEnabled** is false, the CAS and SAML protocols will be used for authentication with all URLs.

For Banner Print application

If this application's deployment is intended to be used with the Banner Print Appllication for printing pending job submission output, then **guestAuthenticationEnabled** must be set to true.

Logout URL

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

Provide 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.

Procedure

1. Use the `logout.afterLogoutUrl` setting to configure the logout URL.
2. Define the `logout.afterLogoutUrl` as follows: `logout.afterLogoutUrl='https://<CAS_HOST>:<PORT>/<cas>/ logout?url=http://myportal/main_page.html'`

Example

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

```
// +++ CAS CONFIGURATION +++
*set active = true if the application is configured with
 CAS SSO
************************************************************
banner {
 sso {
  authenticationProvider = 'cas' // Valid values
 are:'default', 'cas','saml'
  authenticationAssertionAttribute = 'UDC_IDENTIFIER'
  if(authenticationProvider != 'default') {

 grails.plugin.springsecurity.failureHandler.defaultFailureUrl=
 '/login/error'
  }
  if(authenticationProvider == 'saml') {
  grails.plugin.springsecurity.auth.loginFormUrl = '/saml/
login'}}
}
grails {
 plugin {
  springsecurity {
   cas {
    active = true
    serverUrlPrefix = 'http://CAS_HOST:PORT/cas'
    serviceUrl = 'http://BANNER9_HOST:PORT/APP_NAME/
j_spring_cas_security_check'
    serverName = 'http://BANNER9_HOST:PORT'
    proxyCallbackUrl = 'http://BANNER9_HOST:PORT/APP_NAME/
secure/receptor'
    loginUri = '/login'
    sendRenew = false
    proxyReceptorUrl = '/secure/receptor'
    useSingleSignout = true
    key = 'grails-spring-security-cas'
    artifactParameter = 'SAMLart'
    serviceParameter = 'TARGET'
    serverUrlEncoding = 'UTF-8'
    filterProcessesUrl = '/j_spring_cas_security_check'
   }
   logout {
    afterLogoutUrl = 'https://cas-server/logout?url=http://
myportal/page.html'
   }
  }
 }
}
```

Provide only a redirect URL

With this method of handling logouts, users automatically go to a redirect URL.

Procedure

1. Configure logout information, replacing `url` with `service` as follows: `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` as follows: `<bean id="logoutController" class="org.jasig.cas.web.LogoutController" p:centralAuthenticationService-ref="centralAuthenticationService" p:logoutView="casLogoutView" p:warnCookieGenerator-ref="warnCookieGenerator" p:ticketGrantingTicketCookieGeneratorref=" 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

If your environment is enabled for Multi-Entity Processing (MEP), two settings (`grails.plugin.springsecurity.logout`) must be configured in the `BannerGeneralSsb_configuration.groovy` configuration file for your environment. 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 on the **Error** page of the application and the user can use the **Logout** or **Return Home** button.

The following 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.

  For example, a portal page has the following settings:

  - `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.

The installation of General Self-Service 9.2 includes adding a configuration to enable Action Item Processing. Clients in a MEP environment will need to be sure that this configuration is present for all VPDI codes and not just the default one.

- On the GUAAPPL page (table GENERAL.GUBAPPL), add the following:
  – Application ID = GENERAL_SS
  – Application name is the app.name that is part of the application.properties file.
- On the GUACONF page (table GENERAL.GUROCFG), add the following:
  – Configuration name = BCMLOCATION
  – Configuration type = string
  – Configuration value is the BCM location you specified for your installation in the `configuration.groovy` file

Theme Editor tool

Use the Theme Editor tool to apply color theme and logo changes to Banner applications.

You can use the following settings to enable the Theme Editor.

Setting	Description
`banner.theme.url="<UPDATEME>"`	The URL of the application hosting the Theme Server.
`banner.theme.name="<UPDATEME>"`	The name for the theme. **Note:** In a Multi-Entity Processing (MEP) environment, the application uses the MEP code as the theme name instead of `banner.theme.name`. You must create a theme with the same name in the Theme Editor on the server specified by `banner.theme.url`.
`banner.theme.template= "BannerGeneralSsb"`	The scss file containing the theme settings in the WAR file.
`banner.theme.cacheTimeOut=120`	Indicates how long to cache the CSS file that was generated using the template and theme. This is an advanced setting that you can use when the application is configured to be the host of the theme server.

After you enable the Theme Editor, the Preview section displays an example of how the theme settings might look. Your theme choices could display differently in some applications. You can configure the following theme parameters in the Theme Editor section.

Parameter	Description
Theme name (Required)	The name is the theme's primary identifier. Create a recognizable name for the theme, for example, an institution name or abbreviation. If your institution is MEP-enabled, you may want to use a MEP code as a theme name. This gives you the ability to create code themes based on MEP code. If you change the name of an existing theme and click **Save**, this creates a new theme.
Primary color	Defines the color of the main title bar. Use the drop down or enter the hex code of the color.

Parameter	Description
	Theme Editor generates several related shades for borders, backgrounds, and text based on your color choice. View the Preview section to see your choices. It is recommended that you choose colors that coordinate with your institution logo.
Secondary color	Defines color of other page elements related to the Primary color.
Accent color	Defines additional color for various page elements. As you modify the primary and secondary colors, the Theme Editor automatically determines an accent color that contrasts with the primary and secondary colors. If you prefer, you can specify the Accent color.
Logo URL	The URL of the to include in pages. Create the logo as an SVG file so that it can scale to fit the page The URL must be accessible to application clients because users' browsers will load the logo from this URL.
Save Theme	Saves the theme based on the current Theme name. The theme is saved in the file system of the application server hosting the Theme Editor and is available to all applications. When you click **Save Theme**, the theme is applied to the current page.
New Theme	Clears the theme settings. You can also create a new theme by copying an existing one, changing the Theme Name, and clicking **Save Theme**.
As you create themes, they display in the **Saved Themes** section where you can perform the following functions.	
Load	Loads the theme into the Theme Editor and applies the theme to the current page.
Delete	Deletes the theme from Theme Server. This action cannot be undone.
JSON	Links to the JSON values that encode the theme. This allows you to save the JSON to another location.
CSS	Links to the CSS file generated for the theme. This lets you save the CSS to another location and optimize performance by configuring

Parameter	Description
	applications to use the static theme. However, if an application uses a statically saved theme file, you must manually update the static file with any theme changes.

Google Analytics

To enable Google Analytics, use this configuration in the configuration.groovy file.

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

Example:

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

Use cases:

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

Action Item Processing configuration

Action Item Processing provides you with the ability to create Action Items for which a specific population needs to respond.

You can schedule when the population will be notified and track the status of the responses. Users that have action items that require a response before they can proceed with their task will be prompted with a notice and redirected to the Action Items page. For more information about Action Item Processing, see the *Banner General Self-Service Handbook 9.2*.

Enable Banner Page Builder for Action Item Processing

To use Action Item Processing, you must enable Banner Page Builder. Requests and responses to and from the user interface are managed by Page Builder.

The Action Item Processing page for non-administrative users uses Page Builder to load the details of an action item. To enable Page Builder, use this configuration in the configuration.groovy file.

```
pageBuilder.enabled = true
if (!pageBuilder.enabled) {
  grails.plugin.springsecurity.securityConfigType =
 grails.plugin.springsecurity.SecurityConfigType.InterceptUrlMap
}
// initial load location on file system. Files located in "pb"
 directory. pb directory located at root of app as sibling to the
 config files
pbRoot = "<UPDATE_ME>" // Example /temp/pb
pageBuilder {
    locations {
      bundle        = "${pbRoot}/i18n"
      page          = "${pbRoot}/page"
      css           = "${pbRoot}/css"
      virtualDomain = "${pbRoot}/virtdom"
    }
    // Uncomment debugRoles to reveal detailed SQL error messages for
    // Virtual domains to users with any of the comma separated roles
    // debugRoles = "ROLE_GPBADMN_BAN_DEFAULT_PAGEBUILDER_M"
}
```

`pageBuilder.enabled`	This will enable Page Builder in the application. For Action Item Processing, this property must always be *True*. Do not change this value.
`pbRoot`	The directory where the Page Builder status configuration is located. On the initial load of the application server, the Page Builder content gets loaded to your Banner database in the CSS, VIRTUAL_DOMAIN and PAGE tables (all owned by SSPBMGR). This information will be used while displaying the action item details and will allow the user to take appropriate action on action items.

Note:

- Create the folders (if not present) as specified under `pageBuilder.locations` on the application server before starting the application.
- The locations are using variable pbRoot to avoid repetition of the root directory in the locations.
- The application server needs to have read/write access in the folders.
- The directories specified under locations are used for import and export of applications built with Page Builder and for storing PageBuilder resource bundles.
- The configured locations must exist in the file system before starting the application.

Copy the contents of pb folder which is present in the release package into the folders created under the pbRoot location. Ensure this action is performed before the war file is deployed to the application server.

Enable Banner Communication Management (BCM) for Action Item Processing

If you are using Action Item Processing functionality, you need to enable Banner Communication Manager. Use this configuration in the configuration.groovy file.

```
BCMLOCATION='http://<HOST_NAME>:<PORT>/CommunicationManagement' // The
 URL of BCM application.


BANNER_AIP_BLOCK_PROCESS_PERSONA= ['EVERYONE', 'STUDENT', 'REGISTRAR',
 'FACULTYINSTRUCTOR', 'FACULTYADVISOR', 'FACULTYBOTH']

BANNER_AIP_EXCLUDE_LIST='aipActionItemPosting|aipAdmin|aip|
aipPageBuilder|BCM|about|cssManager|cssRender|error|excelExportBase|
dateConverter|keepAlive|login|logout|resetPassword|securityQa|
selfServiceMenu|survey|test|theme|themeEditor|userAgreement|
userPreference'
ssconfig.app.seeddata.keys = [['BCMLOCATION'],
 ['BANNER_AIP_BLOCK_PROCESS_PERSONA']]

GENERALLOCATION='http://<HOST_NAME>:<PORT>/BannerGeneralSsb' // Example
 localhost:8080/BannerGeneralSsb
```

Refer to the “User Interface” section of the *Banner General Self-Service Handbook* for more information about how to set up users.

`BCMLOCATION`	The URL of Banner Communication Management application. Banner Communication Management (BCM) application is required for Banner General SS 9.2 for Action Item processing only. You will need to provide the location for your installation. //Example localhost:8080/ CommunicationManagement
`BANNER_AIP_BLOCK_ PROCESS_PERSONA`	This list of personas will be available when associating an action item to process; this is applicable for only certain processes. Do not make changes to this list.
`BANNER_AIP_ EXCLUDE_LIST`	Action Item Processing allows for processes to be halted until an action item is resolved by a user. However, some functionality must be exempt from halting. This is the list of grails controllers that will be exempt.

	Do not make changes to this list.
GENERALLOCATION	This is the url to General Ssb location. This configuration is mandatory for client applications that use Action Item Gate Keeping filters.
ssconfig.app.seeddata.keys	This list will ensure to synchronize the configuration to Banner GURACFG. Any dynamic change can be made through Admin page GUACONF for the **GENERAL_SS** application.

Enable Quartz Jobs settings for Action Item Processing

For Action Item Posting jobs, it is required to set the following Quartz settings.

```
configJob.delay = 60000 // (In Seconds)
configJob.interval = 120000 // (In Seconds)
configJob.actualCount = -1
aip {
    weblogicDeployment = false

    actionItemPostMonitor {
        enabled = false
        monitorIntervalInSeconds = 10 //(In Seconds)
    }

    actionItemPostWorkProcessingEngine {
        enabled = false
        maxThreads = 1
        maxQueueSize = 5000
        continuousPolling = true
        pollingInterval = 2000 //(In Seconds)
        deleteSuccessfullyCompleted = false
    }

    actionItemJobProcessingEngine {
        enabled = false
        maxThreads = 2
        maxQueueSize = 5000
        continuousPolling = true
        pollingInterval = 2000 //(In Seconds)
        deleteSuccessfullyCompleted = false
    }

    scheduler {
        enabled = false
        idleWaitTime = 30000 //(In Seconds)
    }
}
```

weblogicDeployment	Set this to *True* to run Action Item postings and if application is deployed on weblogic.

	This must be set to True when you deploy to weblogic, or the posting will fail.
`actionItemPost Monitor.enabled`	Set this to *True* to run Action Item postings. After enabled, this monitors the Quartz jobs.
`actionItemPostWork ProcessingEngine.enabled`	Set this to *True* to run Action Item postings. After enabled, it starts the Action Item work processing engine.
`actionItemJob ProcessingEngine.enabled`	Set this to *True* to run Action Item postings. After enabled, it starts the Action Item work processing engine.
`scheduler.enabled`	Set this to *True* to run Action Item postings.

Configure application-specific settings by modifying messages.properties file

To configure application-specific settings, in addition to modifying the groovy file, you can also modify the `messages.properties` file, CSS, or JavaScript files within the WAR file. Some of the application-specific settings that you can configure are the name format, date format, time format, multiple calendars, institution name, and java script.

Change the name format

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

About this task

The login user name shows the person’s name in the format First Middle Last.

Procedure

1. In the `BannerGeneralSsb/current/i18n` folder, find the `messages.properties` file for your language.

   The `messages.properties` file is located at `WEB-INF/grails-app/i18n/messages.properties` within the war file.

   The default for American English is the `messages.properties` file.
2. Open the file using a text editor.
3. Add an entry for the `default.name.format` setting, using the following elements: First name: `$firstName`, Middle name: `$mi`, Last name `$lastName`, and Surname prefix: `$surnamePrefix`.

   For example:

default name format can be changed as follows: `default.name.format=$surnamePrefix $firstName $mi $lastName`

Format name

You can change the display of the user's name in the Overview Section using the rules setup on the Name Display Rules Form (GUANDSP).

Procedure

1. Navigate to the Name Display Rules Form (GUANDSP) in INB.
2. Under the Hierarchy tab, find the entry for the Personal Information Overview Section.

   It has the following values:

   - Product: General
   - Application: GeneralSsb
   - Page: PersonalInformation
   - Section: Overview
3. In the Personal Information entry, change the Usage to the desired name display rule.

   See *Banner General User Guide 8.8.5* for more information on creating name display rules.

Format addresses

You can configure up to eight lines of the addresses, but you must include at least the first four lines that are delivered as the default.

About this task

Separator values such as comma (,) and period (.) must not be used if the preceding address attribute can be null.

Procedure

1. In the `BannerGeneralSsb/current/i18n folder`, find the messages.properties file for your language. The `messages.properties` file is located at `WEB-INF/grails-app/i18n/messages.properties` within the war file. The default for American English is the `messages.properties` file.
2. Using the following elements, edit the `default.personAddress.line`*`x`*`.format` properties to define how employee addresses are displayed.

   If data is not required on the fourth line of the address, then the following format must be used:

   ```
   default.personAddress.line4.format=
   ```

Component	Element
House number	$houseNumber
Street line 1	$streetLine1
Street line 2	$streetLine2
Street line 3	$streetLine3
Street line 4	$streetLine4
City	$city
State or province	$state
ZIP or postal code	$zip
County	$county
Nation	$country

For example, you can format the employee addresses as follows:

```
default.personAddress.line1.format=$streetLine1
default.personAddress.line2.format=$streetLine2
default.personAddress.line3.format=$city
default.personAddress.line4.format=$state $zip
```

3. Rebuild the application WAR file to include your customizations.
4. Redeploy the WAR file to your application server.

Change the phone format

The default phone number format is locale specific. The default phone format for U.S. English is `default.personTelephone.format=$phoneCountry $phoneArea $phoneNumber $phoneExtension`.

Procedure

1. In the `BannerGeneralSsb/current/i18n` folder, find the `messages.properties` file for your language.

   The `messages.properties` file is located at `WEB-INF/grails-app/i18n/messages.properties` within the war file.

   The default for American English is the `messages.properties` file.
2. Open the file using a text editor.
3. Add an entry for the `default.personTelephone.format` setting, using the following elements: Area code: `$phoneArea`, Phone number: `$phoneNumber`, Extension: `$phoneExtension`, Telephone country code: `$phoneCountry`, and International access code: `$phoneInternational`.

For example, `default.personTelephone.format=$phoneCountry $phoneArea $phoneNumber $phoneExtension`.

Change the date format

The default date format is locale specific. The default format for U.S. English is `MM/dd/yyyy` and the `js.datepicker.dateFormat` is `mm/dd/yyyy`. Date formats are case sensitive.

Procedure

1. In the `BannerGeneralSsb/current/i18n` folder, find the `messages.properties` file for your language.

   The `messages.properties` file is located at `WEB-INF/grails-app/i18n/messages.properties` within the war file.

   The default for American English is the `messages.properties` file.
2. Open the file using a text editor.
3. Customize the `default.date.format` and `js.datepicker.dateFormat` keys in the `messages_<ISO_language_code>_<ISO_country_code>.properties` file located in the `BannerGeneralSsb\current\i18n` directory.

Related Links

Date format keys on page 41

Date format keys

The `default.date.format` and `js.datepicker.dateFormat` are date format keys that use different specifications to represent the date. The `default.date.format` is associated with the ICU format, which is `dd/MM/yyyy`. The `js.datepicker.dateFormat` is associated with the Java Script or Keith Wood format, which is `dd/mm/yyyy`.

For dates to be displayed properly, the two formats must match. For example, for 1 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.

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 about the ICU specification, see http://userguide.icu-project.org/formatparse/datetime.

For the `default.date.format` for June 1, 2014, use one of the following variables for the year:

Year format	Interpretation	Comment
yy	14	Two digit year
yyyy	2014	Four digit year

For the `default.date.format` for June 1, 2014, 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, 2014, 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

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 the Keith Wood datepicker format, see http://keith-wood.name/datepick.html.

For the `js.datepicker.dateFormat` for June 1, 2014, use one of the following variables for the year:

Year format	Interpretation	Comment
yy	14	Two digit year
yyyy	2014	Four digit year

For the `js.datepicker.dateFormat` for June 1, 2014, 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, 2014, 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

Display multiple calendars

Customization of multiple calendars is implemented for the Arabic language (AR). You can customize and display multiple calendars by using a set of `calendar` keys.

Procedure

1. In the `BannerGeneralSsb/current/i18n` folder, find the `messages.properties` file for your language.

   The `messages.properties` file is located at `WEB-INF/grails-app/i18n/messages.properties` within the war file.

   The default for American English is the `messages.properties` file.
2. Open the file using a text editor.
3. To display multiple calendars in the application, use the following keys in the `messages_ar.properties` file: `default.calendar`, `default.calendar1`, and `default.calendar2`.

   The `default.calendar2` is optional.

   For example:

   By using the following keys, you can 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=arabic`

Customize CSS

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

Procedure

1. Create a CSS file `BannerGeneralSsb/current/instance/css/bannerSelfService-custom.css` containing the custom CSS directives.
2. If the `BannerGeneralSsb/current/instance/css/images` directory structure does not exist, create the directory structure.
3. If you want to provide custom images, save the images in the `BannerGeneralSsb/current/instance/css/images` directory, and in the CSS, specify their paths as a relative URL from the CSS directory.

   For example:

   An image in `css/images/institution-logo.svg` can have the following CSS rule: `background-image: url(images/institution-logo.svg);`.

Change the institution logo

The default layout includes an institutional branding area that displays the institution logo. To customize the system or university name, you must provide a custom CSS file to override the default styling and a replacement logo image.

About this task

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;}
```

Procedure

1. To override the default image, create a CSS file named `BannerGeneralSsb/current/instance/css/bannerSelfService-custom.css`.
2. In the CSS file you have created, enter `.institutionalBranding {background-image: url("./images/institutionLogo.png");}`.
3. Replace the logo image in the following directory: `BannerGeneralSsb/current/instance/css/images/institutionLogo.png`.
4. To deploy your updates, you must rebuild and redeploy the WAR file.

Related Links

Regenerate the WAR file on page 47

Customize JavaScript

You can customize the JavaScript to modify the behavior of the web pages of your application by placing your JavaScript file named bannerSelfService-custom.js in the following location: BannerGeneralSsb/current/instance/js/bannerSelfService-custom.js.

Procedure

1. Create a JavaScript file, `current/instance/css/bannerSelfService-custom.js`.
2. In the JavaScript file you have created, enter the custom JavaScript code.
3. To deploy your updates, you must rebuild and redeploy the WAR file.

Change the disclaimer text

The disclaimer text is now located in the `messages.properties` file for the Banner General Direct Deposit plugin. The `messages.properties` file for this plugin can be found in the following path in the WAR file: `WEB-INF/plugins/banner-general-direct-deposit-ui-0.1/grails-app/i18n`.

Procedure

1. Copy the WAR file of the consolidated application `BannerGeneralSsb-9.2.war` to a working folder.
2. At a command prompt, run the following JAR command to inflate the contents of the WAR to the current folder.

   ```
   jar -xvf BannerGeneralSsb-9.2.war
   ```
3. Delete the existing WAR file.
4. Locate the appropriate `messages.properties` file found in the `WEB-INF/plugins/banner-general-direct-deposit-ui-0.1/grails-app/i18n` folder.

   According to language, the `messages.properties` files are:

   ```
   messages.properties (US English)
   messages_ar.properties
   messages_en_AU.properties
   messages_en_GB.properties
   messages_en_IE.properties
   messages_en_IN.properties
   messages_es.properties
   messages_fr.properties
   messages_fr_CA.properties
   messages_pt.properties
   ```
5. Edit the `directDeposit.disclaimer.text` line in the appropriate `messsages.properties` file.

Note: If you need to add a line break, see Insert a line break in the disclaimer text on page 46.

6. At a command prompt, run the following JAR command. This command will bundle all the files and folders in the current folder.

   ```
   jar –cvf BannerGeneralSsb-9.2.war
   ```

Results

The new WAR file is generated and ready for deployment.

Insert a line break in the disclaimer text

Additional steps are required to insert a line break into disclaimer text.

Procedure

1. Find this line: `directDeposit.disclaimer.text=By checking this box, I authorize the institution to initiate direct credits or debits on my behalf` , and change it to `directDeposit.disclaimer.text=By checking this box, I authorize the institution \n to initiate direct credits or debits on my behalf`

   Note: The "\n" is the carriage return being added to the text.

2. Create or edit the file `BannerGeneralSsb/current/instance/css/bannerSelfService-custom.css` and place these following details in it.

   ```
   #disclaimerTxt {
      white-space: pre-wrap;
   }
   ```

3. To deploy your updates, you must rebuild and redeploy the WAR file.

Results

For this example, your text should now be displayed as,

“By checking this box, I authorize the institution

to initiate direct credits or debits on my behalf.”

Regenerate the WAR file

After the shared and application-specific configurations are complete, the application WAR file can be regenerated to include your customizations and application-specific settings. The WAR file can then be deployed into your specific application server. The `systool` is used to create the WAR file.

About this task

Application uses the configuration files in the WAR file unless you override them by specifying the environment variable. For example, you can override the location of the `banner_congifuration.groovy` file by setting the environment variable as follows: `BANNER_APP_CONFIG=/path/to/banner_configuration.groovy`

Procedure

1. Change your current working directory to the product home directory: `BannerGeneralSsb/current/installer`.
2. Run the `ant` command which builds the `systool` module.

   For UNIX systems, ensure that the `ant` file is executable. For example,

   ```
   chmod +x ant
   ```

   For example,

   ```
   $ cd BannerGeneralSsb/current/installer

   $ ./ant
   ```

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

Operating system	Command
UNIX	`$ bin/systool war`
Windows	`> bin\systool war`

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

Configure the web application server and deploy the WAR file

You can configure a web application server such as Tomcat or WebLogic and deploy the WAR file on the web application server. It is recommended that you can secure the web application traffic by using standard TLS encryption, which is supported by the application server software.

For more information about enabling HTTPS support for web applications, see your application server documentation.

Related Links

Configure the Tomcat server on page 48
Configure the WebLogic server on page 55
Regenerate the WAR file on page 76

Configure the Tomcat server

You can configure the Tomcat server and then deploy the WAR file to the Tomcat server.

Before you begin

Ensure that you download and install either Tomcat 7 or Tomcat 8 versions. For more information about downloading and installing the Tomcat server, see http://tomcat.apache.org.

About this task

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

Procedure

1. Locate the Oracle JDBC jar files, `ojdbc6.jar` and `xdb6.jar` in the `BannerGeneralSsb\current\lib` directory.

   The account that runs the Tomcat application server must configure environment settings to support the application.
2. On a Linux system, ensure that `CATALINA_HOME` is defined to reference your Tomcat software installation location.

   Note: This step must be executed on a Linux system only.

   For example, in the `CATALINA_HOME=/opt/apache-tomcat-8.0.xx`, `xx` indicates the point version of Tomcat you have installed.
3. Define `CATALINA_OPTS` to configure the following JVM settings: `CATALINA_OPTS=-server -Xms2048m -Xmx4g -XX:MaxPermSize=512m`

Note: If you are deploying multiple Banner 9.x applications to the same Tomcat server, `-Xmx` must be increased by `2g` and `-XX:MaxPermSize` must be increased by `128m`. You should deploy Banner 9.x administrative applications to one Tomcat server instance and General Self-Service application to a separate Tomcat server instance.

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

4. **Optional:** If you install Tomcat as a Windows service, few JVM arguments must be specified.
   a) From the Windows **Start** menu, select **Configure Tomcat** application.
   b) Select the **Java** tab.
   c) In the **Java Options** field, add `-XX:MaxPermSize=384m`.
   d) Set the initial memory pool to 2048.
   e) Set the maximum memory pool to 4096.
   f) Save the settings.
   g) Restart the Tomcat Windows service.
5. **Optional:** If you want to monitor or debug the application, enable remove Java Management Extensions (JMX).

   Configure Java management extension on page 51
6. Define the JNDI datasource resource name for the application.
   a) Edit `$CATALINA_HOME/conf/context.xml`.
   b) 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="" />
```

   c) Add the 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"/>
```

d) Save your changes in `context.xml`.

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

f) Add the Resource definitions inside the `<GlobalNamingResources>` element.

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"/>
```

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

g) Save your changes in `server.xml`.

h) Copy the Oracle JDBC jar file `(ojdbc6.jar and xdb6.jar)` from the `BannerGeneralSsb/current/lib` directory to the `$CATALINA_HOME/lib` directory.

i) Start the application server, `$CATALINA_HOME/bin/startup` to validate the configuration of the Tomcat server.

For example: for Linux,

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

For Windows,

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

j) Browse `http://servername:<port>`.

Example

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 -DBANNER_GENERAL_SSB_CONFIG=/PRODUCT_HOME/BannerGeneralSsb/current/instance/config/BannerGeneralSsb_configuration.groovy"`.

Related Links

Configure the web application server and deploy the WAR file on page 48

Configure Java management extension

This is an optional step that enables you to monitor or debug the application.

About this task

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.

Procedure

1. Add the following options to the `catalina.sh` file and 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` to use port 8999.

   Ellucian recommends 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. For more information, 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. You should deploy 9.x administrative applications and General Self-Service application to separate Tomcat servers to increase performance.

About this task

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

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

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

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

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

Username/password combinations are configured in your Tomcat user database `<TOMCAT_HOME>\conf\tomcat-users.xml`. For Tomcat 8.x, you must configure at least one username/password combination with the manager role. For example: `<user username="tomcat" password="tomcat" (your password) roles="manager-gui, manager"/>`.

Note: The roles in Tomcat server changed between point releases in version 8.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.

Procedure

1. Navigate to the `BannerGeneralSsb\current\installer` directory.
2. Deploy Tomcat.

Operating system	Instruction
Unix	`$ bin/systool deploy-tomcat`
Windows	`> bin\systool deploy-tomcat`

3. Enter the URL, `[]: http://localhost:8080/manager` for the Tomcat 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.

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

   For example: `[]: tomcat`
5. Enter the Tomcat password for the user:

   This password will not be persisted.

   For example: `[]: password`
6. Access the web application `http://servername:<port>/BannerGeneralSsb`.

WebLogic server

To configure the web application and deploy the WAR file to the WebLogic server, you must perform certain tasks such as verify certain WebLogic prerequisites, set up the cookie path for WebLogic installation, create a WebLogic machine, and create a WebLogic server.

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

Verify WebLogic prerequisites

Ensure that the WebLogic prerequisites are verified before configuring your WebLogic server.

Procedure

1. Ensure that WebLogic is installed.

   WebLogic can be downloaded and installed from the Oracle web site.
2. Ensure that the minimum requirement for OFM is 11.1.1.7 or 11.1.1.9 with WebLogic 10.3.6.

   Banner 9 web applications are also supported on OFM WebLogic 12c (12cR1, version 12.1.x).
3. Ensure that both the WebLogic node manager and the administration server are started.

The administration server can be accessed using the following URL, `http://server:7001/console`.

Set up the cookie path

For a WebLogic installation, the cookie path needs to match the location where the application is deployed. Otherwise, the cookies will not be found by the application. If this change is not made, users will be prompted to log in each time they switch between applications.

About this task

Add the following in the `weblogic.xml`:

```
<wls:session-descriptor>
<wls:cookie-path>/<WebApp_Root_Context></wls:cookie-path>
</wls:session-descriptor>
```

Procedure

1. On the Shell command line, in a directory containing the WAR file, enter `jar xvf application_context.war WEB-INF/weblogic.xml`.
2. Enter `vi WEB-INF/weblogic.xml`
3. Add the code listed above for the cookie path to the appropriate place in the file.
4. Enter `jar uvf application_context.war WEB-INF/weblogic.xml`.
5. Redeploy the application.

Example

WebLogic file

```
<?xml version="1.0" encoding="UTF-8"?>
<wls:weblogic-web-app
xmlns:wls="http://www.bea.com/ns/weblogic/weblogic-web-app"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://java.sun.com/xml/ns/javaee
 http://
java.sun.com/xml/ns/javaee/web-app_2_5.xsd http://
www.bea.com/ns/
weblogic/weblogic-web-app http://www.bea.com/ns/weblogic/
weblogic-webapp/
1.0/weblogic-web-app.xsd">
<wls:weblogic-version>10.3.0</wls:weblogic-version>
<wls:container-descriptor>
<wls:prefer-web-inf-classes>true</wls:prefer-web-inf-
classes>
<wls:show-archived-real-path-enabled>true</wls:show-
archived-real-pathenabled>
</wls:container-descriptor>
<wls:session-descriptor>
```

```
<wls:cookie-path>/application_context</wls:cookie-path>
</wls:session-descriptor>
</wls:weblogic-web-app>
```

Create a WebLogic machine

For configuring the WebLogic server, a WebLogic machine must be created. You need not create a WebLogic machine definition again if you have already created a machine earlier.

Procedure

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**.

Configure the WebLogic server

You can configure the WebLogic server and then deploy the WAR file to the WebLogic server. If you previously created a WebLogic server for the application, you can use the same server.

Procedure

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.
   You can have the server name as Banner9-SS and server listen port as 8180.
6. Click **Finish**.
7. Click the newly created server link.
8. In the **General** tab, assign the machine to this server.
9. Click **Save**.
10. Select the **Server Start** tab.
11. Add the following to the **Arguments** text area: `-server -Xms2048m -Xmx4g -XX:MaxPermSize=512m`.

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

a) To override the configuration that was added into the WAR file, you can set system properties to point to external configuration files by appending the following to the **Arguments** text area: `-DBANNER_APP_CONFIG=<full file path to banner_configuration.groovy> -DBANNER_GENERAL_SSB_CONFIG=<full file path to BannerGeneralSsb_configuration.groovy>`.

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

Related Links

Configure the web application server and deploy the WAR file on page 48

Configure weblogic.xml file to make Banner 9.x JSession cookie secure

To make the Banner 9.x JSession cookie secure, certain configuration information must be added to the `weblogic.xml` file. This configuration is only specific to WebLogic server.

About this task

The configuration changes should be added based on specifications at your institution. After the cookie has been secured, the same application cannot be accessed through a non-SSL port. These configuration changes should only be applied if the SSL is in use.

Procedure

Add the following configuration information to the `weblogic.xml` file to make the Banner 9.x JSession cookie secure: `<wls:session-descriptor> <wls:cookie-secure>true</wls:cookie-secure> <wls:url-rewriting-enabled>false</wls:url-rewriting-enabled> </wls:session-descriptor>`.

Update Oracle JDBC JAR files on the WebLogic server

WebLogic releases include specific versions of the JDBC JAR files, but the versions used are those that ship with the Banner application release.

About this task

For more information about JDBC JAR files, see https://docs.oracle.com/middleware/1212/wls/JDBCA/third_party_drivers.htm#JDBCA231.

Procedure

1. Copy the Oracle JAR files (`ojdbc6.jar` and `xdb6.jar`) from the `$PRODUCT_HOME/current/lib` directory to the `$MIDDLEWARE_HOME/modules` directory.
    - `$PRODUCT_HOME` is where the application's release zip file is unpacked and installed.
    - `$MIDDLEWARE_HOME` is the location where Oracle WebLogic is installed.
2. For Linux or Unix servers, edit the `setDomainEnv.sh` file under the `$MIDDLEWARE_HOME/user_projects/domains/<CUSTOM_DOMAIN>/bin` folder and update the `ADD EXTENSIONS` comment with additional information.

   For example:

   ```
   #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"
   ```

   If you plan to copy and paste the configuration settings into the `setDomainEnv.sh` file, ensure that 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 log files, there might have been a typo when you edited the `setDomainEnv.sh` file and the `xdb6.jar` or `ojdbc6.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 update the `ADD EXTENSIONS` comment with additional information.

   For example:

   ```
   @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
   ```

   If you plan to copy and paste the configuration settings into the `setDomainEnv.cmd` file, ensure that 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, there might have been a typo when you edited the `setDomainEnv.cmd` file and the `xdb6.jar` or `ojdbc6.jar` file cannot be found during Application startup.

4. Restart the WebLogic Managed Server.

Create an administrative datasource and connection pool

You can use this task to configure an application's connection to the Oracle database for administrative users. If you have already created an administrative datasource and connection pool, you need not create this again.

Procedure

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 datasource name.
   For example, `Banner9DS`.
6. Specify the JNDI name.
   For example: a JNDI name can be `jdbc/bannerDataSource`.
7. Specify `Oracle` for Database Type and then click **Next**.
8. Select **Oracle Driver (Thin) for Service Connections** and then click **Next**.
9. Clear the **Supports Global Transactions** check box and then click **Next**.
10. Enter the database name, host name, port, user name, password, and password confirmation, and then click **Next**.
    For example:

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

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

a) Set the Initial Capacity parameter to specify the minimum number of database connections to be created when the server starts up.
   For example: Initial Capacity = 5
b) Set the Maximum Capacity parameter to specify the maximum number of database connections that can be created.
   For example: Maximum Capacity = 100
c) Scroll down and click **Advanced** to show the advanced connection pool options.
d) Clear the **Wrap Data Types** check box to disable wrapping.

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

You can use this task to configure an application's connection to the Oracle database for self-service users. If you have already created a self-service datasource and connection pool, you need not create this again.

Procedure

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 datasource name.
   For example, `Banner9SsbDs`.
6. Specify the JNDI name.
   A JNDI name can be `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**.

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.
    a) Set the Initial Capacity parameter to specify the minimum number of database connections to be created when the server starts up.
       Initial Capacity = 5
    b) Set the Maximum Capacity parameter to specify the maximum number of database connections that can be created.
       Maximum Capacity = 100
17. Change `Statement Cache Type = LRU`.
18. Change `Statement Cache Size = 20`.
19. Click **Save**.
20. In the **Change Center** frame, click **Activate Changes**.

Configure server communication

Configure the application server to use the administrative and self-service data sources.

Procedure

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

The WebLogic server uses the deployed WAR file and implements the code in the WAR file by starting the application.

Procedure

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

   ```
   BannerGeneralSsb/current/dist/
   BannerGeneralSsb-9.2.war
   ```

   To

   ```
   BannerGeneralSsb/current/dist/
   BannerGeneralSsb.war
   ```

2. Access the administration server at `http://server:7001/console`.
3. In the **Domain Structure** frame, select the **Deployments** link.
4. In the **Change Center** frame, select **Lock and Edit**.
5. Click **Install**.
6. Select the WAR file to be deployed and then click **Next**.
   The file is located at `BannerGeneralSsb/current/dist`.
7. Select **Install this deployment** as an application and then click **Next**.
8. Select the target server on which to deploy this application and then click **Next**.
   For example, `Banner9-SS` is a target server.
9. Click **Finish**.
10. In the **Change Center** frame, click **Activate Changes**.
11. Select the deployed application and then click **Start**.
12. Select **Servicing all request**.
13. Access the application at `http://servername:<port>/<web application>`.
    For example: `http://localhost:8080/BannerGeneralSsb`.
14. Log in to the application using a valid username and password.

Install Banner General Self-Service for SAML 2.0 SSO

This section details the installation steps for the Banner General Self-Service 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

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

Perform the following steps to create a keystore (`*.jks`) file.

Procedure

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).
   a) Fill in the prompts for your organization information.
   b) When asked for your first and last name, enter the domain name of the serverthat 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 generalkeystore.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 a generalkeystore.jks file containing a private key and your self-signed certificate.

3. Place this file in the location specified in the following key value:
   "grails.plugin.springsecurity.saml.keyManager.storeFile"

Create a service provider file

Perform the following steps to create a service provider file.

Procedure

1. Create a file `banner-<short-appName>-sp.xml` at the folder path mentioned in "SAML 2.0 SSO Configuration" on page 71.
2. Edit the `banner-<short-appName>-sp.xml` file for the service provider configuration which will configure the Authentication end point and Logout endpoint.
   a) Replace the parameters below with the configured values for Banner General Self-Service.
      - `<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 91 for more information.)

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

   Example:

```
<?xml version="1.0" encoding="UTF-8"?>
<md:EntityDescriptor xmlns:md="urn:oasis:names:tc:SAML:2.0:metadata"
ID="<ALIAS_NAME>" entityID="<ALIAS_NAME>">
<md:SPSSODescriptor AuthnRequestsSigned="false"
 WantAssertionsSigned="false"
protocolSupportEnumeration="urn:oasis:names:tc:SAML:2.0:protocol">
<md:KeyDescriptor use="signing">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
```

```
<ds:X509Certificate>
<EXTRACTED_DATA>
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:KeyDescriptor use="encryption">
<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">
<ds:X509Data>
<ds:X509Certificate>
<EXTRACTED_DATA>
</ds:X509Certificate>
</ds:X509Data>
</ds:KeyInfo>
</md:KeyDescriptor>
<md:SingleLogoutService
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"
Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/saml/
SingleLogout/alias/
<ALIAS_NAME>"/>
<md:SingleLogoutService
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTPRedirect"
Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/saml/
SingleLogout/
alias/<ALIAS_NAME>"/>
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-
format:emailAddress</
md:NameIDFormat>
<md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:transient</
md:NameIDFormat>
<md:NameIDFormat>urn:oasis:names:tc:SAML:2.0:nameid-
format:persistent</
md:NameIDFormat>
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-
format:unspecified</
md:NameIDFormat>
<md:NameIDFormat>urn:oasis:names:tc:SAML:1.1:nameid-
format:X509SubjectName</
md:NameIDFormat>
<md:AssertionConsumerService
 Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTPPOST"
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:holderof-
key:SSO:browser" Location="http://<HOSTNAME>:<PORT>/
<APPLICATION_NAME>/saml/
SSO/alias/<ALIAS_NAME>"
hoksso:ProtocolBinding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-
Artifact"
index="1" xmlns:hoksso="urn:oasis:names:tc:SAML:2.0:profiles:holder-
ofkey:
SSO:browser"/>
<md:AssertionConsumerService
 Binding="urn:oasis:names:tc:SAML:2.0:profiles:holderof-
```

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

Create an identity service provider file

Perform the following steps to create an identity service provider file.

Procedure

1. Create a file `banner-<short-appName>-sp.xml` at the folder path mentioned in “SAML 2.0 SSO Configuration” on page 71.
2. Edit the `banner-<short-appName>-sp.xml` file for the identity provider configuration.

   The file contains the identity provider information configured in EIS server over which Banner General Self-Service sends the SAMLrequest and receives the SAML response.

   a) Replace the parameters below with the configured values for Banner General Self-Service.

      - `<HOSTNAME>`: EIS server host name
      - `<PORT>`: Deployed EIS server port number
      - `<ALIAS_NAME>`: Identity provider ID set in EIS identity provider setup (a good pattern to use is “`https://<HOSTNAME>:<PORT>/samlsso`”)
      - `<EXTRACTED_DATA>`: Extract X509 certificate Data for `banner-<short-appName>-idp.xml` (See “Extract X509 certificate data” on page 93 for more information.)

   b) 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="<ALIAS_NAME>" 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:2.0:nameid-
format:transient</
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

You must add the IDP certificate entry in the newly created `.jks` file.

Related Links

Add an IDP certificate entry in the newly created keystore file on page 83

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

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

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

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

Configure application-specific settings

Banner XE applications read configuration at startup from a general configuration file (`banner_configuration.groovy`) and from the General Self-Service configuration file (`BannerGeneralSsb_configuration.groovy`), which provides application-specific settings and can override settings in the general configuration file.

About this task

Applications often override general settings for logging and keeping application logs in a file separate from other applications. Some applications also define custom settings. For example, the Banner Student Advisor application allows configuration of roles that may access specific views within the application.

Procedure

1. Copy `BannerGeneralSsb_configuration.example` and change the name to `BannerGeneralSsb_configuration.groovy`.

   The installer creates the `BannerGeneralSsb_configuration.example` file.
2. Place the `BannerGeneralSsb_configuration.groovy` file in the `BannerGeneralSsb\current\instance\config` directory.

   This application-specific configuration file contains settings that you can customize for your specific environment.

JMX MBean name

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

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

In the example, the user needs to update `BannerGeneralSsb-log4j` identifier for each installation of each application. This allows the JMX management to distinguish between different installations of the application.

Location of the logging file

Log4j is the common logging framework used with applications that run on the Java Virtual Machine. You can configure the location at which the log file is saved.

For more information about the Log4j settings, see http://docs.grails.org/2.5.0/guide/conf.html.

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

The following example shows how to configure the location where the log file is saved:

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

The following example shows how to override the log file directory properties:

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

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

Logging level

The root logging level is pre-configured to the `ERROR` level. Multiple class or package level configurations, by default, are set to a status of `off`. You can set a different logging level for any package or class. However, configuration changes for logging take effect when the application is 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 `BannerGeneralSsb_configuration.groovy` file take effect after the application is restarted.

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

Related Links

Configure Java management extension on page 51

Institutional home page redirection support

The institutional home page redirection support configuration allows General Self-Service to provide an institutional home page to which users can navigate back to if they have access issues specific to insufficient privileges when accessing the General Self-Service application.

```
/******************************************************************
* Home Page link when error happens during authentication. *
*****************************************************************/
grails.plugin.springsecurity.homePageUrl='http://URL:PORT/'
```

Proxied Oracle users

When connecting to Self-Service applications, the `ssbOracleUsersProxied` setting specifies whether the Oracle account is used for the database connections. The `ssbOracleUsersProxied` setting also controls whether Value Based Security (VBS) is enabled in the Self-Service application.

The following values can be used for the `ssbOracleUsersProxied` setting:

- `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 can still log in to the Self-Service application.
- `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 cannot log in to the Self-Service application.

Guest authentication

The **guestAuthenticationEnabled** setting determines how authentication should be handled when the application is configured for CAS or SAML sign-on.

When **guestAuthenticationEnabled** is true, the single sign-on authentication will be bypassed for API URLs which allows users to login using Basic authentication with just a username and password. When **guestAuthenticationEnabled** is false, the CAS and SAML protocols will be used for authentication with all URLs.

For Banner Print application

If this application's deployment is intended to be used with the Banner Print Appllication for printing pending job submission output, then **guestAuthenticationEnabled** must be set to true.

Enable SAML

To enable SAML, set the property grails.plugin.springsecurity.saml.active = true.

Set up SAML SSO configuration

The BannerGeneralSsb\current\instance\config directory contains the `BannerGeneralSsb_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 *
* *
*******************************************************/
//
// Set authenticationProvider to either default, cas or saml.
// If using cas or saml, Either the CAS CONFIGURATION or the SAML
CONFIGURATION
// will also need configured/uncommented as well as set to active.
//
banner {
sso {
authenticationProvider = 'default' // Valid values are: 'saml' and
'cas' for SSO to work. 'default' to be used only for zip file
creation.
authenticationAssertionAttribute = 'UDC_IDENTIFIER'
if(authenticationProvider != 'default') {
grails.plugin.springsecurity.failureHandler.defaultFailureUrl = '/
login/error'
}
if(authenticationProvider == 'saml') {
grails.plugin.springsecurity.auth.loginFormUrl = '/saml/login'
}
}
```

Logout URL

You can specify where a user is directed after logging out of the application by updating the `BannerGeneralSsb_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.

When using SAML, logging out directs the user to a custom logout page.

SAML 2.0 SSO configuration

This is a sample of the configuration you can enable for SSO between Banner General Self-Service and an Identity Management System that support SAML 2.0 SSO protocol.

The following properties describe the configurations required for the application to work in SAML 2.0 SSO protocol.

Note: Uncomment this section when SAML 2.0 SSO is enabled.

Property	Description
`banner.sso.authentication.saml.localLogout`	Default value is set to false, indicating that the application will participate in Global logout. An application participating in global logout will notify the Identity Server about logout within the application. If this is set to true, indicating local logout, the application will not notify the Identity Server to log the user out from all applications.
`grails.plugin.springsecurity.auth.loginFormUrl`	Pre-populated to provide the Login URL.
`grails.plugin.springsecurity.saml.afterLogoutUrl`	Pre-populated to provide the Logout URL.
`grails.plugin.springsecurity.saml.keyManager.defaultKey`	Key name used at the time of key-store creation ("Create a keystore (*.jks)

Property	Description
	file" on page 90). Example:generalkeystore.jks
`grails.plugin.springsecurity.saml.keyManager.storeFile`	Location of the keystore file. This could be a classpath (for example, classpath:securitygeneralkeystore.jks) or it could be an absolute location on the machine (for example, file:c://temp/generalkeystore.jks or u02/generalkeystore.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/bannerstudent-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/bannerstudent-idp.xml'. Possible keys

Property	Description
	are adfs, Okta, Shibb, etc.
`grails.plugin.springsecurity.saml.metadata.defaultIdp`	Provide the default IDP to be used from the IDP providers set. This is the same value specified above for the key.
`grails.plugin.springsecurity.saml.metadata.sp.defaults`	• local: Pre-populated to indicate value to be picked up. • alias: An alias name that is unique to this application (for example, banner-<application-shortname>-sp). • securityProfile: Pre-populated value. • signingKey: A key used to sign the messages that is unique to this application (for example, banner-<application-shortname>-sp). • encryptionKey: A key to to encrypt the message that is unique to this application, (for example, banner-<application-shortname>-sp) • tlsKey: A tls key that is unique to this application (for example, banner-<application-shortname>-sp). • requireArtifactResolveSigned: Pre-Populated to set to false indicating

Property	Description
	artifact to be signed or not. • requireLogoutRequestSigned: Pre-Populated to set to false indicating logout request to be signed or not.

Customize the landing page background image

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

Procedure

1. Create two CSS files in the deployment staging directory of Banner General Self-Service. ~/BannerGeneralSsb/current/instance/css)
   - bannerSelfService-custom.css for LTR support
   - bannerSelfService-custom-rtl.css file for RTL support
2. Modify each of these two files by adding the custom CSS styles you want to change as shown below.

   For overriding the landing page background image, use the following CSS class and add your institutional background image to the media query style that supports the specific responsive design orientation.

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

The above images and styles shown support responsive designs for landscape, portrait and mobile views.

3. Make sure your institutional background images you choose maintain the correct aspect ratio for displaying the custom images in landscape and portrait mode along with supporting different device screen resolutions.
   - campus-landscape.jpg supports widescreen desktop devices (supportedresolution size 1920 * 1080)
   - campus-portrait.jpg supports laptops and tablet devices (supportedresolution size 768 * 1024
4. Confirm the custom CSS files and images are in the deployment staging directory of Banner General Self-Service.

   The directory and file structure should be similar as shown below:

```
BannerGeneralSsb
|-----current
|-----instance
|-----css
|-----bannerSelfService-custom.css
|-----bannerSelfService-custom-rtl.css
|-----images
|-----campus-landscape.jpg
|-----campus-portrait.jpg
|-----campus-sml.jpg
```

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

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

Regenerate the WAR file

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

About this task

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

Procedure

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

   PRODUCT_HOME/current/installer
2. Run the ant command, which builds the systool module.

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

   ```
   $ cd PRODUCT_HOME/current/installerPRODUCT_HOME/current/installer
    $ ./ant
   ```
3. Create the WAR file using the systool module.

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

   On Unix:

   $ bin/systool war

   On Windows:

   > bin\systool war

   The WAR file is created in the PRODUCT_HOME/current/dist directory.You can use external configuration files by setting appropriate system properties, although the configuration files are included in the WAR file to make the WAR file self-sufficient.

Related Links

Configure the web application server and deploy the WAR file on page 48

Add service provider in EIS server

You need to add an EIS server as part of the SAML configuration.

Procedure

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 for EIS server

Add the SAML settings for the integrating application.

Procedure

1. On the Service Providers screen, expand the Inbound Authentication Configuration panel, then expand the SAML2 Web SSO Configuration panel.
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 `BannerGeneralSsb_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 when 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 57.
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

Add a resident identity provider as per the `idp_local.xml` configuration.

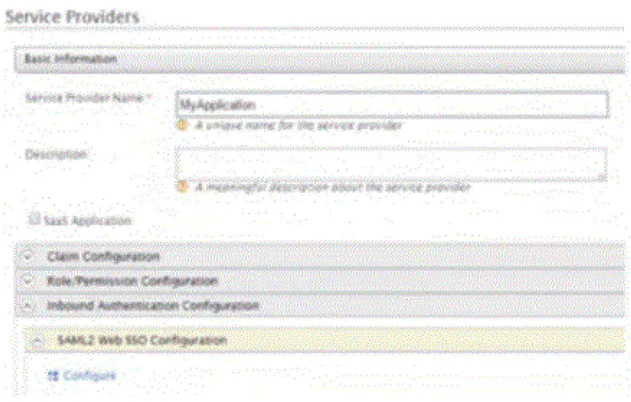
About this task

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, through 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.

Procedure

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:



Service Providers

Basic Information

Service Provider Name *: MyApplication
A unique name for the service provider

Description:
A meaningful description about the service provider

SaaS Application

Claim Configuration

Role/Permission Configuration

Inbound Authentication Configuration

SAML2 Web SSO Configuration

Configure

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

Sub-tasks referred to in the Install Banner General Self-Service for SAML 2.0 SSO chapter are discussed in this section.

Extract a X509 Certificate Key

During SAML configuration, you must extract a X509 certificate key from the keystore for the `banner-BannerGeneralSsb-sp.xml` file.

Procedure

1. With the `generalkeystore.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 90 for more information.

   ```
   C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -list -v -keystore
   generalkeystore.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
SSO
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 generalkeystore.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
generalkeystore.jks
Enter keystore password: password
```

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
ETAPBgNVBAoTCEVsbHVjaWFuMREwDwYDVQQLEwhFbGx1Y2lhbjEZMBcGA1UE
 AxMQU3Bob29ydGkg
QWNoYXJ5YTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAN2cS
+2OX37HioFzwOWLm/S0
F+zt6ldtLfmHc16V9iqkZChkMiXpKgXVPqGLFjBrhwfsWtuMfRy2NYf3forEDFTaV4/
fLXRo+Npd
xTfqWhuZTafDJEyKQc57KY8G3feg1CSjfKkCk1LF
+zbGClHQ0bg1dwUjJlp7eKjWM0rbsKMd5pZ7
```

```
0tGAPcYsi6MtGvJupaVhy3jNTDg+kh4/D92y/mTaLlCR4QQr1qlU9+H
+it3m9jiDrZ7svrdBlsDN
1BVcXDooqUGTuc10IBxYEsb7hFucSFpdJnGJvbg35ll9K9F5S8lEmiQmeOUQ1gQe2Ow01kFl56Qz
4evM0xgeskNid9sCAwEAAaMhMB8wHQYDVR0OBBYEFLKs1wkBFSKjMgiGZOglWhXLoMbZMA0GCSqG
SIb3DQEBCwUAA4IBAQANJbYRTcMwhrETz+mo
+n1okrXIs118AIm7s1yJJd1nyJuaKrn7DcPPLzy/
RjHGKP02uLiupgqar+UUaPqSZjJXSzktLLyq7H6DRrW0Jp2rw48a+Kou
+XOvQ8ZWR9ZXIa1XoAoD
PaSSE2omcVOVGZmQKUYardVeSvQth3IVMW9w9Jl
+DuavXavVjIx5IN6RRhXGfaJjQLKFzIDqZNAp
OcxMEKXHOUqj0ksTRARLpKWSPu7gFOWO/6qapNp5l8rlPjnVxDhqHqCKC3E40VI5n+C
+KJHZQqab
Tfhd6erqEy7S1Cazr655Yq22Jm6L7IXsXgpRwmZnoietLsrFIRyPe1DY
SSO
-----END CERTIFICATE-----
```

Extract X509 certificate data

During SAML configuration, you must extract X509 certificate data for `banner-BannerGeneralSsb-sp.xml`.

Procedure

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-----
```

a) If no certificate file is found, navigate to the `.jks` file. The `.jks` file can be found through `carbon.xml`. By default, it is located at:

   `$EIS_HOME\repository\conf\carbon.xml`

b) 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>
```

c) Create the `saml-idp.cer` file by executing the following command.

   `keytool -export -keystore cacerts -alias mykey -file`

   Note: The `password` and `alias` referenced in this example are also contained in the `carbon.xml` file accessed earlier in this task.

4. Go to the keystore location and execute the following command.

```
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 an IDP certificate entry in the newly created keystore 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 90. Perform the following steps to add the IDP certificate entry to the `.jks` file you created for that sub-task.

Procedure

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 generalkeystore.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
 generalkeystore.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
]
]
*******************************************
*******************************************
```

```
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
]
]
*******************************************
*******************************************
```

Related Links

Add an IDP Certificate entry in the newly created keystore file on page 66

Personal Information configuration

You can use the configuration checklist, which is a quick reference guide that provides you with a list of the decisions you need to make to configure Personal Information application.

Note: These configuration options affect only Personal Information 9.x application. Your choices regarding these settings do not affect any other part of Banner.

For information about the instructions that are required for configuring Personal Information, see *Banner General Self-Service Handbook*.

Personal Information sections

Question	Implementation Options
Do you want to display the Personal Details section in the Personal Information application?	• Yes - No action needed. By default, this section is displayed. • No - This requires a changes to the Integration Configuration Settings (GORICCR) page in the Administrative Banner application.
Do you want to display the Email section in the Personal Information application?	• Yes - No action needed. By default, this section is displayed. • No - This requires a change to the Integration Configuration Settings (GORICCR) in the Administrative Banner application.
Do you want to display the Phone Number section in the Personal Information application?	• Yes - No action needed. By default, this section is displayed. • No - This requires a change to the Integration Configuration Settings (GORICCR) in theAdministrative Banner application.
Do you want to display the Address section in the Personal Information application?	• Yes - No action needed. By default, the Taxes section is displayed. • No - This requires a change to the Integration Configuration Settings (GORICCR) in the Administrative Banner application.
Do you want to display the Emergency Contact section in the Personal Information application?	• Yes - No action needed. By default, this section is displayed.

Question	Implementation Options
	• No - This requires a change to the Integration Configuration Settings (GORICCR) in the Administrative Banner application.
Do you want to display all items in the Additional Details section in the Personal Information application?	• Yes - No action needed. By default, this section is displayed. • No - This requires a changes to the Integration Configuration Settings (GORICCR) page in the Administrative Banner application using the Process PERSONAL_INFORMATION_SSB and the following settings. – ETHNICITY.RACE.MODE= 0-Hidden – ENABLE.VETERAN.CLASSIFICATION= N-Not Visible – ENABLE.DISABILITY.STATUS= N-Not Visible
Do you want to display all items in the Others section in the Personal Information application?	• Yes - No action needed. By default, this section is displayed. • No - This requires a change to the Integration Configuration Settings (GORICCR) in theAdministrative Banner application using the Process PERSONAL_INFORMATION_SSB and the following settings. – ENABLE.DIRECTORY.PROFILE= N-Not Visible – ENABLE.SECURITY.QA.CHANGE= N-Not Visible – ENABLE.PASSWORD.CHANGE= N-Not Visible
Do you want to allow Personal Information and Direct Deposit application access? You have the option to display just one application, but you must edit the GORICCR page.	• Yes - No action needed. By default, the icons are displayed to allow users access to Direct Deposit and Personal Information application. • No - This requires changes to the GORICCR page in the Administrative Banner application.

Overview section

Question	Implementation Options
Do you want to display the user's photo in the Overview section of the Personal Information application?	• Yes - No action needed. By default, photos are displayed in the Personal Information Overview section. This is set

Question	Implementation Options
	up at the time of installation. If a photo is not available, a generic avatar is displayed. • No - This requires a change to the Integration Configuration Settings (GORICCR) in the administrative Banner application.
Do you want to allow the user to update their Preferred email address, which displays in the Overview section of the Personal Information application?	• Yes - No action needed. By default, users can update their Preferred email address, which displays in the Personal Information Overview section. • No - This requires a change to the Integration Configuration Settings (GORICCR) in the Administrative Banner application.
Do you want to define the hierarchy of Address types to display in the Overview section of the Personal Information application?	• Yes - This requires a change to Integration Configuration Settings (GORICCR) in the Administrative Banner application. • No - By default, the first address in the Address section will display.
Do you want to define the hierarchy of Telephone types to display in the Overview section of the Personal Information?	• Yes - This requires a change to Integration Configuration Settings (GORICCR) in the Administrative Banner application. • No - By default, the first phone number in the Phone Number section will display.
Do you want to display the Address in the Overview section of the Personal Information application?	• Yes - No action needed. By default, the first address in the Address section will display. • No - This requires a change to the Integration Configuration Settings (GORICCR) in the Administrative Banner application.
Do you want to display the Email in the Overview section of the Personal Information application?	• Yes - No action needed. By default, the first email in the Email section will display. • No - This requires a change to the Integration Configuration Settings (GORICCR) in the Administrative Banner application.
Do you want to display the Phone Number in the Overview section of the Personal Information application?	• Yes - No action needed. By default, the first phone number in the Phone Number section will display. • No - This requires a change to the Integration Configuration Settings (GORICCR) in theAdministrative Banner application.

Provide access to the Personal Information application

You can allow or deny users' access to the Personal Information application from the landing page.

Procedure

1. Open the GORICCR page in the Administrative Banner application and search for the **GENERAL_SSB** process.
2. Select the **ENABLE.PERSONAL.INFORMATION** setting and go to the next block of information.
3. In the **Value** field, enter one of the following values:
   - N to hide the Personal Information icon and deny access to the application.
   - Y to display the Personal Information icon and allow access to the application.
4. Save the value.

Direct Deposit user and role configuration

Banner Direct Deposit is an Ellucian application that allows employees and students to create, update, and delete bank account records so that they can receive payments through direct deposit.

User setup

Each employee and student who uses the Banner Direct Deposit application must exist in the Banner database with the appropriate records.

At a minimum, employees must have an employee record in the PEBEMPL table, and students must have a record in SGBSTDN.

The user experience will be different between employees and students in that students will not have ability to create direct deposit accounts for Payroll. The Banner Direct Deposit application uses the existing GOVROLE view to gather the information from the Banner database to determine what role(s) users have when they log in. Refer to the Web Roles section in *Banner Web Tailor User Guide* for more details.

Banner ID

Each Banner Direct Deposit user must have a Banner ID. Banner IDs can be created on any Banner identification form (such as PPAIDEN or SPAIDEN).

Oracle ID

Users of the Banner Direct Deposit application do not require an Oracle ID. Typically, only administrative users of the system require an Oracle ID.

Banner Self-Service PIN

Employees and students who will use the Banner Direct Deposit application must have valid login credentials for Self-Service Banner (SSB). PINs for access to SSB are maintained on the Third Party Access (GOATPAC) and Third Party Access Audit (GOATPAD) forms.

Banner Web Tailor role

Employees and students who will use the Banner Direct Deposit application do not need to be manually assigned any specific roles in Banner Web Tailor.

Web Tailor parameters

New direct deposit records created in the Update Direct Deposit Allocation page of Employee Self-Service have a status of Prenote as the default. However, the Banner Direct Deposit application allows clients to set the default status of new direct deposit records to Active if they prefer to by-pass the prenote process.

The ability to configure this behavior is a new feature introduced with Banner Direct Deposit.

To change the default status for new direct deposit records, a user with Web Tailor Administration access in Self-Service Banner can login and navigate to the Web Tailor Parameters item on the Web Tailor Administration tab. After finding the parameter SHOW_USER_PRENOTE_STATUS, click the parameter link. Change the Parameter Value from Y to N and click the **Submit Changes** button.

The Banner Direct Deposit application also allows clients to set a maximum number of Payroll direct deposit accounts that employees can create. By default, the maximum number of Payroll direct deposit account records allowed is 99. The ability to configure this behavior is a new feature introduced with Banner Direct Deposit.

To update the default maximum number of Payroll direct deposit accounts that an employee can create, a user with Web Tailor Administration access can log in to Self-Service Banner and navigate to the Web Tailor Parameters item on the Web Tailor Tab. After finding the parameter MAX_USER_PAYROLL_ALLOCATIONS, click the parameter link. Change the Parameter Value from ‘99’ to the desired value and click the **Submit Changes** button.

Provide access to the Direct Deposit application

You can allow or deny users' access to the Direct Deposit application from the landing page.

Procedure

1. Open the GORICCR page in the Administrative Banner application and search for the **GENERAL_SSB** process.
2. Select the **ENABLE.DIRECT.DEPOSIT** setting and go to the next block of information.
3. In the **Value** field, enter one of the following values:
   - N to hide the Direct Deposit icon and deny access to the application.
   - Y to display the Direct Deposit icon and allow access to the application.
4. Save the value.