Banner Finance Self-Service

Installation Guide

Release 9.1.1
November 2017

Notices

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

Contents

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

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* – found in the Software Downloads section of the Ellucian Support Center. Only users with software download privileges have access to this publication.
- *Application Navigator Installation Guide*
- *Banner 9 Sizing and Configuration Guide*

Hardware requirements

Ellucian recommends these hardware requirements.

CPU and memory

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

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

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

Screen resolution

Minimum: 1024 x 768

Tablets

- iPad, iPad Mini, iOS 8.x, iOS 9.x, and iOS 10.x
- Android OS 4.x
- Microsoft Surface 1.0, RT, and PRO

Software requirements

Ellucian recommends these software requirements.

Banner database

Banner Finance Self-Service requires a minimum version of Common Database Upgrade 9.12.0.1 and baseline Banner Finance patch 8.11.1.3.

Before applying the Banner Database Upgrade 9.12.0.1 patch, you must install Banner Database Upgrade 9.12. For more information, refer to the *Banner Database Upgrade 9.12 Upgrade Guide* that you can download from the Ellucian Support Center in the Banner General Documentation Library.

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 Finance Self-Service applications are supported by WebLogic and Tomcat web servers.

- Oracle WebLogic: 10.3.3, 10.3.4, 10.3.5, 10.3.6, and 12.1.3
- 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 Banner Finance Self-Service applications support 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-FinanceSelfService-9.1.1`)

Single sign-on (SSO) support

Ellucian recommends that you first establish the single sign-on environment before implementing Banner Finance Self-Service.

Central Authentication Service (CAS) and Security Assertion Markup Language (SAML) 2.0 are the Single Sign-On (SSO) protocols supported by Banner Finance Self-Service. Both are certified to run with Ellucian Ethos Identity. You can download Ellucian Ethos Identity from the Ellucian Support Center.

For more information about how to establish a single sign-on environment, refer to the *CAS Single Sign-On Handbook* or *Setting Up Ellucian Identity*. Both are available for download from the Documentation Libraries in 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.

Use the same version of Java to customize and deploy the WAR file. Define the JDK bin directory 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 Support Center.

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

Internet Explorer Compatibility view

When using Internet Explorer, you must be on the Internet Explorer Standard Mode.

About this task

On Internet Explorer 9, you might receive a message that you are in Compatibility View. To disable Compatibility View, perform the following steps:

Procedure

1. **Tools** > **Compatibility View Settings**
2. Clear the **Display intranet sites the Compatibility View** check box.
3. Select the **Display all websites** check box.

Configure Application Navigator

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

You can configure Application Navigator's menu so that a menu item displays for each Self-Service Application. Refer to the *Application Navigator Installation Guide* and *Application Navigator Handbook* for configuration instructions. These guides are available from the Ellucian Support Center in the Banner General Documentation Library.

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).

Upgrade the database

Before proceeding further, make sure that you have reviewed the section Banner database on page 7 for Banner Finance Self-Service 9.1.1 database requirements.

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 `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 BAN_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 `BWFMIGR.TXT` as a starting point for writing your own migration scripts.

Unix

The file `BWFMIGR.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 `bwfmigr.shl` does the appropriate removes, copies, and links. The local *LN* variable at the top of `bwfmigr.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 the background on a Unix platform, perform the following steps:

Note: The directory structure `$BANNER_HOME/finweb/java` must exist. If it does not, use the following command to create this directory before performing the steps to run the migration script:
`mkdir $BANNER_HOME/finweb/java`

If you have installed previous releases of Purchase Requisition, ensure that you move the corresponding release zip file that was migrated to `$BANNER_HOME/finance/java` to the directory `$BANNER_HOME/finweb/java`.

Procedure

1. Ensure that the directory path names in `bwfmigr.shl` are correct.
2. Ensure that the environment variable *$BANNER_HOME* in `bwfmigr.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 bwfmigr.shl >bwfmigr.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 `bwfmigr.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 `bwfmigr.pl` does the appropriate deletes and copies.

About this task

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.
   - If the value of `BANENV` is `REG`, the value used for `BANNER_HOME` will be taken from the registry entry: `HKEY_LOCAL_MACHINE\SOFTWARE\BANNER\BANNER_HOME`.
   - If the value of `BANENV` is `ENV`, the value used for `BANNER_HOME` will be taken from the environment variable `BANNER_HOME`.
2. Ensure that the directory path names in `bwfmigr.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 bwfmigr.pl >bwfmigr.log 2>&1
   ```
6. Review `bwfmigr.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.sql
   ```

2. Review the `versionupdate` listing.

Undeploy the existing application

Before you install Banner Finance Self-Service, you must undeploy any previous 9.x versions of Banner Finance Self-Service, including any standalone instance of Purchase Requisition. If no previous 9.x versions of Banner Finance Self-Service are installed, skip this section.

The sections in this chapter provide the required steps to undeploy existing Banner 9.x applications in Tomcat and WebLogic servers.

Tomcat

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

Undeploy using the Tomcat manager web application

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

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

Undeploy manually

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

Undeploy applications manually on Unix

This task allows you to shut down Tomcat and manually undeploy any previously deployed versions of Banner Finance Self-Service 9.x and Banner Purchase Requisition 9.x on your Unix systems.

Procedure

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

   ```
   $CATALINA_HOME/bin/shutdown.sh
   ```
3. Remove the current deployment and associated WAR file:

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

Undeploy applications manually on Windows

This task allows you to shut down Tomcat and manually undeploy any previously deployed versions of Banner Finance Self-Service 9.x, including any standalone instance of Purchase Requisition 9.x applications on your Windows systems.

Procedure

1. Shut down Tomcat.
   - If you installed Tomcat as a service, use the **Service Control** panel to stop the application.
   - If Tomcat is not installed, use the shutdown script `%CATALINA_HOME%\bin \shutdown.bat`.
2. If you have been using Banner Finance Self-Service 9.x and Purchase Requisition 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\<Finance Self Service 9x Application>
   ```
4. Remove the associated WAR file for each of the applications removed in the previous step:

   ```
   del %CATALINA_HOME%\webapps\<Finance Self Service 9x Application>.war
   ```

Undeploy applications using WebLogic

This task allows you to undeploy any previously deployed versions of Banner Finance Self-Service 9.x, and Banner Finance Purchase Requisition 9.x on your systems by using WebLogic.

Procedure

1. Access the administration server using the following URL:
   `http://server:7001/console`
2. In the **Domain Structure** frame, click **Deployments**.
3. In the **Change Center** frame, click **Lock and Edit**.
4. Select the check box beside the Banner 9.x application.
5. Click **Stop**.
6. Click **Force Stop Now**.
7. In the **Force Stop Application Assistant** page, click **Yes**.
8. Select the check box beside the Banner 9.x application.
9. Click **Delete**.
10. In the **Delete Application Assistant** page, click **Yes**.
11. In the **Change Center** frame, click **Activate Changes**.

Customize the WAR file

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

The `release-FinanceSelfService-9.1.1.zip` must be moved to `$BANNER_HOME\finweb\java` subdirectory during the database upgrade.

Note: JDK 1.7 must be installed on your system.

Related Links

Java dependencies on page 9

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-FinanceSelfService-9.1.1.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-FinanceSelfService-9.1.1.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 four product home directories and a `shared_configuration` directory that support a common test environment.

```
TEST/
|--> FinanceSelfService/
|--> Schedule/
|--> StudentOverall/
|--> 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 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/FinanceSelfService
   ```

   On Windows:

   ```
   []: c:\banner_test_homes\FinanceSelfService
   ```

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 Banner Finance 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 is required as an external authentication provider.

The following components must be configured to authenticate using CAS:

- The Banner Finance 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 Banner Finance 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
```

Note: 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 Banner 9.x self-service application.

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

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

Banner Finance Self-Service 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 Banner Finance 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.ellucian.com:8888/
ssomanager/c/SSB?pkg=http://e009070.ellucian.com:9020/
SMPL_GVU/', BANNER: 'http://e009070.ellucian.com:8888/
ssomanager/c/SSB?pkg=http://e009070.ellucian.com:9020/
SMPL_BANNER/' ]
```

Session timeouts

When you determine the timeout settings for CAS and Banner Finance Self-Service, consider the time limits imposed for each.

For example, if the Banner Finance Self-Service timeout setting is less than the CAS timeout setting, SSO authentication might still be valid even though the user is exited from Banner Finance Self-Service. As long as the CAS authentication remains valid, the user can re-enter Banner Finance Self-Service 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`.

Configure application-specific settings by modifying the groovy file

Banner XE applications read configuration at startup from a general configuration file (`banner_configuration.groovy`) and from a configuration file (`FinanceSelfService_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 Finance Self-Service application allows to configure the vendor address types supported in the application.

Procedure

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

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

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

System properties

The following properties in the `FinanceSelfService_configuration.groovy` file are internal to the application and are required for carrying out its own lifecycle activities. Leave these properties undisturbed.

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

grails.plugin.xframeoptions.deny = true

productName="Finance"
```

```
banner.applicationName="Banner Finance"
footerFadeAwayTime=40000
```

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.

For example:

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

In the example, the user needs to update `FinanceSelfService-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'] ?
"${System.properties['logFileDir']}" : "target/logs"
logAppName = "FinanceSelfService"
loggingFileName = "${loggingFileDir}/
${logAppName}.log".toString()
```

`logAppName = "FinanceSelfService"` must be set by the user.

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 `FinanceSelfService_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 58

Institutional home page redirection support

The institutional home page redirection support configuration allows Employee Self-Service to provide an institutional home page to which users can navigate back to if they have access issues.

Access issues are specific to insufficient privileges when accessing the Employee 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.

Logout URL

You can specify where a user should be directed after logging out of the application by updating the `FinanceSelfService_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'
if (useSingleSignout) {
grails.plugin.springsecurity.useSessionFixationPrevention =
 false
}
}
logout {
afterLogoutUrl = 'https://cas-server/logout?url=http://
myportal/
main_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 `FinanceSelfService_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.

Google Analytics

To enable Google Analytics for the Attendance Tracking application, 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.

Theme Editor tool

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

Use the following settings to enable the tool:

```
banner.theme.URL="<UPDATEME>"
banner.theme.name="<UPDATEME>"
banner.theme.template="FinanceSelfService"
banner.theme.cacheTimeOut=120
```

Settings	Interpretation
`banner.theme.url`	References the URL to the application hosting the Theme Server.
`banner.theme.name`	The desired theme name to use. In a MEP environment, the application uses the MEP code as the theme name instead of `banner.theme.name`.

Settings	Interpretation
	You must create a theme by this name in the Theme Editor on the server specified by `banner.theme.url`.
`banner.theme.template`	The name of the scss file containing the theme settings in the WAR file.
`banner.theme.cacheTimeOut`	Advanced setting for when the application is configured to be the host of the theme server. The value indicates how long the CSS file that was generated using the template and the theme is cached.

After you enable the Theme Editor, the Preview section displays an example of how the theme settings might appear. Your theme choices could be displayed differently in some applications.

You can configure the following theme parameters using the Theme Editor section:

- Theme name (required) – Should be a recognizable name for your theme.

  An institution name or abbreviation is a good example. MEP (Multi-Entity Processing) codes can also be used as theme names for MEP-enabled institutions to allow per-MEP-code theming. The theme name is the theme's primary identifier. If you change the name and click **Save**, a new theme is created with the new name.

- Primary color – Color for the main title bar.

  You can enter the hex code of the color, or choose a color using the dropdown color list. The Theme Editor generates several related shades based on your color choice for borders, backgrounds, and text. As you change colors and shades, the Preview section updates to show your choices.

- Secondary color – Color for other page elements.

  See Primary color.

- Accent color – Additional color for various elements.

  As you modify Primary and Secondary colors, the Theme Editor derives an Accent color that contrasts with the Primary and Secondary colors. If you prefer, you can specify the Accent color.

- Logo URL – Web URL to the logo to include in the pages.

  This logo should be an SVG file to allow scaling to fit the page. It should also be a color that contrasts with the Primary color. The URL must be accessible to application clients, as the URL value that is specified for the theme is included in the CSS files for the applications, so individual users' browsers will load the logo from the specified URL. As you enter or update the URL, if the Theme Editor can resolve the logo, it will appear in the colored bar below the Logo URL field.

- 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 made available to all applications. When you click **Save Theme**, the theme is applied to the current page.

- New Theme – Clears the theme settings.

  Alternatively, you can copy an existing theme by changing the Theme Name and clicking **Save Theme**.

As you create themes, they appear in the Saved Themes section. You can perform the following functions in the Saved Themes section:

- **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 is provided so you can save the JSON to another location if desired.
- **CSS** – Links to the CSS file generated for the theme. This is provided so you can save the CSS to another location if desired. This allows applications to be configured to use the static theme to optimize performance, but if applications are using a statically saved theme file, the static file would have to be updated manually with any updates to the theme.

Hibernate Secondary Level Caching

Banner Finance Self-Service uses Hibernate Secondary Level Caching, which helps in improving the application performance.

The configuration related to this are enabled (`true`) by default. If your institution has Banner Finance module enabled for Multi-Entity Processing (MEP), then the below two settings needs to be disabled (set to `false`).

```
/*************************************************
 *              Hibernate Secondary Level Caching                    *
 *************************************************/
hibernate.cache.use_second_level_cache = true // Default true.
Make it false for the institution that is MEP enabled for Finance.
hibernate.cache.use_query_cache = true // Default true.
Make it false for the institution that is MEP enabled for Finance.
```

Configure My Finance

This section provides configuration information for My Finance.

Consolidated setting for Finance application

This setting enables and disables various My Finance modules.

You can enable all available modules or disable any of the available modules. By default, all available Finance modules are enabled.

Currently the following modules are available for configuration:

- REQUISITION: My Requisition
- FINANCEQUERY: My Finance Query

Do not change the module name key. Only change enable and disable flag. To enable a module, you set the corresponding value of the configuration to `true`. To disable a module, you set the corresponding value to `false`.

```
banner{ consolidated{       applications
    = [REQUISITION:true, FINANCEQUERY:true]
    } }
```

My Finance PDF logo file location

This configuration specifies the file path of the institution logo file.

This logo is displayed in the PDF document. For optimal viewing, use a logo file that has a width of 200 pixels and a height of 90 pixels. The logo file should be of one of the following formats: PNG, JPG, or BMP.

If you need a reference logo, you can find the default Ellucian logo (`ellucian-logo.png`) as part of the WAR file.

Create Unix file path

To create the Unix file path, complete the following steps.

Procedure

1. Create the directory and place the logo file if the directory and file are not present.

   ```
   mkdir /u02/BANXE
   ```

2. Use this configuration:

   ```
   banner.finance.ssb.logoImageFile=    '/u02/BANXE/logo.png'
   ```

   Note: `/u02/BANXE` is a sample path, which can be decided by the administrator.

Create Windows file path

To create the Windows file path, complete the following steps.

Procedure

1. Create the directory and place the logo file if the directory and file are not present.

   ```
   mkdir c:/BanXE
   ```

2. Use this configuration:

   ```
   banner.finance.ssb.logoImageFile='c:\BanXE\logo.png'
   ```

Note: `c:/BANXE` is a sample path, which can be decided by the administrator.

My Requisition: Configurations related to Vendor

There are three configurations related to vendor.

- `chooseVendorForMeChecked` to default the **Choose Vendor For Me** check box as enabled or disabled.
- `corporateAddressTypes` list to filter the address Types of corporate vendors to be displayed in the **Vendor Lookup**.
- `personAddressTypes` list to filter the address Types of person vendors to be displayed in the **Vendor Lookup**.

```
banner{
    vendor{
        corporateAddressTypes = ['MA',’BU’] // Specify corporate vendor
 address type codes to filter vendors lookup
        personAddressTypes = ['']
         chooseVendorForMeChecked = true // Default setting for Choose
 Vendor For me; true is checked , false is unchecked
    }
}
```

Note: Failure to identify these address types will result in My Requisition not returning any results when a user performs a vendor lookup. If person vendors are not required to be listed, it should not include any address types and should be configured as `personAddressTypes = ['']`.

Consult with your functional users to determine whether to have the **Choose vendor for me** check box set to default **ON** or set to default **OFF** in the My Requisition module.

The following are the two options:

- If the `chooseVendorForMeChecked` is set to *`true`* in the configuration file, then the **Choose vendor for me** check box will be set to default **ON** in My Requisition.

  In the My Requisition module, the user will have the option to manually set the **Choose vendor for me** check box to **OFF** upon which the **Vendor**, **Discount**, and **Currency** fields will become enabled and the user may then proceed to select the vendor of the user’s choice.

- If the `chooseVendorForMeChecked` is set to `false` in the configuration file, then the **Choose vendor for me** check box will be set to default **OFF** in My Requisition and the **Vendor**, **Discount**, and **Currency** fields will be enabled and will permit the user to select the vendor.

  In My Requisition, the user will have the option to manually set the **Choose vendor for me** check-box to **ON** upon which the **Vendor**, **Discount**, and **Currency** fields will be disabled and any data previously selected for these fields reverted.

Consult with your functional users to decide the vendor types to be set at the configuration file level to determine which type of vendor and their corresponding address type(s) need to be displayed in the **Vendor** look-up in My Requisition.

The following vendor types may be configured:

- Corporate – If a **Corporate** vendor type needs to be displayed in the **Vendor** look-up in My Requisition, then the corresponding address type or types need to be defined in the `corporateAddressTypes` section in the configuration file.
  - One or more address types may be defined for the corporate vendor upon which only the defined address types will be displayed as part of the vendor look-up in My Requisition.
  - If nothing is defined in the `corporateAddressTypes` in the configuration file, then no corporate vendors will be displayed for selection in the **Vendor** look up in My Requisition.
- Person – If a **Person** vendor type needs to be displayed in the **Vendor** look-up in My Requisition, then the corresponding address type or types need to be defined in the `personAddressTypes` section in the configuration file.
  - One or more address types may be defined for the person vendor, upon which only the defined address types will be displayed as part of the vendor look-up in My Requisition.
  - If nothing is defined in the `personAddressTypes` in the configuration file, then no person vendors will be displayed for selection in the **Vendor** look up in My Requisition.
- Both – In case both **Corporate** and **Person** vendor types are required, then the corresponding address types of both these vendors need to be defined in their respective sections in the configuration file.

My Requisition: Commodity Text configuration

There is one configuration for commodity related to copying commodity text.

```
banner{
        commodity{
           copyItemText = true // copy commodity text to public comments
    }
}
```

Consult with your functional users to determine whether to copy any commodity text from the commodity master in Banner to commodity item text of the Requisition when the commodity is selected during purchase requisition document creation.

If the `copyItemText` is set to `true` in the configuration file, then any commodity text entered in FOATEXT (General Text Entry) in the commodity master in Banner will be copied to the same commodity when selected in the purchase requisition document within the My Requisition module.

Additionally, in FOATEXT, if the **Print** check box against the commodity text line is set to `ON`', the same text will appear in the **Public Comments** text box for the same commodity in the purchase requisition. If the **Print** check box against the text line is set to `OFF`, the same text will appear in the **Private Comments** text box for the same commodity in the purchase requisition.

If the `copyItemText` is set to `false` in the configuration file, then any commodity text entered in FOATEXT (General Text Entry) in the commodity master in Banner will not be copied to the same commodity selected in the purchase requisition document.

My Requisition: Accounting type setting

Consult with your functional users to determine the accounting type default (document or commodity level) for all new requisitions created in My Requisitions module.

In addition, you can control if Requesters can override the default value, based on how you customize these settings in the configuration.

- Disable accounting type change.

  Options are `true` or `false`. Having this setting as `true` means user cannot change the accounting type. Having this setting as `false` means user can change accounting type from document to commodity or other way.

- Default accounting type.

  Document or commodity (case sensitive). User needs to configure any one of them.

```
banner{
    accounting{
        disable = true // Options true or false
        defaultType = 'document' // Options : document or commodity
 ( case sensitive)
    }
}
```

My Requisition: NSF processing configuration

In a scenario where a purchase requisition has non-sufficient funds and approvals are set to `ON`, one of the following will occur based on the configuration.

```
banner{
    nsfChecking{
     bypassChkOnApprovalOn = false // if false while Approval is ON,
 then on NSF Error, document is not allowed to be completed.
    }
}
```

The application will, either:

- Stop the user from completing the document and sending it to approvals

  (See 'Sample scenario 1')

  OR

- Allow the requisition to be completed

  (See 'Sample scenario 2')

> Sample scenario 1:
>
> `Approval On[Explicit Approval] and Requisition NFS checking On and` **`bypassChkOnApprovalOn = false`**

On completing the requisition, the following error message displays:

`"Insufficient budget for sequence x suspending transaction. Requisition is valid but failed available balance check."`

You cannot complete the document; it will be saved as a draft for future use.

Sample scenario 2:

`Approval On[Explicit Approval] and Requisition NFS checking On and` **`bypassChkOnApprovalOn = true`**

On completing the requisition, the following message displays:

`"Requisition Rxxxxxxx completed successfully"`

The document is complete.

My Finance Query: Dashboard Bar Chart

My Finance Query dashboard shows Net Revenue vs. Net Expenditure Bar chart for those queries, which have include revenue account flag selected.

The following configuration allows you to decide whether the commitment component should be considered, along with the YTD component for the calculating the Net Total of Revenue and Expense. If commitment is not included, calculation will happen only on year-to-date component.

So when `includeCommitment` is set to `true`, then

Net Revenue = Net YTD of all Revenue Accounts + Net Commitments of all Revenue Accounts

Net Expenditure = Net YTD of all Expense Accounts + Net Commitments of all Expense Accounts

Whereas when `includeCommitment` is set to `false`, then

Net Revenue = Net YTD of all Revenue Accounts

Net Expenditure = Net YTD of all Expense Accounts

```
financeQuery{
     barChart{
       includeCommitment=true
      }}
```

My Finance Query: Dashboard Radial Progress Chart

My Finance Query uses a Radial Progress chart to display the available balance percentage for those queries that have Include Revenue Accounts set to `false`.

Color of the chart varies based on the available balance budget percentage. The Administrator can set the range of percentage applicable to a particular color. This allows the institution to enforce institution wide policies around Budget enforcement by tweaking this setting based on the time of the year, and so on.

When modifying the range, do make sure you have all possible percentage values for your institution covered. The following is a set of allowed colors and their default percentage range. Do not edit color name.

```
financeQuery{
pieChart{
     colorPercentageRange=[Green:80..1000, Yellow:60..79,
 LightningYellow:40..59, Orange:20..39, Red:0..19, DeepRed:-1000..-1]
}
```

My Finance Query: Query type configuration

You can specify the list of query types supported in My Finance Query module.

By default, Budget Quick Query will always be available to the user while creating a query. Other query types:

- Budget Status by Account
- Encumbrance Query
- Payroll Expense Detail
- Multi Year Query
- Budget Status by Organizational Hierarchy

In the following configuration, you can disable and enable query type by setting the exclude flag to `true` or `false`. Do not change Query key names. Setting the exclude flag to `true` in the following configuration will disable the corresponding query type. By default, the exclude flag for all query types in the configuration are `false`.

```
financeQuery{
queryTypes{
       excludeQueryTypes [BUDGET_QUERY_ACCOUNT : false,
 BUDGET_QUERY_ORG : false, PAYROLL_EXPENSE_QUERY : false,
 ENCUMBRANCE_QUERY : false, BUDGET_MULTI_YEAR_QUERY : false]
}
```

My Finance Query: Disable Health icon

You can disable the Health icon.

In My Finance Query, while displaying the query results information, the system calculates the available balance percentage value and displays the health of the line item visually using an icon called the Health Icon.

```
queryList {
        showHealthColumn = true
    }
```

My Finance Query: Health Icon percentage configuration

In My Finance Query, while displaying the query results information, the system calculates the Available Balance percentage value and displays the health of the line item visually using an icon called a Health Icon.

Available Balance % = Available Balance / Adjusted Budget * 100

You have the ability to configure the percentage ranges for the available color codes (Red/Yellow/Green)

```
healthIcon {
        colorPercentageRange = [Green: 61..1000, Yellow: 21..60, Red:
 -1000..20]
    }
```

My Finance Query: Excel file format

In My Finance Query, users have the ability to download the information in excel format.

As an Administrator, you have the ability to configure the default format of the excel export that you would like to support. The system supports .xls and .xlsx formats.

```
excelExport {
        fileType = 'xls' // options are xls or xlsx. Default is xls
```

Note: When configured with xlsx, in some versions of Excel, when opening the reports downloaded from My Finance Query for the first time, you may be notified like this:

```
We found a problem with some content in 'FileName.xlsx'. Do you want
us to try to recover as much as we can? If you trust the source of this
workbook, Click Yes
```

Please click **Yes** in such cases. We have observed no loss of data and styles in such scenarios.

Finance web app extensibility

This feature allows the spriden user whose oracle user has WTAILORADMIN role the ability to show and hide fields on the user interface.

Web app extensibility for Unix

To grant web app extensibility for Unix, complete these steps.

Procedure

1. Verify that the following directory structure is available. This directory structure is required for extensibility.

Note: `/u02/` is a sample path which can be decided by the installer.

```
/u02/BanXE/Extensions/ss_ext/extensions
```

```
/u02/BanXE/Extensions/ss_ext/i18n
```

If that structure does not exist, use the following command to create the directory:

```
mkdir /u02/BanXE/Extensions/ss_ext/extensions
```

```
mkdir /u02/BanXE/Extensions/ss_ext/i18n
```

2. Go to directories extensions and i18n, provide read and write access configuration:

```
webAppExtensibility {
    locations {
        extensions = "/u02/BanXE/Extensions/ss_ext/extensions/"
        resources = "/u02/BanXE/Extensions/ss_ext/i18n/"
    }
    adminRoles = "ROLE_SELFSERVICE-WTAILORADMIN_BAN_DEFAULT_M"
  }
```

Web app extensibility for Windows

To grant web app extensibility for Windows, complete these steps.

Procedure

1. Verify that the following directory structure is available. This directory structure is required for extensibility.

   Note: `c:/` is a sample path which can be decided by the installer.

   ```
   C:/BanXE/Extensions/ss_ext/extensions
   ```

   ```
   C:/BanXE/Extensions/ss_ext/i18n
   ```

   If that structure does not exist, use the following command to create the directory:

   ```
   mkdir C:/BanXE/Extensions/ss_ext/extensions
   ```

   ```
   mkdir C:/BanXE/Extensions/ss_ext/i18n
   ```

2. Go to directories extensions and i18n, provide read and write access configuration:

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

Note: You must add a new role WTAILORADMIN for an oracle user.

Configure Banner Document Management

This section contains information for implementing Banner Document Management (BDM) for My Requisitions.

Before you begin

Banner Document Management (BDM) integration for My Requisitions module in Banner Finance Self-Service requires the following:

- Banner Document Management 8.6.0.1
- EMC ApplicationXtender 7.0.260 (7.0 SP1 is preferred), including ApplicationXtender Web Access and ApplicationXtender Web Services

  Note: The source code for the My Requisition integration with BDM is included as part of the Banner Finance Self-Service installation and not contained in the BDM source objects delivered in BDM 8.6.

Recommended additional BDM documentation:

- *Banner Document Management Administration Guide 8.6* or later
- *Banner Document Management Installation Guide 8.6* (if upgrading to BDM 8.6)
- *Banner Document Management Release Guide 8.6*
- *pcr-000132439_ext8060001_readme.txt* – Install instructions for the 8.6.0.1 patch

About this task

In addition, the Banner Finance Self-Service configuration contains parameters pertaining to BDM. A Banner Document Management administrator or the person at your institution authorized to support BDM, is required to assist with creating the BDM shared user account and providing some of the parameters contained in the `FinanceSelfService_configuration.groovy` file.

BDM uses the `B-F-DOCS` application to manage the documents uploaded from the Banner Finance Self-Service page. Additional index fields were added to `B-F-DOCS` to support this integration. For more details on this, refer to the *Banner Document Management Administration Guide 8.6* and the *Banner Document Management Install Guide 8.6*.

The BDM specific configuration for Banner Position Description is done within the `FinanceSelfService_configuration.groovy` file. At the end of this file, you will find the placeholders to update with BDM specific configuration.

To successfully connect to BDM and manage documents, complete the following steps:

Procedure

1. Create a folder for keeping temporary uploaded document files on the PC where the Banner Finance Self-Service application is deployed.

   It is a location where the BDM attachments can be uploaded temporarily. Make sure the application server has read/write permissions on the folder so the uploaded files folder can be accessed to create BDM documents.

Note: The folder created is the value for *bdm.file.location* parameter in the `FinanceSelfService_configuration.groovy` file.

Examples of folder location:

- Windows: `C:/BDM_DOCUMENTS_FOLDER/`
- Linux: `/u02/Tomcat7/BDM_DOCUMENTS_FOLDER`

2. Create a BDM user account in ApplicationXtender.

   Note: My Requisition is a component of the Banner Finance Self-Service product. The BDM user account setup in ApplicationXtender must also be a Banner user and must have a valid FOMPROF record with self-service access to view attachments from the My Requisition page.

   Follow the instructions in the *Banner Document Management Administration Guide* (User and Group Management section) to create a new BDM user in ApplicationXtender. This BDM user, and the password assigned to it, are required in the configuration file you will update in the next step.

   Assigning privileges:

   a) At the global level, select **Multiple Logins** check box.
   b) Select **B-F-DOCS** from the Application list.
   c) Assign the following privileges:
      - Scan/Index Online
      - Batch Scan
      - Batch Index
      - Modify Index
      - Display
      - Delete Doc
      - Add Page
   d) Click **Apply**.
   e) Exit ApplicationXtender Generator.

3. Verify settings in ApplicationXtender Generator.

   The `web.config` file is located in your website's ApplicationXtender IIS Website folder, which is typically `C:\Inetpub\wwwroot\AppXtender\`.

   a) Open the `web.config` file for edit.
   b) Search on the following:
      AxWebServicesDocumentPointer
   c) If the value does not equal `2`, then modify the value as follows:
      `<add key="AXWebServicesDocumentPointer" value="2"/>`
   d) Save.
   e) Locate the ApplicationXtender Web Services `web.config` file.

This is typically `C:\inetpub\wwwroot\AppXtenderServices`.

f) Repeat steps **a.** through **d.**

g) Perform an IISRESET to implement the changes to `web.config`.

4. Update the BDM server connection parameters in the `FinanceSelfService_configuration.groovy` file.

   a) Locate the `FinanceSelfService_configuration.groovy` file in the directory where it was deployed during the installation.

   b) Open the file using a text editor.

   c) Search for BDM.

   d) Update the parameters in the BDM Configurations section.

   The following is the list of parameters to be updated and a description of each parameter. This configuration is used to establish a successful connection with the BDM server and to process the documents:

Parameter	Description
bdm.enabled	• True = The integration with Banner Document Management is being used to create, update, and delete documents. • False = The integration with Banner Document Management is not being used.
bdm.file.location	The temporary file upload directory location of the folder created for the “uploaded documents” in Step 1.
AXWebServicesUrl	Web Service URL of ApplicationXtender. For example: `http://<axserver>:port/appxtender/axservicesinterface.asmx`
AXWebAccessURL	ApplicationXtender logon URL. For example: `http://<axserver>:port/appxtender/`
Username	User name of BDM user created in ApplicationXtender (from Step 2) used to connect to the BDM server.
Password	Refer to the section Configure encrypted BDM password on page 45
BdmDataSource	The BDM/ApplicationXtender Data Source name.
AppName	The AppName is *B-F-DOCS*.

Example of BDM Configurations section in `FinanceSelfService_configuration.groovy` file, where *<UPDATEME>* for each parameter is the value as described above:

```
/** ******************************************
                  ++++BDM Configurations ++++
************************************************** **/
bdm.enabled       = false
bdm.file.location = "<UPDATEME>"
bdmserver {
AXWebServicesUrl  = http://<UPDATEME>/appxtender/
axservicesinterface.asmx'
AXWebAccessURL    = 'http://<UPDATEME>/appxtender/’
Username          = '<UPDATEME>'
Password          = '<UPDATEME>'
BdmDataSource     = '<UPDATEME>'
AppName           = '<UPDATEME>'
}
```

Example of the BDM Configurations settings in `FinanceSelfService_configuration.groovy` file where sample information for the *<UPDATE ME>* values are filled in:

```
bdm.enabled       = true
bdm.file.location = "/tmp/bdm_upload_locn"
bdmserver {
AXWebServicesUrl  = 'http://AX_TEST_SERVER/appxtender/
axservicesinterface.asmx'
AXWebAccessURL    = 'http://AX_TEST_SERVER/appxtender/'
Username          = 'FINBDMUSR'
Password          = 'MYPASSWRD'
BdmDataSource     = 'TESTDB'
AppName           = 'B-F-DOCS'
}
```

Configure encrypted BDM password

This section provides an option to update the BDM password when changes to the BDM password are noted within the BDM environment.

About this task

In earlier releases of Banner Finance Self-Service, `BDMPassword` and `BDMKeyPassword` were supplied in clear text format. Banner Finance Self-Service 9.1.1 forward, `BDMPassword` is picked from the `FinanceSelfService_configuration.groovy` file, and `BDMKeyPassword` is picked from a database where it is stored in an encrypted format.

`BDMKeyPassword` is a BDM Crypto key maintained on the **Document Management Systems Settings (EXAINST)** page.

Refer to the Implement Secure Logon configuration step in the *Banner Document Management Administration Guide* for information about managing `BDMKeyPassword` (Crypto Key). This guide

is available for download from the Banner Document Management Documentation Library on the Ellucian Support Center.

Procedure

1. Copy the `encrypt_input_string.sql` script from the `general/plus` directory on `BANNER_HOME` to a location that has SQLPlus access.
2. Connect to SQLPlus as baninst1 user.

   ```
   sqlplus baninst1/<baninst1_password>
   SQL> start encrypt_input_string.sql
   ```

3. When prompted to enter the BDM password, enter the password of the BDM user created in ApplicationXtender that is used to connect to the BDM server.
4. Type the BDM password and press **Enter**.

   The script encrypts the password provided and gives the result as `dbms_output`.
5. Copy the script output and keep this text.

   This is the encrypted password that will be used while updating the password parameter in the BDM configuration in `FinanceSelfService_configuration.groovy`.

Results

When this setup is completed, along with other BDM parameters setup, the Banner Finance Self-Service application is ready to use BDM features.

Note: This script must be rerun whenever a BDM password is updated or an encryption key (Hex Value) is updated in Institution Profile of GSASECR.

Configure application-specific settings by modifying message.properties file

To configure application-specific settings, in addition to modifying the groovy file, you can also modify the `message.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.

Note: For more information about Banner Finance Self-Service configurations, see *Banner Finance Self-Service Handbook*.

Customize the messages.properties file

To configure application settings using the `message.properties` file, perform the following steps.

Procedure

1. Copy the WAR file of the consolidated application `FinanceSelfService-9.1.1.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` FinanceSelfService `-9.1.1.war`
3. Delete the existing WAR file.
4. Locate the UI plugins in the WEB-INF folder:
   - `banner_finance_budget_availability_ui.git`
   - `banner_finance_procurement_ui.git`
5. Modify the application specific `messages.properties` within the appropriate UI plugin as needed.
6. At a command prompt, run the following JAR command.

   `jar -cvf` FinanceSelfService `-9.1.1.war`

   This command will bundle all the files and folders in the current folder.

   The new WAR file generates and is ready for deployment.

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 `FinanceSelfService/current/i18n` folder, find the `message.properties` file for your language.

   The `messages.properties` file is located at `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: `$lastName`
- Surname prefix: `$surnamePrefix`

For example:

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

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 `FinanceSelfService/current/i18n` folder, find the `messages.properties` file for your language.

   The `messages.properties` file is located at `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`
   - 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 `FinanceSelfService/current/i18n` folder, find the `messages.properties` file for your language.

   The `messages.properties` file is located at `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 `FinanceSelfService\current\i18n` directory.

Related Links

Date format keys on page 49

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 `FinanceSelfService/current/i18n` folder, find the `messages.properties` file for your language.

   The `messages.properties` file is located at `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`
   - `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=islamic`
   - `default.calendar2=gregorian`

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 `FinanceSelfService/current/instance/css/bannerSelfService-custom.css` containing the custom CSS directives.
2. If the `FinanceSelfService/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 `FinanceSelfService/current/instance/css/images` directory.

   Also, 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 as follows:
   `FinanceSelfService/current/instance/css/bannerSelfService-custom.css`
2. In the CSS file you have created, enter the following:
   ```
   .institutionalBranding {background-image: url("./images/
   institutionLogo.png");}
   ```
3. Replace the logo image in the following directory:
   `FinanceSelfService/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 54

Customize JavaScript

You can customize the java script to modify the behavior of the web pages of your application by placing your JavaScript file named `bannerSelfService-custom.js` in the following location: `FinanceSelfService/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.

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_configuration.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:
   `FinanceSelfService/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 FinanceSelfService/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 `FinanceSelfService/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. Ellucian recommends that you secure the web application traffic by using the 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:

- Configure the Tomcat server on page 55
- Configure the WebLogic server on page 62

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 `<FinanceSelfService>\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.

   Warning! This step must be executed on a Linux system only.

   For example, in the `CATALINA_HOME=/opt/apache-tomcat-7.0.xx`, the ***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
   ```

   Tip: If you are deploying multiple Banner 9.x applications to the same Tomcat server, increase `-Xmx` by `2g` and `-XX:MaxPermSize` by `128m`.

Note: You should deploy Banner 9.x administrative applications to one Tomcat server instance and Banner 9.x self-service applications to a separate Tomcat server instance.

You can define this variable in the account's profile startup script or you can add this definition into the following:

- `$CATALINA_HOME/bin/catalina.sh` for Linux
- `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).
   Refer to Configure Java management extension on page 58 for more information.
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.

```
<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="30"
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="30"
validationQuery="select 1 from dual"
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 `<FinanceSelfService>/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:

Linux:

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

Windows:

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

j) Browse to `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_FINANCE_SSB_CONFIG= /PRODUCT_HOME/
<FinanceSelfService>/current/instance/
config/FinanceSelfService_configuration.groovy".
```

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 9.x self-service applications 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`.
- User name – 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 6.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 6.x. Refer to the Tomcat documentation specific to your release for information on enabling access to provide the appropriate role to a user account for deployment.

Procedure

1. Navigate to the `FinanceSelfService\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 using the following URL:
   `http://servername:<port>/<application_context_name>`
   `<application_context_name>` is a placeholder for the URL path to which the application is deployed on the web server.

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 the following:
   `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 Tomcat server and then deploy the WAR file to the Tomcat 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
    ```

    Tip: 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`.

    Note: You should deploy Banner 9.x administrative applications to one WebLogic server instance and Banner 9.x self-service applications to a separate WebLogic server instance.

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_FINANCE_SSB_CONFIG=<full file path to
FinanceSelfService_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**.

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.

   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 the `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.

   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 the `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

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, `Banner9DS`.
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

```
FinanceSelfService/current/dist/
FinanceSelfService-9.1.1.war
```

To

```
FinanceSelfService/current/dist/
FinanceSelfService.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 `<FinanceSelfService>/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/<application_context_name>`.

    `<application_context_name>` is a placeholder for the URL path to which the application is deployed on the web server.
14. Log in to the application using a valid username and password.

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. Create a keystore file (`*.jks`) with the name used in step 2.
   See Create a keystore (*.jks) file on page 82 for more information.
2. Place this file in the location specified in the following key value:

```
"grails.plugin.springsecurity.saml.keyManager.storeFile =
 'classpath:security/financessb.jks'"
```

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 Set up SAML SSO configuration on page 77.
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 the specific application.
      - `<HOSTNAME>`: Application host name
      - `<PORT>`: Deployed Application port number
      - `<ALIAS_NAME>`: Service provider ID set in Ellucian Ethos Identity service provider setup
      - `<EXTRACTED_DATA>`:

Extract a X509 Certificate key from the keystore for `banner-<short-appName>-sp.xml` (See Extract a X509 certificate key on page 82 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:HTTP-Redirect"
 Location="http://<HOSTNAME>:<PORT>/<APPLICATION_NAME>/saml/
SingleLogout/alias/<ALIAS_NAME>"/>

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

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

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

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

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

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

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

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

</md:SPSSODescriptor>

</md:EntityDescriptor>
```

Create an identity service provider file

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

Procedure

1. Create a file `banner-<short-appName>-idp.xml` at the folder path mentioned in SAML SSO configuration on page 78.
2. Edit the `banner-<short-appName>-idp.xml` file for the identity provider configuration. The file contains the identity provider information configured in the Ellucian Ethos Identity server over which the configured application sends the SAML request and receives the SAML response.
   a) Replace the parameters below with the configured values for the specific application.
      - `<HOSTNAME>`: Ellucian Ethos Identity server host name
      - `<PORT>`: Deployed Ellucian Ethos Identity server port number
      - `<ALIAS_NAME>`: Service provider ID set in Ellucian Ethos Identity service provider setup
      - `<EXTRACTED_DATA>`:

        Extract X509 certificate Data for `banner-<short-appName>-idp.xml` (See Extract X509 certificate data on page 84 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="https://<HOSTNAME>:<PORT>/samlsso"
 cacheDuration="PT1440M">

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

<md:KeyDescriptor use="signing">

<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

<ds:X509Data>

<ds:X509Certificate>

<EXTRACTED_DATA>

</ds:X509Certificate>

</ds:X509Data>
```



```
</ds:KeyInfo>

</md:KeyDescriptor>

<md:KeyDescriptor use="encryption">

<ds:KeyInfo xmlns:ds="http://www.w3.org/2000/09/xmldsig#">

<ds:X509Data>

<ds:X509Certificate>

<EXTRACTED_DATA>

</ds:X509Certificate>

</ds:X509Data>

</ds:KeyInfo>

</md:KeyDescriptor>

<md:SingleLogoutService Location="https://<HOSTNAME>:<PORT>/
samlsso"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"/>

<md:SingleLogoutService Location="https://<HOSTNAME>:<PORT>/
samlsso"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"/>

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

<md:SingleSignOnService Location="https://<HOSTNAME>:<PORT>/
samlsso"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-POST"/>

<md:SingleSignOnService Location="https://<HOSTNAME>:<PORT>/
samlsso"
Binding="urn:oasis:names:tc:SAML:2.0:bindings:HTTP-Redirect"/>

</md:IDPSSODescriptor>

<md:ContactPerson contactType="administrative"/>

</md:EntityDescriptor>
```

Add IDP certificate entry to the .jks file

During SAML configuration, you must add the IDP certificate entry.

About this task

Add the IDP certificate entry to the new `.jks` file you created as part of this task Create a keystore (*.jks) file on page 82.

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 financessb.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
financessb.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
]
]
*******************************************
*******************************************
```

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

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

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

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

Set up SAML SSO configuration

The `FinanceSelfService\current\instance\config` directory contains the `FinanceSelfService_configuration.groovy` file.

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

Authentication provider name

Authentication provider name is the name identifying the application authentication mechanism. Values are `cas` or `saml`.

Specify `saml` to indicate that the application will use SAML SSO protocol for authentication as shown in the following example:

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

There are three ways the application can handle logouts. When using SAML, logging out directs the user to a custom logout page.

- 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.

SAML SSO configuration

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

> The following properties describe the configurations required for the application to work in SAML SSO protocol.
>
> **Note:** Uncomment this section when SAML 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 it is set to `true`, indicating local logout, the application will not notify the Identity Server to log the user out from all applications.
`grails.plugin.springsecurity.auth.loginFormUrl`	Pre-populated to provide the Login URL.
`grails.plugin.springsecurity.saml.afterLogoutUrl`	Pre-populated to provide the Logout URL.
`grails.plugin.springsecurity.saml.keyManager.defaultKey`	Key name used at the time of key-store creation (Create a keystore (*.jks) file on page 82). Example: `financessb.jks`
`grails.plugin.springsecurity.saml.keyManager.storeFile`	Location of the keystore file. • This could be a classpath. Example: `classpath:financessb.jks` -OR- • It could be an absolute location on the machine. – Example 1: `file: C://temp/financessb.jks` -OR- – Example 2: `u02/financessb.jks`
`grails.plugin.springsecurity.saml.keyManager.storePass`	Password used to decrypt the keys in the keystore created above.

Property	Description
`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. Example: `security/sp.xml` -OR- • It could be a absolute location on the machine. Example: `file: C://temp/banner-sp file:/home/u02/banner-sp.xml`
`grails.plugin.springsecurity.saml.metadata.providers`	Key value pair mapping to validate the identity provider configured in `banner-<short-appName>-idp.xml`. Example: `adfs : 'security/banner-idp.xml'` Possible keys are adfs, Okta, Shibb, etc.
`grails.plugin.springsecurity.saml.metadata.defaultIdp`	Provide the default IDP to be used from the IDP providers set. This is the same value specified above for the key.
`grails.plugin.springsecurity.saml.metadata.sp.defaults`	• `local`: Pre-populated to indicate value to be picked up. • `alias`: An alias name that is unique to this application. Example: `banner-<application-shortname>-sp` • `securityProfile`: Pre-populated value. • `signingKey`: A key used to sign the messages that is unique to this application. Example: `banner-<application-shortname>-sp` • `encryptionKey`: A key to encrypt the message that is unique to this application. Example: `banner-<application-shortname>-sp` • `tlsKey`: A `tls` key that is unique to this application. Example: `banner-<application-shortname>-sp`

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

SAML SSO configuration code sample

Uncomment this section if you are using SAML.

```
/************************************************************
*                                                           *
*                  SAML2 SSO CONFIGURATION                  *
* Uncomment this section if Registration is configured with SAML2 SSO *
************************************************************/
/*
grails.plugin.springsecurity.saml.active = false
grails.plugin.springsecurity.auth.loginFormUrl = '/saml/login'
grails.plugin.springsecurity.saml.afterLogoutUrl ='/logout/
customLogout'

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

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

grails.plugin.springsecurity.saml.metadata.sp.file = 'se-curity/banner-
sp.xml'               // for unix based '/home/u02/banner-sp.xml'
grails.plugin.springsecurity.saml.metadata.providers = [adfs:
 'security//banner-idp.xml']    // for unix based '/home/u02/banner-
idp.xml'
grails.plugin.springsecurity.saml.metadata.defaultIdp = 'adfs'
grails.plugin.springsecurity.saml.metadata.sp.defaults = [
local: true,
```

```
alias: 'banner-financessb-sp',
                            // banner-financessb-sp is the value set in
 Ellucian Ethos Identity Service provider setup
securityProfile: 'metaiop',
signingKey: 'banner-financessb-sp',
                            // banner-financessb-sp is the value set in
 Ellucian Ethos Identity Service provider setup
encryptionKey: 'banner-financessb-sp',
                            // banner-financessb-sp is the value set in
 Ellucian Ethos Identity Service provider setup
tlsKey: 'banner-financessb-sp',
                            // banner-financessb-sp is the value set in
 Ellucian Ethos Identity Service provider setup
requireArtifactResolveSigned: false,
requireLogoutRequestSigned: false,
requireLogoutResponseSigned: false
]
*/
```

Note: After performing the above configuration changes, rebuild and redeploy the WAR file to the corresponding web application servers. Refer to the following sections for instructions:

- Regenerate the WAR file on page 54.
- Configure the web application server and deploy the WAR file on page 55.

SAML 2.0 configuration sub-tasks

You need to perform few tasks to complete the SAML configuration.

Note: Few Banner applications support SAML configuration while some of them do not support SAML.

Create a keystore (*.jks) file

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

Procedure

1. Create a keystore file (`*.jks`) with the name used in step 2.
2. Place this file in the location specified in the following key value:

```
"grails.plugin.springsecurity.saml.keyManager.storeFile =
 'classpath:security/financessb.jks'"
```

Extract a X509 certificate key

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

Procedure

1. With the `financessb.jks` file you created, execute the command below to check which certificates are in a Java keystone.

   See Create a keystore (*.jks) file on page 82 for more information.

```
C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -list -v -keystore
 financessb.jks
Enter keystore password:
Keystore type: JKS
Keystore provider: SUN
Your keystore contains 1 entry
Alias name: mykey
Creation date: Apr 6, 2015
Entry type: PrivateKeyEntry
Certificate chain length: 1
Certificate[1]:
Owner: CN=John Doe, OU=Ellucian, O=Ellucian, L=Malvern, ST=PA, C=US
Issuer: CN=John Doe, OU=Ellucian, O=Ellucian, L=Malvern, ST=PA, C=US
Serial number: 78548b7
```

```
Valid from: Mon Apr 06 12:52:39 IST 2015 until: Thu Mar 31 12:52:39
 IST 2016
Certificate fingerprints:
MD5:  5D:55:F4:18:3D:CF:AE:5A:27:B8:85:68:42:47:CA:76
SHA1: CD:09:04:F5:01:60:14:CC:DF:48:07:4A:93:99:17:BF:10:83:F3:55
SHA256:
 79:5A:7F:0C:A4:B1:0E:30:9C:B0:DD:87:2C:CA:19:A1:0E:89:29:2F:95:A1:35:
E9:EC:A2:AA:B9: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 financessb.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 financessb.jks
Enter keystore password: password
Certificate stored in file <mykey.crt>
```

3. Execute the following command to get the X509 certificate:

```
C:\Program Files\Java\jdk1.7.0_67\jre\bin>keytool -printcert -rfc -
file mykey.crt
-----BEGIN CERTIFICATE-----
MIIDfTCCAmWgAwIBAgIEB4VItzANBgkqhkiG9w0BAQsFADBvMQswCQYDVQQGEwJJTjELMAkGA1UE
CBMCSU4xEjAQBgNVBAcTCUJhbmdhbG9yZTERMA8GA1UEChMIRWxsdWNpYW4xETAPBgNVBAsTCEVs
bHVjaWFuMRkwFwYDVQQDExBTcGhvb3J0aSBBY2hhcnlhMB4XDTE1MDQwNjA3MjIzOVoXDTE2MDMz
MTA3MjIzOVowbzELMAkGA1UEBhMCSU4xCzAJBgNVBAgTAklOMRIwEAYDVQQHEwlCYW5nYWxvcmUx
ETAPBgNVBAoTCEVsbHVjaWFuMREwDwYDVQQLEwhFbGx1Y2lhbjEZMBcGA1UEAxMQU3Bob29ydGkg
QWNoYXJ5YTCCASIwDQYJKoZIhvcNAQEBBQADggEPADCCAQoCggEBAN2cS
+2OX37HioFzwOWLm/S0
F+zt6ldtLfmHc16V9iqkZChkMiXpKgXVPqGLFjBrhwfsWtuMfRy2NYf3forEDFTaV4/
fLXRo+Npd
xTfqWhuZTafDJEyKQc57KY8G3feg1CSjfKkCk1LF
+zbGClHQ0bg1dwUjJlp7eKjWM0rbsKMd5pZ7
0tGAPcYsi6MtGvJupaVhy3jNTDg+kh4/D92y/mTaLlCR4QQr1qlU9+H
+it3m9jiDrZ7svrdBlsDN
1BVcXDooqUGTuc10IBxYEsb7hFucSFpdJnGJvbg35ll9K9F5S8lEmiQmeOUQ1gQe2Ow01kFl56Qz
4evM0xgeskNid9sCAwEAAaMhMB8wHQYDVR0OBBYEFLKs1wkBFSKjMgiGZOglWhXLoMbZMA0GCSqG
SIb3DQEBCwUAA4IBAQANJbYRTcMwhrETz+mo
+n1okrXIs118AIm7s1yJJd1nyJuaKrn7DcPPLzy/
```

```
RjHGKP02uLiupgqar+UUaPqSZjJXSzktLLyq7H6DRrW0Jp2rw48a+Kou
+XOvQ8ZWR9ZXIa1XoAoD
PaSSE2omcVOVGZmQKUYardVeSvQth3IVMW9w9Jl
+DuavXavVjIx5IN6RRhXGfaJjQLKFzIDqZNAp
OcxMEKXHOUqj0ksTRARLpKWSPu7gFOWO/6qapNp5l8rlPjnVxDhqHqCKC3E40VI5n+C
+KJHZQqab
Tfhd6erqEy7S1Cazr655Yq22Jm6L7IXsXgpRwmZnoietLsrFIRyPe1DY
-----END CERTIFICATE-----
```

Extract X509 certificate data

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

Procedure

1. Go to the deployed WSO2 server / Ellucian Ethos Identity server.
   For example: `C:\>cd C:\work\eis\`
2. Navigate to the server certificate location.
   By default, it is located at: `$EIS_HOME\repository\resources\security`
3. Execute the following command, where `saml-idp.cer` is the certificate file in the server:
   ```
   C:\work\eis\repository\resources\security>keytool -printcert -rfc -
   file saml-idp.cer
   ```
   This would return a value similar to the following:

   ```
   -----BEGIN CERTIFICATE-----
   MIICdDCCAd2gAwIBAgIEEsIIcDANBgkqhkiG9w0BAQsFADBtMRAwDgYDVQQGEwdVbmtub3duMRAw
   DgYDVQQIEwdVbmtub3duMRAwDgYDVQQHEwdVbmtub3duMRAwDgYDVQQKEwdVbmtub3duMRAwDgYD
   VQQLEwdVbmtub3duMREwDwYDVQQDEwhXU08yIElEUDAeFw0xNDA4MTgxMzA3NTFaFw0xNTA4MTgx
   MzA3NTFaMG0xEDAOBgNVBAYTB1Vua25vd24xEDAOBgNVBAgTB1Vua25vd24xEDAOBgNVBAcTB1Vu
   a25vd24xEDAOBgNVBAoTB1Vua25vd24xEDAOBgNVBAsTB1Vua25vd24xETAPBgNVBAMTCFdTTzIg
   SURQMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC44MMcoAPb+aR8l5gtTsSb
   +SzslHFESmKL
   wO1+SAY6p2iiO
   +G9qR/511ufCzKWrMdMMpoCJLC0myDwoUuGvk0dycTpm5NwUX6CnqDtYhtGkYg8
   JT
   +LtG67k6yjXNa9wrE6VBJynDDPnlL8gLUl9ZCIFrmevJ75rOCaLsoFsghHPwIDAQABoyEwHzAd
   BgNVHQ4EFgQULyCNnvW9Ngy5zM7Waf205mR11ZAwDQYJKoZIhvcNAQELBQADgYEApWlSy2GUSaHM
   Kkc8XZmdQ0//SId8DKRKaFaZW388K4dJGTSWUnzq4iCWFrAN9O4D1DBnNE
   +dCDEmV8HvmyQBedsG
   JnAre0VisqKz9CjIELcGUaEABKwkOgLe1YyqV29vS4Y3PuTxAhbkyphFb5PxjHDHH/
   WLQ8pOTbsC
   vX4wO04=
   -----END CERTIFICATE-----
   ```

   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 ~/saml-idp.cer
```

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

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

```
C:\work\eis\repository\resources\security>keytool -export -keystore
 cacerts -rfc -alias mykey
Enter keystore password:
-----BEGIN CERTIFICATE-----
MIICdDCCAd2gAwIBAgIEEsIIcDANBgkqhkiG9w0BAQsFADBtMRAwDgYDVQQGEwdVbmtub3duMRAw
DgYDVQQIEwdVbmtub3duMRAwDgYDVQQHEwdVbmtub3duMRAwDgYDVQQKEwdVbmtub3duMRAwDgYD
VQQLEwdVbmtub3duMREwDwYDVQQDEwhXU08yIElEUDAeFw0xNDA4MTgxMzA3NTFaFw0xNTA4MTgx
MzA3NTFaMG0xEDAOBgNVBAYTB1Vua25vd24xEDAOBgNVBAgTB1Vua25vd24xEDAOBgNVBAcTB1Vu
a25vd24xEDAOBgNVBAoTB1Vua25vd24xEDAOBgNVBAsTB1Vua25vd24xETAPBgNVBAMTCFdTTzIg
SURQMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC44MMcoAPb+aR8l5gtTsSb
+SzslHFESmKL
wO1+SAY6p2iiO
+G9qR/511ufCzKWrMdMMpoCJLC0myDwoUuGvk0dycTpm5NwUX6CnqDtYhtGkYg8
JT
+LtG67k6yjXNa9wrE6VBJynDDPnlL8gLUl9ZCIFrmevJ75rOCaLsoFsghHPwIDAQABoyEwHzAd
BgNVHQ4EFgQULyCNnvW9Ngy5zM7Waf205mR11ZAwDQYJKoZIhvcNAQELBQADgYEApWlSy2GUSaHM
Kkc8XZmdQ0//SId8DKRKaFaZW388K4dJGTSWUnzq4iCWFrAN9O4D1DBnNE
+dCDEmV8HvmyQBedsG
JnAre0VisqKz9CjIELcGUaEABKwkOgLe1YyqV29vS4Y3PuTxAhbkyphFb5PxjHDHH/
WLQ8pOTbsC
vX4wO04=
-----END CERTIFICATE-----
```

Add IDP certificate entry to the .jks file

During SAML configuration, you must add the IDP certificate entry.

About this task

Add the IDP certificate entry to the new `.jks` file you created as part of this task Create a keystore (*.jks) file on page 82.

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 financessb.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
financessb.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
]
]
*******************************************
*******************************************
```

Add service provider in Ellucian Ethos Identity server

You need to add an Ellucian Ethos Identity server as part of the SAML configuration.

Procedure

1. Log in to the Ellucian Ethos Identity Management Console.
2. Click the **Main** tab.
3. Under **Service Providers**, click **Add**.
4. On the **Add Service Provider** screen, enter a service provider name in the **Service Provider Name** field.
   You can also add an optional description in the **Description** field.
5. Click **Register** to add the service provider.

Add SAML settings for Ellucian Ethos Identity server

Add the SAML settings for the integrating application.

Procedure

1. On the **Service Providers** screen, expand the **Inbound Authentication Configuration** panel.
2. Expand the **SAML2 Web SSO Configuration** panel.
3. Click the **Configure** link.
4. 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 `FinanceSelfService_configuration.groovy` file.
5. Enter a valid **Assertion Consumer URL** where the browser redirects the SAML Response after authentication.
6. Enter a valid **NameID** format supported by Ellucian Ethos Identity.
   The following values can be used:
   - `urn:oasis:names:tc:SAML:2.0:nameid-format:persistent`
   - `urn:oasis:names:tc:SAML:2.0:nameid-format:transient`
   - `urn:oasis:names:tc:SAML:1.1:nameid-format:emailAddress`
   - `urn:oasis:names:tc:SAML:1.1:nameid-format:unspecified`
   - `urn:oasis:names:tc:SAML:1.1:nameid-format:X509SubjectName`

- `urn:oasis:names:tc:SAML:1.1:nameid-format:WindowsDomainQualifiedName`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:kerberos`
- `urn:oasis:names:tc:SAML:2.0:nameid-format:entity`

7. Select **Use fully qualified username** in the NameID.

   In most cases, this can be left unselected.

   This lets you preserve the user store domain and user ID in the SAML 2.0 Response.
8. Select **Enable Response Signing** to sign the SAML 2.0 Responses returned after the authentication process.
9. Select **Enable Assertion Signing** to sign the SAML 2.0 Assertions returned after the authentication.
10. Select **Enable Signature Validation in Authentication Requests and Logout Requests**.

    This ensures that the identity provider validates the signature of the SAML 2.0 Authentication and Logout Requests that are sent by the service provider.
11. Select **Assertion Encryption** to encrypt the assertions.
12. 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.
13. Select **Enable Single Logout**.

    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`.

    This terminates all sessions across all authenticated service providers when the user signs out from one server.
14. Select **Enable Attribute Profile**.

    This adds a basic attribute profile where the identity provider can include the user's attributes in the SAML Assertions as part of the attribute statement.
15. 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.
16. Select **Enable Audience Restriction** to restrict the audience.
17. Add audience members using the **Audience** text box.
18. Click **Add Audience**.
19. Select **Enable IdP Initiated SSO**.

    This will not require the service provider to send the SAML 2.0 Request.
20. Click **Register**.

    This will save your settings and return you to the **Service Provider Configuration** page.
21. 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 Ellucian Ethos 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 Ellucian Ethos Identity Management Console.
2. Click the **Main** tab.
3. Under the Identity section, in the Main menu, click **List** under Identity Providers.
4. Enter a service provider name in the **Service Provider Name** field.
   You can also add an optional description in the **Description** field.
5. Click **List** under Identity Providers.
6. Click **Resident Identity Provider**.
7. Expand the **Inbound Authentication Configuration** panel.
8. Expand the **SAML2 Web SSO Configuration** panel.
9. Enter a valid identity provider name or URL to be used by all service providers.
10. Click **Update** to save the settings.