ellucian

Banner Advancement Connector
Installation and Administration Guide

Release 8.7
December 2013

Prepared by: Ellucian
4375 Fair Lakes Court
Fairfax, Virginia 22033
United States of America

Revision History

Publication Date	Summary
December 2013	New version that supports Banner Advancement 8.7 software.

Banner Advancement Connector 8.7 Installation and Administration Guide

Contents

Chapter 1 Preinstallation Considerations 5

What is the Advancement Connector? 5

Components 5

Implementation for iModules Encompass 6

Software requirements 6

Application server considerations 6

Considerations for Banner Advancement 7

Resource requirements 7

Shared directory for data loads 7

Server security 8

Security considerations 8

Encrypted credentials 9

Unencrypted credentials 9

Chapter 2 Installation and Configuration 11

Step overview 12

Step details 12

Preparation 13

Step 1 Extract artifacts from the zip file 13

Phase 1 - Install initiator jar on Forms Server 14

Phase 2 - Configure Connector properties 15

Step 1 Unpack the Connector EAR file 15

Step 2 Configure the Advancement Connector properties 15

Step 3 Configure the Encompass properties 19
Step 4 Configure logging for Oracle Fusion Middleware/WebLogic Server 23
Step 5 Configure unencrypted Encompass credentials 24
Step 6 Configure encrypted Encompass credentials 24
Step 7 Repack the connector ear file 26

Phase 3 - Installation on Oracle WebLogic Server 11g 27

Step 1 Create a Managed Server 27
Step 2 Configure the Managed Server 27
Step 3 Define the data source 28
Step 4 Install the connector 29

Phase 4 - Final configuration and test 31

Step 1 Enable decryption 31
Step 2 Test the Advancement Connector 32
Step 3 Set the Advancement Connector URL in Banner Advancement 32

Chapter 3 Administration 35

Set the decryption password after a restart 35

Test the Encompass implementation 36

View requests made to the Advancement Connector 37

Change the credential provider implementation 38

Change the Encompass operation implementations 39

Change the Advancement Connector implementation 40

Appendix A Initiation Parameters 41

Parameters that are passed 41

Troubleshooting 45

1 Preinstallation Considerations

This chapter introduces the Advancement Connector and discusses some deployment options that you must consider before you install the Advancement Connector.

What is the Advancement Connector?

The Advancement Connector provides a configurable, extensible way to exchange data between Banner® Advancement or Ellucian Advance and a partner system. The connector can perform the following processing:

- Move data from your advancement system to the partner system (push).
- Retrieve data from the partner system (pull). Administrative users can review the retrieved data before the data changes are saved to your advancement system.
- Write advancement system data changes to a delimited file, such as CSV, if that is the desired or necessary mechanism for moving the data changes from your advancement system to the partner system.
- Schedule push and pull operations for automatic execution on a repeating basis.

The Advancement Connector is general purpose. It can be used with different partner systems by implementing the Web service that interacts with that partner system.

Components

The Advancement Connector includes the following components:

- Database tables that are used to transfer data from/to the advancement system, track the status of the transfer transactions, and define the advancement system data elements that are transferred from/to the partner system.
- Advancement system forms that are used to create the profiles, initiate the transfers, monitor the transfers, review data changes from the partner system, and make decisions before the data changes are saved to the advancement system.
- Oracle database package that interacts with the forms to select and place advancement system data into a transfer table, or to take data from a transfer table and update the advancement system after the data has been reviewed.
- A Java web application loaded on an application server that manages the data transfers between the transfer tables and the partner system. This application includes the scheduler.

- A utility to encrypt the login and password credentials that are used to connect to the partner system, when that connection requires credentials (for example, a Web service request that requires credentials).
- For Banner Advancement, a Java library loaded on the Oracle Forms server that enables communication between the Banner form that initiates transfer transactions and the Java application that manages the transfer transactions.

Implementation for iModules Encompass

Ellucian provides an Advancement Connector implementation that exchanges data between Banner Advancement or Ellucian Advance and Encompass, a Web-based alumni community application that is available from iModules Software (www.imodules.com). Encompass is a hosted application that your institution licenses from iModules. The implementation for Encompass uses web services made available by iModules to manage the data transfer from/to the advancement system transfer tables. In this case, the Advancement Connector is a client of the Web services, not a provider of the Web services.

Release 8.6.1 of the Advancement Connector has been updated to use the iModules Encompass web services version 2.1. Previous releases used web services version 2.0. The changeover to the 2.1 services is automatic with the upgrade of the connector.

Software requirements

The Advancement Connector requires the following software:

- Banner Advancement 8.6.1
- Java 6 update 43 or later, or Java 7 update 17 or later
- Oracle Fusion Middleware (OFM) 11g with WebLogic Server 10.3.4 - 10.3.6
- Oracle JDBC drivers for Java 6/7

Application server considerations

The Advancement Connector communicates with the iModules Encompass web services over the Internet. Therefore the server to which you deploy the connector must have access to the Internet. The connector communicates to the Encompass web services using standard HTTPS protocol.

Considerations for Banner Advancement

The Advancement Connector for Banner Advancement 8.6.1 is supported on the following servers:

- Forms Server: Oracle Forms Server 11g
- Application Server: Oracle Fusion Middleware 11g/WebLogic Server 10.3.4, 10.3.5, or 10.3.6.

Ellucian recommends that you use a managed WebLogic Server instance with an Oracle Fusion Middleware 11 instance. Oracle Fusion Middleware 11/WebLogic Server 10.3.5 uses Java 6 by default. WebLogic Server 10.3.6 uses Java 6 or 7.

Resource requirements

Under typical conditions, the Advancement Connector does not require a dedicated, high-performance system. Typically, a limited number of administrative staff members can initiate data transfers manually. It is rare that multiple persons would initiate transfers simultaneously. Data transfers are asynchronous, meaning the administrative user does not have to wait for the entire transfer to complete before starting another task. Scheduled data transfers take place automatically at the scheduled times.

A data transfer usually completes within a few seconds to a few minutes. A heavy load on the partner system might produce a response time at the longer end of the range.

For these reasons, you can deploy the Advancement Connector into an existing server. The Advancement Connector can co-exist with other applications running on the same application server.

However, the Connector Application at times needs memory resources allocated to it for large operations such as writing many entity records to a CSV file. Ellucian recommends running the Connector on a server that has at least 3-4 GB of RAM.

Shared directory for data loads

When large amounts of data must be loaded to the partner system, the user selects the CSV output format option and the Advancement Connector writes a flat file to a shared directory. This typically occurs for the initial data load. The shared directory must be writable by the Advancement Connector and accessible to the administrative users who transmit the file to the partner system. You specify the actual directory in the application configuration process.

Server security

Banner Advancement sends transfer requests to the Advancement Connector via HTTP get or post operations emitted from the forms server. These operations are internal to your institution, between the advancement system forms server and the application server that hosts the Advancement Connector. The information that passes this command does not contain any sensitive or personally identifiable information. If you deploy the Advancement Connector on the same Oracle Fusion Middleware instance as the Advancement forms server, then the communication can be arranged to be completely internal to that server by specifying `localhost` in the URL.

In turn, the Advancement Connector communicates with the iModules Encompass Web services over the Internet using HTTPS protocol. Therefore, the server you deploy the connector to must have access to the Internet with the ability to communicate via HTTPS. You do not need to configure the application server to negotiate HTTPS. The Encompass web service site is managing the HTTPS communication.

As part of the security configuration, if you elect to encrypt the Encompass Web Service logon credentials, you must supply a password to a Java servlet after starting the application. This task uses a browser, and should communicate to the application server using HTTPS. The application server then needs to be configured for HTTPS.

Security considerations

When used with Encompass, the Advancement Connector must hold login and password credentials for accessing the iModules Web services. The credentials should be stored securely, but the connector must pass the credentials in the Web service request. The connector offers alternatives for storing the credentials. Your institution's choice depends on the operational and security trade-offs of each alternative.

The connector asks a credential provider to provide the login and password credentials. The credential provider offers different ways to store and retrieve the credentials, hiding these differences from the connector. In other words, the connector asks the credential provider for the credentials, without having to know where or how the credentials are stored or retrieved. Without changing the connector, you can specify, change, or customize the storage and retrieval mechanism to suit your security needs.

The shipped version of the credential provider provides a standard storage mechanism with an option to encrypt the credentials. The standard mechanism stores the login and password credentials in a property file, which is a text file in a particular location within the deployed application. The credentials stored in this file can be encrypted or unencrypted. If the standard credential provider, with encrypted or unencrypted credentials, is acceptable for your institution, then getting started with iModules Encompass is a straightforward installation task.

Encrypted credentials

You can optionally change the implementation of the Advancement Connector to use encrypted credentials. The provider library includes a utility for creating the encrypted Encompass credentials. See "Configure encrypted Encompass credentials" on page 24 for instructions.

The iModules Web services expect the credentials to be supplied in cleartext form. Therefore, encrypted credentials must be decrypted before they are supplied with the Web service request. This is accomplished by manually supplying the decryption key (a password key) to the running application at startup via a function located at this URL:

```
http(s)://<server>:<port>/advconnector/config
```

where `<server>` is the application server and `<port>` is the port number of the application server. See "Enable decryption" on page 31 for instructions.

The decryption key is stored in the memory of the running application, not in a user accessible file. *Therefore, you must supply this key every time you restart the connector, before you start data transfers.* As long as the connector continues running within the application server instance, it can use the decryption key you supplied at the beginning of the session.

Using encrypted credentials and entering the decryption key after every restart may be acceptable in environments, such as production environments, where restarts are infrequent.

Unencrypted credentials

By default, the Advancement Connector uses cleartext (unencrypted) credentials in the property file. This is suitable for environments where the connector is restarted frequently or in environments where a lot of automation is desired (such as automatic restarts). You must configure the application server so that the property file containing the clear text credentials is secure, but accessible to the connector.

2 Installation and Configuration

This chapter provides detailed instructions for installing and configuring the Advancement Connector on Oracle WebLogic Server 11g. Make sure that you review Chapter 1, "Preinstallation Considerations" before you install the connector as described in this chapter.

Note

iModules currently has three different sites where the Encompass web services are available. These three sites are in the United States, Canada and Australia. If you are instructed by iModules to connect to a site other than the United States, please contact Action Line in order to ensure you have the necessary information and instructions for connecting to the appropriate site.

Step overview

The following steps are used to install and configure the Advancement Connector:

"Preparation"

- Step 1, "Extract artifacts from the zip file"

"Phase 1 - Install initiator jar on Forms Server"

"Phase 2 - Configure Connector properties"

- Step 1, "Unpack the Connector EAR file"
- Step 2, "Configure the Advancement Connector properties"
- Step 3, "Configure the Encompass properties"
- Step 4, "Configure logging for Oracle Fusion Middleware/WebLogic Server"
- Step 5, "Configure unencrypted Encompass credentials"
- Step 6, "Configure encrypted Encompass credentials"
- Step 7, "Repack the connector ear file"

"Phase 3 - Installation on Oracle WebLogic Server 11g"

- Step 1, "Create a Managed Server"
- Step 2, "Configure the Managed Server"
- Step 3, "Define the data source"
- Step 4, "Install the connector"

"Phase 4 - Final configuration and test"

- Step 1, "Enable decryption"
- Step 2, "Test the Advancement Connector"
- Step 3, "Set the Advancement Connector URL in Banner Advancement"

Step details

The remainder of this chapter provides details for each step that is used to install and configure the Advancement Connector.

Preparation

Step 1 Extract artifacts from the zip file

Unzip `AdvConnector_v861.zip` to a directory on a system that can access the server instance. The zip file contains the following artifacts:

Artifact	Description
`AdvancementConnector-8.6.1.ear`	Java application portion of the Advancement Connector. This file is deployed to the WebLogic Server instance.
`auainit-8.6.1.jar`	Java code that enables a Banner® form to initiate connector operations through HTTP or HTTPS. This file is deployed to the Oracle Forms server for Banner Advancement.
`jasypt-1.9-dist.zip`	Distribution of Java Simplified Encryption, which includes a command line utility that is used to encrypt the credentials for the iModules Web service.

Phase 1 - Install initiator jar on Forms Server

Use the following steps to install the Java code that enables a Banner form to initiate connector operations through HTTP or HTTPS. This code is installed on the Oracle Forms server.

Note

The CLASS_PATH on your form server (11g) must include this auainit-8.6.1.jar file.

1. Use Oracle Enterprise Manager 11g Fusion Middleware Control to connect to the Fusion Middleware Server that contains the Oracle Forms server used for Banner Advancement. The Fusion Middleware Home page is displayed.

2. Expand the Forms folder and click the **Forms** drop-down menu and select **Environmental Configuration**.

3. In the Show drop-down menu, select the environment file for the instance where the jar file will be installed (for example, prod.env).

4. Go to the CLASSPATH parameter and note the directory path for the Forms Java files (for example, `C:\Oracle\Middleware\as_1\forms\java\`).

5. Copy the `auainit-8.6.1.jar` file to the directory noted in step 4. (The `auainit-8.6.1.jar` file was extracted from `AdvConnector_v861.zip`.)

6. Return to the Forms Edit Environment File page in Oracle Enterprise Manager 11g Fusion Middleware Control.

7. Select the CLASSPATH parameter and add the directory path and file name for the `auainit-8.6.1.jar` file (for example, `C:\Oracle\Middleware\as_1\forms\java\auainit-8.6.1.jar`).

8. Click **Apply**.

Phase 2 - Configure Connector properties

Step 1 Unpack the Connector EAR file

1. Ensure that the JDK bin directory is in your command shell environment path. For example, `C:\Program Files\Java\jdk1.6.0_43\bin` or `C:\Program Fiels\Java\jdk1.7.0_17\bin`.

2. Copy **AdvancementConnector_8.6.1.ear** to a temporary location. This location is referred to as `<EAR_HOME>`.

3. Navigate to `<EAR_HOME>` and execute the following command.

   ```
   jar xvf AdvancementConnector_8.6.1.ear
   ```

   The extract contains a Web archive named advconnector.war.

4. Create a folder under `<EAR_HOME>` and name it `war_home`.

5. Navigate to `war_home` and execute the following command.

   ```
   jar xvf <EAR_HOME>/advconnector.war
   ```

6. Navigate to `war_home\WEB-INF\classes`. This directory contains all of the properties files that may need changes.

Step 2 Configure the Advancement Connector properties

The Advancement Connector properties files (`connector.properties` and `connsched.properties`) contain the following configuration properties. Most of these properties are configured with initial values that facilitate immediate data exchange with Encompass. In most cases, you do not need to make decisions about changing these values.

Property	Description	Initial Value
`connector.properties`		
`connector.delimiter`	Delimiter character used to separate elements in data transfer files, when flat files are used for data upload	, (comma) (Required value for Banner/Encompass integration)
`connector.entityidlabel`	Column label for the entity identifier property when written to a delimited file format such as CSV	*SPRIDEN_PIDM* (Required value for Banner/Encompass integration)

Property	Description	Initial Value
`connector.implementation class`	ConnectorService implementation loaded by the Factory The initial value should be changed only if you customize the Encompass connection or connect with a different partner system.	*com.ellucian.advancement.connector.encompass.EncompassService*
`connector.writefilepath`	File directory or path on the application server where data transfer files are written when a flat file is used to transmit data Your institution's system administrator sets the value for this property and optionally makes this location available to the Banner Advancement administrative user who transfers data files to Encompass for upload.	*/home/oracle/AdvanceConnector* (Assumes a Unix or Linux operating system)
`connector.postprocessingstoredproc`	The name of the stored procedures that will be used to provide gift and bio pull post processing operations. Do not change this unless you are changing the way the post processor functions.	*aukcdrv.P_Post_Process*
`connsched.properties`		
`connector.scheduledjobclass`	Scheduled job implementation that the Quartz Scheduler uses to push or pull information.	*com.ellucian.banner.advancement.utility.scheduler.ConnectorScheduledJob*
`connector.schedulervalidator`	Indicator that determines whether the Connector should set up a simple Scheduler validator job that runs more frequently than regular Connector scheduled jobs, and determines whether the Scheduler is running in an easily verifiable way. *true* - Schedule validator job *false* - Do not schedule validator job. Remove the job from the Scheduler, if the job exists.	*true*

Property	Description	Initial Value
`connector.validatorjobclass`	Connector Scheduler validator job implementation that the Scheduler uses to report that it is running.	*com.ellucian.advancement.scheduler.validator.ConnectorSchedulerValidator*
`connector.validatorrepeats`	Number of times that the validation job should be run. *1* - Run the job once, as soon as the Scheduler is started. *Any other numeric value* - Run the job this number of times on the frequency set by the `connector.validatiorfrequency` property. * - Schedule the job to run indefinitely on the frequency set by the connector.validatiorfrequency property.	*
`connector.validatorfrequency`	Frequency to run the validator job, in minutes. This value is meaningful only if the value of the `connector.validatorrepeats` property is an asterisk (*) or greater than 1.	*60*
`connector.emailhost`	Host address of your institution's SMTP server that sends e-mail notices to designated e-mail addresses after the Scheduler runs scheduled jobs. Use either the Fully Qualified Domain Name (preferred) or the IP address.	*No initial value. A value must be supplied for e-mail notifications to function.*
`connector.emailnotif`	E-mail address that is provided in e-mail notices as the sender address. Used with the connector.emailpersname to identify the message sender.	*connectornotifications@ellucian.com*
`connector.emailpersname`	Personal name (friendly name) that is provided in e-mail notices as the sender. Used with the connector.emailnotif to identify the message sender.	*Advancement Connector Notifications*

Use the following steps to review the Advancement Connector properties.

1. Open the `connector.properties` file with a text editor.
2. Review the configuration properties.
3. Determine whether the path in the `connector.writefilepath` property is suitable for your application server:
   - 3.1. If the path is suitable, then create the directory indicated.
   - 3.2. If the path is not suitable, then change the property value to a suitable path, save the properties file, and create the directory indicated.

     Unix example - `/home/oracle/AdvanceConnector/`

     Windows example - `C:/advconnector/`
4. Ensure that the directory created in step 3.2 is writable by the Java container/ Advancement Connector application and accessible to the administrators who need to transmit the flat files to the partner system.
5. Save and close the `connector.properties` file.
6. Open the `connsched.properties` file with a text editor.
7. Review the configuration properties.
8. Set the `connector.emailhost` property value to the Fully Qualified Domain Name (FQDN) or IP address of your institution's SMTP e-mail server.
9. Change the value of the `connector.emailnotif` property to be appropriate for your institution. Change the value of the `connector.emailpersname` property as desired.
10. Save and close the `connsched.properties` file.

Step 3 Configure the Encompass properties

The Encompass properties file (`encompass.properties`) contains the following configuration properties that are used to synchronize data changes between Banner and Encompass. These properties are configured with initial values that provide a reasonable working environment for data exchange with Encompass. In most cases, you do not need to make decisions about changing these values.

Property	Description	Initial Value
`encompass.allowadd constituents`	Flag that determines whether new Banner constituent records can be pushed to Encompass: *true* — Push new constituents to Encompass. Use this setting if you create new constituent records in Banner and you want to transfer these new constituents to Encompass. *false* — Do not push new constituents to Encompass. This property applies only to push and update transactions.	*true*
`encompass.batchsize`	Maximum number of constituent records in a transfer batch. A constituent record can have any number of individual data changes. A transfer is divided into one or more batches, based on this property. The initial value is *800*, which experience has determined to be a reasonable starting value that balances data transfer size with performance of the Web service. You can change this value if the data transfer situation at your institution warrants it.	*800*

Property	Description	Initial Value
`encompass.debug`	Flag that determines whether the Encompass service operations are in debug or operational mode: *true* Debug mode *false* Operational mode Keep the initial value unless your institution created a different implementation of an operation.	*false*
`encompass.ignoremessages`	List of Encompass message codes that the Advancement Connector should ignore and not capture in the transfer status records or logs. If the list is empty, all messages are captured and recorded. This list does not apply to Encompass error codes, which are those elements found in the <Error> nodes of a response. **Note:** You can comment out this property to temporarily enable messages without deleting the list.	*2207*
`encompass.includenonmembers`	Flag that determines whether constituent records are pulled if they are not members of the community in Encompass: *true* Pull non-members. *false* Do not pull non-members. This property applies only to pull and get member transactions.	*true*

Property	Description	Initial Value
`encompass.includeblankids`	Flag that determines whether constituent records are pulled if they do not have a constituent ID: *true* — Pull records without a constituent ID. *false* — Do not pull records without a constituent ID. This property applies only to pull and get member transactions. The Advancement Connector does not support the processing of constituents that do not have a Banner PIDM. Therefore, this property value should remain *false*, unless you customize the behavior of Banner Advancement and the Advancement Connector.	*false*
`encompass.timeout`	Maximum number of seconds the service client waits for a response from the iModules Web service. This property is used with the batch size to provide optimum performance. The initial value is *90*, which iModules determined to be a reasonable starting value.	*90*
`encompass.credentialprovider`	Credential provider implementation class for Encompass service which determines whether encrypted or unencrypted credentials will be kept in the encompassconnect.properties file. The property value used for the encrypted Encompass credentials is *com.ellucian.integration.credential.property.EncryptedPropertyProvider.* The property value used for the unencrypted Encompass credentials is *com.ellucian.integration.credential.property.PropertyProvider.*	*com.ellucian.integration.credential.property.Property Provider*

Property	Description	Initial Value
`encompass.implementationgetclass`	Implementation class for the bio pull action of the Encompass service.	*com.ellucian.advancement.connector.encompass.EncompassGetMembersChanged*
`encompass.implementationcontrolgetclass`	Implementation class for the gift pull action of the Encompass service.	*com.ellucian.advancement.connector.encompass.control.EncompassGetMembersChangedControl*
`encompass.implementationinfoclass`	Implementation class for the service information action of the Encompass service.	*com.ellucian.advancement.connector.encompass.EncompassGetInfo*
`encompass.implementationputclass`	Implementation class for the bio push action of the Encompass service.	*com.ellucian.advancement.connector.encompass.EncompassUpdate*
`encompass.implementationwriteclass`	Implementation class for the bio write to file action of the Encompass service.	*com.ellucian.advancement.connector.encompass.EncompassWriteToFile*
`encompass.generalquery.url`	WSDL location and web service URL for Encompass general query service. The initial value is for the Encompass US site. Other Encompass sites, such as Canada and Australia, have different URLs.	*https://api.iModules.com/ws/21/generalquery.asmx*
`encompass.controlquery.url`	WSDL location and web service URL for Encompass control query service. The initial value is for the Encompass US site. Other Encompass sites, such as Canada and Australia, have different URLs.	*https://api.iModules.com/ws/21/controlquery.asmx*

Property	Description	Initial Value
`encompass.transactionsquery.url`	WSDL location and web service URL for Encompass transactions query service. The initial value is for the Encompass US site. Other Encompass sites, such as Canada and Australia, have different URLs.	*https://api.iModules.com/ws/21/transactionsquery.asmx*

Use the following steps to review the Encompass properties.

1. Open the `encompass.properties` file with a text editor.
2. Review the configuration properties and change as necessary.

Step 4 Configure logging for Oracle Fusion Middleware/WebLogic Server

The Advancement Connector uses Apache's log4j to log the activities performed by the application at runtime. Log4j uses an XML properties file to establish specific runtime options. The following options should be reviewed and modified as appropriate:

- The default location for the WebLogic Server log for the Advancement Connector is `oracle\middleware\user_projects\domains\<domain_name>\servers\<server_name>\logs`.

  where `<domain_name>` is the WebLogic Sever Domain name, and `<server_name>` is the name of the managed server. This location should not be changed.

- Logging level. The default level is **INFO**, resulting in relatively little logging information being stored in log files. This logging level is appropriate for normal operations, however, you may want or need to obtain more detailed logging information. To do this, you should modify the logging level for the com.ellucian logger to **DEBUG**.
- Logging appender in use. The default logging appender (the actual element that writes out the log) is named "weblogicserver".

Use the following steps to modify the logging options as appropriate.

1. Edit **log4j.xml**.

Property	Description	Initial Value
`<logger name="com.ellucian"> level element`	Info	Leave as is or supply one of debug, info, warn, error, fatal

2. Save **log4j.xml**.

Step 5 Configure unencrypted Encompass credentials

Note

You can choose to store the Encompass Web service logon credentials in unencrypted or encrypted form.

1. Contact iModules (www.imodules.com) to obtain a login username (GUID) and password for your institution to use to access the iModules Web service.

2. Open the `encompassconnect.properties` file with a text editor.

3. Supply the Encompass Web Service login id and password as the values for the properties in the section labeled `For CLEARTEXT (UNENCRYPTED) Encompass Credentials.`

Step 6 Configure encrypted Encompass credentials

The `jasypt-1.9-dist.zip` file, which was extracted from AdvConnector_v861.zip, provides a command line utility that is used to encrypt credentials. Use the following steps to encrypt and store your iModules Web service credentials.

1. Open the `encompass.properties` file with a text editor.

2. Comment out the `encompass.credentialprovider` property in the unencrypted credentials section, and uncomment the `encompass.credentialprovider` property in the encrypted credentials section.

3. Save the `encompass.properties` file.

4. Contact iModules (www.imodules.com) to obtain a login username (GUID) and password for your institution to use to access the iModules Web service.

5. Unzip the `jasypt-1.9-dist.zip` file that was extracted from `AdvConnector_v861.zip`.

6. Ensure that the path environment variable references Java 6 or Java 7. This is an example on Windows:

   ```
   C: set PATH=C:\Program Files\Java\jdk1.6.0_43\bin;%PATH%
   ```

7. Choose a password that will function as a symmetric key for encrypting and decrypting the Encompass Web Service credentials.

   The password can have any length or complexity. It should be something that system administrators can remember or keep secure, because it is needed for final configuration of the connector and any time the connector is restarted.

8. In a command shell, navigate to the `bin` directory in the `jasypt` directory.

9. Enter the following on a command line:

   ```
   encrypt algorithm=PBEWithSHA1AndDESede password=encryptionpwd
   input=encompass-login-guid > logincipher.txt
   ```

   where `encryptionpwd` is your chosen password key and `encompass-login-guid` is the login GUID supplied by iModules.

   The output of this command is in the `logincipher.txt` text file. The login cipher text is under the OUTPUT section of the file.

10. Enter the following on a command line:

    ```
    encrypt algorithm=PBEWithSHA1AndDESede password=encryptionpwd
    input=encompass-password > passwordcipher.txt
    ```

    where `encryptionpwd` is your chosen password key and `encompass-password` is the password supplied by iModules.

    The output of this command is in the `passwordcipher.txt` text file. The password cipher text is under the OUTPUT section of the file.

11. Enter the encrypted credentials in the `encompassconnect.properties` file:

    11.1. Open `encompassconnect.properties` with a text editor.

    You can encrypt either one or both of the Encompass Web Service login and password credentials. If you choose to encrypt either one, you can leave the unencrypted properties commented out.

11.2. Use the following properties if you are encrypting both the login id and password:

```
loginid=ENC(encrypted Encompass login id cipher text)

password=ENC(encrypted Encompass password cipher text)
```

Note

The login cipher text is contained in the `logoincipher.txt` text file that was created in step 9. The password cipher text is contained in the `passwordcipher.txt` text file that was created in step 10. ■

11.3. Use the following properties if you are unencrypting the login id and encrypting the password:

```
loginid=login cipher text

password=ENC(password cipher text)
```

Warning

Some text editors, such as Notepad on Windows, may display one or more "box" characters at the end of each line of the cipher text file. These box characters are not part of the cipher text. Do not copy the box characters as part of the cipher text. ■

Warning

The login and password entries must use ENC followed by the cipher text in parentheses. Otherwise, the credentials will be treated as cleartext, and no decryption attempt will be made on them. ■

11.4. Save the `encompassconnect.properties` file.

Step 7 Repack the connector ear file

1. From `war_home,` execute the following command to rebuild the Web archive file.

```
jar cvf <EAR_HOME>/advconnector.war META-INF/* WEB-INF/*
index.html
```

2. From `<EAR_HOME>` execute the following command to rebuild the enterprise archive file.

```
jar cvf AdvancementConnector_8.6.1.ear advconnector.war META-
INF/*
```

The rebuilt `AdvancementConnector_8.6.1.ear` is used for installation.

Phase 3 - Installation on Oracle WebLogic Server 11g

The Banner Advancement Connector can be installed on an existing Oracle WebLogic Domain, either a Basic Domain or a Classic Domain that comes with WLS_FORMS and WLS_REPORTS. However, since the Connector application must have access to the internet to communicate with iModules Encompass, Ellucian recommends careful consideration of whether installing the Connector on a new Oracle WebLogic Domain, with appropriate internet access, makes for a better overall network security configuration.

The Managed Server to which the application is deployed should be dedicated to the Advancement Connector so it can be managed independently from other applications.

Use the following steps to install the connector on Oracle WebLogic Server 11g.

Step 1 Create a Managed Server

Create a new Managed Server for the Advancement Connector so that it can be independently managed. Refer to Oracle WebLogic Server Documentation Library for details.

Note

This step only needs to be performed once. If you previously performed this step, do not perform the steps below.

Step 2 Configure the Managed Server

Use the following steps to configure the managed server:

Note

This step only needs to be performed once. If you previously performed this step, do not perform the steps below.

1. Connect to the Oracle WebLogic Server Administration Console.

   The Home Page is displayed.

2. Click **Lock &Edit** from the Change Center.

3. In the Domain Structure pane, click **Environment**, then **Servers**.

   The Summary of Servers is displayed.

4. Click the name of the managed server

   The Settings for `<servername>` is displayed, where `<servername>` is the name of the managed server.

5. Click SSL tab under **Configuration** tab.

6. Open **Advanced** Section.

   At the top of the Advanced section, item **Hostname Verification**: use the drop-down to change the value from **BEA Hostname Verifier** to **None**.

7. Click Save.

8. Click the **Server Start** tab under the **Configuration** tab.

9. In the item **Arguments**: add the following line:

   ```
   -Xmx2048m -XX:PermSize=100m -XX:MaxPermSize=150m
   ```

10. Click Save.

11. In the Change Center pane, click **Activate Changes**.

Step 3 Define the data source

Note

This step only needs to be performed once. If you previously performed this step, do not perform the steps below.

1. Click **Lock & Edit** from the Change Center.

2. In the Domain Structure Pane, click **Services** -> **Data Sources**.

   The summary of the JDBC Data Sources is displayed.

3. Click New, which will display a list of data source types.

   3.1. Select **Generic Data Source**.

   3.2. The Create a new JDBC Data Source pane is displayed.

4. Define the following data source:

   Name: banadvconnector

   JNDI Name: jdbc/banadvconnector

   Database Type: Oracle

   4.1. Click Next

   Database Driver: Oracle's Driver (Thin) for Instance Connections; Versions 9.0.1 and later

   4.2. Click Next

4.3. Uncheck **Supports Global Transactions**

4.4. Click Next

5. The **Connection Properties** pane is displayed

   Database Name: enter your Banner Advancement database SID

   Host Name: enter the database server name

   Port: leave at 1521 unless configured differently for your database

   Database User Name: advconn

   Password: password used for advconn,

   Confirm Password: same password

   5.1. Click Next

6. Click **Test Configuration**.

   If the Configuration test is not successful, confirm the correct values are set in the data source and repeat the test until it is successful.

7. Click Next

   The Select Targets pane is displayed.

   7.1. In the Servers list, check the connector managed server.

8. Click Finish

9. In the Data Sources list, select **banadvconnector**.

   9.1. Under the **Configuration** tab, select the **Connection Pool** Tab

   9.2. Open the **Advanced** section

   9.3. Un-check **Wrap Data Types**

   9.4. Click Save

Step 4 Install the connector

1. If the managed server is not started, start it:

   1.1. In the **Domain Structure** pane, select **Environment** -> **Servers**.

   1.2. Select the **Control** tab.

   1.3. Check the managed server instance and click **Start**.

2. Transfer the rebuilt `AdvancementConnector_8.6.1.ear` to the WebLogic Server's file system, using any suitable tool, such as SFTP. Optionally, you can upload your files through the **Install Applications Assistant**. To perform this task, in the **Domain Structure** pane, select **Classic Domain** -> **Deployments** -> **Install** and click the **upload your files** link.

3. Connect to the Oracle WebLogic Server Administration Console:

   The Home Page is displayed.

4. Click **Lock & Edit** from the Change Center.

5. In the **Domain Structure** pane, click **Deployments**.

   The Summary of Deployments pane is displayed.

6. Click Install.

   The Install Application Assistant is displayed.

7. Enter the local server path to the ear file, if it is not the default path.

   7.1. Select the `AdvancementConnector_8.6.1.ear`

   7.2. Click Next

8. Select, if not selected, **Install this deployment as an application**.

   8.1. Click Next

9. The **Select Deployment Targets** pane is displayed.

   9.1. In the **Servers** list, select the managed server instance for the connector.

   9.2. Click Next.

10. The default name of the application will be `AdvancementConnector_8`. Change the name, if desired.

    10.1. In the Security section, select **DD Only: Use only roles and policies that are defined in the deployment descriptors**.

    10.2. Click next, then click Finish.

11. When configuration page appears, click **Activate Changes** in the **Change Center**.

12. In the **Domain Structure** Pane, select **Deployments**.

    12.1. The **Summary of Deployments** appears.

13. Check the box next to the connector deployment. The Connector application status should be **Prepared**.

14. Click **Start** -> **Servicing all requests**.

15. Click Yes when the **Start Deployments** page appears.

16. When the summary of deployments appears the **Connector Application** status should be **Active**.

Phase 4 - Final configuration and test

Step 1 Enable decryption

Use the following steps to enable the Advancement Connector to decrypt the credentials when needed to connect to the Encompass Web service.

1. Open a browser and connect to the following URL. Perform this step only if you are using encrypted Encompass Web Service credentials.

   ```
   http(s)://<server>:<port>/advconnector/config
   ```

   The Web PBE Configuration page is displayed:

2. Enter the following values:

Validation word	*advConn*
Password	Password used to encrypt the credentials
Retype password	Same password that was entered in **Password**

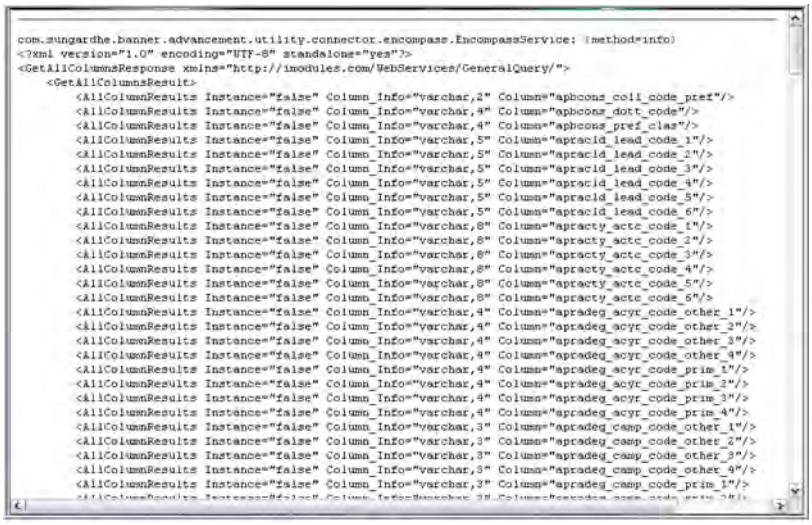
3. Click **Submit**. A confirmation message is displayed.

Step 2 Test the Advancement Connector

Open a browser and connect to the following URL:

```
http(s)://<server>:<port>/advconnector/init?method=info
```

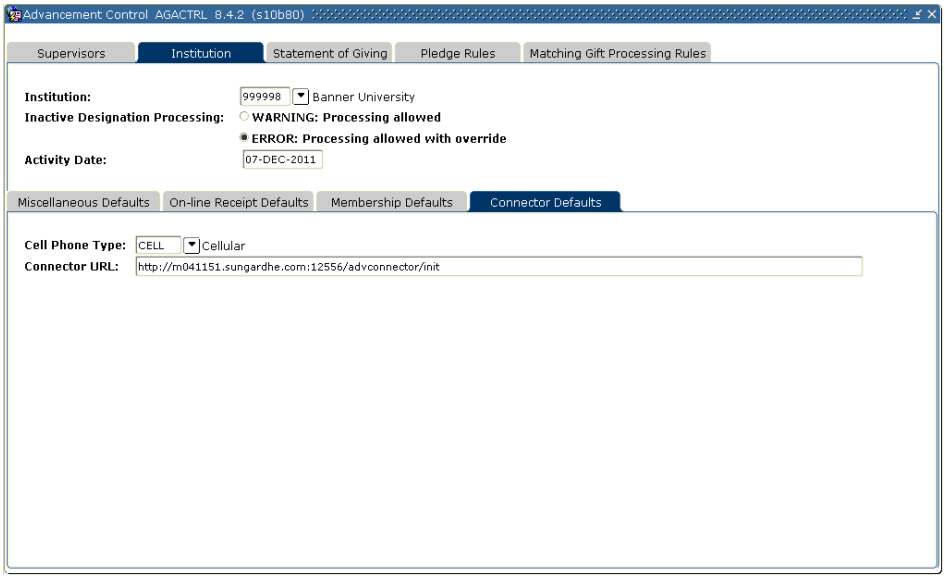
If the Advancement Connector is properly configured and connected to the iModules Web service and the Banner database, a response similar to the following is displayed after a few seconds:



Step 3 Set the Advancement Connector URL in Banner Advancement

Use the following steps to set the URL that Banner uses to call the Advancement Connector.

1. Log in to Banner.
2. Access the Advancement Control Form (AGACTRL).
3. Select the Institution tab.
4. Select the Connector Defaults sub-tab.



5. Enter the base URL in the **Connector URL** field using this format:

   `http(s)://<server>:<port>/advconnector/init`

   where <port> is the port number you set and recorded for the WebLogic Server instance.

6. Click Save.

3 Administration

This chapter provides instructions for the following administration tasks:

- "Set the decryption password after a restart"
- "Test the Encompass implementation"
- "View requests made to the Advancement Connector"
- "Change the credential provider implementation"
- "Change the Encompass operation implementations"
- "Change the Advancement Connector implementation"

Set the decryption password after a restart

If you use the encrypted credentials option for the Advancement Connector, the credentials must be decrypted before the credentials can be supplied to the partner system. The decryption key is stored in the memory of the running connector. Whenever you restart the connector, you must set the decryption key so it is available for data transfer operations.

Use the following steps to set the decryption password.

1. Open a browser and connect to the following URL. This step is required only if you are using encrypted credentials.

   ```
   http(s)://<server>:<port>/advconnector/config
   ```

   The Web PBE Configuration page is displayed:

2. Enter the following values:

Validation word	*advConn*
Password	Password used to encrypt the credentials
Retype password	Same password that was entered in **Password**

3. Click **Submit**. A confirmation message is displayed.

Test the Encompass implementation

You can test basic functionality and connectivity for your Encompass implementation directly from a browser. The info method implementation for Encompass performs the following processing:

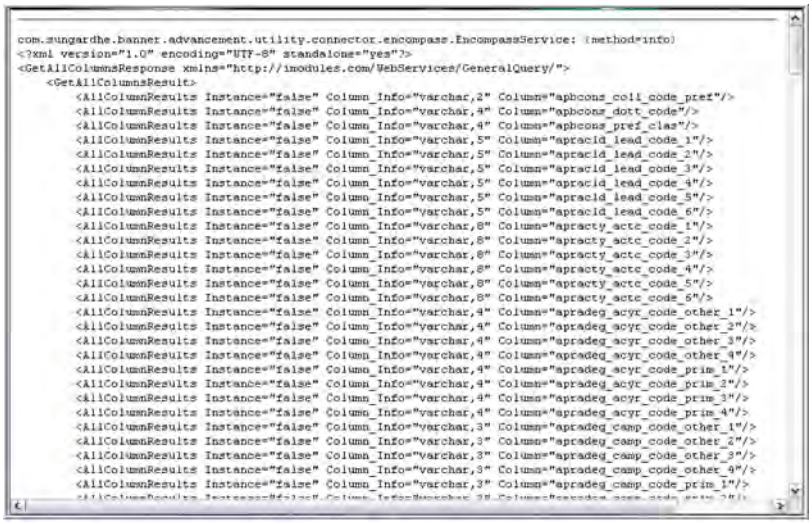
- Connects to the iModules Web service.
- Retrieves a list of columns defined within Encompass for your institution.
- Writes this column information and status information to the Advancement Connector Transfer Status Table (AUBTRST).

This processing provides a full cycle test of connectivity between the Advancement Connector and Encompass, as well as connectivity between the Advancement Connector and the Banner® database.

To perform the test, open a browser and connect to the following URL:

```
http(s)://<server>:<port>/advconnector/init?method=info
```

If the Advancement Connector is properly configured and connected to the iModules Web service and the Banner database, a response similar to the following is displayed after a few seconds:



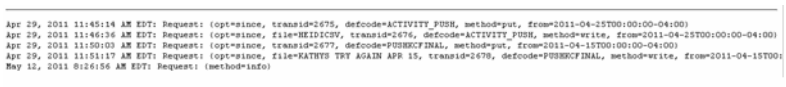
Any other response, such as an exception message, usually indicates a connectivity issue with Encompass or the Banner database. You should check the log for the Advancement Connector to diagnose the issue.

View requests made to the Advancement Connector

The Advancement Connector Initiation Form (AUAINIT) is used to initiate a connector operation with the Advancement Connector by passing parameter values to the connector's Java servlet via HTTP(S). To view a summary of all requests made to the connector since its last restart, open a browser and connect to the following URL:

```
http(s)://<server>:<port>/advconnector/init
```

A response similar to the following is displayed:



If the Advancement Connector did not receive any requests since its last restart, the connector replies with the message *No Requests Logged*.

Refer to Appendix A, "Initiation Parameters" for a description of each parameter that is passed to the connector's Java servlet via HTTP(S).

Change the credential provider implementation

If, for security and operational purposes, you wish to use a different credential provider than the one provided with the default implementation, you can change a property in the `encompass.properties` file.

1. Unpack the Advancement Connector EAR file as described in "Phase 2 - Configure Connector properties" on page 15.

2. Open the `encompass.properties` file with a text editor.

3. Change the following property:

   ```
   encompass.credentialprovider=<fully qualified provider class>
   ```

   where `<fully qualified provider class>` is one of the following values:

   3.1. To use the standard property provider that uses encryption, use the following default class name:

   ```
   com.ellucian.integration.credential.property.
   EncryptedPropertyProvider
   ```

   3.2. To use the standard property provider that does not use or require encryption, use the following class name:

   ```
   com.ellucian.integration.credential.property.PropertyProvider
   ```

   3.3. To use an alternate credential provider, use any class that you have deployed to the application's classpath that implements the `com.ellucian.integration.credential.CredentialProvider` interface, and is a subclass of `com.ellucian.integration.credential.AbstractProvider`. Developers should refer to the Advancement Connector Developer's Guide and API Documentation for more information.

4. Save the `encompass.properties` file.

5. Repack the Advancement Connector EAR file, and redeploy the application.

Change the Encompass operation implementations

Five operations are available in the Encompass implementation of the Advancement Connector. Each operation is implemented by a different class in the package `com.ellucian.advancement.connector.encompass`. The following properties specify the class that implements each operation. The default class name for each operation is also provided.

Operation	Property	Default Class Name
get bio (pull)	`encompass.implementationgetclass`	`com.ellucian.advancement.connector.encompass.EncompassGetMembersChanged`
get gift (pull)	encompass.implementationcontrolgetclass	`com.ellucian.advancement.connector.encompass.control.EncompassGetMembersChangedControl`
put (push)	`encompass.implementationputclass`	`com.ellucian.advancement.connector.encompass.EncompassUpdate`
info	`encompass.implementationinfoclass`	`com.ellucian.advancement.connector.encompass.EncompassGetInfo`
write (to a file)	`encompass.implementationwriteclass`	`com.ellucian.advancement.connector.encompass.EncompassWriteToFile`

Your institution can change the class that implements an operation by changing the associated property in the `encompass.properties` file. Use the following steps to change operation properties in the properties file.

1. Unpack the Advancement Connector EAR file as described in "Phase 2 - Configure Connector properties" on page 15.
2. Open the `encompass.properties` file with a text editor.
3. Change the properties that implement Encompass operations using the following format:

   ```
   <property>=<package name>.<class name>
   ```

   where `<property>` is the property listed in the preceding table, `<package name>` is the package that contains the class, and `<class name>` is the class. The property value must include the package name and class name.

4. Save the `encompass.properties` file.

5. Repack the Advancement Connector EAR file, and redeploy the application.

Refer to the *Advancement Connector Developer's Guide*, included with the connector API documentation, for details on creating new implementations of the operations and requirements for them.

Change the Advancement Connector implementation

If your institution wishes to use the Advancement Connector to transfer data between Banner Advancement and a partner system other than iModules Encompass, you can specify an alternate implementation of the connector. This is accomplished by changing the value of the `connector.implementationclass` property in the `connector.properties` file. Use the following steps to change the value of the property.

1. Unpack the Advancement Connector EAR file as described in "Phase 2 - Configure Connector properties" on page 15.

2. Open the `connector.properties` file with a text editor.

3. Change the `connector.implementationclass` property to the fully qualified name of the desired implementation class.

   The new implementation must implement the `com.ellucian.advancement.connector.ConnectorService` interface.

4. Save the `connector.properties` file.

5. Repack the Advancement Connector EAR file, and redeploy the application.

Refer to the *Advancement Connector Developer's Guide*, included with the connector API documentation, for details on creating new implementations of the connector.

A Initiation Parameters

The Advancement Connector Initiation Form (AUAINIT) is used to initiate a connector operation with the Advancement Connector. This form passes parameter values to the connector's Java servlet at the `advconnector/init` path. This appendix describes each parameter that can be passed.

Parameters that are passed

Parameter Name	Possible Value	Used With Method	Description	Default Value
method	*get*		Service get (pull) method	*info*
	put		Service put (push) method	
	write		Service writeToFile method	
	info		Service getInfo method	
	schedule		Service schedule method	
action	*delete*	schedule	removes a scheduled push or pull	*list*
	insert	schedule	schedules a push or pull	
	list	schedule	lists scheduled jobs/defcodes	
	run	schedule	runs a scheduled job/defcode immediately	
	query	schedule	queries scheduler for a specified job/defcode	
	update	schedule	reschedules a scheduled push or pull	

Parameter Name	Possible Value	Used With Method	Description	Default Value
transid	Character representation of number	get put write	Transaction ID on which the method is performed	none
defcode	Profile codes from `AUBCDEF.AUBCDEF_CODE`	get put write	Profile code that identifies the set of data elements to pull, push, write	*NO_DEFINITION*
file	text file name	write	Name of the file where data is written	`<transid>output.<file extension>`
opt	*during*	get put write	Option that specifies the get request will get all changes during the from/to date range, or that the put request will preserve all Encompass changes made during the from/to date range	*since*
	since	get put write	Option that specifies the get request will get all changes since the from date, or that the put request will preserve all Encompass changes made between the from date and the time when the request was initiated	
from	Character representation of a date/time	get put write	Starting date/time for the changes included in the get request or preserved in the put request	None
to	Character representation of a date/time	get put write	Ending date/time for the changes included in the get request or preserved in the put request	None
cronexpr	Representation of a push or pull schedule in 'cron' expression format	schedule	Schedule (time, date, and repeating characteristics) of a push or pull to be scheduled	None

Note

The “from” and “to” parameters are represented in the format specified in W3C XML Schema 1.0 Part 2, Section 3.2.7-14. This format captures date and time as follows:

yyyy-mm-ddThh:mm:sszzzzzz

where zzzzzz is the time zone represented as (+|-)hh:mm. For example, July 19, 2010 1:30 pm EDT is represented as 2010-07-19T13:30:00-04:00.

Troubleshooting

The following information can be used to help troubleshoot errors or issues encountered during the process of loading data through the Advancement Connector.

Error	Message Name	Object	Location	Comments
Credential file {0} not found.	`filenotfound.message`	`PropertyProviderResource.properties`	`credentialprovider\src\resource\locale`	The shipped version of the credential provider stores the login and password credentials in a property file, which is a text file in a particular location within the deployed application. May need to change location of the property file so it can be found by the connector.
Credential decryption password not set.	`encryptedpropertynodecrypt.message`	`PropertyProviderResource.properties`	`credentialprovider\src\resource\locale`	Your institution uses un-encrypted credentials. As a result you must configure the application server so that the property file containing the clear text credentials is secure, accessible to the connector and contains the credential decryption password.
{0} not found on classpath: {1}	`classpath.message`	`ExceptionResource.properties`	`service\src\resource\locale`	File name in error message could not be found in your classpath. Classpath is a parameter set up during the installation process. Verify the file name referenced in the error message is in a location that can be found by the classpath parameter.
Connector {0} is missing.	`credentialmissing.message`		`service\src\resource\locale`	Checks that the login id and password credentials for connecting to Encompass are available (not null) in this object instance's properties. Verify credential values in the credential property file.

Error	Message Name	Object	Location	Comments
No connector definition with code {0}.	defcodemissing.message	ExceptionResource.properties	service\src\resource\locale	Definition with the supplied definition code not found. Verify you have a defintion code with this name defined on AUACDEF.
Connector definition code is null or cannot be found.	defcodenull.message	ExceptionResource.properties	service\src\resource\locale	Definition code can not be null. Verify you are entering a defintion code on AUAINIT.
Parameter {0} cannot be null.	parameternull.message	ExceptionResource.properties	service\src\resource\locale	Parameter referenced in the error message is null and cannot be.
Scheduler request resulted in the following parser error:{0}	parseexception.message	ExceptionResource.properties	service\src\resource\locale	There are many possible causes for this error. In this case, the user is going to need to work with IT to check the server logs to understand the actual nature of the error. For the most part, these are going to be unusual runtime errors.
Scheduler request resulted in the following error:{0}	schedulerexception.message	ExceptionResource.properties	service\src\resource\locale	There are many possible causes for this error. In this case, the user is going to need to work with IT to check the server logs to understand the actual nature of the error. For the most part, these are going to be unusual runtime errors.
Scheduler request resulted in the following IO error:{0}	schedulerioexception.message	ExceptionResource.properties	service\src\resource\locale	There are many possible causes for this error. In this case, the user is going to need to work with IT to check the server logs to understand the actual nature of the error. For the most part, these are going to be unusual runtime errors.

Error	Message Name	Object	Location	Comments
No database connection found.	noconnection.message	ExceptionResource.properties	service\src\resource\locale	This exception is in the ConnectorPutPrepService, in the private initConnection method. The PutPrepService executes the aukconn package procedure for a push. Because it is executing a procedure instead of interacting with the persistence directly, it uses a regular JDBC connection rather than working through a persistence manager. Normally, the container injects the data source (database connection) into the class. However, if there is some problem, and the class doesn't have a connection object, it throws this exception. The container is specifically looking to inject the jdbc/banadvconnector data source. So the error could be thrown if, for example, the jdbc/banadvconnector data source was missing, deleted, or misconfigured, or misnamed during installation.
No scheduler record defined with code {0}.	schdcodemissing.message	ExceptionResource.properties	service\src\resource\locale	Scheduler Code referenced in error message could not be found. Verify you have a profile code defined on AUACDEF and this is the code that you have set up a schedule for.
No request properties for this scheduled action.	requestpropsmissing.message	ExceptionResource.properties	service\src\resource\locale	The properties that represent the connector initiation request parameters for a scheduled job could not be found.
No scheduler instance available. No schedule action taken.	noscheduler.message	ExceptionResource.properties	service\src\resource\locale	No instance of the scheduler was found. The scheduler isn't running. In this case, the corrective action would be to restart the advconn application. The scheduler is supposed to be automatically started when the application starts up.

Error	Message Name	Object	Location	Comments
Job class {0} does not implement org.quartz.Job interface.	noschedulejobclass.message	ExceptionResource.properties	service\src\resource\locale	This is more a possibility with a client who has extended the connector in some way. The way quartz works is it runs a specified class at the scheduled times. The class can do anything. Quartz doesn't know what it's doing, it just runs it. The only requirement of the class is that it must implement the org.quartz.Job interface. In the connector, the particular class that constitutes the job to run at the specified times is supplied as a property value, located in connsched.properties. Clients can write their own job class and change the name of the class in the properties file. Since the code is picking up the class name this way, it has to do a basic check that the class supplied will function with the Quartz Scheduler, and the requirement is that it implements the org.quartz.Job interface. Similarly, we also have a little validation job that runs every so often as an easy way to detect that the scheduler is running. That class is also stipulated in the property file, and similarly, clients could create a different validator job, if desired, and change the name in the property file.
User email notification failed. Contact your system administrator.	notiffail.message	ExceptionResource.properties	service\src\resource\locale	This represents a failure of the user notification, but the underlying push or pull transaction succeeded. User notification failure is logged, and a note is tacked to the status summary. View user logs or response summary on AUACRVW for additional details.

Error	Message Name	Object	Location	Comments
Element {0} not found in connector definition.	elementmissing.message	EncompassExceptionResource.properties	encompass\src\resource\locale	This message indicates a data element appears to be missing from the profile used for the push or pull request. If the referenced element name does not exist in the profile, it should be updated to include the referenced element.
{0} is null.	isnull.message	EncompassExceptionResource.properties	encompass\src\resource\locale	This message is thrown from a class that processes the response from an Encompass update method request. This message indicated that the results of the update response is null.
{0} cannot be null.	notnull.message	EncompassExceptionResource.properties	encompass\src\resource\locale	This message indicates that a field is null but cannot be in order for the connector to function properly. The message will indicate which field is null and needs to be corrected.
XML write failed.	xmlwrite.message	EncompassExceptionResource.properties	encompass\src\resource\locale	This message is thrown from a class that returns a string containing an XML representation of an Encompass client object. This error is thrown because the creation of that string failed. The class was unable to return a string XML representation of the object and its property values
Timeout: The Encompass service did not respond within the allowed {0} second response time. This response time value is set in the {1} property in the encompass.properties file.	timeout.message	EncompassExceptionResource.properties	encompass\src\resource\locale	The Encompass service did not respond within the allowed response time. The response time value is set in the encompass.properties file. Try increasing the response time value.
Request ID: {0}	transaction.message	MessageResource.properties	encompass\src\resource\locale	
Type: {0}	typecode.message	MessageResource.properties	encompass\src\resource\locale	

Error	Message Name	Object	Location	Comments
Profile Code: {0}	definitioncode.message	MessageResource.properties	encompass\src\resource\locale	
Frequency: Weekly	freqweekly.message	MessageResource.properties	encompass\src\resource\locale	
Frequency: Daily	freqdaily.message	MessageResource.properties	encompass\src\resource\locale	
Date Run: {0}	daterun.message	MessageResource.properties	encompass\src\resource\locale	
Status: {0}	status.message	MessageResource.properties	encompass\src\resource\locale	
Error Info: {0}	info.message	MessageResource.properties	encompass\src\resource\locale	
Error Info: PLEASE SEE BATCH LEVEL RESPONSE SUMMARY FOR THIS REQUEST ID ON AUAINIT.	error.message	MessageResource.properties	encompass\src\resource\locale	
Scheduler Request Initiated Status: {0} Profile: {1}	subject.message	MessageResource.properties	encompass\src\resource\locale	
Job {0} deleted: {1}	jobdeleted.message	MessageResource.properties	encompass\src\resource\locale	
Job {0} scheduled. First start will be {1}.	jobscheduled.message	MessageResource.properties	encompass\src\resource\locale	
Job {0} rescheduled. First start will be {1}.	jobrescheduled.message	MessageResource.properties	encompass\src\resource\locale	

Error	Message Name	Object	Location	Comments
Job {0} triggered at {1}.	jobrun.message	MessageResource.properties	encompass\src\resource\locale	
Job group {0} contains the following jobs:	jobgroups.message	MessageResource.properties	encompass\src\resource\locale	
Job {0} does not exist in scheduler.	jobnotfound.message	MessageResource.properties	encompass\src\resource\locale	

Error	Message Name	Object	Location	Comments
EJBException: An exception occurred during transaction completion: ; nested exception is: javax.transaction.RollbackException: returning error in transaction: javax.persistence.OptimisticLockException: Exception [EclipseLink-5006] (Eclipse Persistence Services - 2.0.2.v20100323-r6872): org.eclipse.persistence.exceptions.OptimisticLockException Exception Description: The object [com.ellucian.banner.advancement.connector.model.ConnectorScheduler@75f4b48d] cannot be updated because it has changed or been deleted since it was last read. Class> com.ellucian.banner.advancement.connector.model.ConnectorScheduler Primary Key> [475]	`javax.persistence.OptimisticLockException`			This error can result from two known conditions: 1. If there is more than one instance of the connector running, they can conflict with each other when they try to fire off a Schedules transaction. Solution: Shut down all but one instance of the connector. There should never be more than on instance of the connector running per database. 2. If a user executes a manual transaction at approximately the same time as a scheduled transaction with the same Definition Code fires, the two requests can conflict with each other. This will be very rare and impossible to prevent. If the manual transaction is the one that fails, wait until the Scheduled transaction completes and resubmit the manual transaction. If the Scheduled transaction is the one that fails, no action is necessary, the transaction will fire correctly at the next scheduled interval.