



Ellucian Degree Works

Configure, 5.1.2

February 21, 2024

Notices and Privacy

© 2024 Ellucian.

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

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

Ellucian's Privacy Statement is available at: www.ellucian.com/privacy.

Ellucian shall have the right to (a) use, store, process, modify, reproduce, distribute and display customer data, and to grant sublicenses to third parties, for the sole purposes of providing the software, performing Ellucian's obligations under its agreements with customers and complying with applicable law or legal requirements; (b) use, store, process, modify and reproduce customer data for Ellucian's internal business purposes, including development, diagnostic, forecasting, planning, analysis and corrective purposes in connection with the software, and for otherwise improving and enhancing the software; and (c) use, store, process, modify, reproduce, display, perform, distribute, disclose and otherwise exploit in any manner Aggregated Data for Ellucian's business purposes, including disclosure within its public statements and marketing materials describing or promoting Ellucian or the software. "Aggregated Data" means any data obtained or generated by Ellucian, including data pertaining to the software, Ellucian's systems and software, and the use of any of the foregoing, and includes data derived from customer data, which in all instances (i) does not identify any individual and (ii) is not attributed or attributable to a specific customer. Aggregated Data includes data that has been combined into databases which include third party data.

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

Contents

Composer Administration.	15
Initial Composer setup and configuration.	15
Composer deployment.	15
Prerequisites.	15
Java.	15
SSL.	15
Composer environment settings.	16
Deployment.	18
Authentication.	18
Composer user access.	19
Shepherd settings.	19
UCX tables.	19
Initial load of resources.	19
Initial load of Shepherd scripts.	20
Composer Database Table Structure.	20
Composer navigation.	21
Notification Center.	21
Composer Editor.	21
Scripts.	23
Script selection.	23
Localized scripts.	23
Shepherd script details.	25
Resources.	25
Resources selection.	26
Localize resources.	26
Resource details.	28
Localize an application.	28
Composer debugging.	32
Controller Administration.	33
Initial Controller setup and configuration.	33
Authentication.	33
User access to Controller.	33

Shepherd settings.	34
Controller deployment.	35
Prerequisites.	35
Java.	35
SSL.	35
Controller environment settings.	35
Deployment.	38
Example Startup Script.	38
Example Stop Script.	39
Authorization.	39
Users.	39
User search.	39
Create and edit users.	40
User Information.	40
User Access.	41
User Keys.	42
User Groups.	42
Delete users.	43
Groups.	43
Search for groups.	44
Create and edit groups.	44
Group Details.	44
Group Keys.	45
Delete groups.	45
Configuration.	46
Search for configurations.	46
Shepherd settings.	46
Add specifications.	47
Delete specifications.	47
Add setting.	47
UCX tables.	48
Create and edit UCX entries.	48
Delete UCX entries.	48
Import and export UCX entries.	49
Properties.	49

Properties search.	49
Properties localization.	50
Scribe.	52
UCX Tables.	52
Manage Scribe Blocks.	53
Search for a block.	53
Search results.	54
Create a New Block.	55
Open Local Block.	55
Edit a Block.	56
Parse.	57
Block Details.	58
Save a Block.	58
Save.	59
Save As.	59
Save Local File.	59
Delete a Block.	60
Block printing.	60
Controller debugging.	60
Enable debugging.	60
Debug file access.	61
Ellucian Experience Integration.	62
Whitelist Ellucian Experience IP addresses.	62
Configure Ellucian Experience access to Degree Works.	63
Set up Degree Works cards in Ellucian Experience.	63
Troubleshooting the integration between Ellucian Experience and Degree Works	65
Responsive Dashboard Administration.	68
Initial Responsive Dashboard setup and configuration.	68
Authentication.	68
Responsive Dashboard user access.	68
Shepherd settings.	69
Responsive Dashboard deployment.	69
Prerequisites.	69

Responsive Dashboard environment settings. . . .	70
Deployment examples.	74
Requirements for the classic server.	75
Integration with portals.	75
Course Information.	78
Course Link.	78
Class registration from Course Link.	78
External course information.	80
API access from external applications.	81
Navigation.	82
Student.	83
Student search.	83
Student data.	85
Student data localization.	86
Custom student data display.	86
Student data refresh.	88
Tools.	90
Contact.	91
GPA Calculator.	91
Graduation Calculator.	92
Term Calculator.	92
Advice Calculator.	93
Class History.	93
Petitions.	93
Petition email notification (petsend).	94
Petition approval with Banner Workflow.	95
Notes.	96
Worksheets.	97
Academic.	98
Academic Audit.	98
Format.	98
Degree progress.	99
Process audit.	99
Historic audits.	99
Diagnostics.	100

Student Data.	100
Save audit.	100
Delete audit.	101
What-If.	101
What-If Analysis with no filtering.	101
What-If Analysis with curriculum-rule filtering.	103
What-If Analysis with degree-drives-major filtering.	105
What-If Analysis with major-drives-degree/school/college filtering.	107
What-If Analysis with major-drives-college filtering.	109
Future classes.	110
Historic audits.	111
Save audit.	111
Delete audit.	111
Athletic Eligibility.	112
Athletic Eligibility audit data setup.	114
Athletic Eligibility audit rule examples.	119
Athletic Eligibility audit Scribe reserved words.	123
Athletic Eligibility audit special topics.	125
CreditsAppliedTowardsDegree calculation.	125
CompletedTermCount calculation.	128
Transfer student athletes.	129
Football rule.	130
Batch Athletic Eligibility audits.	130
Financial Aid.	131
Financial Aid audit data setup.	132
Financial Aid audit rule examples.	134
Financial Aid audit Scribe reserved words.	136
Exceptions.	138
Add exception.	139
Exception types.	140
Force Complete.	140
Substitute.	140
Also Allow.	141
Apply Here.	141

Remove Course.	142
Qualifier exceptions.	143
Remove exception.	143
Exception Management.	143
Manage Petitions.	144
Awaiting Approval.	144
Approved Petitions.	145
Rejected Petitions.	145
Applied Petitions.	145
Fix Petition Status.	145
Exceptions Report.	146
Exception status codes.	147
Plans.	147
Printed plan.	148
Plan List.	148
Plan creating and editing.	149
Terms creating and editing.	151
Requirements creating and editing.	152
Course.	153
Choice.	155
GPA.	157
Non-course.	157
Test.	158
Placeholder.	158
Notes creating and editing.	159
Sidebar.	160
Courses.	160
Still Needed list.	161
Requirements.	161
Requirement drawer.	162
Planner audit.	162
Planner What-If audit.	163
Create Block.	164
Plan approval.	166
Course validation.	166

Prerequisite checking.	167
Tracking.	167
Current term.	168
Requirement tracking status.	169
Course and choice requirement tracking	169
GPA requirement tracking.	170
Non-course requirement tracking. . .	171
Test requirement tracking.	171
Term tracking status.	171
Overall plan tracking status.	171
Batch processes.	172
Scribe pointers.	172
Template Management.	177
Template list.	178
Flat view.	179
Tree view.	179
Template creating and editing.	180
Template tags.	181
Term schemes.	181
Requirements creating and editing.	182
Course.	183
Choice.	184
GPA.	185
Non-course.	186
Test.	186
Placeholder.	187
Notes creating and editing.	187
Sidebar.	188
Courses.	189
Requirements.	189
Transfer Finder.	189
Admin.	190
Debug.	190
Debug enabling.	190
Debug file access.	191

Theme.	191
Other localizations.	191
Links.	192
Transfer Equivalency Admin Administration.	194
Initial Transfer Equivalency Admin setup and configuration.	194
Authentication.	194
Transfer Equivalency Admin user access.	194
Shepherd settings.	195
UCX tables.	195
Database table structure.	196
Localizations.	197
Transfer Equivalency Admin deployment.	197
Prerequisites.	197
Transfer Equivalency Admin environment settings	198
Deployment examples.	201
Classic server requirements.	202
Navigation.	202
Student search.	203
Admin Mappings.	203
Transfer institution selection.	204
School mappings.	204
Mapping details.	205
Add a mapping in Transfer Equivalency Admin.	206
Modify an existing mapping in Transfer Equivalency Admin.	207
Test mappings.	208
Add a test mapping in Transfer Equivalency Admin	209
Modify a test mapping in Transfer Equivalency Admin	210
Define transfer conditions in Transfer Equivalency Admin.	211
Transcript.	211
Edit student information.	212
Manage test scores.	212
Manage transcripts.	213
Add school transcripts.	213

Articulation.	214
Articulation summary.	215
Articulated classes.	215
Audit.	218
Roll.	218
Roll process.	219
Roll results to Banner.	219
Debug.	220
Enable debug.	220
Access debug files.	221
Transfer Equivalency Self-Service Administration.	222
Initial Transfer Equivalency Self-Service setup and configuration.	222
Transfer Equivalency Self-Service user access.	222
Shepherd settings.	223
UCX tables.	224
Database table structure.	224
Transfer Equivalency Self-Service Deployment.	225
Prerequisites.	225
Transfer Equivalency Self-Service environment settings	226
Deployment examples.	229
Example startup script for	
TransferEquivalencyUI.	229
Example stop script for TransferEquivalencyUI.	
.	229
Monitoring.	229
Transfer Equivalency Self-Service navigation.	230
Log out.	230
Start over.	230
Save.	231
Process waypoint bar.	231
Landing page.	231
Create an account.	232
Log in with an existing account.	232
Goals page.	233
Your answers panel.	233

Use the standard UCX filter.	233
Use the curriculum rule filter.	234
Transfer page.	236
Add a transfer class.	236
Select a school.	236
Select a class.	236
Class information.	236
Add an exam.	237
Exam information.	237
My transfer work panel.	238
Results page.	238
Transfer audit.	239
Auto resolution.	239
Resolution options.	240
Download PDF.	243
Other options.	243
Transfer Equivalency Self-Service localization.	243
Application text, labels, and messages.	244
Graphics and styles.	244
Debug Transfer Equivalency Self-Service.	244
Debug access.	244
Debug enabling.	245
Debug file access.	245
Transfer Finder Administration.	247
Initial Transfer Finder Setup and Configuration.	247
Central server.	247
Central services user.	248
Central server Shepherd settings.	248
Central server UCX tables.	249
Global configuration resources.	250
Global directory.	251
School directory.	252
Transfer directory.	254
Local schools.	257

Global services user.	257
Local school Shepherd settings.	258
Local environments synchronization with local codes	259
Local rad_ets_mst synchronization with global translations.	260
User access to Transfer Finder.	261
Student extracts for student system ID.	261
Class schedule adapter.	261
classScheduleAdapter configuration.	262
Adapter implementation for other systems.	262
Transfer Finder scribing.	266
AuditAction=TRANSFER.	266
CATEGORY RuleTag.	267
Local block scribing.	267
Transfer block scribing.	271
Transfer block header.	272
Transfer Finder use.	273
Acknowledgment.	273
Classwork History.	273
Local classwork.	274
Partner school courses.	274
Student system ID.	275
Non-Partner schools.	275
Manually added classes.	275
Transfer Snapshot.	276
Program search.	276
Transfer Audit Summary.	277
Transfer Audit Detail.	278
Course Equivalencies.	278
Resolution options.	279
Transfer audit.	281
Apply now.	282
Find a transfer advisor.	282
Find Class Equivalent at Partner Schools.	283
Partnership Transfer Program Information.	284

Transit Administration.	285
Initial Transit setup and configuration.	286
Authentication.	286
Transit user access.	286
Shepherd settings.	287
Transit user interface deployment.	287
Prerequisites.	288
Transit environment settings.	288
Transit Batch Executor.	293
Transit Batch Executor configuration.	293
Job parameters.	294
Job scheduling.	295
The tbeshow script.	296
The tbestart script.	297
The tbestop script.	297
The tberestart script.	297
The launchjob script.	297
The jobstatus script.	299
The tbedelete script.	299
Job request flow through Transit.	300
Debug Transit.	301
Debug the Transit user interface.	301
Debug the Transit Executor.	302

Composer Administration

Updated: March 25, 2022

Composer is a tool for managing localizations in Degree Works.

With it, technical users can modify the Shepherd Scripts and XSL that are used to build the pages of the functions in the Degree Works Dashboard or change the images, labels, messages, colors, and styles used in most Degree Works Java applications.

The Composer deployment artifact is a JAR file which is executable using java. It includes an embedded container. A standalone container like Tomcat is not required.

Initial Composer setup and configuration

Composer deployment

Updated: March 25, 2022

Composer is designed to deploy equally well in on-premise and cloud hosted scenarios on all supported platforms. It provides a mobile-ready, responsive user interface communicating with the servers using lightweight, stateless, restful API over HTTPS SSL.

Prerequisites

Java

Updated: March 25, 2022

Java version 11 or above is the only third party dependency required to run this application.

SSL

Updated: March 25, 2022

In general, SSL is a best practice and is recommended for Composer because the security implementation uses stateless tokens.

Using signed certificates is recommended for all deployment scenarios, even test or development environments. Self-signed certificates, even for internal test deployments, can be problematic because of browser requirements and warnings. The SSL certificate needs to be configured in a keystore and that keystore will need to be accessible in the deployment environment. There are various options for how to do this, for example using Java's keytool command.

Composer environment settings

Updated: September 29, 2023

Several environment variables are required to be configured before running Composer.

Depending on the desired deployment platform there are many options for how these environment variables can be configured. The only requirement is that they are available in the environment where the product's JAR file will be executed. For example, it may be desirable to configure them in a shell script, in a specific user's profile, a startup instruction like init.d, or in a continuous deployment tool like Jenkins.

The minimum required variables are documented here. But, many optional variables are provided in the Spring Boot platform, for example tuning the embedded container. Please refer to Spring documentation for those: <http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>.

Environment Variables are the recommended method to configure these properties. But, there are various other ways these can be configured. Please refer to this documentation for more information: <http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html>.

Required environment variables

- **SPRING_DATASOURCE_URL**: A JDBC connection string for the Degree Works database. For example: `jdbc:oracle:thin:@{SERVER}:{PORT}/{SERVICE_NAME}` where `{SERVER}` is the address, `{PORT}` is the port of the listener, and `{SERVICE_NAME}` is the service name of the database.
- **SPRING_DATASOURCE_USERNAME**: The database username.
- **SPRING_DATASOURCE_PASSWORD**: The database password.
- **SERVER_PORT**: The desired port for Composer.
- **SERVER_SERVLET_CONTEXT_PATH**: An optional context path. If this is blank, the application will be deployed at the root URL like `https://myserver.edu:NNNN` (where NNNN is the **SERVER_PORT**). If a value is specified like `/my-context-path`, the application would be accessible at a URL like `https://myserver.edu:NNNN/my-context-path` (where NNNN is the **SERVER_PORT**).
- **SERVER_SSL_KEY_STORE**: The path to a keystore file you created to contain your SSL certificate.
- **SERVER_SSL_KEY_STORE_PASSWORD**: The password for your keystore file.
- **SERVER_SSL_KEYSTORETYPE**: The type of your keystore. JKS is the default.
- **SERVER_SSL_KEY_ALIAS**: The alias of the key you created.

Optional environment variables

- **JAVA_OPTIONS**: The memory allocation for the application can be defined using this variable, and is recommended. The value defines the initial heap size and max heap size and is in the

following format: `-Xms1024m -Xmx1024m`. The heap size values should be adjusted as appropriate for your server.

- **SERVER_MAX_HTTP_HEADER_SIZE**: This setting allows you to set the maximum size of the HTTP message header. By default, this is 8KB, and for some users with a large number of Shepherd access keys, this can be too small. It's recommended to use this variable and set it to at least 12KB. For example: `export SERVER_MAX_HTTP_HEADER_SIZE=12288`
- **SERVER_TOMCAT_REDIRECT_CONTEXT_ROOT**: Embedded Tomcat for Spring Boot has a default behavior to redirect a request without a trailing slash to one with a trailing slash. However, that redirect happens without evaluating forward headers like X-Forwarded-Proto. So, when SSL offloading is in place, the redirect happens to http instead of https. This can be an issue especially for load balanced environments. To disable the default behavior, set this variable to **false**.

Warning! You should do this only if the default behavior is not working. If you are offloading SSL at a proxy or load balancer, make sure to follow the instructions in the [Java application load balancing](#) topic. Then, if you have problems accessing the application in the browser when you do not include a trailing slash at the end of the request, try setting this variable to **false**.

- **DEBUG**: A boolean **true** or **false** which will enable or disable debug logging. The log file is by default named `Composer.log`. But, the location and name of that file can be customized by setting the `LOGGING_PATH` and `LOGGING_FILE` environment variables.
- **DEPLOY_LOCATION**: the location of the `Composer.jar` file. This location should be set then referenced in the `java -jar` command that starts the application.
- **LOGGING_FILE**: the name of the file including the path location. If this is specified, the `LOG_PATH` should not be used.
- **LOG_PATH**: the location of the log file. The default PATH location is the Java temporary java folder from the Java property `java.io.tmpdir`. This is usually `/tmp`. The file name will be `controller.log`.
- **LOG_DEFAULT_THRESHOLD**: If set to **DEBUG**, the application will output more verbose logging. Debug output will be sent to the log during the startup process and for all requests thereafter. Warning: this will result in very large log files. If not set, the debug threshold is **INFO**.
- **DW_ENABLE_MONITOR** - This environment variable can be set to enable the `/health`, `/metrics`, `/prometheus`, and `/status` endpoints for health and monitoring metrics. By default, this monitoring functionality is not enabled. Adding this variable to an application's startup script and setting it to true would enable these endpoints for the application. See individual API documentation in the API Catalog for additional information.
- **DW_RABBITMQ_SSLALGORITHM**: Specify the SSL algorithm to use when connecting to the RabbitMQ broker using SSL. If not specified, it uses the current RabbitMQ driver's default (at least TLSv1.2). This is only considered if the Shepherd setting `core.amqp.useSsl` is set to true.

SSL offloading

When SSL offloading is in use (SSL is terminated before reaching the application) with a load balancer, proxy server, and so on, some additional configuration needs to be done to make sure

Degree Works applications behave as expected. See the [Java application load balancing](#) topic for more information about this setup.

Deployment

Updated: March 24, 2023

When the prerequisites and configuration are in place, start up the application using a `java -jar` command like this one: `java $JAVA_OPTIONS -jar Composer.jar`.

Example startup script

If it is desirable to employ a shell script to start the application, here is an example. Note that the `Composer.jar` file must be staged at the `DEPLOY_LOCATION`:

```
#!/bin/ksh
export SPRING_DATASOURCE_URL="jdbc:oracle:thin:@MyServer:1521/service_name"
export SPRING_DATASOURCE_USERNAME="my-user"
export SPRING_DATASOURCE_PASSWORD="my-password"
export SERVER_PORT="8461"
export SERVER_SERVLET_CONTEXT_PATH=""
export SERVER_SSL_KEY_STORE="/usr/local/my-keystore.jks"
export SERVER_SSL_KEY_STORE_PASSWORD="my-key-password"
export SERVER_SSL_KEYSTORETYPE="jks"
export SERVER_SSL_KEY_ALIAS="my-keystore-alias"
export DEPLOY_LOCATION="/path/to/jarfile"
export JAVA_OPTIONS="-Xms1024m -Xmx1024m";
export DEBUG="false"
# To enable full-time debug log output, remove the # from the line below
#export LOG_DEFAULT_THRESHOLD=DEBUG
java -jar $JAVA_OPTIONS DEPLOY_LOCATION/Composer.jar > startup-log-file.log &
```

Example stop script

If using a start script like the example above, here is a corresponding example stop script:

```
#!/bin/ksh
pkill -f Composer
```

Authentication

Updated: March 25, 2022

Composer supports SHP, CAS, and SAML authentication methods and external access managers such as Oracle Access Manager.

Composer user access

Updated: March 25, 2022

The Shepherd key required for Composer is COMPOSER.

See [Users](#) for information about assigning this key to users.

Shepherd settings

Updated: March 25, 2022

Most configurations to manage the behavior of Composer can be maintained using the **Configuration** tab of Controller.

You must review the following Shepherd settings to ensure they are configured correctly for your environment:

```
localization.css.enable
localization.generatedPages.enable
localization.images.enable
localization.internationalization.enable
localization.internationalization.cacheMilliseconds
localization.xsl.enable
core.security.cas.callbackUrl.description spec=composer (if you use
CAS authentication)
```

UCX tables

Updated: March 25, 2022

Review the SHP080 UCX table to ensure that it is configured correctly for your Composer environment.

Initial load of resources

Updated: March 25, 2022

Before using Composer, Ellucian's standard properties, CSS, XSL, and images must be loaded into the database.

This initial load should have been done as part of the update or installation process and can be verified by querying the database to ensure the *shp_resource_mst* is not empty:

```
SQL> select count(*) from shp_resource_mst;
```

If the count returned is zero, please contact the Action Line for assistance with loading the baseline resources.

If you have any localized application properties, CSS, XSL, or images, you will need to manually apply those localizations using the Composer application.

Initial load of Shepherd scripts

Updated: March 25, 2022

Before using Composer, Ellucian's standard Shepherd scripts must be loaded into the database.

This initial load should have been done as part of the update or installation process and can be verified by querying the database to ensure the *shp_composer_mst* is not empty:

```
SQL> select count(*) from shp_composer_mst;
```

If the count returned is zero, please contact the Action Line for assistance with loading the baseline resources.

If you have any localized Shepherd scripts, make sure that they are in */local/shpscripts* and then run the *shpscriptsupdatelocal* script to update the *shp_composer_mst* with the localized source.

Composer Database Table Structure

Updated: March 25, 2022

Brief descriptions of the tables used in Composer are provided in the sub-sections that follow.

SHP_COMPOSER_MST

Used to store SHP Script data. Formerly in flat files under the */shpscripts* directory, this data now resides in the *shp_composer_mst* table. The *baseline_source* is delivered by Ellucian and is uneditable. The *baseline_version* reflects the release in which the script was delivered. Client localizations are stored in *local_source*, and when a localized script is activated, the *is_local_enabled* field is updated.

SHP_RESOURCE_MST

Used to store localized application resources such as properties, CSS, XSL and images. Formerly in flat files that were accessed by the application *WAR* or *JAR* file, this data now resides in the *shp_resource_mst* table. The *resource_type* for application properties is **PROPS** and the properties data is stored in the source field. The *resource_type* for cascading style sheets is 'CSS' and the CSS data is stored in the source field. The *resource_type* for extensible style sheets is **XSL** and the XSL data is stored in the source field. The *resource_type* for images is **IMAGE** and the image is stored in the source field. A copy of the Ellucian baseline is loaded into a database record separate from client localized versions and has the *baseline_version* set to

the release in which the property file was delivered. The baseline is uneditable and for reference only; the application uses the resources compiled in the *WAR* or *JAR* as the baseline and any localized resources in the database will override the baseline. Client localizations can be saved with a locale defined, but the default is a blank locale so that the localized resources will overwrite the baseline for all users, rather than just users of a particular region, language, or both. When a localized resource is activated, the *is_enabled* field is updated.

Composer navigation

Updated: March 25, 2022

A navigation bar at the top of Composer helps users navigate through the application. The user can access the navigation menu, online help and the notification center and log out of the application from the navigation bar.

Navigation menu

Access the navigation menu by clicking the Navigation Menu button on the left of the navigation bar. From the menu you can navigate to other pages in the application. **Home**, **Language**, **CSS**, **Scripts**, **XSL**, **Images**, and **Debug** (if the user has access to this functionality) are options on all pages.

Online help

The online help for Composer can be accessed by selecting **Help** from the Persona drop-down on the right of the navigation bar.

Logout

To log out of Composer, select Sign Out from the Persona drop-down button on the right of the navigation bar.

Notification Center

Updated: March 25, 2022

The Notification Center displays informational, success and error messages for Composer operations.

After taking an action such as saving a script or resource, the Notification Center will display a success or failure message. Additionally, a message advising that localizations aren't enabled or an existing resource could not be found may also appear in the Notification Center.

Composer Editor

Updated: March 25, 2022

Composer uses ACE (Ajax.org Cloud9 Editor) as its built-in editor. It provides a simple user interface to create and maintain the localizable resources used by Degree Works applications.

Find and Replace

To find and replace text, put the cursor focus in the editor window and issue the Find (Ctrl-F) or Replace (Ctrl-H) shortcut to open the *Find/Replace* toolbar. Note that because a Baseline record is not editable, the Replace toolbar will not open in this editor; it will only work in the Localized editor.

Shortcuts

Shortcuts are available in the Composer editor. Some are standard editor shortcuts and some are specific ACE.

Shortcut	Action
Ctrl-/	toggle comment; adds/removes # from front of selected lines, or on the current line if no lines are selected
Ctrl-A	select all text in the editor
Ctrl-C	Copy to clipboard
Ctrl-D	delete the current line, or the selected text
Ctrl-Down-arrow	scroll line down
Ctrl-End	go to the last line
Ctrl-F	find
Ctrl-H	replace
Ctrl-Home	go to the first line
Ctrl-L	go to line number
Ctrl-P	print
Ctrl-Shift-U	change to lower case – either the current word or the selected text
Ctrl-U	change to upper case – either the current word or the selected text
Ctrl-Up-arrow	scroll line up
Ctrl-V	paste from clipboard
Ctrl-X	cut selected text
Ctrl-Z	undo last change

Additional shortcuts for the ACE editor are available at <https://github.com/ajaxorg/ace/wiki/Default-Keyboard-Shortcuts>; however, all of these ACE shortcuts may not work in Composer and are not supported by Ellucian.

Scripts

Updated: March 25, 2022

Shepherd Scripts are used to create the Degree Works web interface.

They contain HTML and JavaScript code with special commands in a propriety format that are interpreted by the Degree Works software. While the colors, images, and styles used in the Degree Works web interface are configurable in CSS and properties files, any content localization is controlled in the Shepherd Scripts.

Shepherd Scripts are stored in the *shp_composer_mst*. A baseline version of these Shepherd Scripts is delivered by Ellucian at installation or in an update, and is not modifiable. When a Shepherd Script is localized, these changes are stored in a different field on the *shp_composer_mst* so the baseline version is always available for comparison and troubleshooting. A localized Shepherd Script needs to be enabled before it will be used by the Degree Works web interface.

Script selection

Updated: March 25, 2022

After selecting the Scripts from the navigation menu, a list of available Shepherd Scripts will display.

The list is populated with the Shepherd Scripts in *shp_composer_mst*. Shepherd Scripts are listed in alphabetical order by script name, and the list can be filtered by script name or description. Initially, only 10 results will be displayed but as the user scrolls down the page, additional results will be loaded.

The Localization column shows a non-editable indicator of whether or not the localized script has been enabled.

A counter at the top of the list of Shepherd Scripts indicates how many scripts records have been found.

Selecting a script will open it in the Editor page.

Localized scripts

Updated: March 25, 2022

Localized Shepherd Scripts are maintained on the Editor page. The user can create a localized version of a baseline script, update a localized script, enable a localized script or view script details.

The script **Description** will display with the script **Name** above the editor.

To edit a Shepherd Script, click on the **Editor** tab. To view script information such as **Last Modified** data, **Created** data or **Baseline Version**, click on the **Details** tab.

Create a localized script

Copy the existing baseline script into the localized editor by clicking the **Copy** button then make your modifications. You can also paste text from another source into the localized editor.

A script may have an unlimited number of lines, and each line is of unlimited length. However, due to limitations in the classic parser engine, we recommend a maximum line length of 200 characters.

Enable a localized script

To globally enable localized Shepherd Scripts in Degree Works, the `localization.generatedPages.enable` setting should be set to true. When this setting is true, then any script with `is_local_enabled = Y` will use the localized source; all other scripts will continue to use the baseline source. This setting should typically always be enabled, but it can be useful when troubleshooting issues or processing an update to set this to false temporarily.

Additionally, a localized script will not be used by Degree Works unless it has been enabled. To enable a localized script, click on the **Localization** toggle so it switches from Disabled to Enabled. Note that changing the **Localization** toggle triggers an auto save of the script.

Review the following Shepherd setting to ensure it is configured correctly for your Scribe environment:

```
localization.generatedPages.enable
```

Save a localized script

Changes to a localized Shepherd Script can be saved by clicking on the **Save** button or switching the **Localization** toggle. If the save is successful, a confirmation message will be displayed in the **Notification Center**.

Note that just clicking on **Save** only updates the database record; it does not enable the localization unless the **Localization** toggle was already enabled.

Delete a localized script

To delete a localized Shepherd Script, switch the **Localization** toggle to Disabled, select all the text in the Localized editor and delete it, then **Save** your changes.

Add a custom Shepherd script

If you have created a new custom Shepherd Script (not a localized version of a standard Ellucian Shepherd Script), follow these steps to add it to the `shp_composer_mst` so the Degree Works software will use it and you can use the Composer application to maintain it.

1. Make sure your custom script is in `/local/shpscripts`.
2. Use SQL to add a `shp_composer_mst` record for each custom Shepherd Script, specifying the correct values for NAME, DESCRIPTION, MODIFY_WHO, and CREATE_WHO:

```
SQL> INSERT INTO SHP_COMPOSER_MST (NAME,DESCRIPTION,BASELINE_SOURCE,
BASELINE_VERSION,LOCAL_SOURCE,IS_LOCAL_ENABLED,MODIFY_WHO,MODIFY_
DATE,
MODIFY_WHAT,CREATE_WHO,CREATE_DATE,CREATE_WHAT)
VALUES (trim('YOURCUSTOMSCRIPT'),
trim('Your Custom Script Description'),
EMPTY_CLOB(),' ',EMPTY_CLOB(),'N','DGWMGR',sysdate,'CUSTOM','DGWMGR',sysdate,'CUSTOM');
```

3. Run the *shpscriptsupdatelocal* script to update the custom script *shp_composer_mst* with the source.
4. Your custom script should now appear in the list of Shepherd Scripts in the Composer.

Shepherd script details

Updated: March 25, 2022

Shepherd Script details can be accessed by selecting the Details tab on the Editor page.

This display-only information shows the **Last Modified Date** and **Who**, **Created Date** and **Who** and the **Baseline Version**.

The **Last Modified Date** and **Who** will be updated when changes are saved to the localized script.

The **Created Date** and **Who** reflect when the script was initially loaded into the database.

The **Baseline Version** is the Degree Works release in which that version of the baseline source was delivered.

Resources

Updated: March 25, 2022

Resources like properties, CSS, XSL, and images are used to localize the user interface of Degree Works applications.

These resources are not only used to change the default values for an application, but also create a custom experience for users in different countries. Properties contain all of the text displayed in an application. CSS – or cascading style sheets – defines the style of the text displayed in an application. XSL defines the appearance of a page.

Resources are stored in the *shp_resource_mst*. A baseline version of these resources is delivered by Ellucian at installation or in an update, and is not modifiable. When a resource is localized, these changes are saved in a separate *shp_resource_mst* record with a specific locale defined, and the baseline version is always available for comparison and troubleshooting. A localized resource needs to be enabled before it will be used by the Degree Works web interface.

Resources selection

Updated: March 25, 2022

After selecting the desired resource from the navigation menu – Language, CSS, XSL or Images – a list of available resources will display.

The list is populated with the resources in shp_resource_mst for the selected resource type. Resources are listed in alphabetical order by name, and the list can be filtered by resource name or description. Initially, only 10 results will be displayed but as the user scrolls down the page, additional results will be loaded.

A counter at the top of the list of resources indicates how many resource records have been found.

Selecting a resource will open it in the Editor page.

Localize resources

Updated: March 25, 2022

Localized resources are maintained on the Editor page. The user can create a localized version of a baseline resource, update a localized resource, enable a localized resource or view resource details.

The resource **Description** will display with the properties file **Name** above the editor.

To edit a resource, click on the **Editor** tab. To view resource information such as **Localization Last Modified** data, **Localization Created** data or **Baseline Version**, click on the **Details** tab.

When a resource is initially loaded in the Editor, the baseline version will load in the **Baseline Resource** editor, while the localized resource for the default locale will load in the **Localized Resource** editor, if it exists. The user can select a different locale from the **Locale** dropdown under the **Localized Resource** editor to either load that properties file or create it.

If a localized resource for the selected locale cannot be found when loading a resource in the Editor, a message for the user will display in the notification center.

Locales

The list of Locales is populated from SHP080, and contains a Description and the actual locale code. The locale code is the IETF language tag, and typically consists of language and country codes, or just a language code. Based on settings in a user's browser, the correct properties file will be loaded for that user so that the application text is appropriate for their location.

Examples:

- jp – for the Japanese language translation
- es – for the Spanish language translation

- `es_MX` – for the Spanish language translation, specific to Mexico
- `es_CL` – for the Spanish language translation, specific to Chile

Note that a default localized resource can be saved without a locale. The values in this default localized resource will overwrite the values in the baseline resource.

Create a localized resource

To create a localized version of a resource, copy the existing baseline resource into the localized editor by clicking the **Copy** button then make your modifications. Ideally, any values that are not localized should be removed from the localized resource so it is clear which values are expected to be localized and which should be baseline. You can also paste text from another source into the localized editor.

A resource may have an unlimited number of lines, and each line is of unlimited length.

The **Copy** option is not available for Images. Instead, click on **Choose File** and navigate to the image file on your computer. Click **Open** to load the image into the localized editor.

Enable a localized resource

To globally enable localized resources in Degree Works, the applicable enable setting should be set to true.

- `localization.css.enable` – enables localized CSS
- `localization.images.enable` – enables localized images
- `localization.internationalization.enable` – enables localized language properties
- `localization.xsl.enable` – enables localized XSL

When these settings are true, then localized resources of the corresponding type with `is_enabled = Y` will be used by the application. These settings should typically always be enabled, but it can be useful when troubleshooting issues or processing an update to set them to false temporarily. Two related settings, `localization.images.cacheMilliseconds` and `localization.internationalization.cacheMilliseconds`, determine for how long resources will be cached. Wait at least this long after making modifications to properties in Composer before refreshing browser to see those changes reflected.

Additionally, a localized resource will not be used by Degree Works unless it has been enabled. To enable a localized resource, click on the **Localization** toggle so it switches from Disabled to Enabled. Note that changing the **Localization** toggle triggers an auto save of the script.

Review the following Shepherd setting to ensure it is configured correctly for your Scribe environment:

```
localization.css.enable
localization.generatedPages.enable
localization.images.cacheMilliseconds
localization.images.enable
localization.internationalization.cacheMilliseconds
```

```
localization.internationalization.enable  
localization.xsl.enable
```

Save a localized resource

Changes to a localized resource can be saved by clicking on the **Save** button or switching the **Localization** toggle. If the save is successful, a confirmation message will be displayed in the **Notification Center**.

Note that just clicking on **Save** only updates the database record; it does not enable the localization unless the **Localization** toggle was already enabled.

Delete a localized resource

To delete a localized resource, switch the **Localization** toggle to Disabled, select all the text in the Localized editor and delete it, then **Save** your changes.

Resource details

Updated: March 25, 2022

Resource details can be accessed by selecting the Details tab on the Editor page.

This display-only information shows the **Localized Last Modified Date** and **Who**, **Localized Created Date** and **Who** and the **Baseline Version**.

The **Localized Last Modified Date** and **Who** will be updated when changes are saved to the localized resource. These fields will not display unless a localized resource for the selected locale exists in the database.

The **Localized Created Date** and **Who** reflect when the localized resource was initially created. These fields will not display unless a localized resource for the selected locale exists in the database.

The **Baseline Version** is the Degree Works release in which that version of the baseline source was delivered.

Localize an application

Updated: March 25, 2022

The baseline versions of language properties, CSS, XSL, and images are embedded with the application's WAR or JAR file.

With Composer, you can localize these resources or create an internationalized version that is saved as a database table, and when the application is run, your localized values will overwrite the baseline values for the user, based on how their default language is configured in their browser. There is a hierarchy to how baseline and localized resources are merged together:

- First it looks at the baseline
- Next, the default (blank locale) localization
- Then the one code locale (e.g. "en" without the _US)
- And finally, for the most specific, two code locale (e.g. en_US)

To modify or to create a customized resource, select the appropriate resource from the Resource List, open it in the Editor and choose the desired locale. The initial locale when first loading a resource in the editor is Default, or a blank locale. You can cut and paste the properties you want to localize from the Baseline Editor into the Localized Editor.

Warning! Ellucian strongly recommends copying only the properties you want to localize. Copying the entire baseline resource to the localized resource will likely cause problems with future upgrades.

In special cases, you might want to override localized properties in the consuming application. For example, in Responsive Dashboard, you can override localized properties defined in AcademicGoalsMessages.properties (used in what-if) from within DashboardMessages.properties. In that case, the localization in the latter takes precedence.

Language properties

The text, labels and messages that appear in most Degree Works Java-based applications – Dashboard Servlet, Student Educational Planner, Scribe, Composer, Transfer Equivalency Self-Service and Transfer Finder and all APIs – come from language properties.

Each of the entries in language properties is made up of two sections, the KEY and the TEXT. For example, the following entry is for the label that is displayed for the School (Level) drop down list is as follows:

```
dw.scribe.blockSearchResults.label.school=School
```

In the example, `dw.scribe.blockSearchResults.label.school` is the KEY and `School` is the TEXT part of the entry.

To change the message, modify the TEXT part of the entry. The modified text will be displayed to users.

Note: Do not modify the KEY, this is a reference used by the application, and if modified, the message will not be displayed.

For example, if you want to change the default school name in Transfer Equivalency Self-Service for all users, you would create a default localization. In the Editor, select the Default locale and then add the properties you want to localize:

```
app.site.title=My University
```

Save and enable this record, and then load Transfer Equivalency Self-Service – for all users, regardless of their browser language preference, the school name in the header should be “My University”, and not the default “University Name”.

If you like a text in the language user prefer, you can create separate records specific to their locale. For example, to have dialogue box buttons in SEP be in Dutch for a student in The Netherlands and in casual English for a student in the United States, you need to create a localization record for each locale. First, select the *Dutch - nl* locale and add these values:

```
dw.sep.button.label.yes=Ja  
dw.sep.button.label.no=Nee
```

Save and enable this record. Then create a new record with the “English (United States) – en_US” locale and add these values:

```
dw.sep.button.label.yes=Sure  
dw.sep.button.label.no=Nah
```

Save and enable this record. Now, a student in The Netherlands will see “Ja/Nee” on the dialog box buttons, a student in the United States will see “Sure/Nah” on these buttons, and a user in English-speaking Canada will see **Yes/No** – which is the baseline text for these buttons.

Note that you can add whichever locales you want to support in SHP080.

CSS

The colors, styles and graphics in Degree Works Java-based applications – Dashboard Servlet, Student Educational Planner, Scribe, Composer, Transfer Equivalency Self-Service and Transfer Finder and all APIs – can be configured through style sheets (CSS).

Styles sheets contain a series of classes that define an application’s components. For example, the following class is for the header bar at the top of the Dashboard Servlet:

```
.HeaderBar  
{  
    min-height: 55px;  
    background-color: #5C798F; /* Ellucian blue*/  
}
```

To change the color of this header bar, modify the background-color part of the entry.

For example, if you want to change the color of the block header in the audit worksheet for all users, you would create a default localization. In the Editor, select the Default locale and then add the class you want to localize and change the background-color value:

```
.BlockHeader  
{  
    font-family: Tahoma, helvetica, arial, serif;  
    font-size: 10pt;  
    font-weight: bold;  
    background-color: # FF5733; /*My School Color*/  
    background-image: url("common/bg-header.gif");  
    background-repeat: repeat-x;  
    color: #FFFFFF;  
    text-align: left;  
    padding-right: 2px;
```

```
padding-left: 2px;
}
```

Save and enable this record, and then load Dashboard Servlet – for all users, regardless of their browser language preference, the block header color in the audit worksheet should be a shade of orange, and not the default “Ellucian Blue”.

XSL

The formatting of XML content in the Degree Works web interface can be configured through XSL files. XSL stylesheets contain one or more template elements, which describe what action to take when a given XML element or query is encountered in the target file. For example,

```
<xsl:variable name="vLabelDegree" >Degree<xsl:variable>
```

will replace any instances of the <vLabelDegree> variable in the target XML file with “Degree”. If you want to change the text of this variable for all users, you would create a default localization. In the Editor, select the Default locale and copy the existing baseline XSL into the localized editor by clicking the **Copy** button then make your modifications:

```
<xsl:variable name="vLabelDegree" >Program of Study<xsl:variable>
```

Save and enable this record, and then load Dashboard Servlet – for all users, regardless of their browser language preference, the header in the audit worksheet will display “Program of Study”, and not the default “Degree”.

Please note that the Responsive Dashboard does not make use of stylesheets and thus modifying the stylesheets in Composer has no effect on the Responsive Dashboard.

Images

The images used in Degree Works can be localized by creating a localization with the new image which will be used in all instances where the image is referenced. For example, the Ellucian logo in the upper left of the Dashboard is Icon_DegreeWorks.gif and is referenced in several places, such as the SD2GENHDR Shepherd script:

```
<td align="left" valign="middle">

</td>
```

If you want to change this image to your school logo for all users, you would create a default localization. In the Editor, select the Default locale, click on **Choose File** and navigate to your logo image file on your computer, click **Open** to load the image into the localized editor. Save and enable this record, and then load Dashboard Servlet – for all users, regardless of their browser language preference, your school logo will display in the upper left, and not the Ellucian logo.

It is recommended that your localized image has the same file format as the source image. Some browsers may have trouble rendering the image correctly if the format is different.

Please note that the Ellucian logo in the Responsive Dashboard cannot be localized using Composer. Instead, use the Theme editor in the Responsive Dashboard to specify the URL of your school image. Composer cannot be used to localize any of the images in Responsive Dashboard.

Composer debugging

Updated: March 25, 2022

Composer users can enable debugging to troubleshoot issues for their session from the application.

Users with access to this functionality will be able to enable and disable debug from a page in Composer. All debugging output for the application is aggregated into one file, even if multiple applications or APIs are involved. Access to the application server is not needed and the application does not need to be restarted. However, access to the server is required to access the generated log file, which will be in the logs directory of the application server.

Access the debugging functionality

To access the debugging functionality, users should be assigned the **DEBUG** key either through the Users function of Controller or in SHPCFG. Users with this key will see the Debug option in the Navigation Menu. Clicking on this link will take the user to the Debug page.

Enable debugging

On the Debug page, toggle the Enable Debug switch to turn debugging on for the Composer application. A random Debug Tag will be generated, and the user can copy this to their clipboard by clicking on the page icon. This Debug Tag will be appended to each line of debug in the log file so that the user can easily identify which lines are from their session, as opposed to other users using the application at the same time.

After enabling debug, the user can then resume using Composer and execute the steps to duplicate the issue they are experiencing. Debug should be turned off by going back to the Debug page and toggling the Enable Debug switch when the issue has been duplicated. Debug for this user's session will also be turned off when they log out of the application.

Access debug files

Debug will appear in the Composer log file on the application server. This is typically defined by the \$LOGGING_FILE environment variable.

Searching for the user's Debug Tag will isolate the lines of debug specific to their actions. For example:

```
2017-10-24 16:22:23,144 68b4e19d-3fb4-4975-9815-83eea01266ff DEBUG [
41-exec-17]
.h.d.s.s.StatelessPreAuthenticatedFilter : Checking secure context t
oken: null
```

The packdebug script can be used to collect the log files generated for a user session's Debug Tag. For more information on using this script, see the [packdebug script](#) topic.

Controller Administration

Updated: March 25, 2022

Controller is a tool Degree Works administrators use to manage user authorization, application configuration settings, validation tables, and application language properties, and to code and store degree and program requirements.

The Controller deployment artifact is deployed as a jar on any supported Unix server and is independent of the classic software. It includes an embedded container so a separate Java container like Tomcat is not required.

Initial Controller setup and configuration

Authentication

Updated: March 25, 2022

The Controller user interface supports SHP, CAS and SAML authentication methods, and external access managers such as Oracle Access Manager.

User access to Controller

Updated: March 25, 2022

The CONTROL Shepherd key grants access to the Controller application.

A user also needs at least one of the keys that grant access to the functionality menus – CTLUSERS for Authorization - Users, CTLGROUP for Authorization - Groups, CTLCONF for Configurations, CTLPROP for Properties, or SCRIBE for Scribe. Additional Shepherd keys grant access to adding, modifying, and deleting.

The CONTROL Shepherd group can be used to assign access to users for Authorization, Configuration, and Properties.

Standard Shepherd Groups that give access to Scribe functionality are SCRIBREG, SCRIBAID, and SCRIBAEA.

The Shepherd keys required for Scribe are SCRIBE and SCRPARSE. Additionally, users should be given the keys for the block types they have access to modify. Most users can be provided the SCRBLALL key that grants access to modify all block types. Users with limited access should be given the keys for the appropriate block type(s).

The list of all Controller keys and groups is available in the [Keys and keyrings](#) topic.

Key	Block type
SCRBLATH	ATHLETE
SCRBLAWR	AWARD
SCRBLCOL	COLLEGE
SCRBLCON	CONC
SCRBLDEG	DEGREE
SCRBLID	ID
SCRBLLIB	LIBL
SCRBLMAJ	MAJOR
SCRBLMIN	MINOR
SCRBLOTH	OTHER
SCRBLPRG	PROGRAM
SCRBLREQ	REQUISITE
SCRBLSCH	SCHOOL
SCRBLSPC	SPEC

Shepherd settings

Updated: March 25, 2022

Most configurations to manage the behavior of Controller can be maintained using the Configuration function of Controller.

For Scribe, review and configure the following settings.

```
classicConnector.dap08.port
```

```
classicConnector.serverNameOrIp
```

```
scribe.*
```

If you use CAS authentication, review the following settings. You will need to restart Controller after changing these.

```
core.security.cas.callbackUrl spec=controller
```

```
core.security.authenticationType = CAS for spec=controller (if spec=default is not already configured this way)
```

Controller deployment

Updated: March 25, 2022

Controller is designed to deploy equally well in on-premise and cloud hosted scenarios on all supported platforms. It provides a mobile-ready, responsive user interface communicating with the servers using lightweight, stateless, restful API over HTTP SSL.

Prerequisites

Java

Updated: October 28, 2020

Java version 11 or above is the only third-party dependency required to run this application. Oracle Java and OpenJDK are supported.

SSL

Updated: October 28, 2020

SSL is in general a best practice and is recommended for Controller because the security implementation uses stateless tokens. Using signed certificates is recommended for all deployment scenarios, even test or development environments.

Self-signed certificates, even for internal test deployments, can be problematic because of browser requirements and warnings. The SSL certificate needs to be configured in a keystore and that keystore will need to be accessible in the deployment environment. There are various options for how to do this, for example using Java's keytool command.

Controller environment settings

Updated: September 29, 2023

Several environment variables are required to be configured before running the application.

Depending on the desired deployment platform there are many options for how these environment variables can be configured. The only requirement is that they are available in the environment where the product's JAR file will be executed. For example, it may be desirable to configure them in a shell script, in a specific user's profile, a startup instruction like init.d, or in a continuous deployment tool like Jenkins.

The minimum required variables are documented here. But, many optional variables are provided in the Spring Boot platform, for example tuning the embedded container. Please refer to Spring documentation for those: <http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>.

Environment Variables are the recommended method to configure these properties. But there are various other ways these can be configured. Please refer to this documentation for more information: <http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html>.

Required environment variables

- **DW_DATASOURCE_URL:** A JDBC connection string for the Degree Works database. For example: `jdbc:oracle:thin:@{SERVER}:{PORT}/{SERVICE_NAME}` where `{SERVER}` is the address, `{PORT}` is the port of the listener, and `{SERVICE_NAME}` is the service name of the database. This is the same database and schema used by the classic Degree Works software.
- **DW_DATASOURCE_USERNAME:** The Degree Works database username.
- **DW_DATASOURCE_PASSWORD:** The Degree Works database password. This can be encoded. To get the encoded value, use the `showdbpasswords --password` command in the classic environment. It will display the encoded value to place here (e.g. `ENC(skdfjldjs)`).
- **SERVER_PORT:** The desired port for Controller.
- **SERVER_SERVLET_CONTEXT_PATH:** An optional context path. If this is blank, the application will be deployed at the root URL like `https://myserver.edu:NNNN` (where `NNNN` is the `SERVER_PORT`). If a value is specified like `/my-context-path`, the application would be accessible at a URL like `https://myserver.edu:NNNN/my-context-path` (where `NNNN` is the `SERVER_PORT`).
- **SERVER_SSL_KEY_STORE:** The path to a keystore file you created to contain your SSL certificate.
- **SERVER_SSL_KEY_STORE_PASSWORD:** The password for your keystore file.
- **SERVER_SSL_KEYSTORETYPE:** The type of your keystore. JKS is the default.
- **SERVER_SSL_KEY_ALIAS:** The alias of the key you created.

Optional environment settings

- **JAVA_OPTIONS:** The memory allocation for the application can be defined using this variable, and is recommended. The value defines the initial heap size and max heap size and is in the following format: `-Xms1536m -Xmx1536m`. The heap size values should be adjusted as appropriate for your server.
- **SERVER_MAX_HTTP_HEADER_SIZE:** This setting allows you to set the maximum size of the HTTP message header. By default, this is 8KB, and for some users with a large number of Shepherd access keys, this can be too small. It's recommended to use this variable and set it to at least 12KB. For example: `export SERVER_MAX_HTTP_HEADER_SIZE=12288`
- **SERVER_TOMCAT_REDIRECT_CONTEXT_ROOT:** Embedded Tomcat for Spring Boot has a default behavior to redirect a request without a trailing slash to one with a trailing slash. However, that redirect happens without evaluating forward headers like `X-Forwarded-Proto`. So, when SSL offloading is in place, the redirect happens to `http` instead of `https`. This can be an issue especially for load balanced environments. To disable the default behavior, set this variable to false.

Warning! You should do this only if the default behavior is not working. If you are offloading SSL at a proxy or load balancer, make sure to follow the instructions in the [Java application load balancing](#) topic. Then, if you have problems accessing

the application in the browser when you do not include a trailing slash at the end of the request, try setting this variable to **false**.

- **DEBUG**: A boolean true or false which will enable or disable debug logging. The log file is by default named `Controller.log`. But, the location and name of that file can be customized by setting the `LOGGING_PATH` and `LOGGING_FILE` environment variables.
- **DEPLOY_LOCATION**: the location of the `Controller.jar` file. This location should be set then referenced in the `java -jar` command that starts the application.
- **LOGGING_FILE**: the name of the file including the path location. If this is specified, the `LOG_PATH` should not be used.
- **LOG_PATH**: the location of the log file. The default PATH location is the Java temporary java folder from the Java property `java.io.tmpdir`. This is usually `/tmp`. The file name will be `controller.log`.
- **LOG_DEFAULT_THRESHOLD**: If set to `DEBUG`, the application will output more verbose logging. Debug output will be sent to the log during the startup process and for all requests thereafter. Warning: this will result in very large log files. If not set, the debug threshold is `INFO`.
- **DW_ENABLE_MONITOR** - This environment variable can be set to enable `/health`, `/metrics`, `/prometheus`, and `/status` endpoints for health and monitoring metrics. By default, this monitoring functionality is not enabled. Adding this variable to an application's startup script and setting it to true would enable these endpoints for the application. See individual API documentation in the API Catalog for additional information.
- **DW_RABBITMQ_SSLALGORITHM**: Specify the SSL algorithm to use when connecting to the RabbitMQ broker using SSL. If not specified, it uses the current RabbitMQ driver's default (at least TLSv1.2). This is only considered if the Shepherd setting `core.amqp.useSsl` is set to true.

Database pool configuration for Microservice applications

For production deployments of java applications, it is important to be able to configure the size of the database connection pool. In Tomcat this is done through the Resource definition in `server.xml`. For microservice applications like Controller, there are environment variables that can be used for this configuration.

These environment variables start with `DW_DATAPOOL_`. They end with the name of the datasource pooling attribute you want to control. These are the most common ones:

Attribute	Meaning
<code>initialSize</code>	The initial number of connections that are created when the pool is started.
<code>maxActive</code>	The maximum number of active connections that can be allocated from this pool at the same time.
<code>maxIdle</code>	The maximum number of connections that can remain idle in the pool, without extra ones being released.
<code>minIdle</code>	The minimum number of active connections that can remain idle in the pool, without extra ones being created.

The name of the attribute should be given in all uppercase, with an underscore separating the words (denoted by a capital letter in the table above). To set the maximum number of active connections in the pool, for example, put the following line in your startup script:

```
export DW_DATAPPOOL_MAX_ACTIVE=100
```

You can set any of the properties of a version 1.4 Apache Commons BasicDataSource object. These are documented at <http://commons.apache.org/proper/commons-dbcp/api-1.4/org/apache/commons/dbcp/BasicDataSource.html>.

The number of active connections and the current configured values for the limits can be monitored using any JMX (Java Management Extensions) client, such as JConsole, attached to the microservice JVM. The values will appear under the mbean net.hedtech.degreeworks/JmxBasicDataSource/datasource. The getState operation will return a formatted report of the current pool state.

SSL offloading

When SSL offloading is in use (SSL is terminated before reaching the application) with a load balancer, proxy server, and so on, some additional configuration needs to be done to make sure Degree Works applications behave as expected. See the [Java application load balancing](#) topic for more information about this setup.

Deployment

Updated: October 28, 2020

When the prerequisites and configuration are in place, start up the application using a "java -jar" command.

```
java $JAVA_OPTIONS -jar Controller.jar
```

Example Startup Script

Updated: March 24, 2023

You can employ a shell script to start the application.

That the *Controller.jar* file must be staged at the DEPLOY_LOCATION.

```
#!/bin/ksh
export DW_DATASOURCE_URL="jdbc:oracle:thin:@MyServer:1521/service_name"
export DW_DATASOURCE_USERNAME="my-dw-user"
export DW_DATASOURCE_PASSWORD="my-dw-password"
export SERVER_PORT="8501"
export SERVER_SERVLET_CONTEXT_PATH=""
export SERVER_SSL_KEY_STORE="/usr/local/my-keystore.jks"
export SERVER_SSL_KEY_STORE_PASSWORD="my-key-password"
export SERVER_SSL_KEYSTORETYPE="jks"
export SERVER_SSL_KEY_ALIAS="my-keystore-alias"
export DEPLOY_LOCATION="/path/to/jarfile"
export JAVA_OPTIONS="-Xms1536m -Xmx1536m";
```

```
export DEBUG="false"
# To enable full-time debug log output, remove the # from the line below
#export LOG_DEFAULT_THRESHOLD=DEBUG
java -jar $JAVA_OPTIONS DEPLOY_LOCATION/Controller.jar > startup-log
-file.log &
```

Example Stop Script

Updated: March 25, 2022

If you use a start script, you must use a corresponding example stop script.

```
#!/bin/ksh
kill -f Controller
```

Authorization

Updated: March 25, 2022

Under the Authorization menu, user access to Degree Works applications and functionality can be managed. This is either done for an individual user in the Users tab, or for a user role in the Groups tab.

Users

Updated: March 25, 2022

In the Users tab, user access records can be created or modified. These user access records are used to manage user access to Degree Works applications, and in conjunction with the core.security.rules.shpcfg Shepherd setting help construct a user's authorization keyring.

Typically, user access records for students, advisors and applicants are created through the data bridge extract from the student records system. These records can be maintained in Controller. For other users of Degree Works who might not have data in the student records system, such as staff, a user access record can be created in Controller so they can be granted access to Degree Works functionality.

For additional information about user authorization, please see the [Access control \(authorization\)](#) topic.

Configuration

Access to Users is granted if the user has the CTLUSERS Shepherd key.

User search

Updated: March 25, 2022

Existing user access records can be searched for by username, access ID, SSO ID, or Degree Works ID.

These correspond to the *shp_name*, *shp_access_id*, *shp_sso_id*, and *shp_box_id* columns in the *shp_user_mst* database table, respectively. To search for user access records, type a keyword string and hit Enter. A paginated list of matching records will be returned. The list is initially sorted alphabetically ascending by the name, but the grid can be sorted by any column by clicking on the column header. An arrow will appear next to the column header to indicate that it is being used for the sort, and the direction of the arrow indicates whether it is ascending or descending.

Existing user access records may also be searched by entering a few characters allowing a suggestive set of records to display below the search bar. Clicking a particular user will populate the respective details into the search results grid.

The pagination details at the bottom of the search results grid allow the user to page to additional matching records, control how many records per page should display and how many total matching records were found.

Create and edit users

Updated: March 25, 2022

A new user access record can be created by clicking on the Add new user button on the User search page. A user access record can be edited by clicking on the edit icon in the search results grid.

To edit a user access record, click the edit icon. A card with all the access details for this user will load.

After modifying the details for the user, click the Save button to keep your changes.

Configuration

Access to the following is granted if the user has the specified Shepherd key:

- Add user access records - CTLUSRAD
- Edit user access records - CTLUSRMD

User Information

Updated: March 25, 2022

Information that identifies the user is detailed in the User Information section.

The **User name** that displays in Degree Works applications can be added or modified; this is typically initially loaded for most users through the data bridge.

A user access record is usually linked to a **Degree Works ID**. This is typically done when a user is loaded through the data bridge, but can be changed or added when manually creating a new user access record. Click the **Search** icon to search the Degree Works database by either the *rad_id*

or `rad_name` on the `rad_primary_mst`. Select the correct record to link the **Degree Works ID** to the user access record. The `rad_name` will display as the **Name** for reference.

When adding a user, a unique **Access ID** must be provided. This is the ID that the user will use to access Degree Works applications. It can be up to 14 alphanumeric characters in length. When a user access record has been saved, the **Access ID** cannot be changed.

If your environment uses SHP authentication, you can use Controller to manage the user's password. Other methods of authentication such as CAS or SAML will not use this password. This password cannot be displayed after the user access record has been saved, but to view the password while entering it, click "Show". The password will be saved as encrypted when password encoding is enabled.

If you are using single sign-on authentication, the **Single sign-on ID** is an optional field that stores the user's ID. For example, the ID used in CAS authentication.

If you are using Transfer Finder, the **Transfer Finder ID** is an optional field that stores the common system ID used by Transfer Finder.

If you are using Transfer Equivalency Self-Service, the **Alternate username** is an optional field that stores the prospective student's email address used for logging in.

Configuration

Review the `core.security.passwordEncoding.enabled` Shepherd setting to ensure it is configured correctly.

User Access

Updated: September 30, 2022

Details about how the user can access Degree Works applications are stored in the User Access section.

For any Degree Works application, the date and time of the user's last successful logon and their last failed logon attempt are displayed, and the Fail count displays the number of times the user has entered an Access ID or Password that is not valid during logon. The count can be reset by clicking the **Reset count** button.

User class is a required field, and this defines the services that the user can access. The user class drop-down is populated by AUD012.

Certain users may be granted access to have students from a designated major(s), school(s), or college(s) pre-load in the Responsive Dashboard. To add an advisee filter, click the **Add filter** button and select a Department, School, or College filter type. A user can only be assigned one advisee filter type. Then select the major, school, or college from the drop-down. Up to 10 filter codes can be added by clicking the add filter icon. Filters can be removed by clicking the delete icon. Assigned filters for the user will display in the interface. These can be deleted individually or en masse by clicking **Clear all**.

You can restrict a user's access to Degree Works by enabling the **Deny** flag, which prevents a user from being able to access any service on their next logon. If you want to set an expiration date for the account, specify it in the **User expire** field.

Use the Timeout increment to specify the time after which a user's session should automatically end, if the user is inactive. When the user first gets authenticated, the timeout is set to the current time plus the timeout increment. This value is the time when the session will end if the user is inactive. Inactive is defined as not requesting a Degree Works service within a specific time frame. If the user stays active then the timeout value will be updated. The timeout value is updated by taking the current time and adding on the timeout increment. The Timeout maximum specifies the maximum time for which a user session can be inactive before ending. The timeout value cannot exceed the initial authentication time plus the timeout maximum. As long as the user stays active, the session will not end within the maximum time allotted. If the timeout fields are not filled in, the timeouts are from the user's groups or from the `core.security.passport.timeout*` Shepherd settings. If you want to keep the user logged in irrespective of the inactive time, select the **Never timeout** check box.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

```
core.security.passport.timeoutIncrementDefault
core.security.passport.timeoutMaximumDefault
core.security.passport.timeoutPrecedence
core.security.passport.timeoutPreference
core.authorization.adviseeFiltering.college.enabled
core.authorization.adviseeFiltering.department.enabled
core.authorization.adviseeFiltering.school.enabled
```

User Keys

Updated: March 25, 2022

User access to Degree Works services is managed by Shepherd keys.

Typically, users are assigned Shepherd keys explicitly by role through the `core.security.rules.shpcfg` Shepherd setting. But refined access can be assigned to or removed from the user implicitly using Controller.

To add a key, click the **Add key** button. Select the key that you want to add to the user's keyring. From the **Operator** drop-down, choose whether you want to grant or deny access to this key for this user. If you want to set a time limit for this key, click the calendar icon for the Expire date and set a date after which the user will no longer have permissions to the service and the key will be removed from the user's keyring.

All keys assigned to a user will display in a grid and are initially sorted alphabetically ascending by the key. This grid can be filtered by Shepherd key or description. Keys can also be sorted by clicking any of the column headers. To modify a key, click on the **Edit** icon. To delete a key, click on the **Delete** icon.

Configuration

Review the `core.security.rules.shpcfg` Shepherd setting to ensure it is configured correctly.

User Groups

Updated: March 25, 2022

Shepherd keys related to a particular service or role can be grouped together in Shepherd groups so user access can be more easily assigned.

Typically, users are assigned Shepherd groups explicitly by role through the `core.security.rules.shpcfg` Shepherd setting. But refined access can be assigned to or removed from the user implicitly using Controller.

To add a group click the **Add group** button. Select the group that you want to add to the user's keyring. From the **Operator** dropdown, choose whether you want to grant or deny access to this group for this user. If you want to set a time limit for this group, click the calendar icon for the **Expire date** and set a date after which the group will be removed from the user's keyring.

All groups assigned to a user will display in a grid sorted alphabetically ascending by group. This grid can be filtered by Shepherd group or description. Groups can also be sorted by clicking any of the column headers. To modify a group, click the **Edit** icon. To delete a group, click the **Delete** icon.

Configuration

Review the `core.security.rules.shpcfg` Shepherd setting to ensure it is configured correctly.

Delete users

Updated: March 25, 2022

A user access record can be deleted from the search results grid or from the user detail page.

To delete a user access record, click on either the delete icon in the user search results grid or the **Delete user** button on the user detail page. Multiple records can be deleted by selecting all the required rows and clicking the delete button on the card. Note that this will delete only the user access record. For information about deleting data in other Degree Works database tables, see [Run RAD40 - Student Data Delete Processor](#).

Configuration

Access to delete user access records is granted if the user has the **CTLUSRDL** Shepherd key.

Groups

Updated: March 25, 2022

In the Group tab, Shepherd group records can be created or modified.

Groups are collections of Shepherd keys and can be assigned to users to grant them access to that collection of Degree Works services.

Configuration

Access to Groups is granted if the user has the **CTLGROUP** Shepherd key.

Search for groups

Updated: March 25, 2022

Existing Shepherd groups can be searched for by group or description.

To search for a group, type a keyword string and press Enter. A paginated list of matching records will be returned. The list is initially sorted alphabetically ascending by the group, but the grid can be sorted by any column by clicking on the column header. An arrow will appear next to the column header to indicate that it is being used for the sort, and the direction of the arrow indicates whether it is ascending or descending.

Existing Shepherd group records may also be searched by entering a few characters allowing a suggestive set of records to display below the search bar. Clicking a particular group will populate the respective details into the search results grid.

The pagination details at the bottom of the search results grid allow the user to page to additional matching records, control how many records per page should display and how many total matching records were found.

Create and edit groups

Updated: March 25, 2022

A new group can be created by clicking the **Add new group** button on the Group search page.

A group can be edited by clicking on the edit icon in the search results grid. A card with all the details for this group will load. After modifying the details for the group, click the **Save** button to keep your changes.

Configuration

Access to the following is granted if the user has the specified Shepherd key:

- Add user access records - CTLGRPAD
- Edit user access records - CTLGRPMD

Group Details

Updated: March 25, 2022

Information about the group is detailed in the Group Details section.

When adding a group, a unique **Group** name must be provided. It can be up to 8 alphanumeric characters in length. When a group has been saved, the Group name cannot be changed.

A **Description** of the group's function can be entered. It can be up to 30 alphanumeric characters in length.

If you want to set an expiration date for the group, specify it in the **Expire date** field.

Use the **Timeout increment** to specify the time after which the session of a user with this group should automatically end, if the user is inactive. When the user first gets authenticated, the timeout

is set to the current time plus the timeout increment. This value is the time when the session will end if the user is inactive. Inactive is defined as not requesting a Degree Works service within a specific time frame. If the user stays active, then the timeout value will be updated. The timeout value is updated by taking the current time and adding on the timeout increment. The **Timeout maximum** specifies the maximum time for which a user session can be inactive before ending. The timeout value cannot exceed the initial authentication time plus the timeout maximum. As long as the user stays active, the session will not end within the maximum time allotted. If the timeout fields are not filled in, then the timeouts from the user's groups or from the `core.security.passport.timeout*` Shepherd settings. If you want to keep the user logged in irrespective of the inactive time, select the **Never timeout** check box.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

```
core.security.passport.timeoutIncrementDefault
core.security.passport.timeoutMaximumDefault
core.security.passport.timeoutPrecedence
core.security.passport.timeoutPreference
```

Group Keys

Updated: March 25, 2022

A key can be added to a group.

To add a key to the group, click the “Add key” button. Select the key that you want to add to the group. From the **Operator** dropdown, choose whether you want to grant or deny access to this key for this group. If you want to set a time limit for this group, click the calendar icon for the **Expire date** and set a date after which the key will be removed from the group.

All keys assigned to a group will display in a grid initially sorted alphabetically ascending by the key. This grid can be filtered by Shepherd key or description. To modify a key, click on the **Edit** icon. To delete a key, click on the **Delete** icon.

Groups are stored in the `shp-group-mst` record. There is a limit of 50 keys that can be added to a `shp_group_mst` record, but you can assign more than 50 keys to a group by sequencing the group name. The format of the sequence is `<group>-nn`. After adding the 50th key to the group, save your changes and then create a new group that has the same name plus a sequence number – that is, **SRNREG-2**, **SRNREG-3**, etc. Note that a group must have 50 keys assigned to it before a new sequence record is added. If one of the group sequence records has less than 50, the software will interpret that as the last sequence record and keys in the following sequence records will not be assigned to the user. You do not need to assign each group sequence record to a user. For example, assigning just SRNREG to a user assigns all the keys on SRNREG, SRNREG-2, SRNREG-3, etc.

Delete groups

Updated: March 25, 2022

A group can be deleted from the group search results grid or from the group detail page.

To delete a group, click on either the delete icon in the group search results grid or the **Delete group** button on the group detail page.

Configuration

Access to delete user access records is granted if the user has the CTLGRPDL Shepherd key.

Configuration

Updated: March 25, 2022

In the Configuration menu, Shepherd settings and UCX tables can be managed.

Shepherd settings are used to configure Degree Works application behavior and UCX tables contain validation data.

Configuration

Access to Configuration is granted if the user has the CTLCONF Shepherd key.

Search for configurations

Updated: March 25, 2022

Configurations can be searched for by setting key, or UCX table, or description.

To search for a configuration, type a keyword string and press Enter. A paginated list of matching records will be returned. The list is initially sorted alphabetically ascending by the configuration, but the grid can be sorted by any column by clicking on the column header. An arrow will appear next to the column header to indicate that it is being used for the sort, and the direction of the arrow indicates whether it is ascending or descending.

Configurations may also be searched by entering a few characters allowing a suggestive set of records to display below the search bar. Clicking a particular configuration will populate the respective details into the search results grid.

The pagination details at the bottom of the search results grid allow the user to page to additional matching records, control how many records per page should display and how many total matching records were found.

To view or edit a configuration, click the view icon in the search results row.

Shepherd settings

Updated: March 25, 2022

When viewing or editing a setting, a card will display the setting key and description along with a grid of all the defined specifications for that setting.

To edit a value of a setting, click on the edit icon. An editable value field will load. Click the **Save** button to keep your changes or **Cancel** to return to the setting card.

Configuration

Access to edit settings is granted if the user has the CTLSETMD Shepherd key.

Add specifications

Updated: March 25, 2022

While most of the settings are specific to a particular application, there are many that generally apply to all applications (for example, core settings).

It may be desirable, in certain situations, to specify a different value for these general settings to different applications. The specification field allows you to create or modify configurations based on the application. Each application has a unique spec value. The application will first use the setting that contains that specification. If it is not found, then the specification with the default value will be used.

To add a specification for a setting, click the **Add specification** button, select the desired specification and specify the correct value. Click the **Save** button to keep your changes.

Configuration

Access to add additional specifications for settings is granted if the user has the CTLSETAD Shepherd key.

Delete specifications

Updated: March 25, 2022

Specifications can be deleted from the setting card.

To delete a specification, click on the delete icon next to the specification on the setting card. Note that default specifications cannot be deleted.

Configuration

Access to delete specifications for settings is granted if the user has the CTLSETDL Shepherd key.

Add setting

Updated: March 25, 2022

The need to add a new setting should be rare. Ellucian delivers new settings through software update packages. But there are some scenarios where you may need to add a new setting, such as localizing the next steps displayed in Transfer Equivalency Self-Service.

To add a new setting, click the **Add a Shepherd setting** button. Enter the key and value then click the **Save** button to keep your changes. The setting will be added with a default specification. You can define a description for this setting by adding a localized property to `ShpSettingsMessages.properties` on the Properties tab in Controller. This property should be in the format of `"shpSetting.description." + SHP setting key`. For example, if your new setting is `treq.selfService.nextSteps.10`, your description property should be `shpSetting.description.treq.selfService.nextSteps.10`.

Configuration

Access to add a new setting is granted if the user has the CTLSETAD Shepherd key.

UCX tables

Updated: March 25, 2022

When viewing or editing a UCX table, a card with a paginated list of all the entries in the table will display.

The list is initially sorted alphabetically ascending by the key, but the grid can be sorted by any column by clicking on the column header. An arrow will appear next to the column header to indicate that it is being used for the sort, and the direction of the arrow indicates whether it is ascending or descending.

The pagination details at the bottom of the grid allow the user to page to additional matching records, control how many records per page should display and how many total matching records were found.

Create and edit UCX entries

Updated: March 25, 2022

A new UCX entry can be created by clicking the Add record button on the UCX table page.

An existing UCX entry can be edited by clicking on the edit icon in the grid. The values of an existing UCX entry can be copied to a new key by clicking on the copy icon. A card with all the details for this record will load. After modifying the details for the UCX entry, click the **Save** button to keep your changes.

Configuration

- Access to add UCX entries is granted if the user has the CTLUCXAD Shepherd key.
- Access to edit UCX entries is granted if the user has the CTLUCXMD Shepherd key.

Delete UCX entries

Updated: March 25, 2022

A UCX entry can be deleted from the grid.

To delete a UCX entry, click on the delete icon in the grid. Note that an entire UCX table cannot be deleted, just individual entries.

Configuration

Access to delete specifications for settings is granted if the user has the **CTLUCXDL** Shepherd key.

Import and export UCX entries

Updated: March 25, 2022

UCX tables can be modified in bulk using the Import and Export functionality. UCX entries can be transferred to a file which can be modified using a plain text editor and then loaded back into table.

Click the **Export** button to export all records in a UCX table out to a file on the user's computer. The user will be prompted for a location to where the file should be saved.

Click the **Import** button to import a file of records on the user's computer into a UCX table. All the records in the UCX table will be overwritten on an Import, so care should be taken when using this functionality. Controller will validate the file for the correct UCX table, duplicate entries, tab characters, and so on and will prevent the file from being imported if errors are found. These must be corrected before the file can be imported successfully.

Configuration

Access to import and export UCX tables is granted if the user has the CTLUCXBK Shepherd key.

Properties

Updated: March 25, 2022

In the Properties menu, the labels, text, and messages in Degree Works applications can be localized. You can also create a custom experience for users in different countries.

Language properties are stored in the shp_resource_mst. A baseline version is delivered by Ellucian at installation or in an update, and is not modifiable. When a resource is localized, these changes are saved in a separate shp_resource_mst record with a specific locale defined, and the baseline version is always available for comparison and troubleshooting. A localized resource needs to be enabled before it can be used in Degree Works applications.

Configuration

Access to Properties is granted if the user has the CTLPROP Shepherd key.

Properties search

Updated: March 25, 2022

A paginated list of language properties will display initially, and it can be filtered by name or description. Properties are listed in alphabetical order by name and selecting a property will open it in the editor page.

Properties localization

Updated: March 25, 2022

Localized properties are maintained on the editor page.

You can create a localized version of a baseline language property, update a localized language property, enable a localized language property, or view property details.

The language property name and description will display above the editor.

To view information such as Localization Last Modified data, Localization Created data, or Baseline Version, click the history button in the upper right.

When a language property is initially loaded in the editor, the baseline version will load in the Baseline Resource editor, while the localized resource for the default locale will load in the Localized Resource editor, if it exists. You can select a different locale from the Locale dropdown under the Localized Resource editor to either load that properties file or create it.

If a localized version for the selected locale cannot be found when loading a language property in the editor, a message will display.

Locales

The list of Locales is populated from SHP080 and contains a Description in addition to the actual locale code. The locale code is the IETF language tag, and typically consists of language and country codes, or just a language code. Based on settings in your browser, the correct properties file will be loaded so that the application text is appropriate for your location.

Examples:

- jp – for the Japanese language translation
- es – for the Spanish language translation
- es_MX – for the Spanish language translation, specific to Mexico
- es_CL – for the Spanish language translation, specific to Chile

Note: A default localized resource can be saved without a locale. The values in this default localized resource will overwrite the values in the baseline resource.

Create a localized language property

To create a language property localizations, select and copy the items you want to localize from the baseline resource into the localized, and then make your modifications.

Warning! Ellucian strongly recommends copying only the properties you want to localize. Copying the entire baseline resource to the localized resource will likely cause problems with future upgrades.

Enable a localized language property

To globally enable localized language properties in Degree Works, the `localization.internationalization.enable` Shepherd setting should be set to true.

When this settings is true, localized language properties with `is_enabled = Y` will be used by Degree Works applications. This setting should always be enabled, but it can be useful when troubleshooting issues or processing an update to set it to false temporarily. A related setting, `localization.internationalization.cacheMilliseconds`, determines for how long resources will be cached. Wait at least this long after making modifications to language properties before refreshing your browser to see those changes reflected.

Additionally, a localized language property will not be used by Degree Works unless it has been enabled. To enable a localized language property, click on the localization toggle so it switches from disabled to enabled. Note that changing the localization toggle on triggers an auto save of the script.

Review the following Shepherd settings to ensure they are configured correctly:

- `localization.internationalization.cacheMilliseconds`
- `localization.internationalization.enable`

Save a localized language property

Changes to a localized language property can be saved by clicking on the Save button or switching the localization toggle. If the save is successful, a confirmation message will be displayed.

Delete a localized language property

To delete a localized language property, switch the localization toggle to disabled, select all the text in the Localized resource editor and delete it, then Save your changes.

Language property details

Language property details can be accessed by clicking the history icon in the upper right. This display-only information shows the Last modified date and who, Created date and who, and the Baseline version.

The Last modified date and who will be updated when changes are saved to the localized resource. These fields will not display unless a localized resource for the selected locale exists in the database.

The Created date and who reflect when the localized resource was initially created. These fields will not display unless a localized resource for the selected locale exists in the database.

The Baseline version is the Degree Works release in which that version of the baseline source was delivered.

Localize language properties in an application

Each of the entries in language properties is made up of two sections, the key and the text. For example, the following entry is for the label on the over the limit section of the Responsive Dashboard audit:

```
dash.worksheet.section.overTheLimit.title=Over The Limit
```

In the example, “dash.worksheet.section.overTheLimit.title” is the key and “Over The Limit” is the text part of the entry.

To change the message, modify the text part of the entry. The modified text will be displayed to users.

Note: Do not modify the key. This is a reference used by the application, and if modified, the message will not be displayed.

If you want to change the default text for all users, you would create a default localization. In the editor, select the Default locale and then add the properties you want to localize. If you would like text to be in the user's native language, you can create separate records specific to their locale by creating a localization record for the specific locale. The localized values will overwrite the baseline values for the user, based on how their default language is configured in their browser.

Baseline and localized resources are merged together following this hierarchy:

1. The application first loads the baseline.
2. The default (blank locale) localization is loaded.
3. The one code locale (for example, "en" without the _US) is loaded.
4. The most specific, two-code locale (for example, en_US) is loaded.

Note: You can add whichever locales you want to support in SHP080.

Scribe

Updated: March 25, 2022

Scribe is used to code and store degree and program requirements.

Requirements are stored in units called blocks. Using Scribe, you can create new blocks or edit existing requirement blocks. When Degree Works produces a student worksheet, its auditor matches the courses on a student's record against the requirements in scribe blocks that contain the student's degree requirements.

UCX Tables

Updated: March 25, 2022

Several UCX tables are used for block values and tags in Scribe.

These UCX tables should be reviewed to ensure they contain the necessary entries and the descriptions are clear and accurate. After updating one of these UCX tables, refresh the page in the browser to load the changes. It is not necessary to restart the application.

Review the following UCX tables to ensure that they are configured correctly for your Scribe environment.

- CFG020, DAP14, and DAP13
- AUD031, AUD033, SCR002, SCR003, SCR004, SCR007, STU016, STU023, STU024, STU035, STU307, STU316, STU323, STU324, STU350, STU560, and STU563

Manage Scribe Blocks

Updated: March 25, 2022

As a best practice, blocks are typically updated and tested in a non-production environment. After they have been verified, they can be transferred to a production environment in bulk.

To transfer blocks from one environment to another, use the DAP40 and DAP41 jobs. See the [Blocks transfer between two different environments](#) topic for more information.

To bulk insert blocks from files, use the dapblockinsert script. See the [Degree Works special scripts](#) topic for more information.

Search for a block

Updated: March 25, 2022

A user can search for existing blocks either by the Requirement ID or using selection criteria on the Block Search page.

To open a block by the Requirement ID, the user should enter the numeric value of the Requirement ID in the text box. The field will automatically left zero-fill. That is, if you type 1234 you will see 001234 appear. The RA prefix will default and is applicable to most blocks. To search for a block created in the student planner, select **RB** from the **Prefix** drop-down list. Clicking **Open block requirement ID** will open the Edit page in a new tab, and the block will display in the editor. If a Requirement ID that is not valid is provided, the user will be shown an error message.

A user can also open a block by first searching for blocks based on selection criteria. The more criteria provided, the smaller the number of matching blocks. When a **Block type** is selected, the **Block value** drop-down list will load with the values from the UCX validation table for that block type, unless the block type is Other, Requisite, or Student ID. For Other and Requisite block types, when the user starts typing in the field a drop-down of matching blocks will be displayed, and the user can select one of these values to search on. For a Student ID block type, a valid Student ID should be entered in the **Block value** field.

A start or stop catalog year can also be provided when searching.

Note: If a Requisite block value is selected, the period labels will be **Start term** and **Stop term** instead of **Start catalog year** and **Stop catalog year**.

The user can also include Secondary Tags in the search. The **Student ID** is a text field and the user should enter a valid student ID. For **School, Degree, College, Major 1, Major 2, Concentration, Minor, Liberal Learning, Specialization** and **Program**, the user should select from the drop-down of values from the corresponding UCX validation table.

The search may be limited to only those blocks with parsing errors by checking the **Only blocks with errors** check box. If the block has `parse_status = NO`, it will be included in the search results.

To perform the search by criteria, click **Search**. Clicking **Clear** will reset the selected criteria back to the initial values.

The number of matching Requisite blocks that will be returned and shown in the autocomplete drop-down list is limited by the `scribe.query.threshold.requisite` setting. This is delivered initially as 500, but can be adjusted as necessary. Increasing this value can impact performance of the drop-down list.

The Secondary Tags that display in the Block Selection Criteria section can be configured. To remove a secondary tag from the display, set the corresponding `scribe.secondaryTag.show*` setting to **false**.

Review the following Shepherd settings to ensure they are configured correctly for your Scribe environment:

```
scribe.query.threshold.requisite
scribe.secondaryTag.showCollege
scribe.secondaryTag.showConcentration
scribe.secondaryTag.showDegree
scribe.secondaryTag.showLiberalLearning
scribe.secondaryTag.showMajor1
scribe.secondaryTag.showMajor2
scribe.secondaryTag.showMinor
scribe.secondaryTag.showProgram
scribe.secondaryTag.showSchool
scribe.secondaryTag.showSpecialization
scribe.secondaryTag.showStudentId
scribe.upshift.codes
```

Search results

Updated: March 25, 2022

The results of a block search by criteria will be returned in a card on the search page.

The results are initially sorted by title with a secondary sort on start catalog year/term. If no matching blocks are found based on the selection criteria provided, the user will be shown an error message.

The **Title**, **Block type**, **Block value**, **Requirement ID**, and **Start/Stop catalog year** will always display in the results grid. If a search includes the Requisite block type as criteria, the results grid will display **Start/Stop term** instead of **Start/Stop catalog year**. The secondary tags that are displayed can be configured. To remove a secondary tag from the display, set the corresponding `scribe.secondaryTag.show*` setting to **false**. The results grid can be sorted in ascending or descending order by any column by clicking on the column header.

Selecting a block in the results grid will open a new tab with the block in the editor.

Multiple blocks can be opened at the same time.

A user can delete one or more blocks from the results grid by selecting the check box to the left of the block title and clicking **Delete**. The user will be prompted to confirm the delete before the block is deleted.

Create a New Block

Updated: March 25, 2022

A new requirement block can be created by clicking the **New Block** button. A new block template will be opened in the editor.

This new block template can be localized in the `scribe.template.newBlock` setting in Controller.

The user can define the block details by clicking the **Block details** link.

To add the new block to the database, click the **Save** button on the Edit page. A block details edit window will be displayed and the user can either confirm the values they have already defined or can enter the appropriate values. Clicking **Save** will save the block to the database and clicking **Cancel** will take the user back to the editor. Before a block can be saved to the database, it will be parsed and validated. If any parsing or validation errors are found, a message will be displayed in the editor and the block will not be saved to the database. These errors must be corrected before the block can be successfully saved. When a block is successfully saved, the **Access Details** fields will be updated and displayed in the **History** link.

Review the `scribe.template.newBlock` Shepherd setting to ensure it is configured correctly for your Scribe environment.

Open Local Block

Updated: March 25, 2022

A plain text file of requirements previously saved to the user's computer can be opened in Scribe for editing and can then be saved to the database.

Click the **Open Local Block** button, then navigate to the local file you want to open. Click **Open** to load the file in a new unsaved block editor window.

Edit a Block

Updated: March 24, 2023

Block requirements and details can be modified on the edit page.

The user can update an existing block, save as a new block, create a new block, export the block to a file on the user's computer, print the block, and parse the block to check for syntax errors.

The block's title will display with the requirement ID above the editor and in the browser tab. If the block is new and has not yet been saved to the database, the title will display as Untitled. If the block has unsaved edits, an asterisk appears in the block title after the requirement ID.

To edit requirement text, click the **Editor** tab. To edit or view block information such as block type and value click the **Block details** link. To view the parse status, click the **History** link.

In the Editor, syntax highlighting exists on comments (lines beginning with #), labels, remarks, and so on (text contained in quotes), and Scribe Reserved Words. A block may have an unlimited number of lines, and each line is of unlimited length. However, Labels, Display, and ProxyAdvice each have a maximum length of 200 characters.

The user can use a utility to find and replace text in a block. In the editor window, Ctrl-F opens the find toolbar or Ctrl-H opens the find/replace toolbar. You can then enter a string of text in the **Search for** field and press the **Enter** key or click the down or up arrow to find the first instance of that string. The string search is case-insensitive. All matching instances of that string will be returned. To replace a string of text, you can enter the string you want to find in the **Search for** field and the string you want to replace it with in the **Replace with** field. Clicking **Replace** finds the first instance of that string, and then clicking **Replace** again makes the text substitution. Alternatively, you can select **Replace All** from the drop-down list and all instances of the find string will be replaced with the replace string.

The **Block details** link shows the Block details (the editable Primary and Secondary Tags for the block), while the **History** link shows the access details (display-only create, modify, and parse information).

The following shortcuts are available in the Scribe editor. Some are standard editor shortcuts and some are specific to the third-party editor tool, ACE, used by Scribe.

Shortcut	Action
Ctrl-/	Toggle comment; adds/removes # from front of selected lines, or on the current line if no lines are selected.
Ctrl-A	Select all text in the editor.
Ctrl-C	Copy to clipboard.
Ctrl-D	Delete the current line, or the selected text.

Shortcut	Action
Ctrl-Down-arrow	Scroll line down.
Ctrl-End	Go to the last line.
Ctrl-Home	Go to the first line.
Ctrl-L	Go to line number.
Ctrl-P	Print
Ctrl-Shift-U	Change to lower case – either the current word or the selected text.
Ctrl-U	Change to upper case – either the current word or the selected text.
Ctrl-Up-arrow	Scroll line up.
Ctrl-V	Paste from clipboard.
Ctrl-X	Cut selected text.
Ctrl-Z	Undo last change.
Triple-click	Triple-click on a line to highlight the entire line allowing you to copy the text (using Ctrl-C or right-click plus Copy)
F3	Open block details window.
F8	Parse on demand without saving.
F9	Maximize browser window. Press F9 again to reset.
F11	Toggle removal of the browser controls for absolute maximization of the page.
Windows-right-arrow	Puts the current window on the right half of the screen. Use in conjunction with Windows-left-arrow to get two windows lined up next to each other.
Windows-left-arrow	Puts the current window on the left half of the screen. Use in conjunction with Windows-right-arrow to get two windows lined up next to each other.

Additional shortcuts for the ACE editor are available at <https://github.com/ajaxorg/ace/wiki/Default-Keyboard-Shortcuts>; however, not all of these ACE shortcuts necessarily work in Scribe and are not supported by Ellucian.

Parse

Updated: March 25, 2022

A block can be checked for parse errors before saving by clicking the **Parse** button.

If no errors are found, a success message will display. If errors are found, an error message will display and the errors will be highlighted in the editor. An arrow will display at the beginning of

each line that has a parse error. Hovering over this arrow will display a detailed error message. Additionally, the specific text in the line with the error will be underlined. All parse errors must be corrected before a block can be successfully saved.

Review all flags in UCX-CFG020 DAP13 to ensure they are configured correctly for your Scribe environment.

Block Details

Updated: March 25, 2022

Block details can be accessed from the Block details link on the edit page.

For an existing block, clicking **Block details** will display only the items that have been defined for the block. Clicking **Edit Details** opens a window with all fields. When editing block details, the **Title**, **Block type**, **Block value**, and **Start/Stop catalog year** are required fields. When a **Block type** is selected, the **Block value** drop-down list will load with the values from the UCX validation table for that block type, unless the block type is Other, Requisite, or Student ID. For Other and Requisite Block Types, when the user starts typing in the field, a drop-down list of matching blocks will be displayed and the user can select one of these values or can enter a new value. The `scribe.upshift.codes` setting controls if what the user has typed in the **Block value** field should be upshifted. For a Student ID Block Type, the **Block value** is a text field and the user should enter a valid student ID.

If a Requisite block value is selected, the period labels will be **Start** and **Stop term** instead of **catalog year**.

The user can also apply Secondary Tags to the block. The **Student ID** is a text field, and the user should enter a valid student ID. For **School**, **Degree**, **College**, **Major 1**, **Major 2**, **Concentration**, **Minor**, **Liberal Learning**, **Specialization** and **Program**, the user should select from the drop-down of values from the corresponding UCX validation table. The Secondary Tags that display can be configured. To remove a Secondary Tag from the display, set the corresponding `scribe.secondaryTag.show*` setting to false.

Save a Block

Updated: March 25, 2022

After creating or editing a block, the user can either update the existing block, add a new block to the database, or save the block to a file on their computer by clicking the **Save** drop-down button.

When updating an existing block or saving a new block, before the block will be written to the database, it will be parsed for syntax and rule structure and checked to ensure it does not overlap catalog years or terms with other blocks. If any errors are encountered, these will be shown at the top of the page or next to the line of the scribe information containing the error. The errors need to be resolved before the block can be saved to the database.

If a user does not have access to modify a block type, the **Save** and **Block details** edit buttons will be inactive. The user can still view the block, but will not be able to parse it or make or save any changes to it.

Save

Updated: March 25, 2022

To save changes made to an existing block, users can click the **Save** button.

The user may be prompted to confirm that they want to update the existing block if the `scribe.save.enable.confirmation` setting is 'true'. When this setting is 'false', no confirmation message will be displayed. If enabled, clicking **Yes** will save the block and clicking **Cancel** will take the user back to the editor.

When saving a new block, the block details edit window displays after clicking **Save**. The user should review and fill in the block details and then click **Save**. When saving an existing block, the block details edit does not display. In both instances, a save confirmation appears when the block is saved to the database.

Review the `scribe.save.enable.confirmation` Shepherd setting to ensure it is configured correctly for your Scribe environment.

Save As

Updated: March 25, 2022

To save changes made to an existing block as a new block rather than updating the existing block, users can click the **Save As** button. The block details edit window will be displayed and the user should review and fill in the block details and then click **Save**.

Save Local File

Updated: March 25, 2022

Requirement text can be saved as a file on the user's computer for archival purposes or to edit and add to the database at a later time.

Click Save Local File – depending on your browser configurations, you will either be prompted for a location where the file is to be saved or the file will be saved in the default location configured in the browser. The requirement text for the block is saved in the form of a text (.txt) file. The default filename is a combination of the block type, value, requirement ID, and title. The filename will be 'untitled' if the block details have not yet been assigned.

Note: On the Safari browser, clicking Save Local File will open the block text on a new window. You will have to press the Command key + S to save the block as a local file. In addition, the default filename for the block will not be generated on Safari browser instances.

Delete a Block

Updated: March 25, 2022

A requirement block can be deleted by an authorized user in several places.

To delete a requirement block, either click **Delete** on the More menu when on the edit page, or select the block(s) you want to delete and click the **Delete** button on the search page. A delete confirmation message will display. Users can click **OK** to delete the block or **Cancel** to return to the page.

Block printing

Updated: March 25, 2022

A requirement block can be printed from the Scribe editor.

To print a block's requirement text, users can select **Print** from the More menu when on the edit page. The browser's print window will open, allowing the user to select the desired printer, preview the output, change the headers and footers, and so on. The print output will show the block's Requirement ID and Title, and then all of the requirement text. Long lines of text will wrap in the printed output.

Controller debugging

Updated: March 25, 2022

Controller allows users to enable debugging to troubleshoot issues in the Controller application for their session.

Users with access to this functionality will be able to enable and disable debug from the Debug page, which is accessed from the User Profile image in the upper right of the header. Access to the application server is not needed to enable debugging. However, access to the server is required to access the generated log files. Debug files can be found in the logs directory of the application server.

Enable debugging

Updated: March 25, 2022

Debugging can be enabled from the Debug page.

On the Debug page, toggle the Enable Debug switch to turn debugging on for the application. A random Debug Tag will be generated, and the user can copy this to their clipboard by clicking on the page icon. This Debug Tag will be appended to each line of debug in the log file so that the

user can easily identify which lines are from their session, as opposed to other users using the application at the same time.

After enabling debug, the user can then resume using Controller and execute the steps to duplicate the issue they are experiencing. Debug should be turned off by going back to the Debug page and toggling the Enable Debug switch when the issue has been duplicated. Debug for this user's session will also be turned off when they log out of the application.

Debug file access

Updated: March 25, 2022

Debug will appear in the Controller log file on the application server. This is typically defined by the \$LOGGING_FILE environment variable.

Searching for the user's debug tag will isolate the lines of debug specific to their actions. For example:

```
2017-10-24 16:22:23,144 68b4e19d-3fb4-4975-9815-83eea01266ff DEBUG [
41-exec-17] .h.d.s.s.StatelessPreAuthenticatedFilter : Checking secu
re context token: null
```

For more information about collecting the log files generated for a user session's debug tag, [packdebug script](#).

Configuration

Access to Debug is granted if the user has the DEBUG Shepherd key.

Ellucian Experience Integration

Updated: February 27, 2023

Ellucian Experience users can view academic plans and degree progress information from Degree Works. You must set up both Ellucian Experience and Degree Works to support the integration.

The table below shows the Ellucian Experience cards that display information from Degree Works.

Card type	Description
Academic Plans	Displays active educational plans defined in Degree Works for this student, along with the overall status (On-Track or Off-Track). The student can expand each plan to view the status for each term. The user can also link into a view of the complete plan.
Degree Progress	Displays the GPA, requirements progress, and credits progress for all active degrees defined in Degree Works for this student. The user can also link into a view of the complete audit.

When a user accesses Ellucian Experience with one of the Degree Works cards on the dashboard, Ellucian Experience obtains the data from Degree Works to display in the cards. The data retrieval uses the following information, which you must enter in Ellucian Experience settings.

- The full URL of the Degree Works Responsive Dashboard application.
- The username and password for a Degree Works user who has permissions to access the Degree Works data displayed in the Degree Progress and Academic Plans cards. You must create this user in Degree Works and assign the required permissions.
- The RPT036 report format that should be used for configuring audit data shown in the Degree Progress card.

Whitelist Ellucian Experience IP addresses

Updated: February 14, 2023

Ellucian Experience calls the Responsive Dashboard application to retrieve data. Your firewall rules must allow for inbound requests from the Ellucian Experience IP addresses to the server where you deployed the Degree Works Responsive Dashboard.

You must whitelist the IP addresses for your region. See [Experience IP addresses](#).

Configure Ellucian Experience access to Degree Works

Updated: April 28, 2023

Ellucian Experience requires a user that is authorized to pull Degree Works information into the Experience cards.

Procedure

1. Configure Responsive Dashboard to allow basic authentication through SHP. In Controller, create a dashboard specification for the core.security.shp.authentication.enabled Shepherd setting and set it to True.
2. Use the Users function of Controller to set up a user who can access the Responsive Dashboard from Ellucian Experience.
 - a) Assign the user to the API user class.
 - b) Assign the user the following Shepherd keys:
 - ANYSTUID
 - NOREFER
 - RSAUDIT
 - RSPLAN
 - SDAUDREV
 - SDWORKS

Note: This user does not need LDAP or other credentials.

Set up Degree Works cards in Ellucian Experience

Updated: February 13, 2024

Set up either or both of the Ellucian Experience cards (Academic Plans and Degree Progress) that display information from Degree Works.

Procedure

1. Access Ellucian Experience as a user with the Card Management administrative permission.
2. Select **Ellucian Experience > Configuration**.
3. Click **Card Management**.
4. In the **Actions** column, click **Edit** next to the card type you want to configure (Academic Plans or Degree Progress).
5. In the Details step of the wizard, enter basic card settings.

- a) In the Card Information section, enter descriptive information.

Setting	Description
Card title	This title displays at the top of the card.
Card description	This description does not display on the card, but does factor in to search.
Card tags	Both search and filters use tags in displaying cards on the Discover page. You can enter multiple tags separated by commas.
Mini card icon	This icon represents card content when viewing in smaller view ports.

Card information helps Ellucian Experience users find cards of interest on the Discover page. Tags appear at the top of the page. Users can click a tag to filter the display to cards only with that tag. When a user searches for cards, the search includes the text that you enter for the title, description, and tags.

- b) In the Card Placement section, choose a category for the card.
The category groups cards together for ease of access. The category link appears on the Experience main menu and in card management.
Enable **Show as default card on home page** if you want this card to be the default for new users. It will automatically appear on the dashboard of new users.
- c) Optional: In the External Link section, you can add a link to Degree Works that will display in the card action menu.

Setting	Instruction
External link label	Enter the text for the link as it will appear in the card.
External link URL	Enter the link target URL.

- d) Click **Next**.
6. In the Configuration step of the wizard, enter the Degree Works parameters for the selected card. There are two types of parameters for each card.
- Parameters that allow you to customize the labels on the card.
The following labels can be localized on the Degree Works Experience cards by entering a new parameter value.
 - Degree Progress card: Getting Started Header, Credits, Degree, GPA, and Requirements

- Academic Plans card: Tracking status (NOT EVALUATED, ON TRACK, OFF TRACK), Modified, Getting Started Header
- Parameters that allow you to connect with the Degree Works Responsive Dashboard. The following parameters apply to the entire extension and may already have a value if you have previously configured another card.

Setting	Instruction
Responsive Dashboard URL	Enter the URL for the Degree Works Responsive Dashboard application located on the Degree Works server. The URL has the form <code>https://{hostname}:{port}/{path}</code>
Username and Password	Enter the username and password of the Degree Works user that you created who can access Degree Works APIs from Ellucian Experience. For more information, see the Configure Ellucian Experience access to Degree Works topic.
Worksheet - RPT036	The report format from RPT036 that should be used to configure the display of the audit information in the Degree Progress card. If blank or not valid, WEB31 will be used as a default. This field is case sensitive.

- Click **Next**.
- In the Roles step of the wizard, do either of the following to specify which users will see this card.
 - Click **Select all users** to assign this card to all Ellucian Experience users.
 - Select roles to assign this card only to users with those roles. Your institution defines the roles available in Ellucian Experience setup.
- Select **Enable card** to specify that the card is available to Ellucian Experience users with the roles you selected to access this card.
- Click **Finish**.

Related tasks

- [Create a shared secret and API token for the extension](#)
- [Set up an Experience Card](#)
- [Add the roles in Ellucian Experience Setup](#)

Troubleshooting the integration between Ellucian Experience and Degree Works

Updated: January 12, 2024

Troubleshooting the integration between Ellucian Experience and Degree Works addresses common issues that can occur when performing integration functions.

Problem	Possible solution
Degree Progress or Academic Plans cards showing errors or not showing data that we know exists for the test student in Degree Works.	<p>Make sure you are logged in as a valid student user, as the Degree Works integration in Experience currently only works for that persona. View your ID in Experience on the Profile page by clicking in the top right corner. This is the ID that the Degree Works API looks up.</p> <hr/> <p>Check the Degree Works API user credentials and access keys.</p> <ul style="list-style-type: none"> • Make sure the user has been assigned the API user class and the required Shepherd keys. For more information, see Configure Ellucian Experience access to Degree Works. • Make sure the user and password you entered in Controller are the same as what you entered in the Experience card configuration. The system uses the user's Access ID from Controller. <hr/> <p>Check the Responsive Dashboard URL in the Experience card configuration. Make sure it matches the correct Responsive Dashboard URL with which you are able to access Responsive Dashboard. There should be no additional path, only your standard Responsive Dashboard URL. The URL has the form <code>https://hostname:port/path</code>.</p> <hr/> <p>Check the <code>core.security.shp.authentication.enabled</code> Shepherd setting to make sure it is set to true for the Responsive Dashboard specification. The applicable specification depends on your version of Degree Works.</p> <ul style="list-style-type: none"> • 5.1.0 or earlier - newdash • 5.1.1 or later - dashboard <hr/> <p>Check for referer issues.</p> <ul style="list-style-type: none"> • If you are on Degree Works release 5.0.7 or earlier, you should add the Experience dashboard URL values for your region to the <code>core.security.refererAllowableUrls</code> Shepherd setting. In the United States, these URLs are <code>experience-test.elluciancloud.com</code> and <code>experience.elluciancloud.com</code>. If you are on Degree Works release 5.1.0 or later, you do not need to add those URLs. • If you have the ability to view the Responsive Dashboard log, check it for CSRF Attack errors. These errors indicate that an attempt is being made to reach the Responsive Dashboard from a server not listed under <code>core.security.refererAllowableUrls</code>.

Problem	Possible solution
	<p>Test the API endpoints directly through curl or Postman. This action verifies the app is running, the URL is correct, the credentials are correct, and the data in question is available. Be sure to include the credentials with the request. If you are not sure of the appropriate school (that is, academic level) and degree to test, you can access the student in Degree Works, view the Student Data Report, and then look under the Goal-Dtl section.</p> <p>Endpoint patterns:</p> <ul style="list-style-type: none"> • <code><resdash-url>/api/students?studentId=XXX</code> • Degree Progress card endpoint - <code><resdash-url>/api/audit?studentId=XXX&school=XX&degree=XX</code> • Academic Plans card endpoint - <code><resdash-url>/api/plans?studentId=XXX</code>
Plans card does not show any plans.	<p>The student does not have any plans that are both active and locked (approved), depending on the <code>studentPlanner.planner.planApprovalMethod</code> Shepherd setting. Ensure your student has an active and locked/approved plan with current requirements.</p>
Plans card shows not evaluated.	<p>Either there has been no progress after the plan was created, the plan has no trackable requirements, or the requirements are for a future term.</p>

Responsive Dashboard Administration

Updated: March 25, 2022

The Responsive Dashboard is the primary user interface for students, advisors, and other users to process and view individual degree audits, run what-if scenarios, add and view notes and petitions, manage exceptions processing, and create and track educational plans.

The Responsive Dashboard contains both the front-end (browser-based) application and the back-end API that is used by the client code. It is deployed as a jar on any supported Unix server and is independent of the classic software. It includes an embedded container – a separate Java container like Tomcat is not required.

Initial Responsive Dashboard setup and configuration

To use Responsive Dashboard, you must set up and configure the tool.

Authentication

Updated: March 25, 2022

The Responsive Dashboard user interface supports SHP, CAS, and SAML authentication methods, in addition to external access managers such as Oracle Access Manager.

For more information, please see the [Authentication security](#) topic.

Responsive Dashboard user access

Updated: March 25, 2022

Access to the features of the Responsive Dashboard is managed by assigning Shepherd keys and groups to users.

This can either be done globally based on a user's role within `core.security.rules.shpcfg`, or individually on a user's `shp_user_mst`. For example, all students may be assigned the SRNSTU and SEPSTUVW Shepherd groups, which will grant limited access to Worksheets and Plans functionality. An advisor, however, might be assigned SRNADV and SEPADV, granting broader access to Worksheets, editing capabilities in Plans, and the ability to process exceptions. For more information, see the [Access control \(authorization\)](#) topic.

Shepherd settings

Updated: March 24, 2023

Most configurations to manage the behavior of Responsive Dashboard can be maintained using the Configuration tab of Controller.

Review the following Shepherd settings to ensure they are configured correctly for your Responsive Dashboard environment.

- `classicConnector.serverNameOrIp`
- `core.audit.*`
- `core.credit.format`
- `core.gpa.format`
- `core.requisite.validate.enable`
- `core.security.cas.callbackUrl spec=dashboard` (if you use CAS authentication)
- `core.security.referrerAllowableUrls`
- `core.whatIf.*`
- `dash.*`
- `studentPlanner.*`

Responsive Dashboard deployment

Updated: March 25, 2022

The Responsive Dashboard is designed to deploy equally well in on-premise and cloud hosted scenarios on all supported platforms.

It provides a mobile-ready, responsive user interface communicating with the servers using lightweight, stateless, restful API over HTTP SSL.

The deployment artifacts are two JAR files which are executable using java and include an embedded container: Dashboard and APIServices. Dashboard is required while APIServices is only needed if using the Transfer Finder functionality or integrating with external applications to provide degree audit data. This section provides information on deploying Dashboard and APIServices.

Prerequisites

Updated: March 25, 2022

Java is a prerequisite, and SSL is a best practice and recommended for Transfer Equivalency Admin.

Java

Java version 11 or above is the only third-party dependency required to run this application. Oracle Java and OpenJDK are supported.

SSL

SSL is in general a best practice and is recommended for Responsive Dashboard because the security implementation uses stateless tokens. Using signed certificates is recommended for all deployment scenarios, even test or development environments. Self-signed certificates, even for internal test deployments, can be problematic because of browser requirements and warnings. The SSL certificate needs to be configured in a keystore and that keystore will need to be accessible in the deployment environment. There are various options for how to do this, for example using Java's keytool command.

Responsive Dashboard environment settings

Updated: September 29, 2023

Several environment variables are required to be configured before running the application.

Depending on the desired deployment platform there are many options for how these environment variables can be configured. The only requirement is that they are available in the environment where the product's JAR file will be executed. For example, it may be desirable to configure them in a shell script, in a specific user's profile, a startup instruction like init.d, or in a continuous deployment tool like Jenkins.

The minimum required variables are documented here. But many optional variables are provided in the Spring Boot platform, for example tuning the embedded container. Please refer to Spring documentation for those: <http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>.

Environment Variables are the recommended method to configure these properties. But there are various other ways these can be configured. Please refer to this documentation for more information: <http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html>.

Required environment variables for Responsive Dashboard

- **SPRING_DATASOURCE_URL**: A JDBC connection string for the Degree Works database. For example: `jdbc:oracle:thin:@{SERVER}:{PORT}/{SERVICE_NAME}` where `{SERVER}` is the address, `{PORT}` is the port of the listener, and `{SERVICE_NAME}` is the service name of the database. This is the same database and schema used by the classic Degree Works software.
- **SPRING_DATASOURCE_USERNAME**: The Degree Works database username.
- **SPRING_DATASOURCE_PASSWORD**: The Degree Works database password. This can be encoded. To get the encoded value, use the `showdbpasswords -password` command in the classic environment. It will display the encoded value to place here (e.g. `ENC(skdfjldjs)`).
- **SERVER_PORT**: The desired port for the Responsive Dashboard. This must be different from the ports used by other applications on this server.

- **SERVER_SERVLET_CONTEXT_PATH:** An optional context path. If this is blank, the application will be deployed at the root URL like `https://myserver.edu:NNNN` (where NNNN is the **SERVER_PORT**). If a value is specified like `/my-context-path`, the application would be accessible at a URL like `https://myserver.edu:NNNN/my-context-path` (where NNNN is the **SERVER_PORT**).
- **SERVER_SSL_KEY_STORE:** The path to a keystore file you created to contain your SSL certificate.
- **SERVER_SSL_KEY_STORE_PASSWORD:** The password for your keystore file.
- **SERVER_SSL_KEYSTORETYPE:** The type of your keystore. JKS is the default.
- **SERVER_SSL_KEY_ALIAS:** The alias of the key you created.

Required environment variables for APIServices

- **DW_DATASOURCE_URL:** A JDBC connection string for the Degree Works database. For example `jdbc:oracle:thin:@{SERVER}:{PORT}/{SERVICE_NAME}` where {SERVER} is the address, {PORT} is the port of the listener, and {SERVICE_NAME} is the service name of the database.
- **DW_DATASOURCE_USERNAME:** The Degree Works database username.
- **DW_DATASOURCE_PASSWORD:** The Degree Works database password.
- **SERVER_PORT:** The desired port for APIServices. This must be different from the ports used by other applications on this server.
- **SERVER_SERVLET_CONTEXT_PATH:** An optional context path. If this is blank, the application will be deployed at the root URL like `https://myserver.edu:NNNN` (where NNNN is the **SERVER_PORT**). If a value is specified like `/my-context-path`, the application would be accessible at a URL like `https://myserver.edu:NNNN/my-context-path` (where NNNN is the **SERVER_PORT**).
- **SERVER_SSL_KEY_STORE:** The path to a keystore file you created to contain your SSL certificate.
- **SERVER_SSL_KEY_STORE_PASSWORD:** The password for your keystore file.
- **SERVER_SSL_KEYSTORETYPE:** The type of your keystore. JKS is the default.
- **SERVER_SSL_KEY_ALIAS:** The alias of the key you created.

Optional environment variables

- **JAVA_OPTIONS:** The memory allocation for the application can be defined using this variable, and is recommended. The value defines the initial heap size and max heap size and is in the following format: `-Xms1536m -Xmx1536m`. The heap size values should be adjusted as appropriate for your server.
- **SERVER_MAX_HTTP_HEADER_SIZE:** This setting allows you to set the maximum size of the HTTP message header. By default, this is 8KB, and for some users with a large number of Shepherd access keys, this can be too small. It's recommended to use this variable and set it to at least 12KB. For example:

```
export SERVER_MAX_HTTP_HEADER_SIZE=12288
```

- **SERVER_TOMCAT_REDIRECT_CONTEXT_ROOT:** Embedded Tomcat for Spring Boot has a default behavior to redirect a request without a trailing slash to one with a trailing slash. However, that redirect happens without evaluating forward headers like X-Forwarded-Proto. So, when SSL offloading is in place, the redirect happens to http instead of https. This can be an issue especially for load balanced environments. To disable the default behavior, set this variable to false.

Warning! You should do this only if the default behavior is not working. If you are offloading SSL at a proxy or load balancer, make sure to follow the instructions in the [Load balancing and proxies](#) topic. Then, if you have problems accessing the application in the browser when you do not include a trailing slash at the end of the request, try setting this variable to false.

- **DEBUG:** A boolean true or false which will enable or disable debug logging. The log file is by default named dashboard.log. But, the location and name of that file can be customized by setting the LOGGING_PATH and LOGGING_FILE environment variables.
- **DEPLOY_LOCATION:** the location of the Dashboard.jar file. This location should be set then referenced in the java -jar command that starts the application.
- **LOGGING_FILE:** the name of the file including the path location. If this is specified, the LOG_PATH variable should not be used.
- **LOG_PATH:** the location of the log file. The default PATH location is the Java temporary java folder from the Java property java.io.tmpdir. This is usually /tmp. The file name will be dashboard.log for Responsive Dashboard and api-services.log for APIServices.
- **LOG_DEFAULT_THRESHOLD:** If set to DEBUG, the application will output more verbose logging. Debug output will be sent to the log during the startup process and for all requests thereafter. Warning: this will result in very large log files. If not set, the debug threshold is INFO.
- **SERVER_FORWARD_HEADERS_STRATEGY:** If you are running applications behind proxies or load balancers, you may need to set this environment variable to framework.
- **DW_ENABLE_MONITOR** - This environment variable can be set to enable the /health, /metrics, /prometheus, and /status endpoints for health and monitoring metrics. By default, this monitoring functionality is not enabled. Adding this variable to an application's startup script and setting it to true would enable these endpoints for the application. See individual API documentation in the API Catalog for additional information.
- **DW_RABBITMQ_SSLALGORITHM:** Specify the SSL algorithm to use when connecting to the RabbitMQ broker using SSL. If not specified, it uses the current RabbitMQ driver's default (at least TLSv1.2). This is only considered if the Shepherd setting core.amqp.useSsl is set to true.
- **DW_SECURITY_CLEANUP_ENABLE:** Enables or disables the periodic cleanup of old TOKEN_REVOCATION_MST and SHP_SESSION entries in the database. The default is enabled (1). To disable, set this to 0 (zero). This should not be set without consulting the Ellucian Action Line.
- **DW_SECURITY_CLEANUP_REVOKEDTOKENS_SCHEDULE:** the frequency at which entries in the TOKEN_REVOCATION_MST database table are scanned for expired entries that can be

removed. The value is in an extended UNIX cron format. See [https://docs.spring.io/spring-framework/docs/current/javadoc-api/org.springframework.scheduling.support/CronExpression.html#parse\(java.lang.String\)](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org.springframework.scheduling.support/CronExpression.html#parse(java.lang.String)) for details. The default is every 4 hours. This should not be set without consulting the Ellucian Action Line.

- **DW_SECURITY_CLEANUP_EXPIREDSSESSIONS_SCHEDULE**: the frequency at which entries in the SHP_SESSION database table are scanned for expired entries that can be removed. The value is in an extended UNIX cron format. See [https://docs.spring.io/spring-framework/docs/current/javadoc-api/org.springframework.scheduling.support/CronExpression.html#parse\(java.lang.String\)](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org.springframework.scheduling.support/CronExpression.html#parse(java.lang.String)) for details. The default is every 4 hours. This should not be set without consulting the Ellucian Action Line.

Database pool configuration

For production deployments of java applications, it is important to be able to configure the size of the database connection pool. In Tomcat this is done through the Resource definition in server.xml. For Responsive Dashboard and API Services, there are environment variables that can be used for this configuration.

For Responsive Dashboard, these environment variables start with DW_DATASOURCE_, while for API Services, these environment variables start with DW_DATAPOOL_. They end with the name of the datasource pooling attribute you want to control. These are the most common ones:

Attribute	Meaning
initialSize	The initial number of connections that are created when the pool is started.
maxActive	The maximum number of active connections that can be allocated from this pool at the same time.
maxIdle	The maximum number of connections that can remain idle in the pool, without extra ones being released.
minIdle	The minimum number of active connections that can remain idle in the pool, without extra ones being created.

To set the maximum number of active connections in the pool, for example, put the following line in your startup script:

```
export DW_DATAPOOL_MAX_ACTIVE=100
```

You can set any of the properties of a version 1.4 Apache Commons BasicDataSource object. These are documented at <http://commons.apache.org/proper/commons-dbcp/api-1.4/org/apache/commons/dbcp/BasicDataSource.html>.

The number of active connections and the current configured values for the limits can be monitored using any JMX (Java Management Extensions) client, such as JConsole, attached to the microservice JVM. The values will appear under the mbean net.hedtech.degreeworks/JmxBasicDataSource/datasource. The getState operation will return a formatted report of the current pool state.

SSL offloading

When SSL offloading is in use (SSL is terminated before reaching the application) with a load balancer, proxy server, etc. some additional configuration needs to be done to make sure Degree Works applications behave as expected. Please see the [Java application load balancing](#) topic for more information about this setup.

Deployment examples

Updated: March 25, 2022

After the prerequisites and configuration are in place, start the application using a "java -jar" command.

```
java $JAVA_OPTIONS -jar ResponsiveDashboard.jar
java $JAVA_OPTIONS -jar APIServices.jar
```

Example startup script for Responsive Dashboard

If it is desirable to employ a shell script to start the application, here is an example. Note that the Dashboard.jar file must be staged at the DEPLOY_LOCATION.

```
#!/bin/ksh
export SPRING_DATASOURCE_URL="jdbc:oracle:thin:@MyServer:1521/service_name"
export SPRING_DATASOURCE_USERNAME="my-dw-user"
export SPRING_DATASOURCE_PASSWORD="my-dw-password"
export SERVER_PORT="8491"
export SERVER_SERVLET_CONTEXT_PATH=""
export SERVER_SSL_KEY_STORE="/usr/local/my-keystore.jks"
export SERVER_SSL_KEY_STORE_PASSWORD="my-key-password"
export SERVER_SSL_KEYSTORETYPE="jks"
export SERVER_SSL_KEY_ALIAS="my-keystore-alias"
export DEPLOY_LOCATION="/path/to/jarfile"
export JAVA_OPTIONS="-Xms1536m -Xmx1536m";
export DEBUG="false"
# To enable full-time debug log output, remove the # from the line below
#export LOG_DEFAULT_THRESHOLD=DEBUG
java -jar $JAVA_OPTIONS DEPLOY_LOCATION/ResponsiveDashboard.jar > startup-log-file.log &
```

Example stop script for Responsive Dashboard

If using a start script like the example above, here is a corresponding example stop script:

```
#!/bin/ksh
kill -f ResponsiveDashboard
```

Example startup script for APIServices

If it is desirable to employ a shell script to start the application, here is an example. Note that the APIServices.jar file must be staged at the DEPLOY_LOCATION:

```
#!/bin/ksh
export DW_DATASOURCE_URL="jdbc:oracle:thin:@MyServer:1521/service_name"
export DW_DATASOURCE_USERNAME="my-dw-user"
export DW_DATASOURCE_PASSWORD="my-dw-password"
export SERVER_PORT="8541"
export SERVER_SERVLET_CONTEXT_PATH=""
export SERVER_SSL_KEY_STORE="/usr/local/my-keystore.jks"
export SERVER_SSL_KEY_STORE_PASSWORD="my-key-password"
export SERVER_SSL_KEYSTORETYPE="jks"
export SERVER_SSL_KEY_ALIAS="my-keystore-alias"
export DEPLOY_LOCATION="/path/to/jarfile"
export JAVA_OPTIONS="-Xms1536m -Xmx1536m";
export DEBUG="false"
java -jar $JAVA_OPTIONS DEPLOY_LOCATION/APIServices.jar > startup-log-file.log &
```

Example Stop Script for APIServices

If using a start script like the example above, here is a corresponding example stop script:

```
#!/bin/ksh
pkill -f APIServices
```

Requirements for the classic server

Updated: March 25, 2022

On the classic server you must have web07 running.

Run `webrestart` to start the web07 daemons. The number of web07 daemons that serve dashboard requests is configured in the `classic.daemons.web07.count` setting. In production you will need perhaps 20 or more web07s running; you will need to test to figure out how many your system can handle. View the `$ADMIN_HOME/logdebug/web.log` for issues web07 may encounter while either starting up or processing dashboard requests.

Integration with portals

Updated: September 29, 2023

You can integrate with portals in many ways, from ID to role pass-along when linking.

Student ID pass-along to Degree Works

When connecting from your school's portal to the Responsive Dashboard, you can pass in the ID of the student that is to appear in Degree Works. This may be the student user who has clicked the link to Degree Works from your portal, or it may be an advisor user viewing this student in your portal. The student ID needs to be passed in the URL used to connect from your portal to Degree Works using this format:

```
https://{server}/{deployment-path}?studentId=<someid>
```

Only users who have been given permission to access this student's records will be successful in seeing this student loaded when they connect to Degree Works. If the user is an advisor with a fixed set of advisees, after the advisees have been loaded, this specific student will appear as the selected student. If the user does not have the SDFIND, SDSTUMY, or SDSTUANY keys, they will not be able to switch to a new student within the Responsive Dashboard; switching students would then have to occur within your portal.

Link to the Worksheets tab in Responsive Dashboard:

```
https://{server}/{deployment-path}?studentId=<someid>
```

Link to the Exceptions tab in Responsive Dashboard:

```
https://{server}/{deployment-path}/exceptions?studentId=<someid>
```

Link to the Plans tab in Responsive Dashboard:

```
https://{server}/{deployment-path}/plans?studentId=<someid>
```

User role pass-along to Degree Works

An advisor or staff member who is also a student may want to access Degree Works as their student persona. To do this, the as-student parameter should be used to log the user in with a STU user class. Note that the access granted to the user will be that which is defined in `core.security.rules.shpcfg`, where `DGWUSERCLASS = STU`. However, if the user has also been assigned Shepherd keys or groups on their `shp_user_mst` through Controller, they will also be granted access to that functionality.

The as-student parameter needs to be passed in the URL used to connect from your portal to Degree Works using this format:

```
https://{server}/{deployment-path}?as-student=true
```

Specific worksheet linking

Linking directly to a student's worksheet is supported with a path parameter that is the key of the desired format from RPT036.

The format code parameter needs to be passed in the URL using this format:

```
https://{server}/{deployment-path}/worksheets/<RPT036 code>
```

The studentId and as-student query string arguments are supported as described above. For students with multiple degrees, a degree code may also be provided so the correct audit is returned.

Link to the Graduation Checklist (RPT036 key = WEB33):

```
https://{server}/{deployment-path}/worksheets/WEB33
```

Link to the Athletic and Academic Report (RPT036 key = WEB56) for a specific student:

```
https://{server}/{deployment-path}/worksheets/WEB56?studentId=<someid>
```

Link to the Student View (RPT036 key = WEB31) for a specific student's BA degree:

```
https://{server}/{deployment-path}/worksheets/WEB31?
studentId=<someid>&degree=BA
```

Active plan linking

It is possible to link directly to a student's active plan and not just the Plans tab for a student.

The active parameter needs to be passed in the URL using this format:

```
https://{server}/{deployment-path}/plans/active
```

The studentId and as-student query string arguments are supported as described above, in addition to the plan school and degree, if the student has more than one active plan.

Link to a student's active plan:

```
https://{server}/{deployment-path}/plans/active?studentId=<someid>
```

Link to a student's active plan for their BS degree:

```
https://{server}/{deployment-path}/plans/active?
studentId=<someid>&degree=BS
```

Link to a student's active plan for their graduate MFA degree:

```
https://{server}/{deployment-path}/plans/active?
studentId=<someid>&school=GR&degree=MFA
```

Linking with and without navigation

Linking to any URL will include all navigation options according to the user's permissions. For some use cases, it may be desirable to exclude navigation options. A common example of that is to a link in an HTML IFRAME. The isEmbedded query string argument supports that. Send it with a value of true to exclude navigation and most interactive features. For example:

```
https://{server}/{deployment-path}/worksheets/worksheets/WEB31?
studentId=<someid>&degree=BA&isEmbedded=true
```

```
https://{server}/{deployment-path}/plans/active?
studentId=<someid>&degree=BS&isEmbedded=true
```

For links to /worksheets and /plans/active, additional interactive options are excluded when isEmbedded=true because the goal is to show the user is a static view without any ability to navigate elsewhere or make changes. Some of the excluded features are:

- All ability to save, delete, create, modify
- Format and Historic Audit selections
- Process new audit button and check boxes
- Sidebar for Courses and Still Needed in plans

Course Information

Updated: September 29, 2023

Users can view additional course information from either Banner or an external source like Leepfrog's CourseLeaf.

When enabled, still-needed courses on an audit (academic, what-if, or planner) or the Courses list of the requirements drawer in Plans and Template Management appear as a link, which opens the Course Information dialog.

In Plans, to display the Course Information dialog on course and choice requirements or all course and choice requirements on a term, users can select **More information** from the action menu of the item. The section information shown is for the planned term only. In the Still Needed and Courses lists in the sidebar in Plans or the Courses list in Template Management, users can click the information icon.

Course Link

Updated: March 25, 2022

Course Link displays additional information from Banner about a course or courses, such as description, prerequisites, and sections.

Note: This functionality is available only for Banner sites. For additional information, see the [Course Link configuration](#) topic.

Configuration

- Review the Show CourseLink and CourseLink*Order flags in RPT036 to ensure they are configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - dash.courseInformation.enabled
 - dash.courseLink.meetingTime.24hour
 - studentPlanner.courseLink.enable

Class registration from Course Link

Updated: March 25, 2022

Course Link displays additional information about a course or courses, such as description, prerequisites, and sections. Single course registration from Course Link can be enabled for Banner clients.

When enabled, authorized users will be able to select a course section in the Course Link sections area and register for it. If the course section does not have any open seats, it cannot be selected. If the section requires an authorization code (configured in SFAAUTC in Banner), the user will be prompted for that code before they can complete registration for the selected section. For more information, see the Banner documentation on SFAAUTC. Only one class registration request can be submitted at a time, and a confirmation message appears if the registration is successful. If the registration is not successful, the user will receive a notification that registration could not be completed.

This is useful for an advisor to help a student register for a needed requirement, but it does not allow for the full range of features available in Banner Registration. From Course Link, a user cannot register for multiple courses at once, drop a course, change a course section or be added to the waitlist. Additionally, PINS is not currently supported and fees are not assessed. If using pre-requisite checking, the student must be registered for concurrent courses before registering for a course that uses those courses as a concurrent requisite.

Class registration from Course Link is accessible from the following locations:

- Still needed requirements on an audit (all audit types) for a single course in addition to wildcards and ranges
- Courses in the Plans Still Needed sidebar
- Courses in the Plans Courses sidebar
- Courses in the Plans requirement drawer
- Course and Choice requirements on a plan
- Terms on a plan

Note: For all options except "Course and Choice requirements on a plan" and "Terms on a plan," all open sections of registration will be shown. For "Course and Choice requirements on a plan" and "Terms on a plan," only sections for that planned term are shown. In Degree Works, Course Link must be enabled and configured, and the `integration.banner.registration.*` Shepherd settings must be configured. For audits, set the `dash.courseInformation.enabled` Shepherd setting to false and the RPT036 Show CourseLink flag to Yes for all audit formats that should have this functionality enabled. For plans, the `studentPlanner.courseLink.enable` Shepherd setting should be true. For both audits and plans, ensure that the RPT036 CourseLink Sections Order is not zero or blank. (Course Link in Plans uses the RPT036 SEP31 format.) The sections that display in Course Link are determined by the SSBSECT3 section in the `integration.banner.extract.config` Shepherd setting. By default, the extract will show all sections that have been created in Banner regardless of whether the schedule of classes has been made available for viewing in Self-Service Banner. You may want to consider modifying this section to only bring over sections that should be displayed. You may also want to disable registration from Course Link (`integration.banner.registration.enable`) when there are no active registration terms so future sections display for planning purposes. Before the newly registered course can be applied to an audit, it must be bridged into Degree Works. If your site is using dynamic refresh, you may want to set CFG020 REFRESH Run Audit Refresh to Y so that the user can navigate to Worksheets and click Process to refresh data and then run an audit (if the user is authorized to run new audits). Additionally, if the user has

access, they can also use the refresh button to pull new classes and run a new audit. In Banner, the Register Student With ADD and DROP Actions ERP API, which is part of the Student APIs deployment on Banner, must be up and running to allow registration from Course Link. Any Banner registration flags and configurations will be obeyed by the API. This is a POST request API with the endpoint `/api/registration-register`. The parameters that Degree Works uses are:

- `bannerId`
- `Term`
- `courseReferenceNumbers (CRN)`
- `authCode (Authorization Code)`

Some of the messages displayed can be modified using Controller to localize the `dash.courseLink.registration.banner.*` and `dash.error.error.CONFIGURATION.*` text in `DashboardMessages.properties`. However, there are some messages coming directly from the API that cannot be localized.

Configuration

- Review the following Shepherd settings to ensure they are configured correctly:
 - `dash.courseInformation.enabled`
 - `dash.courseLink.meetingTime.24hour`
 - `integration.banner.registration.apiUrl`
 - `integration.banner.registration.enable`
 - `integration.banner.registration.password`
 - `integration.banner.registration.username`
 - `studentPlanner.courseLink.enable`
- Access to register from Course Link is granted if the user has the REGISTER Shepherd key.

External course information

Additional information about a course or courses from an external source can be configured to display in audits, Plans, and Template Management.

When configured to display external course information, a new tab or window will open based on the user's browser configuration. The URL for the external source should be appended with tags defining the course. These optional tags will be replaced with the actual course details when the request is made.

- {courseDiscipline} is replaced with the course discipline
- {courseNumber} is replaced with the course number
- {attributeValue} is replaced with the attribute code that is part of the WITH qualifier. For example, 1 Class in MATH 2@ (With Attribute=ABCD)
- {attributeOperator} is replaced with the attribute operator that is part of the WITH qualifier. Usually this is = (equals) or <> (not equals).

For example, if this URL is configured as:

```
https://externalcourseinformation.myserver.tld/?  
sDisc={courseDiscipline}&sNumber={courseNumber}&attribute={attributeValue}&attributeOperator={attributeOperator}
```

The request URL for CHEM 106 in the worksheet or the planner would be:

```
https://externalcourseinformation.myserver.tld/?  
sDisc=CHEM&sNumber=106&attribute=&attributeOperator=
```

For CHEM 2@ (With Attribute=WRIT) the URL would be:

```
https://externalcourseinformation.myserver.tld/?  
sDisc=CHEM&sNumber=2@&attribute=WRIT&attributeOperator=
```

Configuration

- Review the Show CourseLink flag in RPT036 to ensure it is configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - dash.courseInformation.enabled
 - dash.courseInformation.url

API access from external applications

Updated: September 30, 2022

The Degree Works APIs that provide access to articulations, audits, plans, and requirements can be integrated with external applications like CRM Advise and Banner's Registration and CPoS processes.

See the Ellucian API Documentation on the Customer Center for more information.

Authentication

Stateless token authentication is recommended when accessing Degree Works APIs, but some applications will need to use HTTP Basic authentication over SSL. For these applications, you will need to create a user access record in Degree Works.

For example, if you use Degree Works plans with Banner registration, use the Users function of Controller to create a record for the username and password defined on SOAREST. We recommend assigning the API Rest Users user class, and at a minimum, assigning the RSPLAN, SDSTUANY, and NOREFER keys.

Navigation

Updated: September 29, 2023

The header bar at the top of Responsive Dashboard helps users navigate through the application.

The user can access the functionality to which they have been given access by clicking the link for that item.

Configuration

Access to the following is granted if the user has the specified Shepherd key.

- Worksheets - SDAUDIT and SDAUDREV
- Exceptions - SDEXCEPT
- Plans - SEPPLAN
- Exception Management (under Admin) - SDEXPMGT
- Template Management (under Admin) - SEPTMGMT
- Debug (under Admin) - DEBUG
- Theme (under Admin) - SDADMIN
- Links - EXTLINKS

Mega Menu

Users can also navigate throughout the application using the Mega Menu on the right of the header bar. Note that for users on a mobile device, only the mega menu will be available for navigation.

Help

If authorized, administrative users can access the Ellucian Documentation site from the persona drop-down menu. If users have the DOCDASH Shepherd key, they will see a Help link in the menu. A login to the Ellucian Support Center is also required to access the documentation, and this access should not be given to students.

Logout

To log out of Responsive Dashboard, select Sign Out from the persona drop-down button on the right of the header bar.

Student

Updated: September 29, 2023

Persistent on all pages that focus on a student (that is, Worksheets, Exceptions, and Plans) is a student data card. Depending on the user's access, they can enter a known student ID, see a pre-loaded list of advisees or assigned students, and search for students.

Student search

Updated: March 24, 2023

If a user has access to Advanced search, a dialog will open that allows the user to select criteria specific to the students for which they would like to search for.
[c_responsive_dashboard_student_search.html](#)

The user will always have the option to search by a student's ID, first name, and last name. Curriculum data can also be configured as search options. Degree, Level, Classification, Catalog year, and Degree source will allow for only one value to be selected, while the other curriculum data will allow for multiple values to be selected. Multiple selections in the same data item will be an AND. This means that if you choose majors of both ART and CHEM, the search results will return only students who have both majors – either on the same degree or different degrees. Selections from different data items are also an AND. For example, if you select Degree=BS and Major=ART and Major=CHEM, you are saying to select any student with the BS degree and both the MATH major and the CHEM major, so BS degree students with dual majors of ART and CHEM (and possibly additional majors). You can reset your select options to the default by clicking **Clear**.

In addition to curriculum data, up to 10 custom search items for rad_custom_dtl data can be configured. To add a custom search item:

- Add the rad_custom_value codes to a UCX-CST table.
- Set the corresponding dash.search.custom.cst*.enabled Shepherd setting to true.
- Set the corresponding dash.search.custom.cst*.code Shepherd setting to the rad_custom_code.
- Localize the corresponding dash.studentSearch.custom.cst*.label property.

For example, to add Sport as a custom search item:

- Add all valid SPORT codes to CST001 (for example, FOOTBALL).
- Set dash.search.custom.cst001.enabled = true.
- Set dash.search.custom.cst001.code = SPORT.
- Localize dash.studentSearch.custom.cst001.label = Sport.

In the **Custom data** area, there will be a **Sport** drop-down with the values from CST001. When FOOTBALL is selected, any student with a rad_custom_dtl with CODE = SPORT and VALUE = FOOTBALL will be returned in the search.

Note that if any of your UCX-CST tables are used for other functionality, make sure that the corresponding dash.search.custom.cst*.enabled Shepherd setting is set to false.

The first name search field actually searches on any part of the name following the comma in the student's name. This typically includes the first, middle, and preferred names. A student whose name is Jones, Robert William (Bob) will have his name stored in the name search field in the database as JONES,ROBERTWILLIAMBOB. Searching on the first name field will match against ROBERTWILLIAMBOB and thus searches on Robert, Rob, Bert, Will, Liam, and Bob will all find this student. The last name field searches on the value preceding the comma. Searching on Jones or Jon will find this student, but searching on Ons will not.

Click **Search** to execute the search. All students who meet the search criteria will be displayed in the bottom window of the Find Students search dialog. The list of students is sorted by name in descending order as a default but can be sorted by clicking on any of the column headers in the results grid. By default, all students in the results list are selected; unchecking the check box to the left of the student will deselect them.

If more students than are allowed based on the dash.search.maximum setting are found, only the maximum number of students are returned. A warning will display showing the maximum and how many were found suggesting to the user to refine their search.

You can filter out students who have graduated or dropped out using the UCX-CFG020 WEBPARAMS **Term used for filtering in student search** field. Students whose active term (on the rad_student_mst) is older than this term will be filtered out from the search results. The **Term used for filtering in student search** field can be left blank so that no filtering occurs.

Click **Select** to close the search window and load the list of students into Degree Works. You can then select any student from the list to process a degree audit, apply an exception or manage a plan.

Click **Cancel** to discard your search and go back to the originating page.

Some institutions may limit advisors to having access only to their list of advisees or department heads to having access only to students in their department. In this case, a preselected list of students will appear in the **Select student** drop-down and the Advanced search functionality will not be available. Please see the [Advisee filtering](#) topic for additional information.

Configuration

- Review the following UCX tables that populate the Advanced search drop-downs: STU307 (Degree), STU350 (Level), STU305 (Classification), STU035 (Catalog year), STU023 (Major), STU024 (Minor), STU560 (College), STU323 (Specialization), STU563 (Concentration), STU324 (Liberal learning), STU316 (Program), STU306 (Student Status Codes). The drop-downs will be populated with all values with Show in Advanced Search Picklist = Y.
- Review CFG020 WEBPARAMS **Term used for filtering in student search** field to filter out students who are not active.
- Review the following Shepherd settings to ensure they are configured correctly:

- dash.search.catalogYear.enabled
- dash.search.college.enabled
- dash.search.concentration.enabled
- dash.search.custom.cst*.enabled
- dash.search.custom.cst*.code
- dash.search.degree.enabled
- dash.search.degreeSource.enabled
- dash.search.level.enabled
- dash.search.liberalLearning.enabled
- dash.search.major.enabled
- dash.search.maximum
- dash.search.minor.enabled
- dash.search.program.enabled
- dash.search.school.enabled
- dash.search.specialization.enabled
- dash.search.studentId.mask
- dash.search.studentStatus.enabled
- Access to the following is granted if the user has the specified Shepherd keys:
 - Search by ID - SDFINDID
 - Advanced search - SDFIND

Student data

Updated: September 29, 2023

After a student has been selected, the student's information is loaded into the student data area.

This includes the Student ID, Degree, and other goal data. By default, only goal data items for which the student has a record will display. For example, if the student doesn't have a concentration, the Concentration label will not display. Additionally, custom data can be configured to display in the student data area.

Next to the student's ID and name is the student's degree. If the student is seeking multiple concurrent degrees, they will appear in a drop-down. The order of the degrees appearing here matches the order of how the goal records were bridged to Degree Works from the student system.

Student data localization

Updated: March 25, 2022

The goal items that will display is controlled by the `dash.studentHeader.goals.items` setting. If you don't want a specific goal to display, use Controller to remove that goal from the list.

The full list of goals is:

```
school,classification, major, minor, program, conc, college, spec, li  
bl
```

Here "school" is undergraduate, graduate, etc while "classification" is 1st-year, 2nd-year, etc. The space after the comma is optional.

If you want the goal label to appear when the goal value does not exist, then add the appropriate property with the "none" suffix to your localized `DashboardMessage.properties` in Controller. For example, to always show the major label when the major is missing, add this property: `dash.home.context.MAJORFormat.none={label} (no major)`. Without this property in place the major label will not appear if the student does not have a major. Having a property for each goal allows you to always have a label for some custom items and to have different text showing for each missing goal value. You might want "missing" to show for one goal item and "see advisor" showing for another. See the baseline `DashboardMessage.properties` in Controller for a full list of the "none" properties. You can copy them to your localized properties.

There is a corresponding `dash.worksheets.goals.items` setting to display the goal items in the planner audit. The properties should be localized in `DashboardMessages.properties` in the same manner as described above, except they have a prefix of `dash.worksheets` instead of `dash.home.context`.

There is a corresponding `core.audit.printView.studentHeader.goals.items` setting to display the goal items in the PDF audit output. The properties should be localized in `PDFAuditMessages.properties` in the same manner as described above, except they have a prefix of `worksheets` instead of `dash.home.context`. If you want the goal label to appear when the goal value does not exist, then add the appropriate property with a `_nodata` suffix to your localized `PDFAuditMessage.properties` in Controller. For example, to always show the major label when the major is missing, add this property: `worksheets.major_nodata=(no major)`.

Custom student data display

Updated: September 29, 2023

Data from the `rad_report_dtl` and `rad_custom_dtl` can be configured to display in the student data card.

Any item defined in SCR002 and stored on the `rad_custom_dtl` for the student, or any item defined in RPT046 and stored on the `rad_report_dtl` for the student can be displayed. Like goal data, only custom and report items for which the student has a record will display by default.

Configuration has several steps. First, set the `dash.studentHeader.custom.enabled` setting to true to enable the display of custom data. Then define the items to display and the order in which to

display them in the `dash.studentHeader.custom.items` setting. And finally, use Controller to add localized properties for the custom data labels. These should be on `DashboardMessages.properties` and each custom data item should have a property in the format of `dash.studentHeader.custom.xxxx`, where `xxxx` is the lowercase code from SCR002 or RPT046.

Here is an example of what the `dash.studentHeader.custom.items` setting may look like (spaces are optional):

```
acstanding, gpain, gpaov, admitterm
```

Given these custom items, you would then create properties like this in `DashboardMessages.properties`:

```
dash.studentHeader.custom.acstanding=Academic Standing
```

```
dash.studentHeader.custom.gpain=Institutional GPA
```

```
dash.studentHeader.custom.gpaov=Overall GPA
```

```
dash.studentHeader.custom.admitterm=Admit Term
```

If you do not define the custom item in `DashboardMessages.properties`, you will see `{item}` property not defined as the label. For example, `admitterm` property not defined.

If you want the custom label to appear when the custom value does not exist, then add the appropriate property with the `none` suffix. For example, `dash.studentHeader.custom.xxxxx.none={label} no data`. Having a property for each custom item allows you to always have a label for some custom items and to have different text showing for each missing custom value. You might want missing to show for one custom item and see advisor showing for another.

To display custom data that should appear only for users of a certain user class, you can use Controller to create a new setting with the user class as a suffix. For example, create new setting `dash.studentHeader.custom.items.adv` with the list of the custom data items that should appear only for users in the ADV user class. You may want to include the student's registration PIN in this advisor-only setting so the advisor will see it on the worksheet, but the student will not see it. If you use both the ADV and ADVX user classes, be sure to create a setting for each as needed.

There are corresponding `dash.worksheets.custom.enabled` and `dash.worksheets.custom.items` settings to display the custom items in the planner audit. The properties should be localized in `DashboardMessages.properties` in the same manner as described above except they have a prefix of `dash.worksheets.custom` instead of `dash.studentHeader.custom`.

There is a corresponding `core.audit.printView.studentHeader.custom.items` setting to display the custom items in the PDF audit output. The properties should be localized in `PDFAuditMessages.properties` in the same manner as described above except they have a prefix of `studentHeader.custom` instead of `dash.studentHeader.custom`. If you want the custom label to appear when the custom value does not exist, add the appropriate property with a `.none` suffix to your localized `PDFAuditMessage.properties` in Controller. For example, `studentHeader.custom.xxxxx.none=no data`.

If you have a custom or report value that should display based on the selected degree, you should set up BAN080 to pull over the school/level and degree with the data value. For example, if you

are pulling the admit code from SORLCUR into the rad_report_dtl, you may want to also pull the SORLCUR_LEVL_CODE or SORLCUR_DEGC_CODE values. If one or both of these are pulled into the school and degree fields on the rad_report_dtl, the correct admit code will be displayed in the UI for the selected degree. See the [UCX-BAN080 Banner Custom Data Definitions](#) topic for details about setting up BAN080 to pull these additional fields.

Configuration

- Review SCR002 and RPT046 to ensure they are configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - core.audit.printView.studentHeader.custom.items
 - core.audit.printView.studentHeader.goals.items
 - dash.studentHeader.custom.enabled
 - dash.studentHeader.custom.items
 - dash.studentHeader.goals.items
 - dash.worksheets.custom.enabled
 - dash.worksheets.custom.items
 - dash.worksheets.goals.items
 - localization.internationalization.enable

Student data refresh

Updated: March 25, 2022

A user may be given access to refresh the student's data by performing a dynamic refresh.

If enabled to display, the Data refreshed date and time indicate the last time the student's academic data was copied from the Student Information System (SIS) into Degree Works, while the refresh button allows a user to again pull in this student's data. If you hover over the Data refreshed label, the hover text indicates the last date and time the student's data changed, which may be older than when it was refreshed. After the refresh process completes, the user will get a confirmation message, the Data refreshed date and time will be updated, and a new audit is automatically generated if data changes were detected after the last time the student's data was refreshed. For Banner schools, for additional information on this process, see the [Batch Extract flow diagram](#) topic.

The refresh button is used to make a call to obtain the newest data from the Student Information System (SIS) for the current student.

The refresh button can be used by Banner and non-Banner schools. When the dash.refresh.external.enabled setting is set to false, the Responsive Dashboard API will make a call to the classic server to pull the student's data from Banner. When this setting is set to true, the Responsive Dashboard API will make a call to an external API housed on the school's server. This external API written by the school will retrieve the student's data from its SIS and make a call to

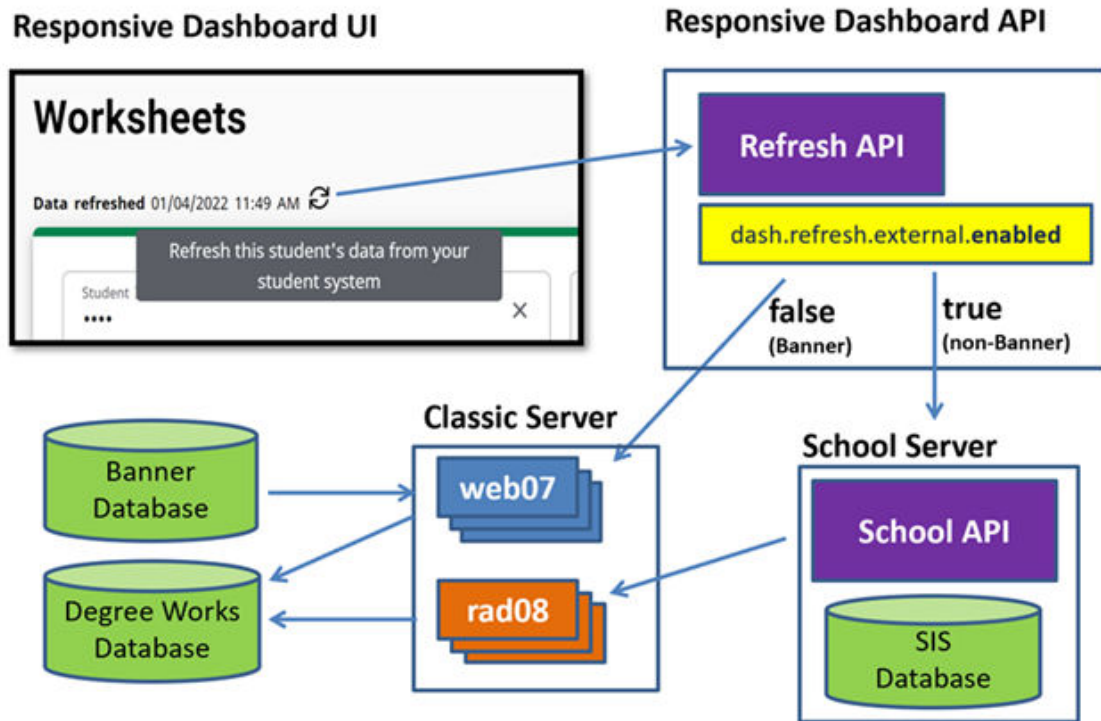
RAD08 on the classic server to have the data written to Degree Works. See the [Degree Works dynamic refresh](#) topic for details about how to call RAD08 with the student data.

When `dash.refresh.external.enabled` is true, the Responsive Dashboard uses the `dash.refresh.external.url` to know how to call the external API. The `dash.refresh.external.username` and `dash.refresh.external.password` settings are used to authenticate against the external API. The URL should include a special token `{studentId}` that will be replaced by the student's Degree Works ID. The optional token `{userId}` may also be part of the URL. This token will be replaced by the user's Degree Works access ID and may be used for logging or other purposes. This URL setting may look something like this: `https://myschool.edu/something/like/this?stuld={studentId}&user={userId}`

The external API will pass the student's data as BIF records to RAD08 to be bridged into Degree Works. After RAD08 processes the student data, an indication of `DATACHANGED=YES` is returned in the response if the Degree Works bridge has detected changes after the last time the student was bridged, either through RAD08 or through the batch process. In this case, the data last changed date is updated to the current date when you hover over the Data refreshed label. If no changes are detected then an indication of `DATACHANGED=NO` is returned instead. The external API should interpret the `DATACHANGED` value and return a status of 0 for the former condition and 4100 for the latter condition. The status response returned by the external API to the Responsive Dashboard should be a JSON object with a single status property and string value of 0 or 4100. For example:

```
{
  "status": "4100"
}
```

If the `studentId` passed to the external API is not valid in the SIS, the external API should return http status 404. A successful http status code of 200 should be returned if the refresh request was successful.



Configuration

- SDREFRES Shepherd key - provides user access to see the refresh data and time.
- SDREFBTN Shepherd key - provides user access to the refresh button.
- Review the following Shepherd settings to ensure they are configured correctly:
 - `dash.refresh.external.enabled`
 - `dash.refresh.external.password`
 - `dash.refresh.external.url`
 - `dash.refresh.external.username`

Tools

Updated: March 25, 2022

Functions that are specific to the student context (that is, Worksheets, Exceptions, and Plans) have a common set of tools that are accessible on each page.

Using a set of icons in the upper right, the user can print audits, get student or advisor contact information, use GPA calculators, view class history data, and manage notes and petitions.

Contact

Updated: March 25, 2022

The Contacts icon opens a dialog that displays the student and advisors' names and email addresses.

Students see their advisors. Non-students see the student contact information, and if they have the CONTADVR key, they also see the student's advisors. The contact icon displays only if the user has the CONTACT key.

Clicking the email address opens the user's default email program with the contact's email address loaded in the To: field.

You can show the advisor-type field from the rad_attach_code on the ADVISOR rad_goalData_dtl record, which is populated from SGRADVR_ADVR_CODE for Banner schools. Non-Banner schools can bridge it. By setting up the properties, you can show either the code (for example, PRIM) or show a description (for example, Primary advisor) for each advisor-type. See the dash.tools.contact properties in DashboardMessages.properties to see how to show the advisor-type.

Configuration

Access to the following is granted if the user has the specified Shepherd key:

- Contact window - CONTACT
- View advisors - CONTADVR (not needed for students)

GPA Calculator

Updated: September 29, 2023

The GPA Calculators are located under the More menu. There are three different GPA calculators: Graduation, Term, and Advice.

The calculators can help students set realistic goals at the beginning of the term or academic career and calculate their end-of-term GPA using actual academic information.

If a student's most recent audit does not include in-progress classes, a new audit will be run but not saved when the user selects this tool.

Configuration

- Review the Graded Attempted and Use in GPA Calculators flags in STU385 to ensure they are configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - dash.gpaCalculator.decimals

- dash.gpaCalculator.round
- Access to the following is granted if the user has the specified Shepherd key:
 - GPA Calculators - SDGPACLC
 - Graduation Calculator - SDGPAGRD
 - Term Calculator - SDGPATRM
 - Advice Calculator - SDGPAADV

Graduation Calculator

Updated: March 25, 2022

This calculator gives the student a general view of the average GPA they will need to earn over their final "X" credits to achieve their desired GPA.

In some cases, the student may be informed that their desired GPA is not possible to achieve, considering their number of credits remaining. This calculator helps students to set long-term general goals.

Graduation Calculator inputs

- Current GPA (defaults from the rad_cum_gpa on the rad_term_dtl)
- Credits Remaining (defaults from the most recent academic audit)
- Credits Required (defaults from the most recent academic audit)
- Desired GPA

Term Calculator

Updated: September 30, 2022

The term calculator allows students to predict what their GPA will be after completing the current term.

If the student has coursework in the current term on their most recent academic audit, these classes and their credits will automatically load into the calculator. Classes can be deleted, and additional classes can be added.

Term Calculator inputs

- Current GPA (defaults from the rad_cum_gpa on the rad_term_dtl)
- GPA credits attempted (sum of gpa-credits from classes applied to the audit)
- Class information
- In-progress classes

Advice Calculator

Updated: September 30, 2022

This calculator is used to figure out how a student can raise or lower their GPA using actual grades as advice.

Note that the Graded Attempted flag in STU385 is used to filter out grades – only those grades with Graded Attempted = Y are used in the Advice Calculator – allowing you to exclude P grades, for example.

Advice Calculator inputs

- Current GPA (defaults from the rad_cum_gpa on the rad_term_dtl)
- GPA credits attempted (sum of gpa-credits from classes applied to the audit)
- Desired GPA

Class History

Updated: September 29, 2023

Class History is located under the More menu. It is a listing of all the student's coursework, organized by term.

By enabling `dash.classHistory.auditSection.enabled`, you will see where each class that is not applying to a requirement appears in the audit.

By enabling `dash.classHistory.termSummary.enabled`, after each term will appear both a set of term and cumulative calculated values. Any item in the summary can be disabled by localizing the properties file. See the `dash.classHistory.termSummary` section of `DashboardMessages.properties`.

If a student's most recent audit does not include in-progress classes, a new audit will be run but not saved when the user selects this tool.

Configuration

- Review the `dash.classHistory.auditSection.enabled` and `dash.classHistory.termSummary.enabled` settings to ensure they are configured correctly.
- Access to Class History is granted if the user has the SDXML31 Shepherd key.

Petitions

Updated: March 25, 2022

Located under the More menu, Petitions allow users to enter requests for exceptions.

A petition can be a request to have a particular requirement modified or waived for a particular student.

A preview of all petitions that have been entered for the student displays initially. The list can be filtered by the petition status, either awaiting approval, approved, applied as exception, or rejected. The date the petition was created, the author of the petition and the petition status will display for each petition.

Click **Add a new petition** to add a new petition. After you have entered the text for the petition, click **Save petition** to save the petition to the database. A message will appear telling you your petition was added successfully. All new petitions have a default status of awaiting approval when first created. After a petition is created, it can be acted upon by registrar-class users or users having access to Exceptions Management.

To see all the text of a petition, select **View petition** from the action menu in the right corner of a petition. Click **Back** to go back to the petition list. From this dialog, you can also edit or delete the petition.

To modify a petition, select **Edit petition** from the action menu in the right corner of a petition. Click **Save petition** to save your changes. Click **Cancel** to discard your changes and go back to the petition list. From this dialog, you can also delete the petition.

To delete a petition, select **Delete petition** from the action menu in the right corner of a petition. Click **Delete petition** to delete the petition. Click **Cancel** to keep the petition and go back to the petition list.

Petitions can be configured to display on an audit. If enabled, they will be in the Notes section.

Configuration

- Review the Show Petitions flag in RPT036 to ensure it is configured correctly.
- Access to the following is granted if the user has the specified Shepherd key:
 - Petitions - SDPETVIEW
 - Add petitions - SDPETADD
 - Modify petitions - SDPETMOD
 - Delete petitions - SDPETDEL

Petition email notification (petsend)

Updated: March 25, 2022

Degree Works can be configured to send an email that notifies the recipient that there are new petitions either waiting approval or approved petitions waiting to be applied as exceptions.

However, if you are a Banner school using Workflow to manage petitions you will not want to use this email notification as Workflow takes care of this for you.

The petsend Perl script is used to configure where the notification email is to be sent. Below is an example on how to run petsend to send notification emails:

Example:

```
petsend someaddress@yourschool.edu WAITING
```

```
petsend someaddress@yourschool.edu APPROVED
```

The Reply To email address is set to the same email address as the Send To address by default as shown below:

```
$ReplyToEmailAddress = $ToEmailAddress;
```

To modify the Reply To e-mail address, change the \$ToEmailAddress in the petsend script to the e-mail address you want replies to be sent to as shown below:

```
$ReplyToEmailAddress = "computer_center\@yourschool.edu";
```

You can also configure petsend to carbon copy (CC) one or more individuals at your institution.

To activate the CC function, modify the following line with the e-mail address of the person you want to CC:

```
$CC = "someone_else\@yourschool.edu";
```

The petsend job can be configured not to send a notification e-mail if there are no petitions either waiting approval or waiting to be applied. To disable sending e-mails under these circumstances, change the value for SEND_ZERO_PETITIONS_MSG to "N" as shown below:

```
$SEND_ZERO_PETITIONS_MSG = "N";
```

You can edit the content of the notification e-mail by modifying the DefineSubjectAndBody function of the petsend script.

Petition approval with Banner Workflow

Updated: March 25, 2022

When Workflow is enabled, Banner Workflow can be used to manage petition approvals instead of Exception Management.

Using this method, when a petition is created it will be sent to the petition Workflow model where the configured approvers will be notified to review and either accept or reject the request. The petition status will then be updated based on the approver's review.

See the [Banner Workflow integration installation](#) topic for additional information.

Configuration

Review the following Shepherd settings to ensure they are configured correctly.

- core.workflow.petition.*
- core.workflow.password

- core.workflow.username
- core.workflow.wsdl

Notes

Updated: March 25, 2022

Located under the More menu, Notes allows users to document academic advising on student records.

Notes can be marked as internal so that students will not see them. Notes made available to the student appear in audit reports in a Notes section at the bottom of the report.

A preview of all notes that have been entered for the student displays initially. The date the note was created, the author of the note and whether the note is internal will display for each note.

Click **Add a new note** to add a new note. Enter your note text or select a predefined note if this functionality has been enabled. Users with the SDNTECHG key can also enter additional text to the predefined note in the note field. To make a note internal, select the **Not available to student** check box if this functionality has been enabled. Click **Save note** to save the note to the database. A message will appear stating your note was added successfully. If the user has access, a new academic audit will be automatically generated when adding a note.

To see all the text of a note, select **View note** from the action menu in the right corner of a note. Click **Back** to go back to the note list. From this dialog, you can also edit or delete the note.

To modify a note, select **Edit note** from the action menu in the right corner of a note. Click **Save note** to save your changes. Click **Cancel** to discard your changes and go back to the note list. From this dialog, you can also delete the note.

To delete a note, select **Delete note** from the action menu in the right corner of a note. Click **Delete note** to delete the note. Click **Cancel** to keep the note and go back to the note list.

Configuration

- Review CFG071 to ensure it is configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - dash.notes.internal.enabled
 - dash.notes.predefined.enabled
 - dash.audit.process.inprogress.defaultValue
 - dash.audit.process.preregistered.defaultValue
- Access to the following is granted if the user has the specified Shepherd key:
 - Notes - SDNOTES

- View notes - SDNTEVUE
- Add notes - SDNTEADD
- Modify notes - SDNTEMOD
- Delete notes - SDNTEDEL
- Add text to predefined notes - SDNTECHG
- Automatically run a new audit after adding a note - SDNTERUN

Worksheets

Updated: September 29, 2023

A user can view and run a variety of audit reports in Worksheets.

Each audit report displays specific information about students and their progress towards degree completion. After a student is selected, the most recent audit for the student will load automatically. If the `core.audit.process.runOnScribeChanges.enabled` Shepherd setting is true, the auditor engine will check to see if any changes were made to the Scribe blocks in the student's audit after the last audit was generated. A new audit will be run if changes are detected.

A PDF of an audit can be generated by clicking on the Print icon that displays on an audit. A dialog prompting for the page dimensions will open, and after making a selection, the generated PDF will display in a browser window. The user can choose to send the output to a printer, save the output as a PDF, or any other option available in the browser. The options in the page dimensions dialog are defined in SYS100. The text and label properties in the PDF audit can be localized in `PDFAuditMessages.properties`.

Configuration

- Review SYS100 to ensure it is configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - `core.audit.printView.dimensions.defaultValue`
 - `core.audit.printView.repeat.codes`
 - `core.audit.printView.showNotesInternal`
 - `core.audit.printView.showStudentId`
 - `core.audit.printView.studentHeader.custom.items`
 - `core.audit.printView.studentHeader.goals.items`
 - `core.audit.process.runOnScribeChanges.enabled`
- Access to Worksheets is granted if the user has the SDAUDIT Shepherd key. The SDAUDREV key is also required to view the actual worksheet.

Academic

Updated: March 25, 2022

The Academic audit is the primary audit used to track and evaluate a student's progress towards a degree.

Academic Audit

Updated: March 25, 2022

The academic audit is a series of blocks that define degree and program requirements.

All blocks initially load as expanded in a desktop view; for mobile users all blocks initially load as collapsed. You can manually expand or collapse individual blocks by clicking the arrow in the upper right of each block or expand/collapse all block by clicking the link at the top of the audit.

While the content of the academic audit is driven by the requirements defined in Scribe, the appearance is controlled by the settings in RPT036 for each report format. For example, Show Label controls whether the completeness icons and labels should display on the audit, while Show Legend and Show Disclaimer control whether to display the legend and disclaimer on the audit. You should review all of these flags to ensure they are set correctly for the audit formats you use.

The default behavior for block header data in each block type can be configured in SCR004. Additionally, HeaderTags can be scribed to override the flags as needed.

Configuration

- Review RPT036 to ensure it is configured correctly for WEB30, WEB31, WEB33, and WEB36.
- Review the Show.* flags in SCR004 for each block type.

Format

Updated: March 25, 2022

The Format drop-down allows the user to select in which format the audit report should be displayed.

There are several different report formats available, although not all formats are appropriate for all user classes. For example, many schools choose to not make the Registrar Report available to students. These report formats can be configured by the client to meet their respective needs.

Configuration

Access to the following is granted if the user has the specified Shepherd key:

- Registrar Report - SDWEB30
- Student View - SDWEB31

- Graduation Checklist - SDWEB33
- Registration Checklist - SDWEB36

Degree progress

Updated: September 29, 2023

Degree progress can be measured in percentage of degree requirements met, the percentage of credits completed, and the overall GPA.

The values are calculated by the auditor based on the requirements and qualifiers defined in the audit's blocks. For each report format, displaying the requirements and credits progress and overall GPA is configurable.

For more information, see the [Percent complete calculation](#) topic.

Configuration

Review the Show Progress Bar Requirements, Show Progress Bar Credits, and Show Overall GPA flags in RPT036 to ensure they are configured correctly for each report.

Process audit

Updated: March 25, 2022

With access, a user can generate a new audit by clicking the Process button.

Options to select whether to include in-progress and preregistered in addition to the default selection for these options can also be configured. Note that if the in-progress and preregistered check boxes are configured to not display but the user can still process a new audit, the auditor will follow the default value settings to determine whether in-progress and preregistered classes should be included.

Configuration

- Review the following Shepherd settings to ensure they are configured correctly:
 - `dash.audit.process.inprogress.defaultValue`
 - `dash.audit.process.inprogress.enabled`
 - `dash.audit.process.preregistered.defaultValue`
 - `dash.audit.process.preregistered.enabled`
 - `dash.audit.repeat.codes`
- Access to process an audit is granted if the user has the SDAUDRUN Shepherd key.

Historic audits

Updated: March 25, 2022

View Historic Audits allows a user to view historical audits for a student.

The maximum number of historical audits to keep per student per School/Degree combination can be configured. With access, a user can view historic audits by selecting the audit to view from the drop-down and it will load using the currently selected report format.

Configuration

- Review the Audit History Depth flag in CFG020 DAP14 to ensure it is configured correctly.
- Access to view historic audits is granted if the user has the SDHIST Shepherd key.

Diagnostics

Updated: March 25, 2022

Diagnostics allows a user to view additional information about an audit and can be used to troubleshoot audit issues.

With access, a user can view diagnostics by clicking the link. A new tab or window with the diagnostics audit will open.

Configuration

Access to Diagnostics is granted if the user has the SDXML30 Shepherd key.

Student Data

Updated: March 25, 2022

Student Data is a report of what is in the Degree Works database for the selected student.

This can be useful in troubleshooting issues with the audit. With access, a user can view student data by clicking on the link. A new tab or window with the student data report will open.

The data reflects the student's current set of classes and curriculum and other data saved in the database. If you are viewing a historic audit and click the Student Data link, the data shown will likely not match the data showing on the historic audit.

Configuration

Access to Student Data is granted if the user has the SDXML32 Shepherd key.

Save audit

Updated: March 25, 2022

Audits may be saved or “frozen” so they do not get deleted when the maximum CFG020 DAP14 Audit History Depth has been reached.

Frozen audits do not count against the history depth value. With access, a user can freeze an audit by clicking Save audit. A dialog will open that prompts for the Freeze type and a Description.

When the user clicks Save, the audit will be saved in the database with the specified freeze type and description, and the screen will be updated with the user who saved the audit, the date, the freeze type, and description. For more information, see the [Freeze audits](#) topic.

Configuration

Access to the following is granted if the user has the specified Shepherd key:

- Freeze audits - AUDFREEZ
- Add a description - AUDDDESCR

Delete audit

Updated: March 25, 2022

Audits that have been saved or frozen can be deleted if the user has access to do so.

Clicking the Delete audit link opens a confirmation dialog. If the user clicks Delete, the audit will be deleted from the database. If the user clicks Cancel, Worksheets appears without deleting the audit.

Configuration

Access to delete audits is granted if the user has the SDAUDEL Shepherd key.

What-If

Updated: March 25, 2022

What-If audits allow you to process speculative degree audits for a student using their current class history.

You can audit a student against the requirements for a different major, minor, degree, catalog year, or other goal data and add classes that may be taken in the future. Responsive Dashboard can be configured to use curriculum rules to filter, use no filtering or use a specific form of filtering for the goal data drop-downs. The `core.whatIf.filterMode` controls which type of filtering will occur in both the worksheets tab and in the planner.

Configuration

- Review the `core.whatIf.filterMode` Shepherd setting to ensure it is configured correctly:
- Access to What-If is granted if the user has the SDWHATIF Shepherd key.

What-If Analysis with no filtering

Updated: March 24, 2023

When Responsive Dashboard is configured for no filtering rules (`core.whatIf.filterMode = N`), standard UCX filtering will be used in the What-If Analysis drop-downs.

Level, Degree, and Catalog year are required goal data items for a What-If and will default to the student's data. The options that display in Areas of study are configurable.

Catalog Year will always display and is a required field. The catalog year drop-down will be populated with all STU035 values with Show in What-If flag = Y, but the student's current catalog year will load as the default value.

By default, the catalog year in the Program area is used for all goal data items in Areas of study and Additional areas of study on the What-If audit. Users can be given the option of selecting different catalog years for individual goal data items by setting the `core.whatIf.enableGoalCatalogYear` Shepherd setting to true. When enabled, a catalog year drop-down displays adjacent to the goal data item. If a catalog year is selected, this will be passed along to the auditor engine. If left blank, the catalog year selected in the Program area will be used.

Degree will always display and is a required field. The degree drop-down will be populated with all STU307 values with Show in What-If flag = Y, but the student's current degree will load as the default value.

Level can be configured to display and is a required field if displayed. The level drop-down will be populated with all STU350 values with Show in What-If flag = Y, but the student's current level will load as the default value.

Program can be configured to display. The program drop-down will be populated with all STU316 values with Show in What-If flag = Y.

Major can be configured to display and can be configured to be a required field. The values in the major drop-down will be all AUD027 values.

Minor can be configured to display. The minor drop-down will be populated with all AUD029 values.

College can be configured to display. The college drop-down will be populated with all STU560 values with Show in What-If flag = Y.

Concentration can be configured to display. The concentration drop-down will be populated with all STU563 values with Show in What-If flag = Y.

Specialization can be configured to display. The specialization drop-down will be populated with all STU323 values with Show in What-If flag = Y.

Liberal Learning can be configured to display. The liberal learning drop-down will be populated with all STU324 values with Show in What-If flag = Y.

If any of these settings are enabled an Additional areas of study section will appear. In the drop-downs appearing in this section the user can select multiple of each goal.

- `core.whatIf.showProgramAdditional`
- `core.whatIf.showMajor`
- `core.whatIf.showMinor`
- `core.whatIf.showCollege`

- `core.whatIf.showConcentration`
- `core.whatIf.showSpecialization`
- `core.whatIf.showLiberalLearning`

Configuration

- Review AUD027 and AUD029 to ensure they are configured correctly.
- Review the Show in What-If flag in the following tables to ensure it is configured correctly: STU035 (Catalog Year), STU307 (Degree), STU350 (Level), STU316 (Program), STU560 (College), STU563 (Concentration), STU323 (Specialization), and STU324 (Liberal learning).
- Review the following Shepherd settings to ensure they are configured correctly:
 - `core.whatIf.filtermode`
 - `core.whatIf.majorRequired`
 - `core.whatIf.showCollege`
 - `core.whatIf.showConcentration`
 - `core.whatIf.showLiberalLearning`
 - `core.whatIf.showMajor`
 - `core.whatIf.showMinor`
 - `core.whatIf.showProgramPrimary`
 - `core.whatIf.showSchool`
 - `core.whatIf.showSpecialization`

What-If Analysis with curriculum-rule filtering

Updated: March 25, 2022

When Responsive Dashboard is configured to use curriculum rules (`core.whatIf.filterMode = R`), the values that appear in the drop-down lists will come from the curriculum rules loaded in `rad_currule_dtl`.

The Catalog year and Degree drop-downs will always display, while Campus, Program, Level, College, Major, Concentration, and Minor can be configured to display. Unless otherwise noted, the student's current goal data will initially populate the drop-downs if it exists on a valid curriculum rule.

Catalog Year will always display in the Program area and is a required field. The catalog years that are in the drop-down come from the codes in STU035 with Show in What-If picklist = Y. If `core.whatIf.defaultCatalogYear` is configured, this value will initially be displayed instead of the student's catalog year. Additionally, when `core.whatIf.changeCatalogYear` is false the default catalog year cannot be changed.

Campus will display in the Program area when `core.whatIf.showCampus` is true and will be a required field. The values in the campus drop-down will only be those from STU576 with Show in What-If picklist = Y and are on the selected curriculum rule.

Level will display in the Program area when `core.whatIf.showSchool` is true and will be a required field. The values in the level drop-down will only be those from STU350 with Show in What-If flag = Y and are on the selected curriculum rule.

College will display in the Program area when `core.whatIf.showCollege` is true and will be a required field. If `core.whatIf.showProgramAdditional` = N or `core.whatIf.mustUsePrimaryRule` = D or N, the college will also display in Additional areas of study. The values in the college drop-down will only be those from STU560 with Show in What-If flag = Y and are on the selected curriculum rule. When `core.whatIf.degreeBeforeCollege` is false, college will display before degree and drive the degree in the curriculum rule filtering.

Degree will always display in the Program area and is a required field. If `core.whatIf.showProgramAdditional` or `core.whatIf.mustUsePrimaryRule` are false, the Degree will also display in Additional areas of study. The values in the degree drop-down will only be those from STU307 with Show in What-If flag = Y and are on the selected curriculum rule. When `core.whatIf.degreeBeforeCollege` is true, degree will display before college and drive the college in the curriculum rule filtering. Note that the what-if audit would only include the degree block based on degree selected in the Program area, even when there's a different degree selected in Additional areas of study.

Program will display in the Program area when `core.whatIf.showProgramPrimary` is true. It will be a required field and the Level, Degree, and College (if displayed) will display but be inactive. The program may also be separately configured to display in Additional areas of study when `core.whatIf.showProgramAdditional` is true. If configured to display in Additional areas of study, when `core.whatIf.programAdditionalRequired` is true the program will be a required field. The values in the program drop-down will only be those from STU316 with Show in What-If flag = Y and are on the selected curriculum rule.

Major will display in both Areas of study and Additional areas of study when `core.whatIf.showMajor` is true. It will be a required field in Areas of study when `core.whatIf.majorRequired` is true. The values in the major drop-down will only be those from AUD027 that are on the selected curriculum rule.

Concentration will display in both Areas of study and Additional areas of study when `core.whatIf.showConcentration` is true. The concentration in Areas of study can be required if the selected major has AUD027 Concentration Required = Y. The values in the concentration drop-down will only be those from STU563 with Show in What-If flag = Y and are on the selected curriculum rule. When `core.whatIf.concTiedToMajor` is true, the concentration drop-down in Areas of study will be populated with only the concentration values on the curriculum rule where the `rad_attach_code` = MAJOR and `rad_attach_value` is equal to the selected major. When `core.whatIf.showAllConcsInPrimary` = true, if there are no concentrations attached to the selected curriculum rule in Areas of study, all values in STU563 where STU563 Show in What-If Picklist= Y will be displayed in the concentration drop-down. When `core.whatIf.showAllConcsInSecondary` = true, if there are no concentrations attached to the selected curriculum rule in Additional areas of study, all values in STU563 where STU563 Show in What-If Picklist= Y will be displayed in the concentration drop-down.

Minor will display in both Areas of study and Additional areas of study when `core.whatIf.showMinor` is true. The values in the minor drop-down will only be those from AUD029 that are on the selected curriculum rule. When `core.whatIf.showAllMinors` is true, if there are no minors attached to the selected curriculum rule in Areas of study and Additional areas of study, all values in AUD029 will be displayed in the minor drop-down.

Configuration

- Review AUD027 and AUD029 to ensure they are configured correctly.
- Review the Show in What-If flag in the following tables to ensure it is configured correctly: STU035 (Catalog year), STU307 (Degree), STU316 (Program), STU350 (Level/School), STU560 (College), STU563 (Concentration), STU576 (Campus).
- Review the following Shepherd settings to ensure they are configured correctly:
 - `core.whatIf.changeCatalogYear`
 - `core.whatIf.concTiedToMajor`
 - `core.whatIf.defaultCatalogYear`
 - `core.whatIf.degreeBeforeCollege`
 - `core.whatIf.filterMode`
 - `core.whatIf.majorRequired`
 - `core.whatIf.mustUsePrimaryRule`
 - `core.whatIf.programAdditionalRequired`
 - `core.whatIf.showAllConcsInPrimary`
 - `core.whatIf.showAllConcsInSecondary`
 - `core.whatIf.showAllMinors`
 - `core.whatIf.showCampus`
 - `core.whatIf.showCollege`
 - `core.whatIf.showConcentration`
 - `core.whatIf.showMajor`
 - `core.whatIf.showMinor`
 - `core.whatIf.showProgramAdditional`
 - `core.whatIf.showProgramPrimary`
 - `core.whatIf.showSchool`

What-If Analysis with degree-drives-major filtering

Updated: March 24, 2023

When Responsive Dashboard is configured for degree-drives-major (`core.whatIf.filterMode = D`), the Major drop-down will be populated with only the values that are valid for the selected Degree.

Catalog Year will always display and is a required field. The catalog year drop-down will be populated with all STU035 values with Show in What-If flag = Y, but the student's current catalog year will load as the default value.

By default, the catalog year in the Program area is used for all goal data items in Areas of study and Additional areas of study on the What-If audit. Users can be given the option of selecting different catalog years for individual goal data items by setting the `core.whatIf.enableGoalCatalogYear` Shepherd setting to true. When enabled, a catalog year drop-down displays adjacent to the goal data item. If a catalog year is selected, this will be passed along to the auditor engine. If left blank, the catalog year selected in the Program area will be used.

Degree will always display and is a required field. The degree drop-down will be populated with all STU307 values with Show in What-If flag = Y, but the student's current degree will load as the default value.

Level can be configured to display and is a required field if displayed. The level drop-down will be populated with all STU350 values with Show in What-If flag = Y, but the student's current level will load as the default value.

Program can be configured to display. The program drop-down will be populated with all STU316 values with Show in What-If flag = Y.

Major will always show and will be a required field regardless of the `core.whatIf.showMajor` and `core.whatIf.majorRequired` settings. The values in the major drop-down will be all AUD027 values that have the selected degree in an AUD027 Degree Filter* field.

Concentration can be configured to display based on `core.whatIf.showConcentration`. The concentration drop-down will be populated with all STU563 values with Show in What-If flag = Y. However, if `core.whatIf.filterConcentrationByMajor` is true, the concentration drop-down will be populated based on the selected major and the filter fields in STU563 and the concentration will show regardless of the `core.whatIf.showConcentration` setting and the STU563 Show in What-If flag. If the selected major is not found in any of the filter fields in STU563 then the concentration drop-down will be empty.

Minor can be configured to display. The minor drop-down will be populated with all AUD029 values.

College can be configured to display. The college drop-down will be populated with all STU560 values with Show in What-If flag = Y.

Specialization can be configured to display. The specialization drop-down will be populated with all STU323 values with Show in What-If flag = Y.

Liberal Learning can be configured to display. The liberal learning drop-down will be populated with all STU324 values with Show in What-If flag = Y.

Configuration

- Review the Degree Filter 1 – 5 fields in AUD027 to ensure they are configured correctly.

- Review the Major Filter 1 – 4 fields in STU563 to ensure they are configured correctly; only used if `core.whatIf.filterConcentrationByMajor` is enabled.
- Review the Show in What-If flag in the following tables to ensure it is configured correctly: STU035 (Catalog Year), STU307 (Degree), STU350 (Level), STU316 (Program), STU560 (College), STU563 (Concentration), STU323 (Specialization), and STU324 (Liberal learning).
- Review the following Shepherd settings to ensure they are configured correctly:
 - `core.whatIf.filterMode`
 - `core.whatIf.filterConcentrationByMajor`
 - `core.whatIf.showCollege`
 - `core.whatIf.showConcentration`
 - `core.whatIf.showLiberalLearning`
 - `core.whatIf.showMinor`
 - `core.whatIf.showProgramPrimary`
 - `core.whatIf.showSchool`
 - `core.whatIf.showSpecialization`

What-If Analysis with major-drives-degree/school/college filtering

Updated: March 24, 2023

When Responsive Dashboard is configured for major-drives-degree (`core.whatIf.filterMode=M`), the Degree, Level (aka School) and College drop-down will be populated with only the values that are valid for the selected Major.

Catalog Year will always display and is a required field. The catalog year drop-down will be populated with all STU035 values with Show in What-If flag = Y, but the student's current catalog year will load as the default value.

By default, the catalog year in the Program area is used for all goal data items in Areas of study and Additional areas of study on the What-If audit. Users can be given the option of selecting different catalog years for individual goal data items by setting the `core.whatIf.enableGoalCatalogYear` Shepherd setting to true. When enabled, a catalog year drop-down displays adjacent to the goal data item. If a catalog year is selected, this will be passed along to the auditor engine. If left blank, the catalog year selected in the Program area will be used.

Major will always show and will be a required field regardless of the `core.whatIf.showMajor` and `core.whatIf.majorRequired` settings. The major drop-down will be populated with all AUD027 values. After a major has been selected the Degree, School, and College fields on AUD027 will be used to return the appropriate values for the respective UCX tables: STU307, STU350, STU560. The Degree, Level, and College drop-downs will show as disabled.

Note that an error will display when the following conditions are encountered:

- The Degree field is blank in AUD027 for the selected major.
- The School field is blank in AUD027 for the selected major and if `core.whatIf.showSchool= true`.
- The College field is blank in AUD027 for the selected major and if `core.whatIf.showCollege= true`.

Level can be configured to display based on `core.whatIf.showSchool`. If the `core.whatIf.showSchool` setting is false the Level drop-down will not display and the student's current level will be used when running the what-if audit.

College can be configured to display based on `core.whatIf.showCollege`. If the `core.whatIf.showCollege` setting is false the College drop-down will not display and the college configured in AUD027 for the selected major will be ignored. If the `core.whatIf.showCollege` setting is true college will be a required field.

Concentration can be configured to display based on `core.whatIf.showConcentration`. The concentration drop-down will be populated with all STU563 values with Show in What-If flag = Y. However, if `core.whatIf.filterConcentrationByMajor` is true the concentration drop-down will be populated based on the selected major and the filter fields in STU563 and the concentration will show regardless of the `core.whatIf.showConcentration` setting and the STU563 Show in What-If flag. If the selected major is not found in any of the filter fields in STU563 then the concentration drop-down will be empty.

Program can be configured to display. The program drop-down will be populated with all STU316 values with Show in What-If flag = Y.

Minor can be configured to display. The minor drop-down will be populated with all AUD029 values.

Specialization can be configured to display. The specialization drop-down will be populated with all STU323 values with Show in What-If flag = Y.

Liberal Learning can be configured to display. The liberal learning drop-down will be populated with all STU324 values with Show in What-If flag = Y.

Configuration

- Review the Degree, School, and College fields in AUD027 to ensure they are configured correctly.
- Review the Major Filter 1 – 4 fields in STU563 to ensure they are configured correctly; only used if `core.whatIf.filterConcentrationByMajor` is enabled.
- Review the Show in What-If flag in the following tables to ensure it is configured correctly: STU035 (Catalog Year), STU307 (Degree), STU350 (Level), STU316 (Program), STU560 (College), STU563 (Concentration), STU323 (Specialization), and STU324 (Liberal learning).
- Review the following Shepherd settings to ensure they are configured correctly:
 - `core.whatIf.filterMode`
 - `core.whatIf.filterConcentrationByMajor`

- `core.whatIf.showCollege`
- `core.whatIf.showConcentration`
- `core.whatIf.showLiberalLearning`
- `core.whatIf.showMinor`
- `core.whatIf.showProgramPrimary`
- `core.whatIf.showSchool`
- `core.whatIf.showSpecialization`

What-If Analysis with major-drives-college filtering

Updated: March 24, 2023

When Responsive Dashboard is configured for major-drives-college (`core.whatIf.filterMode = C`), the College drop-down will be populated with only the values that are valid for the selected Major.

Catalog Year will always display and is a required field. The catalog year drop-down will be populated with all STU035 values with Show in What-If flag = Y, but the student's current catalog year will load as the default value.

By default, the catalog year in the Program area is used for all goal data items in Areas of study and Additional areas of study on the What-If audit. Users can be given the option of selecting different catalog years for individual goal data items by setting the `core.whatIf.enableGoalCatalogYear` Shepherd setting to true. When enabled, a catalog year drop-down displays adjacent to the goal data item. If a catalog year is selected, this will be passed along to the auditor engine. If left blank, the catalog year selected in the Program area will be used.

Degree will always display and is a required field. The degree drop-down will be populated with all STU307 values with Show in What-If flag = Y, but the student's current degree code will load as the default value.

Level can be configured to display based on `core.whatIf.showSchool`. The level drop-down will be populated with all STU350 values with Show in What-If flag = Y. If `core.whatIf.showSchool` is false the Level drop-down will not display and the student's current level will be used when running the what-if audit.

Major will always show and will be a required field regardless of the `core.whatIf.showMajor` and `core.whatIf.majorRequired` settings. The major drop-down will be populated with all AUD027 values. After a major has been selected the College fields on AUD027 will be used to return the appropriate values for the STU560. The College drop-down will show as disabled.

Note that an error will display if the College field is blank in AUD027 for the selected major.

College will always show and will be a required field regardless of the `core.whatIf.showCollege` setting.

Concentration can be configured to display based on `core.whatIf.showConcentration`. The concentration drop-down will be populated with all STU563 values with Show in What-If flag = Y.

However, if `core.whatIf.filterConcentrationByMajor` is true the concentration drop-down will be populated based on the selected major and the filter fields in STU563 and the concentration will show regardless of the `core.whatIf.showConcentration` setting and the STU563 Show in What-If flag. If the selected major is not found in any of the filter fields in STU563 then the concentration drop-down will be empty.

Program can be configured to display. The program drop-down will be populated with all STU316 values with Show in What-If flag = Y.

Minor can be configured to display. The minor drop-down will be populated with all AUD029 values.

Specialization can be configured to display. The specialization drop-down will be populated with all STU323 values with Show in What-If flag = Y.

Liberal Learning can be configured to display. The liberal learning drop-down will be populated with all STU324 values with Show in What-If flag = Y.

Configuration

- Review the College fields in AUD027 to ensure they are configured correctly.
- Review the Major Filter 1 – 4 fields in STU563 to ensure they are configured correctly; only used if `core.whatIf.filterConcentrationByMajor` is enabled.
- Review the Show in What-If flag in the following tables to ensure it is configured correctly: STU035 (Catalog Year), STU307 (Degree), STU350 (Level), STU316 (Program), STU560 (College), STU563 (Concentration), STU323 (Specialization), and STU324 (Liberal learning).
- Review the following Shepherd settings to ensure they are configured correctly:
 - `core.whatIf.filterMode`
 - `core.whatIf.filterConcentrationByMajor`
 - `core.whatIf.showConcentration`
 - `core.whatIf.showLiberalLearning`
 - `core.whatIf.showMinor`
 - `core.whatIf.showProgramPrimary`
 - `core.whatIf.showSchool`
 - `core.whatIf.showSpecialization`

Future classes

Updated: March 25, 2022

The user can also add courses that may be taken in the future to see how they might apply to the student's current program or a program they may be considering.

A course Subject and course Number must be entered. Note that there is no validation on these fields so care should be taken to enter them correctly. Click **Add** to add the course to the What-If Analysis criteria. An unlimited number of courses can be added.

If the user wants to see how the future classes will apply to their current program, they can check Use current curriculum. The Program and Areas of study sections will collapse and the student's current goal data will be used to generate the what-if audit with the manually added future classes.

Historic audits

Updated: March 25, 2022

View historic what-if audits allows a user to view historical What-If audits for a student.

The maximum number of historical What-If audits to keep per student per School/Degree combination can be configured. If the user has access to view historic What-If audits, they can select the audit they want to view from the drop-down and it will load using the currently selected report format.

Configuration

- Review the Maximum What-If audits per student flag in CFG020 DAP14 to ensure it is configured correctly.
- Access to view historic audits is granted if the user has the SDWIFHIS Shepherd key.

Save audit

Updated: March 25, 2022

Audits may be saved or “frozen” so they do not get deleted when the CFG020 DAP14 Maximum What-If audits per student value has been reached.

Frozen What-If audits do not count against the What-If history count value. Depending on the user's access, when they click Save audit a dialog will open that prompts for the Freeze type and Description. When the user clicks Save, the What-If audit will be saved in the database with the specified freeze type and description, and the screen will be updated with the user who saved the audit, the date, the freeze type and description. For more information, see the [Freeze audits](#) topic.

Configuration

Access to the following is granted if the user has the specified Shepherd key:

- Freeze What-If audits - WIFFREEZ
- Add descriptions - WIFDESCR

Delete audit

Updated: March 25, 2022

What-If audits that have been saved or frozen can be deleted if the user has access to do so.

Clicking the Delete audit link opens a confirmation dialog. If the user clicks Delete, the What-If audit will be deleted from the database. If the user clicks Cancel, they will be returned to Worksheets without deleting the audit.

Configuration

Access to delete audits is granted if the user has the SDWIFDEL Shepherd key.

Athletic Eligibility

Updated: March 25, 2022

Athletic Eligibility audits allow you to process degree audits specifically for your student athletes.

Rules for Athletic Eligibility are built using Scribe, and processing an athlete's athletic data against these requirement blocks will verify if a student is eligible to participate in the sport, in accordance with rules laid out by the NCAA, NJCAA, and other organizations.

ATHLETE blocks house athletic eligibility requirements. You may create a single ATHLETE block or create multiple to separate out the different eligibility rules. The ATHLETE block values are controlled by AUD031. The Athletic Eligibility Scribe templates can help you quickly and easily create your ATHLETE blocks. For more information, see [Athletic Eligibility audit rule examples](#).

When running an Athletic Eligibility audit, the In-progress check box should always be checked as the current classes are required and the Preregistered check box should always be unchecked - future classes are never wanted for an athletic eligibility audit. Athletic Eligibility history follows the same CFG020 DAP14 Audit History Depth setting as academic audits when saving audits – if the depth is set to 3, you will have 3 academic audits and 3 athletic audits. However, you may choose to save an unlimited number of frozen audits as they are not impacted by this setting. When an Athletic Eligibility audit is run, all ATHLETE blocks are used. The catalog year and other secondary tags are checked. The ATHLETE blocks appear on the worksheet sorted by their block titles. You can alter the block title to get the blocks in the order you want them to appear. The previous term is set to term before the active term when the active term is a summer term. When you have chosen to include in-progress classes in the audit the previous term is set to the active term; otherwise the previous term is the term before the active term. Including in-progress classes allows you to forecast for future eligibility.

The degree progress on the Athletic Eligibility audit is directly related to the calculations performed for the 40-60-80 rule and accounts for elective credits allowed/needed. The data feeding this progress bar is specifically calculated for athletic eligibility and therefore differs from the data feeding the progress bar on the academic audit worksheet.

The top of the Diagnostics Report can be very helpful in understanding why rules in your ATHLETE blocks were marked as complete or as unsatisfied. The “type” of data is listed on the far left of this grid. Those types in all UPPERCASE letters indicate those that are directly related to a Scribe reserved word you may use in an IF-statement. Those in MixedCase are used in internal calculations and appear here merely to assist you in understanding the data.

- Credits – this field contains the credits value for this type
- Text – describes the data for this type

- Term – the term value used in this calculation
- TermType – the type of term as specified on UCX-STU016

Athletic Eligibility	TotalCreditsAttempted=043.000	CumGradePoints=040.000	CumGPA=003.549	TotalCreditsEarned=042.000	ResidenceCreditsEarned=042.000	CompletedTermCount=5
	LastCompletedTermType=	ResidenceCompletedTermCount=5	Probation=	ActiveTerm=201010	PreviousTerm=201010	ActiveTermLit=First Term 2 PreviousTermLit=First Term
ECA Overall	Credits=45	Text=Credits required by degree block				
ECA BlocksRequired		Text=				
ECA BlocksNotRequired		Text=				
ECA Blocksum	Credits=0	Text=Sum of credits required in all "required" blocks				
ECA Shared	Credits=0	Text=Credits shared between required blocks				
ECA ECA	Credits=45	Text=Credits allowed/needed in fall-through and non-required blocks				
ECA RequiredCreditsApplied	Credits=0	Text=Credits applied to required blocks				
ECA NonrequiredCreditsApplied	Credits=6	Text=Credits applied to non-required blocks				
ECA Fallthru Overflow	Credits=14	Text=Excess fall-through credits				
ECA NonRequired Overflow	Credits=0	Text=Excess non-required credits				
ECA Overflow	Credits=14	Text=Excess fall-through and non-required credits				
CREDITSAPPLIEDTOWARDSDEGREE	Credits=45	Text=Credits applied minus overflow				
FullTimeCredits	Credits=12	Text=Full-time credits from UCX-STU350				
Year1 TermType		Text=Type of term in this year			Term=200510 (BAN WNTR2005)	TermType=FALL
Year1 TermType		Text=Type of term in this year			Term=200620 (BAN SPRG2006)	TermType=SPRING
Year2 TermType		Text=Type of term in this year			Term=200810 (BAN WNTR2008)	TermType=FALL
Year2 TermType		Text=Type of term in this year			Term=200820 (Spring 20082)	TermType=SPRING
FirstTermOfThirdYear		Text=First term of 3rd year			Term=200910 (Fall 2008)	
FirstYear TermType	Credits=6	Text=Type of term in 1st year			Term=200510 (BAN WNTR2005)	TermType=FALL
FirstYear TermType	Credits=6	Text=Type of term in 1st year			Term=200620 (BAN SPRG2006)	TermType=SPRING
FIRSTYEARLEARNEDCREDITS	Credits=12	Text=Credits for first full year starting this term			Term=200510 (BAN WNTR2005)	TermType=FALL
FirstYearRemedialCredits	Credits=0	Text=Remedial credits earned in first full year				
PREVIOUSTERMSEARNEDCREDITS	Credits=6	Text=This is the previous full-time term credits			Term=200910 (Fall 2008)	
PreviousPartTimeTermEarnedCredits	Credits=0	Text=The previous term is part-time			Term=201010 (First Term 2)	
PREVIOUS2TERMSEARNEDCREDITS	Credits=0	Text=This is the total for the previous 2 terms (PT or FT); this is the 2nd prev term			Term=200920 (Spring 2009)	
PREVIOUS3TERMSEARNEDCREDITS	Credits=0	Text=This is the total for the previous 3 terms (PT or FT); this is the 3rd prev term			Term=200915 (Winter 2009)	
PREVIOUSACADEMICYEAREARNEDCREDITS	Credits=6	Text=Credits for academic year ending this spring term			Term=200920 (Spring 2009)	

In addition to the data above, the Diagnostics Report for athletic audits also contains a summary of classes by term. When applicable, a notation next to each term appears to connect the information above to a specific term.

Spring 20082 (200820)				
FUTURE	101	Intro to Statistics	A	3
FUTURE	102	Intro to Statistics	A	3
HISTORIC	101	Intro to Statistics	A	3
Transferred from - University of Dublin				
HISTORIC	102	Intro to Statistics	A	3
Transferred from - Deans College				
IMPROG	102	Intro to Statistics	A	3
Fall 2008 (200910) [Previous full-time term] [First term of 3rd year]				
ACCT	021	Intro to Accounting	B	3
ACCT	022	Intro to Accounting	B	3
ACCT	223	Intro to Accounting	B	3
ACCT	224	Intro to Accounting II	B	3
Winter 2009 (200915) [Previous 3rd term]				
SPAN	001	Intro to Spanish	B	2
SPAN	102	Intro to Spanish II	B	2
Spring 2009 (200920) [Previous 2nd term]				
BUS	021	Intro to Statistics	B	2
MATH	222	Intro to Statistics	B	2
MTH	333	Intro to Finite Math	B	2
First Term 2 (201010) [Previous term]				
ART	100	Intro to Art	B	2

Configuration

- Review BAN080 to ensure the required Athletic Eligibility records exist. For more information, see [Athletic Eligibility audit data setup](#).
- Review AUD031 - Athletic Eligibility types

- Review RPT036 to ensure it is configured correctly for WEB55 and WEB56.
- Review SCR002 to ensure the required Athletic Eligibility records exist. For more information, see [Athletic Eligibility audit data setup](#).
- Review STU016 to ensure that the Term Type field is set to "SUMMER" for summer terms, "FALL" for fall terms, "SPRING" for spring terms and "WINTER" for winter terms.
- Review STU350 to ensure that the Full-time Credits field is populated for each school
- Review the `dash.audit.athletic.football.enabled` Shepherd setting to ensure it is configured correctly.
- Access to the following is granted if the user has the specified Shepherd key:
 - Athletic Eligibility - SDATHAUD
 - Athletic Eligibility Report - SDWEB55
 - Athletic and Academic Report - SDWEB56
 - Delete audits - SDATHDEL
 - View athletic eligibility audits - SDATHREV
 - View historic athletic eligibility audits - SDATHHIS
 - Process athletic eligibility audits - SDATHRUN

Athletic Eligibility audit data setup

Updated: March 25, 2022

The Athletic Eligibility audit requires specific data to be bridged into the Degree Works database.

At least four values must be bridged to the `rad_custom_dtl`:

- ACADSTANDING – indicates if the student is in good academic standing.
- AEAFIRSTTERM – first official term that can be counted; mapped from first date of attendance; this date is stored on SGRATHE for Banner schools.
- AEAFIRSTDATE – first official date of attendance; this date is stored on SGRATHE for Banner schools.
- SPORT – one or more participating sports; this value is stored on SGRSPRT for Banner schools. Multiple SPORT values can be bridged if the student is participating in multiple sports.

AEASTATUS should be bridged if you would like to display the Football 9 HR/APR E Point value. See examples below for configuring BAN080 to extract this data. If you are a non-Banner site and use RAD11 to bridge student information, you need to alter your extract to obtain information about your student athletes into the `rad_custom_dtl`.

Two values should be bridged to the `rad_report_dtl`. If they are not found, the corresponding values from the `rad_custom_dtl` will be used for the worksheet display.

- ACADSTANDESC – description - indicates if the student is in good academic standing.
- SPORTDESC – one or more participating sport descriptions; this value is stored on SGRSPRT for Banner schools. Multiple SPORTDESC values can be bridged if the student is participating in multiple sports.

You need to bridge each student's Academic Standing description to the rad_report_dtl with a code of ACADSTANDESC and the Academic Standing value to the rad_custom_dtl with a code of ACADSTANDING. The description will appear in the Athletic eligibility details section of the worksheet, while the value can be used in any IF-statement that you may have scribed. If ACADSTANDESC description is not found, the worksheet will use the ACADSTANDING code on the rad_custom_dtl.

You also need to bridge each student's participating Sport description to the rad_report_dtl with a code of SPORTDESC and the Sport value to the rad_custom_dtl with a code of SPORT. The description will appear in the Athletic eligibility details section of the worksheet, while the value can be used in any IF-statement that you may have scribed. If the student is participating in more than one sport activity, bridge each sport as a different rad_custom_dtl and rad_report_dtl record. If SPORTDESC description is not found, the worksheet will use the SPORT code on the rad_custom_dtl.

SCR002 setup

Create a SPORT record that looks similar to the one shown in the screenshot that follows. The SPORT record is needed if you will be scribing against the sport code. It is also needed to gather the data for display on the worksheet.

SCR002 Custom Data

Custom Data Item* SPORT	
Description Sport code for Athletic Elig.	Data Element R323
Edit Element 1 R322	Edit Type 1 Use Edit Value 1
Edit Value 1 SPORT	Edit Element 2
Edit Type 2	Edit Value 2
Edit Element 3	Edit Type 3
Edit Value 3	

Create an ACADSTANDING record that looks similar to the one shown in the screenshot that follows. Likely you will be scribing against the academic standing and will probably need this

record. This too is displayed on the worksheet. Ensure the Value field has the ending G, which is hidden below.

SCR002 Custom Data

Custom Data Item* ACADSTANDING	
Description Academic Standing	Data Element R323
Edit Element 1 R322	Edit Type 1 Use Edit Value 1
Edit Value 1 ACADSTANDING	Edit Element 2
Edit Type 2	Edit Value 2
Edit Element 3	Edit Type 3
Edit Value 3	

Create an AEAFIRSTTERM record that looks similar to the one shown in the screenshot that follows. Likely you will not be scribing against this value, but this record is still needed so the value can be passed to the auditor to be used when calculating the term count. Ensure the Value field has the ending M, which is hidden below.

SCR002 Custom Data

Custom Data Item # AEAFIRSTTERM	
Description AEA: First official term	Data Element R323
Edit Element 1 R322	Edit Type 1 Use Edit Value 1
Edit Value 1 AEAFIRSTTERM	Edit Element 2
Edit Type 2	Edit Value 2
Edit Element 3	Edit Type 3
Edit Value 3	

Create an AEAFIRSTDATE record that looks similar to the one shown in the screenshot that follows. Likely you will not be scribbling against this value, but this record is still needed so the value can be passed to the worksheet for display. Ensure the Value field has the ending E, which is hidden below.

SCR002 Custom Data

Custom Data Item # AEAFIRSTDATE	
Description AEA: First date of attendance	Data Element R323
Edit Element 1 R322	Edit Type 1 Use Edit Value 1
Edit Value 1 AEAFIRSTDATE	Edit Element 2
Edit Type 2	Edit Value 2
Edit Element 3	Edit Type 3
Edit Value 3	

Create an AEASTATUS record that looks similar to the one shown in the screenshot that follows. Likely you will not be scribing against this value, but this record is still needed so the value can be passed to the worksheet for display. If you do not want to show the Football 9 HR/APR E Point value on the report, you do not need to add this record.

SCR002 Custom Data

Custom Data Item * AEASTATUS	
Description Status from student-mst	Data Element 2407
Edit Element 1	Edit Type 1 ▼
Edit Value 1	Edit Element 2
Edit Type 2 ▼	Edit Value 2
Edit Element 3	Edit Type 3 ▼
Edit Value 3	

BAN080 setup

Suggested entries for BAN080 ACADSTANDING:

BAN080ACADSTANDING: COLUMN	SHRTTRM_ASTD_CODE_END_OF_TERM
BAN080ACADSTANDING: ORDERBY	SHRTTRM_ASTD_CODE_END_OF_TERM
BAN080ACADSTANDING: TABLE	SHRTTRM α
BAN080ACADSTANDING: WHERE_1	α.SHRTTRM_TERM_CODE =
BAN080ACADSTANDING: WHERE_2	(SELECT MAX(b.SHRTTRM_TERM_CODE)
BAN080ACADSTANDING: WHERE_3	FROM SHRTTRM b
BAN080ACADSTANDING: WHERE_4	WHERE b.SHRTTRM_PIDM = α.SHRTTR
M_PIDM)	

Suggested entries for BAN080 SPORT:

BAN080SPORT: COLUMN	SGRSPRT_ACTC_CODE
BAN080SPORT: ORDERBY	SGRSPRT_ACTC_CODE
BAN080SPORT: TABLE	SGRSPRT α
BAN080SPORT: WHERE_1	α.SGRSPRT_SPST_CODE = 'AC'

These ACADSTANDING and SPORT records (along with those built in SCR002) allow you to Scribe an IF-statement against these codes. To get the Academic Standing description and the Sport description to appear on the report, you should create additional BAN080 entries to pull the corresponding descriptions from the Banner STV tables. The Athletic Eligibility worksheet will look

for ACADSTANDESC and SPORTDESC report entries and will display them if found. If they are not found, the corresponding ACADSTANDING and SPORT custom entries will be used.

BAN080ACADSTANDESC:REPORT	STVASTD
BAN080ACADSTANDESC:COLUMN	SHRTTRM_ASTD_CODE_END_OF_TERM
BAN080ACADSTANDESC:ORDERBY	SHRTTRM_ASTD_CODE_END_OF_TERM
BAN080ACADSTANDESC:TABLE	SHRTTRM α
BAN080ACADSTANDESC:WHERE_1	α.SHRTTRM_TERM_CODE =
BAN080ACADSTANDESC:WHERE_2	(SELECT MAX(b.SHRTTRM_TERM_CODE)
BAN080ACADSTANDESC:WHERE_3	FROM SHRTTRM b
BAN080ACADSTANDESC:WHERE_4	WHERE b.SHRTTRM_PIDM = α.SHRTTR
M_PIDM)	
 BAN080SPORTDESC:REPORT	 STVACTC
BAN080SPORTDESC:COLUMN	SGRSPRT_ACTC_CODE
BAN080SPORTDESC:ORDERBY	SGRSPRT_ACTC_CODE
BAN080SPORTDESC:TABLE	SGRSPRT α
BAN080SPORTDESC:WHERE_1	α.SGRSPRT_SPST_CODE = 'AC'

Ensure that you grant access to these tables for the Degree Works Banner database user.

Athletic Eligibility audit rule examples

Examples of Athletic Eligibility-specific RULE Scribe requirements

6/18/24 Rule

```
#-- 6/18/24 Rule

#-- 6 credits earned in the previous term
if (PreviousTermEarnedCredits >= 6 ) then
    RuleComplete
    Label 6a "Previous Term - at least 6 credits earned required"
else
    RuleIncomplete
    ProxyAdvice "You did not earn 6 credits your previous term"
    Label 6b "Previous Term - at least 6 credits earned required";

#-- 18 credits earned in the previous year
# for Semester schools
if (CompletedTermCount <= 1) then
    RuleComplete
    Label 18a "2 terms have not yet been completed"
else if (Previous2TermsEarnedCredits >= 18 or
         PreviousAcademicYearEarnedcredits >= 18) then
    RuleComplete
    Label 18b "Previous Year: at least 18 credits earned required"
else
    RuleIncomplete
    ProxyAdvice "You did not earn 18 credits your previous year"
    Label 18c "Previous Year: at least 18 credits earned required";
```

```
# for Quarter schools
if (CompletedTermCount <= 2) then
  RuleComplete
  Label 18e "3 terms have not yet been completed"
Else if (Previous3TermsEarnedCredits >= 27 or
  PreviousAcademicYearEarnedcredits >= 27) then
  RuleComplete
  Label 18f "Previous Year: at least 27 credits earned required"
else
  RuleIncomplete
  ProxyAdvice "You did not earn 27 credits your previous year"
  Label 18g "Previous Year: at least 27 credits earned required";

#-- 24 credits earned in the first year
if (FirstYearEarnedCredits >= 24 ) then
  RuleComplete
  Label 24a "First Year: at least 24 credits earned required"
else
  RuleIncomplete
  ProxyAdvice "You did not earn at least 18 credits your First Year"
  Label 24b "First Year: at least 24 credits earned required";
```

40/60/80 Rule

```
#-- 40/60/80 Rule

#-- 80% complete with course requirements by start of 5th year
if (CompletedTermCount >= 8 And
  CreditsAppliedTowardsDegree >= 80% of DegreeCreditsRequired) then
  RuleComplete
  Label 80a "80% complete by start of 5th year"
else if (CompletedTermCount >= 8) then
  RuleIncomplete
  ProxyAdvice "You did not complete 80% by the start of the 5th year"
  Label 80b "80% complete by start of 5th year"

#-- 60% complete with course requirements by start of 4th year
Else if (CompletedTermCount >= 6 And
  CreditsAppliedTowardsDegree >= 60% of DegreeCreditsRequired) then
  RuleComplete
  Label 60a "60% complete by start of 4th year"
else if (CompletedTermCount >= 6) then
  RuleIncomplete
  ProxyAdvice "You did not complete 60% by the start of the 4th year"
  Label 60b "60% complete by start of 4th year"

#-- 40% complete with course requirements by start of 3rd year
Else if (CompletedTermCount >= 4 And
  CreditsAppliedTowardsDegree >= 40% of DegreeCreditsRequired) then
```



```

n
  RuleComplete
  Label 40a "40% complete by start of 3rd year"
else if (CompletedTermCount >= 4) then
  RuleIncomplete
  ProxyAdvice "You did not complete 40% by the start of the 3rd year"
"

  Label 40b "40% complete by start of 3rd year"
else # not yet started 3rd year
  RuleIncomplete
  ProxyAdvice "By the start of your 3rd year you need to be 40% complete"
Label 3rd "3rd year not started ";

```

90/95/100 GPA Rule

```

# 90/95/100 GPA Rule

#-- Entering 2nd year - GPA must be at least 90% of 2.0
If (CompletedTermCount <= 3 and SISGPA >= 1.8) then
  RuleComplete
  Label GPA-1 "Entering 2nd year - GPA must be at least 1.8"
Else If (CompletedTermCount = 3 and SISGPA < 1.8) then
  RuleIncomplete
  ProxyAdvice "Your GPA is less than 1.8 - you are ineligible"
  Label GPA-2 "Entering 2nd year - GPA must be at least 1.8"
Else If (CompletedTermCount < 3 and SISGPA < 1.8) then
  RuleIncomplete
  ProxyAdvice "Your GPA needs to be at least 1.8 by the start "
  ProxyAdvice "of your 3rd term."
  Label GPA-3 "Entering 2nd year - GPA must be at least 1.8"

#-- Entering 3rd year - GPA must be at least 95% of 2.0
Else If (CompletedTermCount <= 5 and SISGPA >= 1.9) then
  RuleComplete
  Label GPA-4 "Entering 3rd year - GPA must be at least 1.9"
Else If (CompletedTermCount <= 5 and SISGPA < 1.9) then
  RuleIncomplete
  ProxyAdvice "Your GPA is less than 1.9 - you are ineligible"
  Label GPA-5 "Entering 3rd year - GPA must be at least 1.9"

#-- Entering 4th year - GPA must be at least 2.0
#-- And beyond 4th year also
Else If (CompletedTermCount >= 6 and SISGPA >= 2.0) then
  RuleComplete
  Label GPA-6 "4th year and beyond - GPA must be at least 2.0"
Else If (CompletedTermCount >= 6 and SISGPA < 2.0) then
  RuleIncomplete
  ProxyAdvice "Your GPA is less than 2.0 - you are ineligible"
  Label GPA-7 "4th year and beyond - GPA must be at least 2.0";

```

Declared Degree by Third Year Rule

```

#-- Before 3rd year - degree/major must be declared
If (NumMajors >= 1) then # regardless of how many terms have been c

```

```

ompleted
  RuleComplete
  Label Major1 "Declare a degree/major by 3rd year"
else if (CompletedTermCount < 5) then
  RuleIncomplete
  ProxyAdvice "You need to declare a degree/major by your Third Year"
"
  Label Major2 "Declare a degree/major by 3rd year"
else # student has already started her 3rd year
  RuleIncomplete
  ProxyAdvice "You did not declare a degree/major by your Third Year"
"
  Label Major3 "Declare a degree/major by 3rd year";

```

Academic Standing Rule

```

# Student must be in good academic standing
if (AcadStanding = "GOOD") then
  RuleComplete
  Label 1 "You must be in good academic standing"
else # not so good
  RuleIncomplete
  Proxyadvice "You are currently not in good academic standing"
  Label 2 "You must be in good academic standing";

```

Football Rule

```

##### Football rule - 9 credits in prior fall season
Begin
;
Remark "Football players must earn 9 credits in the prior fall season."
Remark "You can regain eligibility if you earn 27 credits before the ";
Remark "next fall season (includes summer term credits). ";

# If the student does not have FOOTBALL as a SPORT this block will be
# hidden from the worksheet.

if (SPORT=FOOTBALL) then
  BeginSub
    if (PreviousTermEarnedCredits-Fall < 9) then
      RuleIncomplete
      ProxyAdvice "You did not earn at least 9 credits in your last fall term"
      Label "Earned less than 9 credits last fall"

    else # if (PreviousTermEarnedCredits-Fall >= 9) then
      RuleComplete
      Label "Earned at least 9 credits last fall";

  # ResidenceCompletedTermCount does not count summer terms
  # FirstYearEarnedCredits includes prior and ending summer terms
  # PreviousFullYearEarnedCredits includes ending summer

```

```

if (PreviousTermEarnedCredits-Fall < 9) then
  # If a 1st year student and not enough credits
  if (ResidenceCompletedTermCount < 3 and
      FirstYearEarnedCredits < 27) then
    RuleIncomplete
    ProxyAdvice "You earned less than 27 credits and thus did not"
    ProxyAdvice "regain eligibility"
    Label "1st year: less than 27 credits"
  # If a 1st year student and has enough credits
  else If (ResidenceCompletedTermCount < 3 and
      FirstYearEarnedCredits >= 27) then
    RuleComplete
    Label "1st year: at least 27 credits; regained eligibility"
  # If past 1st year and not enough credits
  else If (ResidenceCompletedTermCount >= 3 and
      PreviousFullYearEarnedCredits < 27) then
    RuleIncomplete
    ProxyAdvice "You earned less than 27 credits and thus did not"
    ProxyAdvice "regain eligibility"
    Label "Less than 27 credits this past year"
  # If past 1st year and has enough credits
  else If (ResidenceCompletedTermCount >= 3 and
      PreviousFullYearEarnedCredits >= 27) then
    RuleComplete
    Label "27 credits in the last year; regained eligibility"

  else # just in case
    RuleIncomplete
    ProxyAdvice "Unexpected situation; need to review 27 credit re
quirement"
    Label "27 credits check; should never see this";

EndSub
Label "Football: 9 credits in fall";

# Quarter schools should change 9 to 8 above
# Quarter schools should change 27 to 40 above
# Quarter schools may also want to change the term count

End.

# These may also be useful...
# if (DegreeCreditsRequired >= 120) then
# if (CreditsAppliedTowardsDegree >= 48 ) then

```

Athletic Eligibility audit Scribe reserved words

Updated: March 25, 2022

These Scribe reserved words are allowed in an IF statement in an Athletic Eligibility audit, usually in an ATHLETE block.

	Alias	Data Source
CompletedTermCount	TermCount	Includes the current, in-progress term; starting at the first full-time term; either in residence or as a transfer; does not count summer terms
TotalCreditsEarned	TotalCreditsCompleted	rad_term_dtl.rad_cum_tot_earn
ResidenceCreditsEarned	ResidenceCreditsCompleted	rad_term_dtl.rad_cum_cr_earn
TotalCreditsAttempted	CreditsAttempted	rad_term_dtl.rad_cum_gr_att
ResidenceCompletedTermCount		Count the terms in residence – starting at the first full-time term. Does not include summer terms.
DegreeCreditsRequired	CreditsRequired	Credits required from starting block - allowed on right-hand-side of an IF-statement in addition to the left-hand side
CreditsAppliedTowardsDegree		Total credits applied towards degree completion. Takes into account the special elective credits allowed/needed discarded the excess fall-through credits.
PreviousTermEarnedCredits		After 4 th full-time term these must be degree applicable credits. Previous term must be full-time but can be the in-progress term. Does not include summer terms.
PreviousTermEarnedCredits-Fall		Earned credits for the previous fall term. The Football rule dictates that the student must have earned 9 semester credits in the previous fall term.
Previous2TermsEarnedCredits		After 4 th full-time term these must be degree applicable credits. Previous 2 terms can include the current in-progress term – and both can be part-time or full-time terms. Does not include summer terms. Per NCAA proposal 2018-68, both the ECA okay and

	Alias	Data Source
		overflow credits are included in this calculation.
Previous3TermsEarnedCredits		<p>After 4th full-time term these must be degree applicable credits. Previous 3 terms can include the current in-progress term – and all can be part-time or full-time terms. Does not include summer terms.</p> <p>Per NCAA proposal 2018-68, both the ECA okay and overflow credits are included in this calculation.</p>
PreviousAcademicYearEarnedCredits		<p>After 4th full-time term these must be degree applicable credits. Does not include summer terms.</p> <p>Per NCAA proposal 2018-68, both the ECA okay and overflow credits are included in this calculation.</p>
PreviousFullYearEarnedCredits		<p>Includes fall, winter, spring and ending summer term credits. Needed for the Football rule.</p> <p>Per NCAA proposal 2018-68, both the ECA okay and overflow credits are included in this calculation.</p>
FirstYearEarnedCredits		Credits earned in the first full academic year. If first term is summer it is counted. The summer ending term of that year is also counted.

Athletic Eligibility audit special topics

Updated: March 25, 2022

There are several calculations and evaluations specific to determining athletic eligibility.

CreditsAppliedTowardsDegree calculation

Updated: March 25, 2022

The Credits Applied Towards Degree calculation is used to support the 40/60/80 rule and present an accurate progress bar on the Athletic Eligibility audit worksheet.

The Credits Applied Towards Degree calculation does not count any credits placed into the Over-The-Limit section (also referred to as Not Counted) but when counting Fall Through (also referred to as Electives) credits, the calculation is more complicated. For most schools, the Gen Ed and Major blocks do not account for all of the credits the student needs to take to graduate. In most cases the student does have to take (non-major) elective credits that appear in the Fall Through section of the audit. However, while some elective credits are required, the student should not take more than that which is required. It is these “excess” elective credits that cannot be counted as Credits Applied Towards Degree.

If credits applied to a block are greater than the credits required, we use the credits applied. For example, if the major block requires 50 credits but it contains several credit range rules (for example, 6:8 Credits in ...), it is possible the student will actually apply more than 50 credits to the major. These extra credits applied to the major should be subtracted from the elective (fall-through) credits the student must/can take, known as Elective Credits Allowed or ECA.

If credits are shared between two blocks then those credits are added to the total elective (fall-through) credits the student must/can take. For example, if 7 credits are shared between the major and Gen Ed blocks, the student must then apply 7 additional credits towards the degree, and those additional credits must be taken as elective (fall-through) classes.

To correctly calculate how many elective credits are allowed/needed for the degree, each block must have a block header credits qualifier. In some cases, however, you may not want to specify a fixed value – perhaps because of several credit range requirements or because of variable credit classes. In this situation you may use the Pseudo keyword to essentially hide this requirement from the user. For example:

31 Credits Pseudo # informational only – not checked by the auditor

The auditor ignores this qualifier when checking to see if the block is complete but will use this credit value when figuring out how many electives are needed to satisfy the overall degree credits. If you have a block that does not represent any credits you can use "0:1 Credits Pseudo."

The credits for a block containing the Optional header qualifier are not included in the calculation.

Example 1

A student needs 120 credits to graduate with 98 (50 + 30 + 18) credits coming from required blocks. This means the student must take 22 elective (fall-through) credits to meet the 120 required by the degree. However, if the student has more than 22 credits in fall-through the auditor will not count those “excess” credits giving a better measure of how close the student is to reaching the 120 credit goal.

Block	Hours
Degree Block	120 hrs required to graduate
General Education Block	50 hrs
Major Block	30 hrs
Conc Block (required by major)	18 hrs

Example 2

A student needs 120 credits to graduate with 100 (50 + 50) credits coming from required blocks; the Minor was added by the student but is not required by the degree or by the major. This means the student must take 20 elective credits to meet the 120 required by the degree. However, because the Minor block is not required any classes applying to the Minor are considered as “elective” credits for this calculation – along with any appearing in fall-through. As soon as the student applies more than 20 credits to the Minor and fall-through the extra credits are considered “excess” and are not counted towards degree completion. So if the student has 25 credits applying to the Minor and 10 showing in fall-through only 20 of these 35 credits will be counted.

Block	Hours
Degree Block	120 hrs required to graduate
General Education Block	50 hrs required
Major Block	50 hrs required
Minor Block (voluntary)	30 hrs required

Example 3

A student also needs 120 credits to graduate but 5 credits are being shared between two required blocks.

Block	Hours
Degree Block	120 hrs required to graduate
General Education Block	50 hrs
Major Block	30 hrs
Minor Block (required)	20 hrs

Because 5 credits are shared between major and minor the student must take 25 elective credits instead of only 20.

Example 4

A student needs 120 credits to graduate with 100 (50 + 50) credits coming from required blocks. The Conc is included (called in) by the Major and so the Conc credits are not counted twice. The Major block's credits required and credits applied values include those credits from the Conc block. For this reason, the Gen Ed and Major block are only included when calculating the credits required. This means that the student must take 20 elective credits to meet the 120 required by the degree.

Block	Hours
Degree Block	120 hrs required to graduate
General Education Block	50 hrs required
Major Block	50 hrs required
Conc Block (included by major)	15 hrs required

Example 5

In this situation, the student needs 120 credits to graduate with 100 (50 + 50) credits coming from required blocks. However, 9 credits can be shared between the major and Gen Ed blocks. If no classes are shared between the two blocks then the student must take 20 elective credits to meet the 120 required by the degree. The ShareWith specifies how many credits may be shared but does not dictate that all or some of these credits will be shared. For this reason the auditor assumes no credits will be shared until the student actually takes a class that does apply to both blocks and is shared. Only then does the auditor increase the ECA credits from 20 to the amount that is being shared. At one point this ECA value will be 20 but then later increased to 23, if 3 credits are shared, and then finally to 29 if the maximum of 9 credits are shared. The ECA calculation does not predict what credits may be shared; the calculation only reacts to what credits are actually being shared.

Block	Hours
Degree Block	120 hrs required to graduate
General Education Block	50 hrs required - Share 9 Credits (major)
Major Block	50 hrs required

In the Class Summary report, any fall-through credits not being counted against degree completion are marked with an “excess” notation.

CompletedTermCount calculation

Updated: March 25, 2022

Calculating completed terms is important for determining Athletic Eligibility.

For “completed terms” we count only terms from the first full-time term. So when we are asking if a student is entering the 5th semester, we need to count the completed terms after that first full-time term, but summer terms don’t count. The terms may also be transfer terms or in-resident terms.

Additionally, the first full-time term cannot precede the first official term as bridged to Degree Works on the rad_custom_dtl as AEAFIRSTTERM. For Banner schools this is mapped from the first date of attendance as recorded in SGRATHE. For non-Banner schools this value needs to be the first term you want Degree Works to consider in the term count calculation. The issue here is that high school students who have taken college classes may have transfer work recorded in the system, but the terms for these classes cannot count for athletic eligibility. This AEAFIRSTTERM, or first date of attendance, must represent post-high school terms. For reporting purposes, an AEAFIRSTDATE value must also be bridged to the rad_custom_dtl with a date in the form of ccyyymmdd (for example, 20221231 for December 31, 2022). This value appears on the worksheet to aid the user in understanding the calculations performed.

Banner schools should ensure you record the student athlete’s first date of attendance at any college/university in the SGRATHE_ATTEND_FROM_DATE field. Because of this, Degree Works cannot pull this first date/term of attendance and will have to calculate it using the data it has starting at the first term it finds, which could be data from the student’s high school coursework. However, if you can store the AEAFIRSTTERM in some other place in Banner, you can set up a UCX-BAN080 record to pull it over.

As part of the 6/18/24 hour rule, the full-time term also comes into play but only with regard to the 6 hour rule. When we look at the previous term attended, we have to go backwards in time until

we find the first full-time term. For the 18 hour and 24 hour rules, however, the terms may be part-time or full-time terms.

First year remedial credits

No more than 6 earned credits can be counted in the first year and none can be counted after the first year.

Review:

- 6 credits earned in the previous term
- 18 credits earned in the previous two terms, or the previous academic year
- 24 credits earned in the first year

For the 6-credit rule, a check is performed to see if that previous term is in the first year; if it is up to 6 credits remedial are allowed; if it is not in the first year then no remedial credits are allowed.

For the 18-credit rule, a check is performed to see if the previous 2 terms are in the first year; if they are up to 6 credits remedial are allowed; if they are not in the first year then no remedial credits are allowed. Same idea applies when counting the previous year credits.

For the 24-credit rule, up to 6 credits remedial are allowed.

Remedial classes are those with a course number below 100 (for example, MATH 99, ENGL 098) or start with a zero (for example, ART 01224, ANTH 0893).

First year earned credits

A small difference to the calculation for FirstYearEarnedCredits is that the auditor also counts all credits earned before the first official term in the first year.

This means that transfer credits earned before the first year are counted, but the auditor includes credits for all terms (including summer terms) before the first term, transfer or not. In addition, summer terms in the first year are also counted. The top of the Diagnostics Report has information that details how this calculation was made for the given student.

Transfer student athletes

Updated: March 25, 2022

The requirements for determining if a transfer student-athlete is eligible to compete at the certifying institution are complex.

For example, there are a lot of different rules depending on whether the student-athlete is a qualifier or non-qualifier through the NCAA and whether they are coming from a 2-year institution or a 4-year institution, and these often need to be evaluated on a case by case basis. At this time, the Athletic Eligibility audit does not evaluate whether the transfer student-athlete meets these specific transfer regulations. This verification needs to be done by the institution. However, after the transfer student-athlete has been determined eligible to compete at the certifying institution, an Athletic Eligibility audit can be generated on the student-athlete to certify they meet the general eligibility requirements.

Football rule

Updated: March 25, 2022

Adding the football rule into its own ATHLETE=FOOTBALL block with an if-statement around your set of rules is important for the Athletic Eligibility audit.

If the student is not a football player (that is, does not have FOOTBALL as a sport), the subset will be removed from this block. The auditor recognizes this situation and marks the block as not-needed and the block is hidden from the worksheet display. This means that this FOOTBALL block will be included for all athletes but will only display on the worksheet for true football players.

```
if (SPORT=FOOTBALL) then
  BeginSub
    # Place your requirements here; see sample rules below in th
is document
  EndSub
  Label "Football: 9 credits in fall";
  Remark "Football players must earn 9 credits in the prior fall season.";
```

The NCAA states that the student can only regain eligibility one time. Degree Works does not keep track of whether or not the student has already regained eligibility, believing the compliance office on your campus keeps track of this information. Degree Works can tell you if the student has earned 27 credits in the last year, but it is the compliance office which is the final word on whether the student really is allowed to regain eligibility.

Furthermore, the 9 HR/APR E Point status is displayed in the Athletic eligibility details section of the report for students who have SPORT=FOOTBALL on the custom-dtl table. The status appears as either Passed or Failed. If the student's active term is a fall term then the status is determined based on the completeness of all ATHLETE blocks: if all blocks are complete then the status appears as Passed; otherwise it appears as Failed. If the active term is not a fall term then the status is pulled from the dap_student_mst's dap_aea_status field through the SCR002 AEASTATUS record. When an audit is run in a fall term the status calculated based on the completeness of the ATHLETE blocks is saved to the dap_student_mst.

To summarize, if the audit term is FALL the status is based on completeness of ATHLETE blocks and saved to the database. If the audit term is not FALL, the status shown is based on what was saved to the database in the previous fall term. If for some reason no status is found in the database or it is neither PASSED nor FAILED the report will show "(unknown)". This situation will happen for your initial set of audits until you have run audits in a fall term.

Batch Athletic Eligibility audits

Updated: March 25, 2022

You can run batch Athletic Eligibility audits from DAP22 in Transit like you do with Academic audits.

Choose Athletic Eligibility Audit as the audit type as the first question on the Questions section. You can choose to have them converted to PDF and choose either the RPT55 - Athletic Eligibility report or RPT56 - Athletic and Academic report. You can also freeze the new audits you run by choosing a freeze type.

In the Selection criteria section, you can select any student, but it makes sense to select your student athletes. This can most easily be done by searching on students with a SPORT record on the rad_custom_dtl, but you can decide to use additional criteria as needed.

Financial Aid

Updated: March 25, 2022

Financial Aid audits allow you to process degree audits to assist with reviewing aid requirements.

Rules for Financial Aid awards are built using Scribe, and processing a student's financial aid data against these requirement blocks will verify if a student has met the requirements for an award.

When running a Financial Aid audit, the In-progress check box should always be checked as the current classes are required and the Preregistered check box should always be unchecked - future classes are never wanted for a Financial Aid Audit. Financial Aid history obeys the same CFG020 DAP14 Audit History Depth setting as academic audits when saving audits – if the depth is set to 3 you will end up with 3 academic audits and 3 financial aid audits. When running a Financial Aid audit an academic audit is run but any AWARD blocks are also pulled in based on the AWARD values found in the rad_aid_dtl. All rad_aid_dtl records are read when doing a Financial Aid audit - no SCR002 entries are needed.

By default, the award code for any awards the student has displays in the Financial aid awards section of the audit. This can be localized to display a literal instead. For example, to display Pell Award instead of PELL in the Responsive Dashboard, in DashboardMessages.properties, create a localization for dash.worksheet.financial.PELL=Pell Award. To display Pell Award instead of PELL in PDF Audits, add a corresponding entry in PDFAuditMessages.properties:worksheet.financial.PELL=Pell Award.

Note: The Financial Aid audit is not equivalent to the CPoS (Course Program of Study) component of Banner.

Configuration

- Review BAN080 to ensure the required Financial Aid records exist. For more information, see [Financial Aid audit data setup](#)
- Review AUD033 - Financial Aid Award types
- Review RPT036 to ensure it is configured correctly for WEB50 and WEB51.
- Review STU016 to ensure that the Term Type and Financial Aid Year fields are populated; only terms with non-blank aid year fields will appear in the Aid Term drop-down.
- Access to the following is granted if the user has the specified Shepherd key:
 - Financial Aid - SDAIDAUD
 - Financial Aid Report - SDWEB50
 - Aid and Academic Report - SDWEB51

- View student financial aid audits - SDAIDREV
- Delete audits - SDAIDDEL
- View historic financial aid audits - SDAIDHIS
- Process financial aid audits - SDAIDRUN

Financial Aid audit data setup

Updated: March 25, 2022

The Financial Aid audit requires specific data to be bridged into the Degree Works database.

The data structure for holding data required for the Financial Aid audit is the rad_aid_dtl, with one name value pair stored in each record. The structure is:

RAD_AID_DTL		
Column name	Type	Size
srn_id	Integer	
rad_id	Char	10
rad_aid_code	char	12
rad_aid_value	Char	12
unique_id	Integer	
unique_key	Char	36

At least two values should be bridged to the rad_aid_dtl:

- AIDYEAR - all active aid years for each student should be bridged to the rad_aid_dtl.
- AWARD - all active awards for each student should be bridged to the rad_aid_dtl.

You may also want to bridge an ENROLLSTATUS and AIDSTATUS for each student for reporting purposes, but neither is required. See examples below for configuring BAN080 to extract this data. If you are a non-Banner site and use RAD11 to bridge student information, you need to alter your extract to obtain information about your students into the rad_aid_dtl.

There are a few items related to awards which are repeatable, and require special coding for the rad_aid_code. Any record related to an award should have the fund or award code as part of the code. For example, to store the cumulative amount of the Pell award, use the rad_aid_code of PellCumAmt.

Any item that is a number will be stored in ASCII encoded digits, not as binary. Numeric items of the same type (e.g. EstFamContrib) should be zero padded to the same length.

Additional information you may consider extracting and bridging from your student system:

Item	Type	Size	Year specific	Notes
Year in College	INTEGER		Y	1st, 2nd
Terms Completed	INTEGER		Y	#
Enrollment Status	CHAR	2	Y	Full Time, Part time
Class Level	CHAR	2	Y	FR, SO
Program Type	CHAR	12	Y	5-year, AS, BA, non-degree seeking
SAP Status	CHAR	2	Y	Probation, Suspension
Citizenship	CHAR	2	Y	US, Eligible, Not
TAP Payments	INTEGER		Y	#
TAP Pts Used	INTEGER		Y	#
TAP Waiver Flag	CHAR	1	Y	Y/N
TAP Waiver Term	CHAR	12	Y	Term
TAP Waiver Date	DATE		Y	Date
Award Source	CHAR	12	Y	Pell, ACG, etc
Award Type	CHAR	12	Y	Federal, State, Institution
Cumulative Award Amount	INTEGER		Y	Cumulative, no decimal
Current Award Amount	INTEGER		Y	This year, no decimal
Secondary Program of Study	CHAR	12	N	Rigorous
Previous Undergrad Experience	CHAR	1	Y	Y/N
Pell Recipient	CHAR	1	Y	Y/N
Residence	CHAR	4	Y	State, County, Non
Matriculated	CHAR	1	Y	Y/N
1st time enrollment term	CHAR	12	N	Term
Disability Code	CHAR	4	Y	3C
Estimated Family Contribution	INTEGER		Y	EFC
Education Level	CHAR	4	Y	HS, Some College, BA
Dependency	CHAR	2	Y	Dep / Indep

BAN080 setup

Suggested entries for BAN080 AIDAWARD:

AIDAWARD:AID	AWARD
AIDAWARD: COLUMN	RPRAWRD_FUND_CODE
AIDAWARD: ORDERBY	RPRAWRD_FUND_CODE
AIDAWARD: TABLE	RPRAWRD
AIDAWARD: WHERE_1	RPRAWRD_AIDY_CODE = '0506'
AIDAWARD: WHERE_2	AND RPRAWRD_AWST_CODE = 'ACPT'

Suggested entries for BAN080 AIDYEAR:

AIDYEAR:AID	AIDYEAR
AIDYEAR: COLUMN	RPRAWRD_AIDY_CODE
AIDYEAR: ORDERBY	RPRAWRD_AIDY_CODE
AIDYEAR: TABLE	RPRAWRD
AIDYEAR: WHERE_1	RPRAWRD_AWST_CODE = 'ACPT'
AIDYEAR: WHERE_2	AND RPRAWRD_AIDY_CODE in
AIDYEAR: WHERE_3	(SELECT b.ROBINST_AIDY_CODE FROM ROBINST b
AIDYEAR: WHERE_4	WHERE b.ROBINST_STATUS_IND = 'A')

Suggested entries for BAN080 AIDENRSTATUS

AIDENRSTATUS:AID	ENROLLSTATUS
AIDENRSTATUS: COLUMN	SGBSTDN_FULL_PART_IND
AIDENRSTATUS: ORDERBY	SGBSTDN_FULL_PART_IND
AIDENRSTATUS: TABLE	SGBSTDN α
AIDENRSTATUS: WHERE_1	α.SGBSTDN_TERM_CODE_EFF =
AIDENRSTATUS: WHERE_2	(SELECT MAX(b.SGBSTDN_TERM_CODE_EFF
AIDENRSTATUS: WHERE_3)
AIDENRSTATUS: WHERE_4	FROM SGBSTDN b
	WHERE b.SGBSTDN_PIDM = α.SGBSTDN_PIDM

Suggested entries for BAN080 AIDSTATUS

AIDSTATUS:AID	AIDSTATUS
AIDSTATUS: COLUMN	RORSAPR_SAPR_CODE
AIDSTATUS: ORDERBY	RORSAPR_SAPR_CODE
AIDSTATUS: TABLE	RORSAPR
AIDSTATUS: WHERE_1	RORSAPR_SAPR_CODE IS NOT NULL

Financial Aid audit rule examples

Updated: March 24, 2023

AWARD blocks house Financial Aid requirements.

AWARD header qualifiers must have Labels to display their advice to the right of the label. AWARD blocks should always contain the current aid year's requirements, and be saved with open-ended catalog years. As long as your AIDAWARD BAN080 item is up-to-date, that block will

only ever be given to students who have that award in the *current* aid year, because only those students will have that AWARD code in rad_aid_dtl.

HEADER examples

```

if (TotalCreditsEarned > 30) then
  beginif
    MinGPA 3.0
    ProxyAdvice "Your CrEarned > 30 - so you must meet the 3.0 requirement"
  endif
if (LastCompletedTermType = Spring) then
  beginif
    MinGPA 2.5
    ProxyAdvice "You need a GPA of 2.5 now that you have completed Spring"
  endif
if (CompletedTermCount > 3) then
  beginif
    MinGPA 3.0
    ProxyAdvice "You need a GPA of 3.0 now that you have 3 terms completed"
  endif

if (ResidenceCompletedTermCount > 3) then
  beginif
    MinGPA 3.2
    ProxyAdvice "You need a GPA of 3.2 now that you have 3 terms completed at UW"
  endif
MinGPA 2.5 in (MAJOR)
  ProxyAdvice "GPA not good enough for Major block"
Under 30 Credits in a (With Attribute=DEV)
  ProxyAdvice "You have more than 30 developmental credits"
  Label "No more than 30 developmental credits"
if (EnrollStatus=F) then # F=full-time
  beginif
    MinTerm 12 Credits
    ProxyAdvice "You did not take at least 12 credits per term"
  endif
else if (EnrollStatus=P) then # P=part-time
  beginelse
    MinTerm 6 Credits
    ProxyAdvice "You did not take at least 6 credits per term"
  endelse

```

RULE examples

```

if (CreditsAttemptedThisTerm > 12) then
  RuleComplete
  Label "12 credits attempted this term - met"
else
  RuleIncomplete
  ProxyAdvice "You have not yet attempted 12 credits"

```

```

Label "12 credits attempted this term - not met";

if (CreditsEarnedThisTerm >= 75% of CreditsAttemptedThisTerm) then
  RuleComplete
  Label "Term credits earned satisfied"
else
  RuleIncomplete
  ProxyAdvice "You have not yet earned 75% of attempted credits"
  Label "Term credits earned- not met";
if (CreditsAttemptedThisAidYear > 40) then
  RuleComplete
  Label "Aid Year attempted - met"
else
  RuleIncomplete
  ProxyAdvice "You have not yet attempted 40 credits in the aid year"
  Label "Aid Year attempted - not met";
if (CreditsEarnedThisAidYear >= 75% of CreditsAttemptedThisAidYear)
then
  RuleComplete
  Label "Aid year credits earned satisfied"
else
  RuleIncomplete
  ProxyAdvice "You have not yet earned 75% of attempted credits in the aid year"
  Label "Aid year credits earned- not met";
if (TotalCreditsAttempted > 30) then
  RuleComplete
  Label "total-cr-attempted - met"
else
  RuleIncomplete
  ProxyAdvice "You have not yet attempted 30 credits"
  Label "total-cr-attempted - not met";
if (ResidenceCreditsEarned >= 75% of TotalCreditsAttempted ) then
  RuleComplete
  Label "Credits earned is at least 75% of credits att"
else
  RuleIncomplete
  Label "Credits earned is NOT 75% of attempted";
if (TotalCreditsEarned < 150% of DegreeCreditsRequired) then
  RuleComplete
  Label "Credits earned is less than 150% of required"
else
  RuleIncomplete
  Label "Credits earned more than than 150% of required";
12 Credits in @ (With DWTerm = CURRENT) Label "CURRENT term";
12 Credits in @ (With DWTerm = PREVIOUS) Label "PREVIOUS term";

```

Financial Aid audit Scribe reserved words

Updated: March 24, 2023

These Scribe reserved words are allowed in an IF statement in a Financial Aid audit, usually in an AWARD block.

The following reserved words are allowed only in an IF statement in a Financial Aid audit, usually in an AWARD block.

Scribe word	Alias	Data source
CompletedTermType	TermType	
CompletedTermCount	TermCount	
TotalCreditsEarned	TotalCreditsCompleted	rad_term_dtl.rad_cum_tot_earn
ResidenceCreditsEarned	ResidenceCreditsCompleted	rad_term_dtl.rad_cum_cr_earn
TotalCreditsAttempted	CreditsAttempted	rad_term_dtl.rad_cum_gr_att
CreditsAttemptedThisTerm		look at the classes on the aid-term - count the rad_class_dtl.rad_credits (not earned); this does not include any credits for withdrawn classes – unless they are bridged with non-zero credits.
CreditsEarnedThisTerm		look at the classes on the aid-term - count the rad_class_dtl.rad_gpa_credits
CreditsAttemptedThisAidYear		look at the classes on the terms in the aid-year specified - count the rad_class_dtl.rad_credits (not earned); this does not include any credits for withdrawn classes – unless they are bridged with non-zero credits.
CreditsEarnedThisAidYear		look at the classes on the terms in the aid-year specified - count the rad_class_dtl.rad_gpa_credits
CompletedTermCount		count the terms where at least one class was taken either in residence or as a transfer
ResidenceCompletedTermCount		count the terms where at least one class was taken in residence
LastCompletedTermType		else go backwards in time and find the first completed term
DegreeCreditsRequired	CreditsRequired	credits required from starting block - only allowed on right-hand-side of an IF-statement

The following reserved words are allowed in both Academic and Financial Aid audits, but it makes the most sense to use them in an AWARD block on a Financial Aid audit.

Scribe word	Alias	Data source
MinCredits 12 in @ (With DWTerm=PREVIOUS)		previous term the student took classes - before active
MinCredits 12 in @ (With DWTerm=CURRENT)		active-term
MinTerm X Credits/Classes		look at credits ATTEMPTED - anywhere in the audit - including failed and OTL only works in the starting block or an AWARD block If the MinTerm is in the starting block or an AWARD block we then look at the entire audit - otherwise we just look at classes in this scope.
Under 30 Credits in @ (With Attribute=DEV)	Below	new header qualifier As long as the student has less than or equal to the number of credits/classes specified = OK All attempted classes/credits are applied here
MinGPA 2.0 in (MAJOR) MinGPA 2.0 in (MAJOR = MATH)		These MinGPA scopes can be used in any block - but normally used in an AWARD block "MinGPA 2.0" in AWARD block looks at the overall GPA

Exceptions

Updated: March 25, 2022

The Exceptions function allows users to apply or remove exceptions to the requirements for degree completion for a specific student.

Configuration

- Review RPT036 to ensure it is configured correctly for WEB44.
- Review the following Shepherd settings to ensure they are configured correctly:
 - dash.exceptions.details.option
 - dash.exceptions.runAuditOnEnter
 - dash.exceptions.saveNewAudit
- Access to the following is granted if the user has the specified Shepherd key:
 - Exceptions - SDEXCEPT
 - Add exceptions - SDEXPADD
 - Delete exceptions - SDEXPDEL
 - Also Allow exceptions - EXPALLOW
 - Apply Here exceptions - EXPAPPLY
 - Change the Limit/Remove Course exceptions - EXPCHANG
 - Force Complete exceptions - EXPFORCE
 - Substitute exceptions - EXPSUBST
 - View exception details - EXPVWDTL

Add exception

Updated: September 29, 2023

A user can add an exception through the Add Exception dialog.

After selecting a student, click the plus sign where you want to add an exception, whether globally or individually and the Add Exception dialog will display. If configured, a new audit will be run in the background. Select the exception type you want to add from the drop-down. Enter a description of the exception being added, which is a required field and will display on the audit. If a description is not entered, a default is provided by the system and can be localized in **Controller properties > DashboardMessages.properties > dash.exceptions.description.*** properties.

A Details field may be configured to display, allowing the user to provide additional details about the exception. This text doesn't display directly on the audit but can be viewed as a tool tip when hovering over the exception description on the audit but only if the user has the EXPVWDTL Shepherd key.

Exceptions are both student-specific and block-specific. In other words, an exception applies to a specific block used in a student degree audit. If an exception is processed in a major block for a student and that student then changes major, the previous exception will no longer apply to the student's new major. However, if the CFG020 DAP14 **Apply exception to a similar block** flag is set to Y, the exception can be used on the new major but only if the label-tag or qualifier tags between the two blocks match. For more information, see the [UCX-CFG020 DAP14](#) topic. Unused exceptions will appear at the bottom of the audit report. Exceptions are also used with What-If audits. Exceptions will only appear in the exception screen and on audit reports configured to show Exceptions. By default, the Registrar's Audit report is configured to show exceptions.

Exceptions on rules are tied to the rule's label-tag, or label if no label-tag exists. When a block is modified using Scribe the exception is placed on the correct rule as long as the original rule label can be found in the new block. Qualifiers do not have labels so instead a Tag should be scribed for each header qualifier to prevent the exceptions from becoming unhooked. See the [Label Tags](#) topic for more information on using tags.

A new audit will be automatically run, applying the exception. This audit can be configured to be saved to the database when it is run by setting dash.exceptions.saveNewAudit to true; otherwise, it will only be visible to the user while on this page.

Exception types

Updated: March 25, 2022

Exception types are codes assigned to each Degree Works exception.

Force Complete

Updated: March 25, 2022

Force Complete completes a course rule, subset rule, block qualifier, or rule qualifier without applying additional classes. It is the most powerful exception type available.

The force complete exception can be used on any course rule and most qualifiers. This exception type is completely independent of all student data. It will simply complete a rule on a student degree audit regardless of any qualifiers that might apply. Forcing a qualifier complete tells the auditor to ignore the qualifier the next time an audit is performed for this student.

Substitute

Updated: March 25, 2022

Substitute is used to substitute one course for another. This is distinct from the Also Allow exception type in that one course is exchanged for another.

If the rule contains only a single course then the substituted course is required for completion of the block. If a substitute exception is processed on a rule with more than one course option that can be used to complete the rule, then the substituted course is not required and is an option available to the student. Only qualifiers that list courses support this type of exception.

To create a substitute exception, enter the target course from the course rule in the “Change” fields. The target course must be found on the rule where the exception is to be placed to be saved to the database. Enter the substituted course in the “To” fields.

You can further define the exception using “With” qualifiers. The values listed in the drop-down are Ellucian baseline “With” qualifiers (DW Credits, DW Grade, and so on) in addition to user-defined “With” qualifiers from SCR044. When “With” qualifiers are included as a condition for this exception type, only those courses meeting the WITH qualifier criteria will be evaluated for the exception.

You can also create a global substitution exception by choosing the option at the top of the page. These substitutions will be applied to all requirements found in all blocks.

Also Allow

Updated: March 25, 2022

Also Allow modifies a course rule by appending a course to the course rule. This exception can be used when you want to expand the course options available on a specific rule.

Courses applied using the Also Allow exception are still subject to header qualifiers in the blocks in which they are used. For example, if an Also Allow exception is processed allowing ENGL 215 to be used to satisfy a course rule and there is a block header or rule qualifier preventing the use of ENGL 215 within that block, the exception will be added to the database, but ENGL 215 will not be allowed to be used within that block. You will need to first modify the header qualifier preventing this course from being used before processing the Also Allow exception.

When processing an Also Allow exception to a group rule, the exception can only be processed on the individual rule labels inside the group rule, not the GROUP RULE HEADER LABEL.

To create an also allow exception, enter the course discipline and number in the Allow fields.

You can further define the exception using With qualifiers. The values listed in the drop-down are Ellucian baseline With qualifiers (DW Credits, DW Grade, and so on) in addition to user-defined With qualifiers from SCR044. When With qualifiers are included as a condition for this exception type, only those courses meeting the WITH qualifier criteria will be evaluated for the exception.

The Also Allow exception does not require that the selected course be used on the modified rule.

The best-fit algorithm of the Audit Processor Engine will still function with this type of exception. Consequently, the allowed course may not be applied to rule bearing the exception if there is a better fit for this course elsewhere in the degree audit.

Block header exceptions do not support the Also Allow exception.

Apply Here

Updated: March 25, 2022

Apply Here allows the user to apply a course to a rule even if the course is not listed as an option. This exception is very useful in correcting audits in cases where the user wishes to dictate specifically where courses are to be used within the degree audit.

The Apply Here exception will apply a course to a rule regardless of any scribing, rule or block header qualifiers. For example, suppose a block contains a MAXPASSFAIL 0 CLASSES header qualifier. You can use the Apply Here exception to apply a PASSFAIL class to a rule in this block without having to modify the block header qualifier. The Apply Here exception will also override an EXCEPT list on a rule. Courses applied to rules using this exception type will not be moved around within the audit. When processing an Apply Here exception on a group rule, the exception can only be processed on the individual rule labels inside the group rule. Apply Here exceptions cannot be processed on a Group Rule Header.

To create an apply here exception, enter the course you want applied to a rule. The course must be a course already taken by the student and found on the degree audit or one the student is planning to take. The Apply Here exception cannot be used for courses that have not yet been taken.

You can further define the exception using With qualifiers. The values listed in the drop-down are Ellucian baseline With qualifiers (DW Credits, DW Grade, and so on) in addition to user-defined With qualifiers from SCR044. When With qualifiers are included as a condition for this exception type, only those courses meeting the WITH qualifier criteria will be evaluated for the exception.

When placing an Apply Here exception any existing WITH data is removed so that a course that normally did not meet the WITH qualifiers will be able to fit on the rule. If the Apply Here exception contains a new WITH qualifier that will be applied – replacing the previous WITH data.

Block header exceptions do not support the Apply Here exception.

Remove Course

Updated: March 25, 2022

Remove Course allows the user to remove a course from a course rule or qualifier or to change the limit on a course rule or qualifier.

This exception type is very useful in modifying audit reports when students successfully petition to have a specific course or part of a specific requirement waived. It is also useful in correcting the advice given to students when a specific course is disallowed on a specific rule by a student. For example, if a student receives advice that they can take CSC 113 to fulfill the Technology Requirement but the department chair has disallowed this course on this rule for a specific student, you can correct the advice given to the student by using the Remove Course exception.

The Remove Course exception type can also be used to change the limit on a rule without removing a course from the course list.

To create a remove course exception, enter the course discipline and number you want to remove. Enter a brief description for the exception. If the exception also involves changing the limits on the rule or qualifier for the student, enter the new limit in the “Change” fields.

In situations where a course is applied to a rule or qualifier as a result of a wildcard statement, using the Remove Course exception will remove ALL courses that have been applied as a result of the wildcard statement. For example, if HIST 330 and HIST 427 are both applied to the scribe rule 3 Classes in HIST @, removing HIST @ using the Remove Course exception will remove both HIST 330 and HIST 427.

However, if you remove just HIST 330 the rule gets changed to 3 Classes in HIST @ Except HIST 330 – making sure that HIST 330 does not get applied to this requirement.

Only qualifiers that list courses or specify a number of classes or credits support the Remove Course exception.

Qualifier exceptions

Updated: March 25, 2022

The Force Complete, Substitute, Remove Course, and Change Limit exceptions can be used on block header qualifiers and rule qualifiers.

The header qualifiers appear above the rules in the block and have an option button next to them.

Not all qualifiers support the same types of exceptions. Below are the exceptions and the qualifiers on which they can be placed:

Force Complete	All except NonExclusive (also known as ShareWith), SameDisc, and Optional
Substitute	MaxClasses, MinClasses, MaxCredits, MinCredits, MaxTerm, MinTerm, SpMaxCredit, SpMaxTerm
Remove Course	MaxClasses, MinClasses, MaxCredits, MinCredits, MaxTerm, MinTerm, SpMaxCredit, SpMaxTerm
Change Limit	All except SameDisc, Option, MinGPA, MinGrade

Remove exception

Updated: March 25, 2022

To remove an exception, you can click the **Remove** icon that appears to the right of the applied exception. The exception will be removed and a new audit will be automatically generated. Exceptions can be removed in bulk from the Exceptions section on an audit.

Exception Management

Updated: March 25, 2022

Exception Management allows users to manage petitions and apply exceptions, fix petition statuses, and generate exceptions reports. The Exceptions Management function is located under the Admin menu.

Configuration

Access to the following is granted if the user has the specified Shepherd key:

- Exception Management - SDEXPMGT
- Awaiting Approval - SDEMPEWA
- Approved Petitions - SDEMPEAV
- Rejected Petitions - SDEMPERJ
- Applied Petitions - SDEMPEAL
- Delete rejected petitions - SDEMPERD
- Delete applied petitions - SDEMPEAD
- Fix petition status - SDEMPEFX
- Search for exceptions - SDEMEXSR

Manage Petitions

Updated: March 25, 2022

Manage Petitions allows the user to view, approve, reject, and apply petitions requests for all students.

Awaiting Approval

Updated: March 25, 2022

Awaiting Approval is used to approve, reject or add additional documentation to pending petitions.

Any petition with a pending status will appear on the Petitions Awaiting Approval grid.

To approve an individual petition, expand the petition by clicking on the arrow on the right and click **Approve**. The status of the selected petition will change to approved and the exception will be removed from the list of petitions waiting approval. To approve multiple petitions, select the petitions that should be approved by selecting the check box on the left and then click **Approve** at the top of the grid.

To reject an individual petition, expand the petition by clicking on the arrow on the right and click **Reject**. The status of the selected petition will change to rejected and the exception will be removed from the list of petitions waiting approval. To reject multiple petitions, select the petitions that should be rejected by selecting the check box on the left and then click **Reject** at the top of the grid.

It is also possible to add additional documentation to a pending petition while approving or rejecting an individual petition. With the petition row expanded, enter your text in **My comments** before clicking Approve or Reject.

It is important to remember that an approved petition is not the same as an applied exception. After a petition has been approved, it is still necessary to actually apply the petition as an exception on the student's record.

Approved Petitions

Updated: March 25, 2022

Approved Petitions allows a user to process exceptions for petitions that have been approved.

Any petition with an approved status will appear on the "Apply Approved Petitions" grid.

To process an exception for an approved petition, expand the petition by clicking on the arrow on the right. The exceptions audit will load, and an exception can either be added or removed. For information on how to apply an exception, see the [Exceptions](#) topic. After adding the exception, click the **Update petition and close** button. The status of the selected petition will change to applied and the exception will be removed from the list of approved petitions.

Rejected Petitions

Updated: March 25, 2022

Rejected Petitions allows a user to view all petitions that have been rejected.

Select the petition date range to filter the list of rejected petitions that appear on the Rejected Petitions grid based on the day they were created.

Rejected petitions can be maintained in the database or deleted as necessary. To delete rejected petitions, select one or more petitions by selecting the check box on the left and then click the **Delete** button at the top of the grid. The exception will be removed from the list of rejected petitions.

Applied Petitions

Updated: March 25, 2022

Applied Petitions allows a user to view all petitions that have been applied as exceptions.

Select the petition date range to filter the list of applied petitions that appear on the Petitions Applied as Exceptions grid based on the day they were created.

Applied petitions can be maintained in the database or deleted as necessary. To delete applied petitions, select one or more petitions by selecting the check box on the left and then click the **Delete** button at the top of the grid. The exception will be removed from the list of applied petitions.

Fix Petition Status

Updated: March 25, 2022

Fix Petition Status allows a user to change the status of a petition. Petitions can be filtered by date range and student ID.

All petitions meeting the search criteria will be displayed in the Fix Petition Status grid. To change the status of a petition, select the correct status from the drop-down in the Status column. Click the **Save Changes** button to save your changes to the database.

If a petition has a status of Approved and additional comments were entered when the petition was approved, changing the petition status back to waiting will not affect the additional comments that were originally entered. These comments will display with the petition until the petition is either applied as an exception or deleted from the database.

If a petition has a status of Applied, changing the status of the petition will have no effect on the exception that has already been applied. One should therefore be careful when changing the status of petitions that have already been applied as exceptions as it is possible to apply more than one exception for a given petition.

Click **Cancel** to reverse any changes made and refresh the page, returning all petitions to their original state.

Exceptions Report

Updated: March 25, 2022

Exceptions Report allows users to query the database and create reports on the number and type of exceptions processed on your audits.

You can search for exceptions created by an individual, exceptions applied for a particular student or exceptions applied within a specific block. You can also use this function to identify unhooked exceptions as well.

You can filter the exceptions report by any combination of exception date, type, status, created by ID, student ID, and requirement ID. Click the **Run exceptions report** button to process your query. Your search will be displayed in the grid below.

The search results can be sorted by any of the columns displayed. Click the header label for the column you want to use as your sort and the list will sort based upon that criterion. Sorts can be done in ascending or descending order. To change from descending to ascending, click the column header again and the indicator will change from a down arrow to an up arrow.

By default, the search results are returned to Show exception details. Expanding the row will display additional information about the exception. Select **Show requirement block counts** to show which scribe blocks have exceptions processed against them. Expanding the requirements row will show the count of exceptions for each label tag in the block. This is a useful tool for determining if a scribe block is requiring an excessive number of exceptions.

Configuration

Review the dash.exceptionManagement.exceptionsReport.allowSearchOnAll Shepherd setting to ensure it is configured correctly.

Exception status codes

Updated: March 25, 2022

The OK status code indicates that the exception applied correctly. The UN status code means that the exception became unhooked from its intended requirement during the reparse of the block.

This usually is because the rule on which the exception was placed changed so much that new rule could not be recognized. To prevent this from happening, you should ensure your rules have label tags and your qualifiers have qualifier tags. The other codes listed below indicate that the exception was unenforced meaning that when the audit was run the exception could not be applied as intended.

If you see no value in the Label Tag field for any exception you should strongly consider adding a label tag to that requirement. Please review the [Label Tags](#) topic for help preventing unhooked exceptions.

Status code	Meaning
OK	Exception applied successfully.
UN	Unhooked – typically caused by a Scribe block change (be sure to use label tags).
BK	Block was not in audit – typically caused by a major or catalog year change.
CR	Rule is not a course rule – node-ids are incorrect.
CO	Course to be substituted was not found on the rule.
IF	IF-statement was resolved – this rule was not used.
ND	Node-id not found in block.
PA	Rule is a plus-list and Also Allow is not allowed on a plus-list.
PN	Rule is a plus-list and the course is not on the rule; the course can't be added to the end of the rule.
R1	Not enforced but we don't know why exactly.
R2	Not enforced and an unknown error code was found.

Plans

Updated: September 30, 2022

In Plans, a student or advisor can create an academic plan that can be used in conjunction with the audit to ensure that all degree requirements are planned for.

Plans can be created either from scratch or using a predefined template for the course of study. Tools such as the course list and still needed list help identify courses that can be added to a plan, in addition to GPA, test score, non-course and placeholder requirements. Using their plan, a student can generate an audit to evaluate their intended degree progress or have a custom Scribe block for their degree program created. The tracking feature evaluates whether the planned requirements were completed in the expected term, and a plan can be printed or saved locally.

By default, a user assigned access to Plans can just view existing plans; additional authorization grants access to the other features.

Configuration

Access to Plans is granted if the user has the SEPPLAN Shepherd key.

Printed plan

Updated: September 30, 2022

A plan can be printed by clicking the Print icon.

A printer-friendly calendar view of the student's plan will open in a new browser window. From this view, clicking the **Print** icon will open a dialog in which the user can choose to send the output to a printer, save the output as a PDF, or any other option available in the browser.

When `studentPlanner.planner.showDisclaimer` is true, a disclaimer will show on the printed plan. The disclaimer text can be localized in the `planner.planDetail.disclaimer` property in `PlannerMessages.properties`.

Configuration

Review the `studentPlanner.planner.showDisclaimer` Shepherd setting to ensure it is configured correctly.

Plan List

Updated: March 25, 2022

The Plan List displays all of a student's plans, regardless of degree, school, or status.

Plans in the view can be sorted by any column by clicking on the column header. To open a plan for viewing or editing, click on the plan Description.

From the Plan List, if a user has access they can also create a new plan or delete an existing plan.

The user can return to the Plan List from a plan by clicking the **Plan list** button.

Plan creating and editing

Updated: March 24, 2023

A new plan can be created while on the Plan List or in the plan view.

To create new plans, a user needs the SEPPADD Shepherd key. They will also be able to modify existing plans, create a copy of a plan, add new terms and requirements on plans, and modify terms and requirements (including deleting individual terms and requirements.)

The SEPPMOD Shepherd key can be assigned to a user instead of SEPPADD if they shouldn't have access to create new plans, but can modify existing plans, add new terms and requirements on plans, and modify terms and requirements (including deleting individual terms and requirements.)

To delete plans, including all terms, requirements and notes on that plan, a user should be assigned the SEPPDEL Shepherd key. Limited delete access can be given to users with the SEPPDELL Shepherd key. This restricts the user to only deleting plans that are not active and not locked/not approved, depending on the studentPlanner.planner.planApprovalMethod Shepherd setting.

For users with access to create a plan, when they click the **New Plan** button they can create a new blank plan. The user will be prompted to select a starting term for the plan, and after providing a description and other details, a blank plan with the starting term but no requirements will load. The values in the starting term drop-down will be those that have the STU016 Show in SEP Picklist flag = Y.

If the user also has the SEPPTMP Shepherd key, they will have the option to create a plan from a template. The user first selects the starting term for the plan and then selects the template on which they want to base the plan from a list of all active templates. The values in the starting term drop-down will be those that have the STU016 Show in SEP Picklist flag = Y. The generic term on the template will be replaced with an actual term on the plan, based on the starting term and the template's term scheme. It is important to have enough future terms defined in STU016 with term types to satisfy all the terms in the template's term scheme. The template description will be loaded as the default plan description, but it can be modified by the user.

A plan description is required, whether creating a blank plan or a plan from a template. This field is 80 bytes long and does not need to be unique; that is, the student can have more than one plan with the same description.

The degree and school associated with the plan will default from the selected values in the student header, and if the student has goal data for more than one degree/school combination, the user will be able to assign a different degree for the plan. If a different degree is selected, the corresponding school will automatically load. Only the degrees for which the student has active goal data will be available in the drop-down.

When the studentPlanner.planner.showScope Shepherd setting is optional or required, the plan type will display. This can be used to identify the type of plan and is typically used for state fund reporting. If the user has the SEPSCOPE Shepherd key, they will have access to modify the plan type. The plan type drop-down contains values from SEP011 with Show in SEP Picklist flag = Y. When the plan type is modified, the scope date will be updated with the date modified.

The Total planned credits field displays the sum of the credits planned for each term on the plan.

The active flag identifies which plan is the student's active plan. While students can have multiple plans, they typically have only one active plan per degree/school. Multiple active plans per degree/school can be allowed by setting `studentPlanner.planner.allowMultipleActivePlans` to true, but Ellucian does not recommend this. Note that if configured to allow only one active plan, the students can still have more than one active plan, if they have several degree/school goal combinations. The active flag can be modified by users with access to modify plans and the SEPPACTV Shepherd key if either the locking or no approval method is configured with the `studentPlanner.planner.planApprovalMethod` Shepherd setting. If the approval method is workflow, the active flag will be updated through the Banner Workflow process.

The locked flag will display only if `studentPlanner.planner.planApprovalMethod` is configured for locking. Only users with the SEPPLOCK Shepherd key will be able to modify locked plans, including terms and requirements.

On the plan card, Status will display only if `studentPlanner.planner.planApprovalMethod` is configured for locking or Banner Workflow. When using Banner Workflow for approval, the status cannot be modified in Responsive Dashboard.

The tracking status will initially load as NOTEVALUATED if `studentPlanner.planner.tracking.enable` = true. This will be updated automatically when the plan is modified and saved.

A user can create a copy of an existing plan by clicking **Save as copy**. This will create an identical copy of the plan including the description and all terms, requirements, and notes. However, the active flag will be set to not active, the locked flag will be set to not locked, and the approval status will be set to needs approval. The copied plan will automatically load in the user interface.

Configuration

- Review the Show in SEP Picklist flag in STU016 to ensure it is configured correctly.
- Review the Term Type in STU016 and Term Type in SEP002 to ensure they are configured correctly.
- Review the Show in SEP Picklist flag in SEP011 to ensure it is configured correctly
- Review the following Shepherd settings to ensure they are configured correctly:
 - `studentPlanner.planner.allowMultipleActivePlans`
 - `studentPlanner.planner.planApprovalMethod`
 - `studentPlanner.planner.showScope`
 - `studentPlanner.planner.tracking.enable`
- Access to the following is granted if the user has the specified Shepherd key:
 - Add and modify plans - SEPPADD
 - Modify plans - SEPPMOD
 - Delete all plans - SEPPDEL

- Delete only unlocked plans - SEPPDELL
- Create new plans from a template - SEPPTEMP
- Modify plan scope - SEPSCOPE
- Modify locked plans - SEPPLOCK
- Set active flags on plans - SEPPACTV

Terms creating and editing

Updated: September 30, 2022

An unlimited number of terms can be added to a plan by a user who has access to modify a plan.

A user with the SEPPADD or SEPPMOD Shepherd key can add, reassign, or delete terms on a plan. You can add a new term by clicking the **Add term** button. A drop-down of available terms will be displayed. The values in this drop-down are all terms from STU016 with Show in SEP Picklist flag = Y and that have not already been added to the plan. The student's active term and past terms will be filtered out of this list if the studentPlanner.planner.allowChangesToCurrentTerms and studentPlanner.planner.allowChangesToPastTerms Shepherd settings are false.

All requirements on a term can be moved to another term by selecting **Reassign this term** from the action menu in the right corner of a term and selecting the new term from the drop-down. After clicking **Reassign**, the term and its requirements will be moved to the correct chronological position on the plan.

The total number of planned course credits will be displayed on a term header. If studentPlanner.planner.tracking.enable = true, the tracking status will also be displayed for the term.

Up to three terms will display in the desktop view. Other terms on the plan can be scrolled into view by clicking the left or right arrows on the left side of the plan.

To delete a term, you can select **Delete this term** from the action menu in the right corner of a term.

Configuration

- Review the Show in SEP Picklist flag in STU016 to ensure it is configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - studentPlanner.planner.allowChangesToCurrentTerms
 - studentPlanner.planner.allowChangesToPastTerms
 - studentPlanner.planner.tracking.enable
- Access to add, modify, and delete terms is granted if the user has the SEPPADD and SEPPMOD Shepherd keys.

Requirements creating and editing

Updated: September 30, 2022

An unlimited number of requirements can be added to a term on a plan by a user who has access to modify a plan.

A user with the SEPPADD or SEPPMOD Shepherd key can add, reassign, or delete requirements to a term on a plan. There are six requirement types: Course, Choice, GPA, Non-course, Test, and Placeholder.

Requirements can be added to a plan term by either clicking + on the requirement card in the Requirements list on the sidebar or by dragging and dropping the requirement card to the desired term

When adding a requirement, a dialog will open where the details of the requirement can be added. If the requirement card was dragged and dropped into a term, that term will load automatically. If added from the sidebar, the user can select the term to which the requirement should be added. The term drop down is all terms from STU016 with Show in SEP Picklist flag = Y that have not already been added to the plan. The student's active term and past terms will be filtered out of this list if the studentPlanner.planner.allowChangesToCurrentTerms and studentPlanner.planner.allowChangesToPastTerms Shepherd settings are false.

When studentPlanner.courseReq.showCritical is true, the critical check box will display for users that have the SEPCRIT Shepherd key on all requirement types except Placeholder. It is used to identify requirements that are critical for tracking.

To edit a requirement, you can select **Edit this requirement** from the action menu in the right corner of a requirement.

To delete a requirement, you can select **Delete this requirement** from the action menu in the right corner of a requirement.

To move a requirement to another term, you can select **Reassign this requirement** from the action menu in the right corner of a requirement or drag and drop it to the desired term.

Control over whether requirements on past or current terms can be modified or deleted can be managed with the studentPlanner.planner.allowChangesToCurrentTerms and studentPlanner.planner.allowChangesToPastTerms Shepherd settings. These settings apply to all users regardless of the access granted by Shepherd keys. Note that when tracking is enabled, these settings only apply to ONTRACK requirements. OFFTRACK and not tracked requirements can always be modified or deleted.

Configuration

- Review the Show in SEP Picklist flag in STU016 to ensure it is configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - studentPlanner.courseReq.showCritical

- studentPlanner.planner.allowChangesToCurrentTerms
- studentPlanner.planner.allowChangesToPastTerms
- studentPlanner.planner.tracking.enable
- Access is granted to the following if the user has the specified Shepherd key:
 - Add, modify, and delete requirements - SEPPADD or SEPPMOD
 - Modify the critical indicator on requirements - SEPCRIT

Course

Updated: September 30, 2022

A single class can be planned for on a plan with the course requirement.

Individual courses that need to be planned for to complete a degree program can be added to a plan. They can be used as criteria in tracking and apply to planner audit rules.

In addition to adding individual course requirements from the Requirements menu on the sidebar, multiple course requirements can be added to a term using the requirement drawer. This is accessed by clicking + in a term. Course requirements can also be added by drag and dropping from the Courses and Still Needed lists in the sidebar.

When adding course requirements from the requirement drawer, the user can filter the list of all courses by subject or by course title. Additionally, a list of all requirements that are still needed can be viewed. One or more requirements can be selected while in the requirement drawer, and then added at one time by clicking **Add to plan**.

When adding a course requirement from the Requirements list on the sidebar, enter a string of either the course discipline or course number in the **Course requirement** field. This will start to filter the list of matching courses, which will display in a drop-down below the field. You can continue typing to further refine this list or click **Load more** to see additional courses and select the desired course.

The credits will automatically load from the course record in the database (rad_course_mst). If the course is a variable credit course, the value in rad_credits (typically equal to rad_credits_high, per the extract) for that course is displayed, and the user can change the number of credits to a value within the range for that course. The format of the credits can be configured with the core.credit.format Shepherd setting.

Over time, a course's credits can change. However, when the value changes on the rad_course_mst, the number of credits on a requirement is not updated and this can cause a validation error when saving a plan. The studentPlanner.planner.updateCreditsIfChanged Shepherd setting can be used update the credits on the requirement and allow the user to save a plan. If this setting is set to false, the user will not be able to save the plan if the credits have changed. If it is set to true, the user can save the plan and the credits on the plan requirement will be updated with the value on the rad_course_mst. If it is set to warn, the user will get a warning that the credits are different when saving the plan, but the credits on the plan requirement will not be updated.

If `studentPlanner.courseReq.showHonors` is true, the **Honors** check box will display. The user can check this field to indicate if an honors section is needed. This field is informational only, but may be used for reporting.

If `studentPlanner.courseReq.showGrade` is true, the minimum grade field will display. The user can select the minimum grade needed for the course to be used as criteria in tracking. The values in this drop-down come from the grades in STU385 with Show in Student Educational Planner Picklist flag = Y.

If `studentPlanner.courseReq.showCampus` is true, the campus field will display. The user can select the campus where the course should be taken. This field is informational only, but may be used for reporting. The values in this drop-down come from the location codes in STU576 with Show Student Educational Planner Picklist Show in SEP Picklist flag = Y.

If `studentPlanner.courseReq.showDelivery` is true, the delivery field will display. The user can select a content delivery method for the course. This field is informational only, but may be used for reporting. The values in this drop-down come from the delivery codes in SEP003 with Show in SEP Picklist flag = Y.

When a course is added to a plan more than once, the user will get a warning letting them know that the course already exists. This warning is just information and can be overridden by any user.

Note that the course requirement search results can be filtered by the school associated with the plan and the course end date. When `studentPlanner.planner.filterCoursesByPlanSchool` is true, only courses with a DW-SCHOOL `rad_crs_attr_dtl` equal to the school on the plan will appear in the list. When `studentPlanner.planner.filterCoursesNoLongerOffered` is true, if the student's active term is greater than `rad_course_mst.rad_cat_yr_stop`, the course will not appear in the list. After enabling either filtering by school or by course end date, the course extract needs to be run to generate the data required for this filtering.

Configuration

- Review the Show in SEP Picklist flag in SEP003, STU576, and STU385 to ensure they are configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - `core.credit.format`
 - `studentPlanner.courseReq.showCampus`
 - `studentPlanner.courseReq.showDelivery`
 - `studentPlanner.courseReq.showGrade`
 - `studentPlanner.courseReq.showHonors`
 - `studentPlanner.planner.filterCoursesByPlanSchool`
 - `studentPlanner.planner.filterCoursesNoLongerOffered`
 - `studentPlanner.planner.updateCreditsIfChanged`

Choice

Updated: September 30, 2022

When there are several course options that can fulfill a planned requirement on a plan, a choice requirement can be used.

Choice requirements allow an unlimited number of course options to be defined and are generally used for program requirements that can be fulfilled by more than one course, such as general ed or core requirements. An option can be any combination of actual courses, wildcards, or ranges. For example, STAT 101 OR MATH @. Choice requirements can be used as criteria in tracking and apply to planner audit rules.

Each option can have up to two items. An item can be an actual course such as MATH 101, a wildcard such as @ @, MATH @, or MATH 3@, or a range such as MATH 100:299. When studentPlanner.choiceReq.enableCourseAttribute is true, a user can specify a course attribute on any item. The values in the attribute drop-down come from STU050. STU050 can be loaded through the extract, or manually by the user.

To add an option, you can click the **Add** link and enter a string of either the course discipline or course number in the **Course requirement** field. This will start to filter the list of matching courses, which will display in a drop-down below the field. You can continue typing to further refine this list or click **Load more...** to see additional courses.

To add an additional item to an option, you can click **Add a paired course or lab**.

With the SEPPSEL Shepherd key, one of the options may be selected by clicking the button next to the preferred option. Clicking the **Clear selection** button will remove the selection from the option.

The minimum credits for this requirement can be manually entered but are not required. The format and length of the minimum credits can be configured with the core.credit.format Shepherd setting. If a minimum credits value is entered, this will be included in the total number of planned course credits displayed on the term and used on the planner audit as the number of credits for the selected option. Note that if the selected option is a pair of courses, each course will use the minimum credits value as the number of credits on the planner audit. If the minimum credits is left blank, the number of credits on the rad_course_mst will be used for the selected option on the planner audit.

If studentPlanner.choiceReq.enableScribePointer is true, a user with the SEPPPTR Shepherd key can define a value that links this requirement to a rule in a requirement block. The values in this drop-down come from the pointer codes in SEP009 with Show in SEP Picklist flag = Y. See the [Scribe pointers](#) topic for additional information.

If studentPlanner.courseReq.showHonors is true, the **Honors** check box will display. The user can check this field to indicate if an honors section is needed. This field is informational only but may be used for reporting.

If studentPlanner.courseReq.showGrade is true, the minimum grade field will display. The user can select the minimum grade needed for the course to be used as criteria in tracking. The values in this drop-down come from the grades in STU385 with Show in SEP Picklist flag = Y.

If `studentPlanner.courseReq.showCampus` is true, the campus field will display. Users can select the campus where the course should be taken. This field is informational only but may be used for reporting. The values in this drop-down come from the location codes in STU576 with Show in SEP Picklist flag = Y.

If `studentPlanner.courseReq.showDelivery` is true, the delivery field will display. The user can select a content delivery method for the course. This field is informational only but may be used for reporting. The values in this drop-down come from the delivery codes in SEP003 with Show in SEP Picklist flag = Y.

When a course is added to a plan more than once, the user will get a warning that the course already exists. This warning is just information and can be overridden by any user.

Note that the course requirement search results be filtered by the school associated with the plan and the course end date. When `studentPlanner.planner.filterCoursesByPlanSchool` is true, only courses with a DW-SCHOOL `rad_crs_attr_dtl` equal to the school on the plan will appear in the list. When `studentPlanner.planner.filterCoursesNoLongerOffered` is true, if the student's active term is greater than `rad_course_mst.rad_cat_yr_stop`, the course will not appear in the list. After enabling either filtering by school or by course end date, the course extract needs to be run to generate the data required for this filtering.

Configuration

- Review STU050, in addition to the Show in SEP Picklist flag in SEP003, SEP009, STU576, and STU385 to ensure they are configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly for your SEP environment:
 - `core.credit.format`
 - `studentPlanner.choiceReq.enableCourseAttribute`
 - `studentPlanner.choiceReq.enableScribePointer`
 - `studentPlanner.courseReq.showCampus`
 - `studentPlanner.courseReq.showDelivery`
 - `studentPlanner.courseReq.showGrade`
 - `studentPlanner.courseReq.showHonors`
 - `studentPlanner.planner.filterCoursesByPlanSchool`
 - `studentPlanner.planner.filterCoursesNoLongerOffered`
- Access is granted to the following if the user has the specified Shepherd key:
 - Select a choice requirement option - SEPPSEL
 - Change a choice requirement pointer - SEPPPTR

GPA

Updated: September 30, 2022

Minimum overall, major, and class list GPAs required for a degree program can be planned for on a plan with the GPA requirement.

Four types of minimum GPAs for a term can be planned for with the GPA requirement: Overall Degree Works GPA, Overall Student System GPA, Major GPA, and a Class List GPA. These can be used as criteria in tracking.

The GPA types in the drop-down come from the codes in SEP008 with Show in SEP Picklist flag = Y.

The minimum GPA is a required field, and the format can be configured with the `core.gpa.format` Shepherd setting.

When adding a Major GPA, a major must be specified. The majors in the drop-down come from the major codes in STU023 with Show in Student Educational Planner Picklist flag = Y.

When adding a Class List GPA, a group of classes in which the student needs to have earned a minimum GPA can be defined. Each class should be separated by a plus sign, and ranges and wildcards can be included in the list. Classes are validated, but ranges and wildcards are not.

Configuration

- Review the Show in SEP Picklist flags in SEP008 and STU023 to ensure they are configured correctly.
- Review the `core.gpa.format` Shepherd setting to ensure it is configured correctly.

Non-course

Updated: September 30, 2022

Non-course items that need to be completed as part of a degree can be planned for on a plan with the non-course requirement.

Non-course requirements help students plan for items other than coursework that need to be completed for their program such as recitals or applications. They can be used as criteria in tracking and may apply to planner audit rules.

The non-course requirement types that are in the drop-down come from the codes in SCR003.

The status can be left blank to indicate that the student just needs to have completed the non-course item. It can also be the alphanumeric score or status needed to consider the requirement complete.

Configuration

Review SCR003 to ensure it is configured correctly.

Test

Updated: September 30, 2022

Competency or assessment tests that need to be completed as part of a degree can be planned for on a plan with the test requirement.

Test requirements help students plan for competency or assessment tests they need to take to complete their program. They can be used as criteria in tracking.

The test codes that are in the drop-down come from the codes in SEP006 with Show in SEP Picklist flag = Y.

The minimum score is a required field, and the length, type, and valid score range of the minimum score are defined in SEP006. Note that alpha scores require additional setup in SEP007 to define the sort order for these values.

If you will be using the Create Block functionality, your SEP006 Test Codes must also be in SCR002 Custom Data. If they are not, the creation of the block will fail with a parsing error.

Configuration

- Review the Show in SEP Picklist flag, Score Length, Score Type, Minimum Valid Score, and Maximum Valid Score fields in SEP006 to ensure they are configured correctly.
- Review the Sort Order in SEP007 to ensure that it is configured correctly.

Placeholder

Updated: September 30, 2022

User-defined requirements that need to be completed as part of a degree can be planned for on a plan with the placeholder requirement.

Placeholder requirements are intended to be informational only and are not included in tracking, nor do they apply to planner audit rules. However, they will display in the Planner Placeholders section of the planner audit. Typically, these are used to hold the place of a requirement that has not yet been selected, such as a general education or core requirement, and is replaced with the appropriate requirement after the student decides what they will take to complete that requirement.

The placeholder requirement types that are in the drop-down come from the codes in SEP005 with Show in SEP Picklist flag = Y.

The value is a free-form text field and is a required field.

Configuration

Review the Show in SEP Picklist flag in SEP005 to ensure it is configured correctly.

Notes creating and editing

Updated: September 30, 2022

An unlimited number of notes can be added to requirements, terms, or to the overall plan.

By default, all users have access to view all notes on plans, terms, and requirements that have not been marked as internal. Users can be given access to add, modify, or delete notes they have created on each of these plan components, and additional access can be given to users to modify or delete notes created by another user, internal notes, and notes copied from a template.

To create new notes on a plan, term, or requirement, users need the SEPPNADD Shepherd key. They will also be able to modify existing notes on a plan, term, or requirement that they have created. The SEPPNMOD Shepherd key can be assigned to users instead of SEPPNADD if they shouldn't have access to create new notes, but can modify existing notes on a plan, term, or requirement they have created. To delete notes on a plan, term, or requirement they have created, users should be assigned the SEPPNDEL Shepherd key.

After users have been given basic access to create, modify, and delete their own notes, they can be granted additional notes management access. If users have the SEPPNTOW Shepherd key, they will be able to modify notes created by other users, and if users have SEPPNTGD, they will be able to delete notes created by other users. Users with the SEPINOTE Shepherd key can view and modify notes that have been marked as internal. Additionally, the **Not available to student** check box will display in the note edit dialog for users with this key. Notes that came from the template used to create a plan have a special indicator that can be configured to display with the studentPlanner.planner.notes.showComesFromTemplateIndicator Shepherd setting. These notes are viewable by all users but can only be modified by users with the SEPPNTET Shepherd key or deleted by users with the SEPPNTDT Shepherd key.

To view notes on a plan, term, or requirement, you can click on the note icon to open the note list window. You can choose to view, add, edit, or delete notes, depending on your level of access. You can select **View note** from the action menu in the right corner of a note to see a particular note.

You can add a note by clicking **Add a new note**. Notes can have an unlimited amount of text. To make a note internal, you can select the **Not available to student** check box. You can click **Save note** to keep your note, or **Cancel** to go back to the note list window without keeping your note.

You can edit a note by selecting **Edit note** from the action menu in the right corner of a note.

You can delete a note by selecting **Delete note** from the action menu in the right corner of a note.

Configuration

- Review the studentPlanner.planner.notes.showComesFromTemplateIndicator Shepherd setting to ensure it is configured correctly for your SEP environment.
- Access to the following is granted if the user has the specified Shepherd key:
 - Add and modify your own plan, term and requirement notes - SEPPNADD
 - Modify your own plan, term and requirement notes - SEPPNMOD
 - Delete your own plan, term and requirement notes - SEPPNDEL

- Modify notes copied from a template - SEPPNTET
- Delete notes copied from a template - SEPPNTDT
- Modify notes created by other users - SEPPNTOW
- Delete notes created by other users - SEPPNTGD
- View, create and modify internal notes - SEPINOTE

Sidebar

Updated: September 30, 2022

The Plans sidebar panel displays lists of all courses available, courses the student still needs to plan, and requirement types that can be added to a plan.

Courses from either the Still Needed list or the Courses list in addition to requirement types can be dragged and dropped into a term on the plan.

The Requirements list will always display in the sidebar. When `studentPlanner.planner.sidebar.enableCourses` is true, the Courses list will display in the sidebar. When `studentPlanner.planner.sidebar.enableStillNeeded` is true, the Still Needed list will display in the sidebar.

You can configure whether the Courses list or the Still Needed list should load as the default with the `studentPlanner.planner.sidebar.coursesOpenedByDefault` Shepherd setting. Additionally, the sidebar can be configured to initially load as collapsed or expanded with the `studentPlanner.planner.sidebar.sidebarOpenedByDefault` Shepherd setting.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- `studentPlanner.planner.sidebar.coursesOpenedByDefault`
- `studentPlanner.planner.sidebar.enableCourses`
- `studentPlanner.planner.sidebar.enableStillNeeded`
- `studentPlanner.planner.sidebar.sidebarOpenedByDefault`

Courses

Updated: September 30, 2022

The Courses list displays courses that can be added as requirements to a student's plan.

The Courses list by default shows all courses in the `rad_course_mst` ordered by discipline where the discipline code is in `STU352`. Clicking on the arrow next to the discipline will expand the list to show all available courses in that discipline. If there are more than 10 courses for the discipline, a pagination drop-down will display allowing the user view the additional courses.

You can narrow the list of courses that display by searching by course discipline, course number or title. To clear your search, you can click the **x** in the search field.

The Courses list can be filtered by the school associated with the plan and the course end date. When `studentPlanner.planner.filterCoursesByPlanSchool` is true, only courses with a DW-SCHOOL `rad_crs_attr_dtl` equal to the school on the plan will appear in the list. When `studentPlanner.planner.filterCoursesNoLongerOffered` is true, if the student's active term is greater than `rad_course_mst.rad_cat_yr_stop`, the course will not appear in the list. After enabling either filtering by school or by course end date, the course extract needs to be run to generate the data required for this filtering.

Configuration

- Review STU352 to ensure it is configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - `studentPlanner.planner.filterCoursesByPlanSchool`
 - `studentPlanner.planner.filterCoursesNoLongerOffered`

Still Needed list

Updated: September 30, 2022

The Still Needed list displays remaining courses in a student's program that can be added as requirements to their plan.

The Still Needed list is a mini planner audit that shows only the requirements that are still needed to complete the plan's associated program. The requirements are organized by blocks and labels, and the still needed courses can be dragged and dropped into a term on the plan. When all requirements on a rule have been planned for, the rule will no longer appear in the Still Needed list.

The Still Needed list does not show remarks, proxy-advice, or other non-course information that displays on the audit because these items cannot be added to a plan. However, you can run the planner audit and move that browser tab to the side of the plan browser tab so you can reference that information.

Configuration

Access to generate the still needed audit is granted if the user has the SEPPAUD Shepherd key.

Requirements

Updated: September 30, 2022

The Requirements list displays all the requirement types that can be used on a plan.

The Requirements list is a group of requirement cards that can be used to add requirements to a plan. The options in this list come from the requirement types in SEP004 with Show in SEP Picklist flag = Y. The descriptions for these requirement types can be modified, but the keys should

not be changed or deleted. There are 6 requirement types: Course, Choice, GPA, Non-Course, Test, and Placeholder.

Configuration

Review the Show in SEP Picklist flag in SEP004 to ensure it is configured.

Requirement drawer

Updated: September 30, 2022

The requirement drawer allows multiple courses from the Courses list or the Still Needed list to be added to a term on a plan at one time.

To access the requirement drawer, you can click **+** in a term on a plan. From this window, you can select multiple courses from either the Courses list or the Still Needed list by checking the check box in front of the course. Each course you select will display as a chip at the bottom of the drawer. After selecting all of the courses to add to the term, you can click the **Add to plan** button.

You can narrow the list of courses that display by either selecting a discipline from the Subjects drop-down or by searching by title. To clear your search, you can click **x** in the search field.

When `studentPlanner.planner.sidebar.enableCourses` is true, the Courses list will display in the requirement drawer. When `studentPlanner.planner.sidebar.enableStillNeeded` is true, the Still Needed list will display in the requirement drawer. You can configure whether the Courses list or the Still Needed list should load as the default with the `studentPlanner.planner.sidebar.coursesOpenedByDefault` Shepherd setting.

See the [Courses](#) and [Still Needed list](#) topics for additional information.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- `studentPlanner.planner.sidebar.enableCourses`
- `studentPlanner.planner.sidebar.coursesOpenedByDefault`
- `studentPlanner.planner.sidebar.enableStillNeeded`

Planner audit

Updated: September 30, 2022

Generating audits from students' plans will show their progress towards their degree if all of their planned requirements are completed successfully.

Users will have access to generate planner audits if they have the SEPPAUD Shepherd key.

Class requirements will be applied to rules, as will Choice requirements in which a selection has been made or that have a Scribe pointer associated with a rule. Placeholder requirements will display in the Planned Placeholders section of the planner audit. Test and GPA requirements will not be applied to the audit. Non-course requirements will also be displayed in the planner audit, in rules or in fall-through.

For historical terms, only actual classes will appear on the planner audit; planned requirements in these terms will not be applied. Only planned requirements for future terms will apply on the planner audit. However, when `studentPlanner.planner.audit.usePlannedCoursesOnActiveTerm` is true, students' active terms will include both classes for which they have registered and any planned courses in that term for which they have not registered. In the special situation where a student has not registered for classes but does have planned requirements in an active term, the planned requirements will be applied to the planner audit regardless of how `studentPlanner.planner.audit.usePlannedCoursesOnActiveTerm` is configured.

If `studentPlanner.planner.audit.usePreregistered` is true, the planner audit will show these classes for future terms rather than the planned course.

Student goals and custom data are displayed at the top of the planner audit. See the [Student Data](#) topic for information about how to configure the `dash.worksheets.goals.items` and `dash.worksheets.custom.items` Shepherd settings and their corresponding properties.

Configuration

- Review RPT036 to ensure it is configured correctly for SEP31.
- Review the following Shepherd settings to ensure they are configured correctly:
 - `dash.worksheets.goals.items`
 - `dash.worksheets.custom.items`
 - `dash.worksheets.custom.enabled`
 - `studentPlanner.planner.audit.usePlannedCoursesOnActiveTerm`
 - `studentPlanner.planner.audit.usePreregistered`
- Access to generate a planner audit is granted if the user has the SEPPAUD Shepherd key.

Planner What-If audit

Updated: September 30, 2022

Planner What-If audits allow the student to forecast how their planned and completed coursework would apply to a different goal.

A user will have access to generate planner What-If audits if they have the SEPPWIF Shepherd key.

You can click the **What-If** link on a plan page to open the Planner What-If Analysis dialog window.

Planned requirements will apply to the planner What-If audit as they do on the planner audit. However, Planner What-If audits are not saved to the database. See the [Planner audit](#) and [What-If](#) topics for the various configuration options available.

Configuration

Access to the Planner What-If functionality is granted if the user has the SEPPWIF Shepherd key.

Create Block

Updated: March 24, 2023

Create Block can generate a requirement block from a student's plan.

Users have access to create a Scribe block from a plan if they have the SEPPBLCK Shepherd key.

This is useful when the degree requirements are decided on a student-by-student basis as is the case in many graduate programs and eliminates the need to build individualized blocks in Scribe.

After creating a plan for a student, you can click **Create Block** to save the plan and generate an OTHER requirement block with one rule for each requirement listed in the plan. The value and title for this OTHER block can be configured by the site. If these settings are left blank, the default values of PLAN and Planner Block are used. The student's catalog year is used as the start and stop catalog years on the block, and the student's ID is added as a secondary tag to block. A degree block that calls this OTHER block must exist to generate an audit for this program.

Example

For an MBA degree, you would create a DEGREE=MBA [Create Block](#) block in Scribe and then call in the OTHER=PLAN block.

```
1 Block (OTHER=PLAN)
  Label "Educational Plan requirements";
```

When a new audit is run, the correct OTHER=PLAN block with the specific student ID as the secondary tag will be pulled into the audit.

If the plan is modified, you can click **Create Block** again to regenerate the block. This will overwrite the block that had been previously saved.

After a block is created, Scribe may be used to modify it. If Create Block is used to regenerate the block after Scribe changes have been made, users will be asked to confirm that they want to overwrite the changes made in Scribe with the Create Block modifications.

Blocks created using this option have a block ID that begins with RB instead of RA.

The requirements on a plan can be configured to be grouped together by term in a subset in the block, or they can be single rules in the block. If subsets are used, the text preceding the term literal in the block can be defined by the site.

Course requirements can be configured to use either classes or credits in the rules, and whether to include the number of credits as part of the course title. The text preceding a Placeholder requirement can be defined by the site.

Test requirements will be created as If statements using RuleComplete/RuleIncomplete to evaluate if the minimum score was achieved.

Non-course requirements will be created as a NonCourse rule in the block. If the requirement value is blank, the rule will be written as follows:

```
1 Noncourse (THESIS)
  Label 14 "THESIS required";
```

If the requirement value is alpha, the rule will be written as follows:

```
1 Noncourse (THESIS = XYZ)
  Label 14 "THESIS with a value of XYZ required";
```

If the requirement value is numeric, the rule will be written as follows:

```
1 Noncourse (THESIS >= 123)
  Label 14 "THESIS with a score of at least 123 required";
```

Overall and Major GPA requirements will be created as Remarks in the block, while Class GPA requirements will be written as header qualifiers.

Choice requirements will be created as Group rules in the block. If an attribute is specified on the choice option, these will be written to the block as well.

Configuration

- Review the following Shepherd settings to ensure they are configured correctly:
 - studentPlanner.planner.createBlock.classIndent.numberOfSpaces
 - studentPlanner.planner.createBlock.dateFormat
 - studentPlanner.planner.createBlock.hourFormat
 - studentPlanner.planner.createBlock.labelIndent.numberOfSpaces
 - studentPlanner.planner.createBlock.placeholderLabelPrefix
 - studentPlanner.planner.createBlock.planBlockTitle
 - studentPlanner.planner.createBlock.planBlockValue
 - studentPlanner.planner.createBlock.showCreditsInRuleLabel (applies to course requirements only)
 - studentPlanner.planner.createBlock.subsetLabelPrefix
 - studentPlanner.planner.createBlock.subsetsForTerms
 - studentPlanner.planner.createBlock.useClassesInRules

- Access to the Create Block is granted if the user has the SEPPBLCK Shepherd key

Plan approval

Updated: September 30, 2022

Plan approval can be managed using either locking or Banner Workflow.

Locking

When `studentPlanner.planner.planApprovalMethod = L`, the locking approval method will be enabled. Each plan will have a status that indicates if the plan is locked or unlocked. Typically, a student's active plan is locked after it has been approved by their advisor, and no additional changes can be made to the plan by the student.

The locked status is manually maintained and can be changed only by users that have the SEPPLOCK Shepherd key. Users who does not have access to change the locked status will not be able to modify a locked plan; however, they will be able to save their changes as a new plan.

Banner Workflow

When `studentPlanner.planner.planApprovalMethod = W`, Banner Workflow can be used to manage plan approvals. Users with the SEPPADD or SEPPMOD Shepherd keys will be able to submit new plans for approval. While a plan is in a pending approval status, the plan cannot be modified by any user. If a user needs to make changes, they will have to save their modified plan as a new plan. In Banner Workflow, the plan approver(s) either approves or rejects the plan, which will update the approval status on the plan. Approved plans cannot be modified if `studentPlanner.planner.allowChangesToApprovedPlans` is false. Rejected plans can be modified and resubmitted for approval. See the [Configure Banner Workflow integration in](#) topic for additional information on configuring Banner Workflow for Degree Works.

If users have the SEPPAUTO Shepherd key, they will have an option to save and approve the plan from Plans, bypassing the Workflow process.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- `studentPlanner.planner.allowChangesToApprovedPlans`
- `studentPlanner.planner.planApprovalMethod`

Access to the auto approve plans when using Banner Workflow is granted if the user has the SEPPAUTO Shepherd key.

Course validation

Updated: September 30, 2022

In which terms courses will, or will not, be offered can be configured for Plans.

When a course is added to a Course or Choice requirement, it will be validated against CFG072 if the `studentPlanner.planner.addOnlyCoursesOfferedInTerm` setting is true. If the course is not valid for that term, an error message is displayed. The user will be given the option to add the course even though it is not being offered in that term, if the user has the SEPPCOFF key.

For additional information on how to define when a course will be offered, see the [UCX-CFG072 Planned Courses](#) topic.

Configuration

- Review CFG072 to ensure it is configured correctly.
- Review the `studentPlanner.planner.addOnlyCoursesOfferedInTerm` Shepherd setting to ensure it is configured correctly.
- Access to add a course to a term in which it is not offered is granted if the user has the SEPPCOFF Shepherd key.

Prerequisite checking

Updated: September 30, 2022

Using prerequisite checking helps ensure that courses are not added to a plan if pre- or co-requisite courses are missing.

For information about configuring and using Scribe blocks for prerequisite checking in plans, see [Prerequisite Checking](#).

Tracking

Updated: September 30, 2022

Tracking is a tool that helps students complete requirements and milestones for their degree program by the term for which they planned and to complete their degree program on time.

Tracking will be assessed on an active and locked or approved (depending on `studentPlanner.planner.planApprovalMethod`) plan when it is created or modified if it has trackable requirements. Only course, choice, GPA, non-course, and test requirements in the current term and in past terms are evaluated against completed coursework to determine if they satisfy the planned requirement. A tracking status is calculated for trackable requirements, terms, and the overall plan. Whether a requirement is trackable is determined by the `studentPlanner.planner.tracking.trackCriticalRequirementsOnly` Shepherd setting. When this setting is true, a requirement needs to have the critical check box selected to be trackable. Any requirement that does not have the critical check box checked is skipped. When this setting is false, all requirements are trackable and will be processed.

Tracking is enabled by setting `studentPlanner.planner.tracking.enable` to true. There are three tracking statuses: ONTRACK, OFFTRACK, and NOTEVALUATED.

When a plan is created, all tracking statuses are initially set to NOTEVALUATED. On a tracked plan, future terms and requirements will have a NOTEVALUATED tracking status, as will any requirements that are not trackable or terms that do not have any trackable requirements.

ONTRACK trackable requirements are those that were completed satisfactorily by the planned term. If a term does not have any OFFTRACK trackable requirements, it will be ONTRACK. A plan is ONTRACK if the number of OFFTRACK terms is less than the value defined in the `studentPlanner.planner.tracking.offTrackTermsAllowed` Shepherd setting.

A trackable requirement will be OFFTRACK if it was not completed satisfactorily by the planned term. When `studentPlanner.planner.tracking.trackCriticalRequirementsOnly` is false, OFFTRACK trackable requirements that are not critical will display a WARNING tracking status rather than OFFTRACK. One or more OFFTRACK requirements will make a term OFFTRACK, and a plan will be OFFTRACK if the number of OFFTRACK terms is greater than the value defined in the `studentPlanner.planner.tracking.offTrackTermsAllowed` Shepherd setting.

When `studentPlanner.planner.tracking.showTrackingText` is true, the plan, term and requirement tracking statuses will display on the plan. The status text can be localized with the `planner.tracking.status.*` properties in `PlannerMessages.properties`. Note that the tracking status labels will always be uppercase.

Note: When Banner Workflow is being used as the approval method for a plan, and when the plan is transitioning from a Pending Approval to an Approved state, the tracking status is not evaluated. The tracking status on the requirement, term, and overall plan will not be automatically updated in such a scenario, and DAP58 should be run to update tracking on these plans in batch.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- `studentPlanner.planner.tracking.enable`
- `studentPlanner.planner.tracking.offTrackTermsAllowed`
- `studentPlanner.planner.tracking.showTrackingText`
- `studentPlanner.planner.tracking.trackCriticalRequirementsOnly`

Current term

Updated: September 30, 2022

Tracking can either use a student's active term or calculate it from date values in STU016.

By default, the student's active term is used as the current term by tracking. However, when `studentPlanner.planner.currentTerm.isDeterminedFromUcx` is true, the current term for all students can be calculated based on a term start and stop date. The current date is compared to the Term Start Date and Term Stop Date values in STU016. The term in which the current date falls between that defined start and stop dates is then used as the current term for tracking.

Configuration

Review the `studentPlanner.planner.currentTerm.isDeterminedFromUcx` Shepherd setting to ensure it is configured correctly.

Requirement tracking status

Updated: September 30, 2022

A tracking status is calculated for course, choice, GPA, non-course, and test trackable requirements.

Tracking will be evaluated for all requirements when the `studentPlanner.planner.tracking.trackCriticalRequirementsOnly` Shepherd setting is false or only for requirements that have been marked as critical when `studentPlanner.planner.tracking.trackCriticalRequirementsOnly` is true.

When the `studentPlanner.planner.tracking.showTrackingForCriticalRequirementsOnly` Shepherd setting is false, the tracking status will display for all requirements on a plan. However, when `studentPlanner.planner.tracking.showTrackingForCriticalRequirementsOnly` is true, only the tracking status for critical requirements will be displayed on a plan, even if all requirements are included in the actual tracking evaluation. For this reason, it is recommended that `studentPlanner.planner.tracking.showTrackingForCriticalRequirementsOnly` be set to false.

If a trackable requirement was completed in or before the planned term, and the minimum grade, GPA, or score (if applicable) was met, the requirement will be ONTRACK. If the requirement was completed in a later term, did not meet the minimum grade, GPA, or score (if applicable), or was not completed at all, the requirement will be OFFTRACK.

Course and choice requirement tracking

Updated: September 30, 2022

If a class was completed in or before the planned term, and at least the minimum grade was earned (if the requirement has a minimum grade), a trackable course or choice requirement will be ONTRACK.

If a trackable course or choice requirement was completed in a later term, did not meet the minimum grade or was not completed at all, it will be OFFTRACK.

By default, course and choice requirements must be taken by the planned term to be ONTRACK. When the `studentPlanner.planner.tracking.allowTakenTermDifferentThanPlanned` Shepherd setting is true, tracking allows forgiveness for a student who did not take a class by the planned term but did ultimately take it. If a planned class was taken before, during, or even after the planned term, the term and the plan would still be considered ONTRACK.

When a class is in-progress and not graded, tracking will evaluate it as passed with the minimum grade met so a trackable requirement won't be OFFTRACK. Then when tracking is triggered after the class is graded, the status will be re-evaluated and if the minimum grade on the requirement was not met, the requirement will then be OFFTRACK.

When the `studentPlanner.planner.tracking.passFailClassMeetsMinimumGrade` Shepherd setting is true, pass/fail classes will meet the minimum grade on a course or choice requirement. When the

studentPlanner.planner.tracking.blankGradeTransferClassMeetsMinimumGrade Shepherd setting is true, transfer classes with blank grades will meet the minimum grade on a course or choice requirement.

When the studentPlanner.planner.tracking.incompleteClassMeetsRequirement Shepherd setting is true, incomplete classes will satisfy a course or choice requirement as a completed class would.

There is no limit to the number of requirements that a course can apply to during tracking. The intent of tracking is to determine if a requirement was completed by a defined point-in-time, as defined in the plan. For example, if the plan contains two requirements such as ENGL 101 (A course requirement) and ENGL @ (A choice requirement) and the student takes ENGL 101, both of these requirements would be considered on-track.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- studentPlanner.planner.tracking.allowTakenTermDifferentThanPlanned
- studentPlanner.planner.tracking.blankGradeTransferClassMeetsMinimumGrade
- studentPlanner.planner.tracking.incompleteClassMeetsRequirement
- studentPlanner.planner.tracking.passFailClassMeetsMinimumGrade

GPA requirement tracking

Updated: September 30, 2022

If the planned minimum GPA value is less than or equal to the actual GPA value, a GPA requirement will be ONTRACK.

To evaluate the Overall Degree Works GPA and Major GPA, the planned minimum GPA on the requirement is compared to the actual GPA values calculated by the Degree Works auditor for that term. These values are stored in dap_gpa_history when CFG020 DAP14 Save GPA History = Y. If a dap_gpa_history record cannot be found for the planned term and GPA type, the requirement will be evaluated as OFFTRACK.

For a Class List GPA, a combined GPA for all the classes in the requirement list that have been completed in or by that term is calculated by tracking. This is stored on the GPA requirement, and if the calculated combined GPA is equal to or greater than the minimum GPA on the requirement, the requirement will be evaluated as ONTRACK. If the Class List is a wildcard or a range, the student must have at least one class that falls within the wildcard or range in order for the GPA to be calculated. Note that because in-progress classes do not have a grade, they will not be included in the Class List GPA calculation. Transfer classes will be included in the GPA calculation.

Note that GPA requirements on a student's current term will often be evaluated as OFFTRACK because classes in that term are still in progress and have not been graded. You can force all GPA requirements on the current term to be evaluated as ONTRACK by setting studentPlanner.planner.tracking.currentGpaMeetsRequirement to true. When this setting is true, validation of Overall Degree Works GPA, Major GPA and Class List GPA requirements will be skipped and these requirements will be ONTRACK. When this term becomes a historical term and grading is complete, tracking on these requirements will be evaluated based on the criteria specified above.

Configuration

- Review the Save GPA History flag in CFG020 DAP14 to ensure it is configured correctly.
- Review the studentPlanner.planner.tracking.currentGpaMeetsRequirement Shepherd setting to ensure it is configured correctly.

Non-course requirement tracking

Updated: September 30, 2022

A non-course requirement will be considered ONTRACK if it was completed in or before the planned term.

If a status is specified, this must be equal to the non-course score in order for the requirement to be considered ONTRACK.

Test requirement tracking

Updated: September 30, 2022

A test requirement will be considered ONTRACK if it was completed during or before the planned term and the minimum score was earned.

If a test requirement has non-numeric scores, SEP007 needs to be configured for this test code so that tracking is able to evaluate whether the minimum score was met.

Configuration

Review the Sort Order in SEP007 to ensure it is configured correctly.

Term tracking status

Updated: September 30, 2022

If one or more trackable requirements in a term is OFFTRACK, the term will be considered OFFTRACK.

Trackable requirements are either only critical requirements or all requirements, depending how the studentPlanner.planner.tracking.trackCriticalRequirementsOnly Shepherd setting is configured.

Overall plan tracking status

Updated: September 30, 2022

A plan will be considered OFFTRACK if the number of OFFTRACK terms on the plan is equal to or greater than the number of OFFTRACK terms allowed.

Configuration

Review the studentPlanner.planner.tracking.offTrackTermsAllowed Shepherd setting to ensure it is configured correctly.

Batch processes

Updated: September 30, 2022

The Student Educational Planner relies on three batch processors for data creation and management.

DAP54 - Template processor plan creation

Using DAP54, you can create plans from templates for a group of students who do not already have a plan. A specific template can be specified that will be applied to all students, or the processor can find the best template for students based on matching template tags and student goal data. Temporary plans for students without an active and approved plan can also be created and used in reporting. For more information, see the [DAP54 - Template processor plan creation](#) topic.

DAP58 - Batch tracking processor

DAP58 will update the tracking status on active plans for a group of students. Running the processor in the official mode calculates the requirement, term, and tracking statuses that are then visible on a plan in Responsive Dashboard. By using the unofficial mode of DAP58 with a cutoff term that is in the future, tracking statuses can be forecasted and used in advising reports. For more information, see the [DAP58 - Batch tracking processor](#) topic.

DAP59 - Batch timetabling processor

DAP59 generates reporting data by running audits against student plans and storing the results in the CPA tables. This data can then be used for reporting and schedule forecasting. For more information, see the [DAP59 - Batch timetabling processor](#) topic.

Scribe pointers

Updated: September 30, 2022

Using Scribe pointers, a choice requirement on a plan can be linked to a rule in the student's requirement blocks so that it will satisfy a rule on the planner audit.

Unless a selection has been made on a choice requirement, it will not apply to any rules on the planner audit and it can be difficult to determine if all program requirements have been planned. Using Scribe pointers, if a selection hasn't been made on a choice requirement, the auditor will apply a course to the associated rule, allowing the choice requirement to count towards the program requirements. However, if a selection has been made on a choice requirement, that option will be sent to the auditor instead. If the Scribe pointer is blank or a matching rule can't be found, the auditor will not apply the choice requirement in the planner audit.

When `studentPlanner.choiceReq.enableScribePointer` is true, the Scribe pointer will display on a choice requirement. A user with the SEPPPTR Shepherd key can assign a Scribe pointer from the options in the pointer drop-down. The values in the drop-down are from the codes in SEP009 with Show in SEP Picklist flag = Y.

In addition to assigning a Scribe pointer to a choice requirement, you must add the SEPPOINTER reserved word to the appropriate rules in your requirement blocks. The auditor treats the SEPPOINTER reserved word as a course discipline and the Scribe pointer value as a course number. SEPPOINTER can be used in IF and HIDE statements to associate the Scribe pointer with a rule. Using a wildcard on a Scribe pointer is also allowed.


There are several ways to Scribe a rule with a pointer, depending on what results should appear on the audit. Some examples and the expected output are listed below.

Single class rules example 1

When written as follows, this option allows for a different label on the rule if an unselected choice requirement is applied versus a selected choice requirement or actual class.

```
If (MATH 101 wasnt passed and MATH 102 wasnt passed and MATH 103 was
nt passed and
  SEPPOINTER COREMATH was FOUND) then
  1 class in SEPPOINTER COREMATH
  Label "Core Mathematics Choice - Example 1"
Else
  3 credits in MATH 101, MATH 102, MATH 103
  Label "Core Mathematics - Example 1";
```

Then an unselected choice requirement with a COREMATH Scribe pointer will show on the audit.

	Course	Title	Grade	Credits	Term
 Core Mathematics Choice - Example 1	SEPPOINTER COREMATH	Requirement planned but choice not selected	PLAN	(3)	Fall 2023

Note that the number of credits shown on the audit will come either from the Minimum Credits field on the choice requirement or default to 3.

Using the scribing above, the appropriate MATH class will apply to the 3 Credits requirement even with a COREMATH SEPPOINTER in place when the student takes the MATH class. When no SEPPOINTER is in place and when no MATH class was taken, the default “3 Credits in MATH ...” will appear.

Single class rules example 2

When written as follows, this option will show the rule as completed when an unselected choice requirement is applied.

```
If (SEPPOINTER COREMATH was FOUND) then
  RuleComplete
  Label "Core Mathematics Choice planned for but not selected"
Else
  3 credits in MATH 101, MATH 102, MATH 103
  Label "Core Mathematics - Example 2";
```

Then an unselected choice requirement with a COREMATH Scribe pointer will show on the audit.

	Course	Title	Grade	Credits	Term
✓	Core Mathematics Choice planned for but not selected				

Note that the choice requirement will apply to the Falthrough section and the credits would apply to the degree.

Multiple class rules example 3

When more than one class is needed to satisfy a rule, multiple choice requirements can be created to apply to this rule using sequential Scribe pointer values on the plan and a wildcard in the rule. Using the HIDE keyword will apply the unselected choice requirements to the rule.

```
3 Classes in MATH 101, 102, 103, 104, 105, 106, {HIDE SEPPINTER COREMATHa}
Label "Core Mathematics - Example 3";
```

Then an unselected CHOICE requirement with a COREMATH Scribe pointer will show on the audit.

	Course	Title	Grade	Credits	Term	Repeated
○	Core Mathematics - Example 3	SEPPINTER COREMATH1	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
Still needed:		2 Classes in MATH 101 or 102 or 103 or 104 or 105 or 106				

Multiple class rules example 4

In this option, using an IF statement allows for different labels if the rule is satisfied by an unselected choice requirement or an actual class.

```
If (SEPPINTER COREMATHa was FOUND) then
  3 Classes in MATH 101, 102, 103, 104, 105, 106, {HIDE SEPPINTER COREMATHa}
  Label "Core Mathematics Choice - Example 4"
Else
  3 Classes in MATH 101, 102, 103, 104, 105, 106
  Label "Core Mathematics - Example 4";
```

Then an unselected choice requirement with a COREMATH Scribe pointer will show on the audit.

	Course	Title	Grade	Credits	Term
○	Core Mathematics Choice - Example 4	SEPPINTER COREMATH3	Requirement planned but choice not selected	PLAN	(3) Fall 2023
Still needed:		2 Classes in MATH 101 or 102 or 103 or 104 or 105 or 106			

Multiple class rules - example 5

When the student's plan has multiple unselected choice requirements that match the SEPPINTER value, they will be applied to the associated rule.

```
If (SEPPOINTER COREMATHa was FOUND) then
  3 Classes in MATH 101, 102, 103, 104, 105, 106, {HIDE SEPPOINTER C
OREMATHa}
  Label "Core Mathematics Choice - Example 5"
Else
  3 Classes in MATH 101, 102, 103, 104, 105, 106
  Label "Core Mathematics - Example 5";
```

If the student's plan has choice requirements with COREMATH1 and COREMATH2 Scribe pointers, the audit will look like this:

	Course	Title	Grade	Credits	Term
○ Core Mathematics Choice - Example 5	SEPPOINTER COREMATH1	Requirement planned but choice not selected	PLAN	(3)	Spring 2023
	SEPPOINTER COREMATH2	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
	Still needed: 1 Class in MATH 101 or 102 or 103 or 104 or 105 or 106				

Multiple class rules - example 6

If the student's plan has choice requirements with COREMATH1, COREMATH2, and COREMATH3 Scribe pointers, the audit will look like this:

	Course	Title	Grade	Credits	Term	Repeated
① Core Mathematics Choice - Example 6	SEPPOINTER COREMATH1	Requirement planned but choice not selected	PLAN	(3)	Fall 2022	
	SEPPOINTER COREMATH2	Requirement planned but choice not selected	PLAN	(3)	Spring 2023	
	SEPPOINTER COREMATH3	Requirement planned but choice not selected	PLAN	(3)	Fall 2023	

Group rules - example 7

Scribe pointers can be used in Group rules to allow an unselected choice requirement to apply to the best fit option by creating choice requirements on the student's plan for the Group rule options and associating sequential Scribe pointers to the requirements.

```
2 Groups in
  (3 Classes in MATH 101, 102, 103, 104, 105, 106, {HIDE SEPPOINTER
COREMATHa}
  Label "Core Mathematics - Example 7") OR
  (3 Classes in ART 101, 102, 103, 104, 105, 106, {HIDE SEPPOINTER C
OREARTa}
  Label "Core Humanities and Fine Arts - Example 7") OR
  (3 Classes in ENGL 101, 102, 103, 104, 105, 106, {HIDE SEPPOINTER
COREENGLa}
  Label "Core English - Example 7") OR
  (3 Classes in CHEM 101, 102, 103, 104, 105, 106, {HIDE SEPPOINTER
CORESCIa}
  Label "Core Science - Example 7")
Label "CORE REQUIREMENTS - Example 7";
```

If the student has a choice requirement for each of the corresponding Scribe pointers, the audit might look like this:

	Course	Title	Grade	Credits	Term
<input type="radio"/> CORE REQUIREMENTS - Example 7	Still needed:	Choose from 2 of the following:			
<input type="radio"/> Core Mathematics - Example 7	SEPPOINTER COREMATH1	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
		2 Classes in MATH 101 or 102 or 103 or 104 or 105 or 106			
<input type="radio"/> Core Humanities and Fine Arts - Example 7	SEPPOINTER COREART1	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
		2 Classes in ART 101 or 102 or 103 or 104 or 105 or 106			
<input type="radio"/> Core English - Example 7		3 Classes in ENGL 101 or 102 or 103 or 104 or 105 or 106			
<input type="radio"/> Core Science - Example 7		3 Classes in CHEM 101 or 102 or 103 or 104 or 105 or 106			

Group rules - example 8

If the student has more than one choice requirement for one of the Scribe pointers, the audit might look like this:

	Course	Title	Grade	Credits	Term
<input type="radio"/> CORE REQUIREMENTS - Example 8	Still needed:	Choose from 2 of the following:			
<input type="radio"/> Core Mathematics - Example 8	SEPPOINTER COREMATH1	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
		2 Classes in MATH 101 or 102 or 103 or 104 or 105 or 106			
<input type="radio"/> Core Humanities and Fine Arts - Example 8		3 Classes in ART 101 or 102 or 103 or 104 or 105 or 106			
<input type="radio"/> Core English - Example 8	SEPPOINTER COREENGL2	Requirement planned but choice not selected	PLAN	(3)	Fall 2022
	SEPPOINTER COREENGL3	Requirement planned but choice not selected	PLAN	(3)	Spring 2023
		1 Class in ENGL 101 or 102 or 103 or 104 or 105 or 106			
<input type="radio"/> Core Science - Example 8		3 Classes in CHEM 101 or 102 or 103 or 104 or 105 or 106			

Group rules - example 9

If the student has three choice requirement for one of the Scribe pointers, the audit might look like this:

	Course	Title	Grade	Credits	Term
○ CORE REQUIREMENTS - Example 9	Still needed: Choose from 1 of the following:				
○ Core Mathematics - Example 9	SEPPOINTER COREMATH1	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
	2 Classes in MATH 101 or 102 or 103 or 104 or 105 or 106				
○ Core Humanities and Fine Arts - Example 9	3 Classes in ART 101 or 102 or 103 or 104 or 105 or 106				
① Core English - Example 9	SEPPOINTER COREENGL1	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
	SEPPOINTER COREENGL2	Requirement planned but choice not selected	PLAN	(3)	Fall 2022
	SEPPOINTER COREENGL3	Requirement planned but choice not selected	PLAN	(3)	Spring 2023

Group rules - example 10

If the student has more than one choice requirement for one of the Scribe pointers and either taken or planned for some of the courses, the audit might look like this:

	Course	Title	Grade	Credits	Term
① CORE REQUIREMENTS - Example 10					
① Core Mathematics - Example 10	MATH 101	Quantitative Literacy	PLAN	(4)	Spring 2023
	MATH 103	Plane Trigonometry	PLAN	(3)	Fall 2022
	SEPPOINTER COREMATH1	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
① Core English - Example 10	SEPPOINTER COREENGL1	Requirement planned but choice not selected	PLAN	(3)	Fall 2023
	SEPPOINTER COREENGL2	Requirement planned but choice not selected	PLAN	(3)	Fall 2022
	SEPPOINTER COREENGL3	Requirement planned but choice not selected	PLAN	(3)	Spring 2023

Template Management

Updated: September 30, 2022

Template Management is an administrative tool to create and manage plan templates.

Templates define typical degree paths that will be used as the basis for creating a student's actual plan. The course list can help identify courses that can be added to a plan, in addition to GPA, test

score, non-course, and placeholder requirements. Plans can be created from templates either for an individual in the Plans tab or in batch by running DAP54 in Transit.

By default, a user assigned access to Template Management can just view existing templates. Additional authorization grants access to the other features.

Configuration

Access to Template Management is granted if the user has the SEPTMGMT Shepherd key.

Template list

Updated: September 30, 2022

In the template list, users can search for, select, and delete templates.

There are two view options, tree view and flat view, that can be configured. If both the `studentPlanner.template.showTreeView` and `studentPlanner.template.showFlatView` settings are true, users can switch to the other view by clicking the option button. Additionally, the view that displays initially can be configured in the `studentPlanner.template.showFlatViewByDefault` Shepherd setting. Note that if the `studentPlanner.template.showTreeView` and `studentPlanner.template.showFlatView` settings are both false, the flat view will load by default.

The list of templates that displays in each view can be filtered either by description or by template tags. You cannot filter by both description and template tags. The previous search will be cleared when a new search is initiated.

You can filter by description by entering a word or partial string that appears in the template description.

To filter by template tags, you can click on **Advanced search**. In the search window, select the tags you want to use in your search and the required values from the drop-down lists. Multiple values can be selected for filtering by checking more than one box. As a default, required template tags initially display in the Advanced Search window. Additional template tags to be used for filtering can be added by clicking **+** in the upper right, and tags that are not required can be removed by clicking the delete icon. All selected values can be cleared by clicking **Clear**. To filter based on selected template tags, you can click **Search**. You can go back to the template list by clicking **Cancel**.

You can view a template by clicking on the description link, create a new template by clicking **New Template**, and delete a template by selecting the template and clicking the delete icon.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- `studentPlanner.template.showFlatView`
- `studentPlanner.template.showFlatViewByDefault`
- `studentPlanner.template.showTreeView`

Flat view

Updated: September 30, 2022

The flat view displays all selected templates in a grid.

Templates in the flat view can be sorted on any column by clicking on the column header. The template description and required template tags (as defined in SEP001) will always be displayed, while the modified date, what, who, template ID, and term scheme can be configured to display.

Configuration

- Review the SEP001 Required flag for your template tags to ensure it is configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - `studentPlanner.template.columnsToDisplay.modifyDate`
 - `studentPlanner.template.columnsToDisplay.modifyWhat`
 - `studentPlanner.template.columnsToDisplay.modifyWho`
 - `studentPlanner.template.columnsToDisplay.templateId`
 - `studentPlanner.template.columnsToDisplay.termScheme`

Note: When you are filtering templates by template tags chosen in the advanced search, you are not allowed to sort by the template tag columns. That is, searching based on tags and sorting by tags are not allowed.

Tree view

Updated: September 30, 2022

The tree view presents the selected templates in a hierarchical tree, ordered by template tags.

The hierarchy is defined by SEP001 Hierarchy Order, and the number of open branches initially displayed in the tree can be configured with the `studentPlanner.template.initialDepth` Shepherd setting. Clicking the arrow next to a label will open the next branch in the hierarchy, and the template is at the final branch of the tree. All branches of the tree can be opened by clicking **Expand all**, while clicking **Collapse all** will close the tree down to the first branch.

Configuration

- Review the SEP001 Hierarchy Order for your template tags to ensure it is configured correctly.
- Review the `studentPlanner.template.initialDepth` Shepherd setting to ensure it is configured correctly.

Template creating and editing

Updated: September 30, 2022

A new template can be created while viewing a list of templates or a single template.

To create new templates, users need the SEPTADD Shepherd key. They will also be able to modify existing templates, add new terms and requirements on templates, and modify terms and requirements (including deleting individual terms and requirements).

The SEPTMOD Shepherd key can be assigned to users instead of SEPTADD if they shouldn't have access to create new templates, but can modify existing templates, add new terms and requirements on templates, and modify terms and requirements (including deleting individual terms and requirements).

To delete templates, including all terms, requirements, and notes on that template, users should be assigned the SEPTDEL Shepherd key.

For users with access to create a template, when they click the **New Template** button they can create a new template. They will be prompted to enter several details about template and after clicking **Save**, a blank template with the terms defined in the term scheme but no requirements will load.

A template description is required. This field is 80 bytes long and must be unique.

The term scheme determines how many terms will load on the template. For more information, see the [Term schemes](#) topic.

The active flag indicates whether a template is current and can be used to create plans from within Plans or through DAP54. If this is not checked, the template will not display in the Plans list of templates that can be used to create a plan, nor will it be used by DAP54.

Template tags can be assigned to a template to assist with organizing and searching for templates. Required template tags (as defined in SEP001) must be assigned, while optional template tags can be assigned to further enhance organization and searching, but the template can be saved without them.

Configuration

- Review the Required flag in SEP001 for your template tags to ensure it is configured correctly.
- Review SEP002 to ensure it is configured correctly.
- Access to the following is granted if the user has the specified Shepherd key:
 - Add and modify templates - SEPTADD
 - Modify templates - SEPTMOD
 - Delete templates - SEPTDEL

Template tags

Updated: September 30, 2022

Template tags are used to match a student to the best template when creating plans from templates through DAP54 and to organize templates in the tree view.

Template tags are defined in SEP001. The standard tags Ellucian delivers correspond to student goal data: CATYEAR, COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, PROGRAM, SCHOOL, and SPEC. This allows the best fitting template to be used when creating a plan for a student. For more information, see the [DAP54 - Template processor plan creation](#) topic.

In addition to these standard tags, user-defined tags can be created to be used when organizing or searching for templates. Note that these user-defined tags cannot be used when creating a plan from a template.

All template tags defined in SEP001 will display in the tag drop-down lists. Template tags that aren't used (for example, LIBL - Liberal Learning) can be deleted from SEP001 and they will no longer appear in the drop-downs. Setting **SEP001 Required** to Y will make the template tag required when creating or modifying a template, while **SEP001 Hierarchy Order** defines the order of the tags in the tree view.

Do not change the settings in SEP001 for **Validation Table** or **Match Student Goal Data?** for any of the standard template tags.

Configuration

Review SEP001 to ensure it is configured correctly.

Term schemes

Updated: September 30, 2022

Term schemes define a typical term sequence that should be used as the basis for a template.

A template defines the ideal path for completing the requirements for a degree program. This ideal also includes a prescribed number of terms. While a student's plan may have terms added or deleted, depending on their circumstances, a template has a set number of terms that define the best path. This set of terms makes up the term scheme for templates.

You can have several different term schemes to accommodate the different scenarios at your institution. For example, most of your programs may typically start in a fall term, but some may start in spring terms. Or, you may want a template to accommodate students that are transferring in a spring term. Or, there may be programs that will always require a winter or summer term.

Template term schemes are defined in SEP002. A term scheme needs a sequenced entry for each term that should appear on the template. So for a program that typically has 8 semesters, you would create 8 entries in SEP002 for your term scheme. A common mistake when creating SEP002 entries is putting the sequence number in the incorrect position. The sequence should always be in positions 28, 29, and 30. That is, your term scheme name should be 27 bytes, including trailing blanks, and followed by the 3-byte sequence. For example:

Correct: SEMESTER_FALL_START 001

Not correct: SEMESTER_FALL_START001

The SEP002 Literal displays as the term literal on the template. The SEP002 Term Type is used when creating a plan from a template to match the generic template term to an actual term from STU016, and must be either SUMMER, SPRING, FALL, or WINTER.

A user with the SEPTTRMS Shepherd key can change the term scheme on a template by clicking **Change term scheme** and selecting the new term scheme from the drop-down. If the new term scheme has fewer terms than the term scheme that was used to create the template, the additional terms and all requirements and notes on them will be deleted.

Configuration

- Review SEP002 to ensure it is configured correctly.
- Access to change a template term scheme is granted if the user has the SEPTTRMS Shepherd key.

Requirements creating and editing

Updated: September 30, 2022

An unlimited number of requirements can be added to a term on a template by a user who has access to modify a template.

A user with the SEPTADD or SEPTMOD Shepherd key can add, reassign, or delete requirements to a term on a template. There are six requirement types: Course, Choice, GPA, Non-course, Test, and Placeholder.

Requirements can be added to a template term by either clicking **+** on the requirement card in the Requirements list on the sidebar or by dragging and dropping the requirement card to the desired term.

When adding a requirement, a dialog will open where the details of the requirement can be added. If the requirement card was dragged and dropped into a term, that term will load automatically. If added from the sidebar, the user can select the term to which the requirement should be added. The term drop-down is all terms in the template term scheme.

When `studentPlanner.courseReq.showCritical` is true, the critical check box will display for users that have the SEPCRIT Shepherd key on all requirement types except Placeholder. It is used to identify requirements that are critical for tracking.

To edit a requirement, you can select **Edit this requirement** from the action menu in the right corner of a requirement.

To delete a requirement, you can select **Delete this requirement** from the action menu in the right corner of a requirement.

To move a requirement to another term, you can select **Reassign this requirement** from the action menu in the right corner of a requirement or drag and drop it to the desired term.

Configuration

- Review the studentPlanner.courseReq.showCritical Shepherd setting to ensure it is configured correctly.
- Access is granted to the following if the user has the specified Shepherd key:
 - Add, modify, and delete requirements - SEPTADD or SEPTMOD
 - Modify the critical indicator on requirements - SEPCRIT

Course

Updated: September 30, 2022

A single class can be planned for on a template with the course requirement.

Individual courses that need to be planned for to complete a degree program can be added to a template.

In addition to adding individual course requirements from the Requirements menu on the sidebar, multiple course requirements can be added to a term using the requirement drawer. This is accessed by clicking **+** in a term. Course requirements can also be added by drag and dropping from the Courses list in the sidebar.

When adding course requirements from the requirement drawer, the user can filter the list of all courses by subject or by course title. One or more requirements can be selected while in the requirement drawer and then added at one time by clicking **Add to template**.

When adding a course requirement from the Requirements list on the sidebar, you can enter a string of either the course discipline or course number in the **Course requirement** field. This will start to filter the list of matching courses, which will display in a drop-down below the field. You can continue typing to further refine this list or click **Load more...** to see additional courses.

The credits will automatically load from the course record in the database. If the course is a variable credit course, the value in rad_credits (typically this is equal to rad_credits_high, per the extract) for that course is displayed, and the user can change the number of credits to a value within the range for that course. The format and length of the credits can be configured with the core.credit.format Shepherd setting.

Over time, a course's credits can change. However, when the value changes on the rad_course_mst, the number of credits on a requirement is not updated and this can cause a validation error when saving a template. The studentPlanner.planner.updateCreditsIfChanged Shepherd setting can be used update the credits on the requirement and allow the user to save a plan. If this setting is set to false, the user will not be able to save the template if the credits have changed. If it is set to true, the user can save the template and the credits on the template requirement will be updated with the value on the rad_course_mst. If it is set to warn, the user will get a warning that the credits are different when saving the template, but the credits on the template requirement will not be updated.

If studentPlanner.courseReq.showGrade is true, the minimum grade field will display. The user can select the minimum grade needed for the course to be used as criteria in tracking. The values in this drop-down come from the grades in STU385 with Show in SEP Picklist flag = Y.

If `studentPlanner.courseReq.showCampus` is true, the campus field will display. The user can select the campus where the course should be taken. This field is informational only, but may be used for reporting. The values in this drop-down come from the location codes in STU576 with Show in Student Educational Planner Picklist flag = Y.

If `studentPlanner.courseReq.showDelivery` is true, the delivery field will display. The user can select a content delivery method for the course. This field is informational only, but may be used for reporting. The values in this drop-down come from the delivery codes in SEP003 with Show in SEP Picklist flag = Y.

Configuration

- Review the Show in SEP Picklist flags in SEP003, STU576, and STU385 to ensure they are configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly:
 - `core.credit.format`
 - `studentPlanner.courseReq.showCampus`
 - `studentPlanner.courseReq.showDelivery`
 - `studentPlanner.courseReq.showGrade`
 - `studentPlanner.planner.updateCreditsIfChanged`

Choice

Updated: September 30, 2022

When there are several course options that can fulfill a planned requirement on a template, a choice requirement can be used.

Choice requirements allow an unlimited number of course options to be defined and are generally used for program requirements that can be fulfilled by more than one course, such as general ed or core requirements. An option can be any combination of actual courses, wildcards, or ranges. For example, STAT 101 OR MATH @. Choice requirements can be used as criteria in tracking and apply to planner audit rules.

Each option can have up to two items. An item can be an actual course such as MATH 101, a wildcard such as @ @, MATH @, or MATH 3@, or a range such as MATH 100:299. When `studentPlanner.choiceReq.enableCourseAttribute` is true, a user can specify a course attribute on any item. The values in the attribute drop-down come from STU050. STU050 can be loaded through the extract or manually by the user.

You can add an option by clicking the **Add** link and entering a string of either the course discipline or the course number in the **Course requirement** field. This will start to filter the list of matching courses that will display in a drop-down below the field. Continue typing to further refine this list or you can click **Load more...** to see additional courses.

To add an additional item to an option, you can click **Add a paired course or lab**.

One of the options may be selected by clicking the option button next to the preferred option. Clicking the **Clear selection** button will remove the selection from the option.

The minimum credits for this requirement can be manually entered but are not required. The format and length of the minimum credits can be configured with the `core.credit.format` Shepherd setting.

If `studentPlanner.choiceReq.enableScribePointer` is true, users with the SEPPPTR Shepherd key can define a value that links this requirement to a rule in a requirement block. The values in this drop-down come from the pointer codes in SEP009 with Show in SEP Picklist flag = Y. See the [Scribe pointers](#) topic for additional information.

If `studentPlanner.courseReq.showGrade` is true, the minimum grade field will display. Users can select the minimum grade needed for the course to be used as criteria in tracking. The values in this drop-down come from the grades in STU385 with Show Student Educational Planner Picklist flag = Y.

If `studentPlanner.courseReq.showCampus` is true, the campus field will display. Users can select the campus where the course should be taken. This field is informational only, but may be used for reporting. The values in this drop-down come from the location codes in STU576 with Show Student Educational Planner Picklist flag = Y.

If `studentPlanner.courseReq.showDelivery` is true, the delivery field will display. Users can select a content delivery method for the course. This field is informational only, but may be used for reporting. The values in this drop-down come from the delivery codes in SEP003 with Show in SEP Picklist flag = Y.

Configuration

- Review STU050 and SEP009, in addition to the Show in SEP Picklist flag in SEP003, STU576, and STU385 to ensure they are configured correctly.
- Review the following Shepherd settings to ensure they are configured correctly for your SEP environment:
 - `core.credit.format`
 - `studentPlanner.choiceReq.enableCourseAttribute`
 - `studentPlanner.choiceReq.enableScribePointer`
 - `studentPlanner.courseReq.showCampus`
 - `studentPlanner.courseReq.showDelivery`
 - `studentPlanner.courseReq.showGrade`
- Access to change the choice requirement pointer is granted if users have the SEPPPTR Shepherd key.

GPA

Updated: September 30, 2022

Minimum overall, major, and class list GPAs required for a degree program can be planned for on a template with the GPA requirement.

Four types of minimum GPAs for a term can be planned for with the GPA requirement – Overall Degree Works GPA, Overall Student System GPA, Major GPA, and a Class List GPA. These can be used as criteria in tracking.

The GPA types that are in the drop-down come from the codes in SEP008 with Show in SEP Picklist flag = Y.

The minimum GPA is a required field, and the format can be configured with the core.gpa.format Shepherd setting.

When adding a Major GPA, a major must be specified. The majors that are in the drop-down come from the major codes in STU023 with Show in Student Educational Planner Picklist flag = Y.

When adding a Class List GPA, a group of classes in which the student needs to have earned a minimum GPA can be defined. Each class should be separated by a plus sign, and ranges and wildcards can be included in the list. Classes are validated, but ranges and wildcards are not.

Configuration

- Review the Show in SEP Picklist flags in SEP008 and STU023 to ensure they are configured correctly.
- Review the core.gpa.format Shepherd setting to ensure it is configured correctly.

Non-course

Updated: September 30, 2022

Non-course items that need to be completed as part of a degree can be planned for on a template with the non-course requirement.

A non-course requirement helps a student plan for items other than coursework that need to be completed for their program such as recitals or applications. They can also be used as criteria in tracking and may apply to planner audit rules.

The non-course requirement types that are in the drop-down come from SCR003.

The status can be left blank to indicate that the student just needs to have completed the non-course item. It can also be the alphanumeric score or status needed to consider the requirement complete.

Configuration

Review SCR003 to ensure it is configured correctly.

Test

Updated: September 30, 2022

Competency or assessment tests that need to be completed as part of a degree can be planned for on a template with the test requirement.

A test requirement helps a student plan for competency or assessment tests they need to take to complete their program. These can also be used as criteria in tracking.

The test codes that are in the drop-down come from the codes in SEP006 with Show in SEP Picklist flag = Y.

The minimum score is a required field, and the length, type, and valid score range of the minimum score are defined in SEP006. Note that alpha scores require additional setup in SEP007 to define the sort order for these values.

If you will be using the Create Block functionality, your SEP006 Test Codes must also be in SCR002 Custom Data. If they are not, the creation of the block will fail with a parsing error.

Configuration

- Review the Show in SEP Picklist flag, Score Length, Score Type, Minimum Valid Score, and Maximum Valid Score in SEP006 to ensure they are configured correctly.
- Review the Sort Order in SEP007 to ensure it is configured correctly.

Placeholder

Updated: September 30, 2022

User-defined requirements that need to be completed as part of a degree can be planned for on a template with the placeholder requirement.

Placeholder requirements are intended to be informational only and are not included in tracking nor do they apply to planner audit rules. However, they will display in the Planner Placeholders section of the planner audit. Typically, these are used to hold the place of a requirement that has not yet been selected, such as a general education or core requirement, and is replaced with the appropriate requirement after the student decides what they will take to complete that requirement.

The placeholder requirement types that are in the drop-down come from the codes in SEP005 with Show in SEP Picklist flag = Y.

The value is a free-form text field and is required.

Configuration

Review the Show in SEP Picklist flag in SEP005 to ensure it is configured correctly.

Notes creating and editing

Updated: September 30, 2022

An unlimited number of notes can be added to requirements, terms, or to the overall template.

As a default, all users have access to view all notes on templates, terms, and requirements that have not been marked as internal. Users can be given access to add, modify, or delete notes on each of these template components, and additional access can be given to users to modify or delete internal notes.

To create new notes on a template, term, or requirement, a users need the SEPTNADD Shepherd key. They will also be able to modify existing notes on a template, term, or requirement. The SETPNMOD Shepherd key can be assigned to users instead of SEPTNADD if they shouldn't have access to create new notes but can modify existing notes on a template, term, or requirement. To delete notes on a template, term, or requirement, users should be assigned the SEPTNDEL Shepherd key.

After the user has been given basic access to create, modify, and delete notes, they can be granted additional notes management access. Users with the SEPINOTE Shepherd key can view and modify notes that have been marked as internal. Additionally, the **Not available to student** check box will display in the note edit dialog for users with this key.

To view notes on a template, term, or requirement, you can click the note icon to open the note list window. You can choose to view, add, edit, or delete notes, depending on your level of access.

You can add a note by clicking the **Add a new note** button. Notes can have an unlimited amount of text. To make a note internal, you can select the **Not available to student** check box. You can click **Save note** to keep your note, or **Cancel** to go back to the note list window without keeping the note.

You can edit a note by selecting **Edit note** from the action menu in the right corner of a note.

You can delete a note by selecting **Delete note** from the action menu in the right corner of a note.

Configuration

Access to the following is granted if the user has the specified Shepherd key:

- Add and modify template, term, and requirement notes - SEPTNADD
- Modify template, term, and requirement notes - SEPTNMOD
- Delete template, term, and requirement notes - SEPTNDEL
- View, create, and modify internal notes - SEPINOTE

Sidebar

Updated: September 30, 2022

The Template Management sidebar panel displays lists of all courses available and requirement types that can be added to a template.

Courses from the Courses list and Requirement types can be dragged and dropped into a term on the template.

The Requirements list will always display in the sidebar. When `studentPlanner.template.sidebar.enableCourses` is true, the Courses list will also display in the sidebar. Additionally, the sidebar can be configured to initially load as collapsed or expanded with the `studentPlanner.template.sidebar.sidebarOpenedByDefault` Shepherd setting.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- `studentPlanner.template.sidebar.enableCourses`
- `studentPlanner.template.sidebar.sidebarOpenedByDefault`

Courses

Updated: September 30, 2022

The Courses list displays courses that can be added as requirements to a template.

The Courses list by default shows all courses in the `rad_course_mst` ordered by discipline where the discipline code is in STU352. Clicking on the arrow next to the discipline will expand the list to show all available courses in that discipline. If there are more than 10 courses for the discipline, a pagination drop down will display allowing the user view the additional courses.

You can narrow the list of courses that display by searching by course discipline, course number, or title. To clear your search, you can click the **x** in the search field.

Configuration

Review STU352 to ensure it is configured correctly.

Requirements

Updated: September 30, 2022

The Requirements list displays all the requirement types that can be used on a template.

The Requirements list is a group of requirement cards that can be used to add requirements to a template. The options in this list come from the requirement types in SEP004 with Show in SEP Picklist flag = Y. The descriptions for these requirement types can be modified, but the keys should not be changed or deleted. There are 6 requirement types: Course, Choice, GPA, Non-Course, Test, and Placeholder.

Configuration

Review the Show in SEP Picklist flag in SEP004 to ensure it is configured.

Transfer Finder

Updated: March 25, 2022

Transfer Finder allows a student at one Degree Works school to compare their completed classwork to the requirements at a partner Degree Works school.

The goal is to let a student see their transfer potential at the partner school by sending their local classwork to the partner school and generating transfer audit. For additional information on using Transfer Finder, please see [Transfer Finder Administration](#).

Admin

Updated: March 25, 2022

The Admin menu contains functions that are not in the student context and are typically available to a limited set of users.

Debug

Updated: March 25, 2022

The Responsive Dashboard allows users to enable debugging to troubleshoot issues in the application for their session.

Users with access to this functionality will be able to enable and disable debug from the Debug page, which is located under the Admin menu. All debugging output for the application is aggregated into one file, even if multiple applications or APIs are involved. Access to the application server is not needed to enable debugging. However, access to the server is required to access the generated log files. Debug files for the user interface can be found in the logs directory of the application server, while debug files for functionality like generating audits can be found in the logdebug directory of the classic server.

Configuration

Access to Debug is granted if the user has the DEBUG Shepherd key.

Debug enabling

Updated: March 25, 2022

On the Debug page, toggle the Enable Debug switch to turn debugging on for the application. A random Debug Tag will be generated, and the user can copy this to their clipboard by clicking on the page icon.

This Debug Tag will be appended to each line of debug in the log file so that the user can easily identify which lines are from their session, as opposed to other users using the application at the same time.

After enabling debug, the user can then resume using Responsive Dashboard and execute the steps to duplicate the issue they are experiencing. Debug should be turned off by going back to

the Debug page and toggling the Enable Debug switch after the issue has been duplicated. Debug for this user's session will also be turned off when they log out of the application.

Debug file access

Updated: March 25, 2022

Debug will appear in the Responsive Dashboard log file on the application server. This is typically defined by the \$LOGGING_FILE environment variable.

Searching for the user's Debug Tag will isolate the lines of debug specific to their actions. For example:

```
2017-10-24 16:22:23,144 68b4e19d-3fb4-4975-9815-83eea01266ff DEBUG [
41-exec-17]
.h.d.s.s.StatelessPreAuthenticatedFilter : Checking secure context
token: null
```

The packdebug script can be used to collect the log files generated for a user session's Debug Tag. For more information on using this script, please refer to the [packdebug script](#) topic.

Theme

Updated: September 30, 2022

The color scheme and logo in Responsive Dashboard can be localized for your institution.

Users with access to this functionality can configure the primary, secondary, and call to action colors in addition to the logo using the Theme Editor, which is located under the Admin menu. Changes made in Theme will impact all users so access to this functionality should only be given to users who are authorized to make these changes.

To change the color theme of your Responsive Dashboard deployment, specify the Hex color code for the Primary color and Secondary color. The Call to action color has limited configurability. You must select an option from the drop-down of available colors.

To change the logo, specify the URL of the graphic to be used. This URL must be accessible to users of the Responsive Dashboard so may need to reside outside your network and firewall. The host server also must be added to the Content Security Policy (CSP) by adding it to the Shepherd setting `core.security.contentSecurityPolicy.imgSrc`.

The default Ellucian theme can be restored by clicking **Restore standard theme**.

Configuration

Access to Theme is granted if the user has the SDADMIN Shepherd key.

Other localizations

Updated: March 25, 2022

The Responsive Dashboard text, labels, and messages in `DashboardMessages.properties` can be localized for your institution.

There are additional properties from other Degree Works applications that are used by Responsive Dashboard and can also be localized:

- `PDFAuditMessages.properties` where the property names begin with `worksheet.*` or `worksheets.*`
- `StudentPlannerMessages.properties` and `PlannerDomain.properties` where the property names begin with `dw.sep.*`
- `TransferFinderMessages.properties` where the property names begin with `dw.transferFinder.*`
- `AcademicGoalsMessages.properties` where the property names begin with `academicGoals.*`
Used in Worksheets and Plans What-If.

For more information on how to localize application properties, see the [Localize in an Application](#) topic.

Links

Updated: March 24, 2023

The Links menu contains external links that you can add to connect from Degree Works to other locations on campus or anywhere else.

To configure this functionality, define the items to display and the order in which to display them in the `dash.navigation.links.items` setting. Then use Controller to add localized properties for the link label and URL. These should be on `DashboardMessages.properties` and each link should have a label property in the format of `dash.navigation.links.xxxx.label` and a URL property in the format of `dash.navigation.links.xxxx.url`, where `xxxx` matches the code you entered in the `dash.navigation.links.items` setting. It is best to use lowercase names in that setting and lowercase property names to match. That is, you can use **MySchool** in the setting and **MySchool** in the property, but it is best to use lowercase **myschool** in both the setting and property.

The links configured in `dash.navigation.links.items` will appear for all users. You can also configure links to display only for users of a certain user class by creating a Shepherd setting in Controller with a key like `dash.navigation.links.items.userclass` in lowercase, where `userclass` is a value from AUD012. For example, if you want additional links to appear for your ADV user class, you would create a new setting named `dash.navigation.links.items.adv`. The URL and label for each of these additional links need to be added to the properties as stated above.

Additionally, you can include a student's ID and degree in the link URL. You would add one or both `{studentId}` and `{degree}` tags into the URL property value. For example, the value for the `dash.navigation.links.transcript.url` property could be `https://my.school.edu/transcript?stuld={studentId}°ree={degree}`. These two tags are replaced with the current student's ID and degree code.

Configuration

- Review the dash.navigation.links.items Shepherd setting to ensure it is configured correctly.
- Access to Links is granted if the user has the EXTLINKS Shepherd key.

Transfer Equivalency Admin Administration

Updated: September 30, 2022

Transfer Equivalency Admin is the data entry interface for managing Transfer Equivalency coursework articulation.

With this tool, you can perform the following functions:

- Create mappings of course equivalents from transfer institutions.
- Create transcript information for specific students, including courses and test scores.
- Perform an articulation of processed classes.
- Perform a degree audit on articulation results, audit classes in the student system, and view a degree audit report.

Transfer Equivalency Admin contains both the frontend (browser-based) application and the backend API used by the client code. It is deployed as a jar on any supported Unix server and is independent of the classic software. It includes an embedded container; a separate Java container like Tomcat is not required. It also is dependent on the classic server for performing the roll to Banner and running an audit.

Initial Transfer Equivalency Admin setup and configuration

To use Transfer Equivalency Admin, you must set up and configure the tool.

Authentication

Updated: March 25, 2022

The Transfer Equivalency Admin user interface supports SHP, CAS, and SAML authentication methods in addition to external access managers such as Oracle Access Manager.

Transfer Equivalency Admin user access

Updated: March 25, 2022

The TEAADMIN Shepherd key grants access to the Transfer Equivalency Admin application, and additional keys grant access to specific functionality.

The TEAREG Shepherd group can be used to assign job access to users. The Users function of Controller or the `core.security.rules.shpcfg` Shepherd setting can be used to assign keys or groups to users. The list of all Transfer Equivalency Admin keys and groups is available in the [Keys and keyrings](#) topic.

Shepherd settings

Updated: September 30, 2022

Most configurations to manage the behavior of Transfer Equivalency Admin can be maintained using the Configuration tab of Controller.

Review the following Shepherd settings to ensure they are configured correctly for your Transfer Equivalency Admin environment:

- `core.security.cas.callbackUrl` spec =treqadmin (if you use CAS authentication)
- `treq.applicant.default.treqStatus`
- `treq.goal.catalogYear.defaultValue`
- `treq.goal.degree.defaultValue`
- `treq.goal.school.defaultValue`
- `treq.goal.scr004.codesToShow`
- `treq.treqadmin.applicant.show.articulationConditions`
- `treq.treqadmin.audit.report`
- `treq.treqadmin.default.gradeAssignMode`
- `treq.treqadmin.default.manyPassGrade`
- `treq.treqadmin.testscores.default.gradeType`
- `treq.treqadmin.testscores.default.schoolId`

UCX tables

Updated: March 25, 2022

Several existing UCX tables are used by Transfer Equivalency Admin. These should be reviewed to ensure they contain the necessary entries and the descriptions are clear and accurate.

Review the following UCX tables to ensure they are configured correctly for your Transfer Equivalency Admin environment:

- STU016

- STU023
- STU035
- STU307
- STU314
- STU346
- STU350
- STU351
- STU355
- STU356
- STU362
- STU379
- STU380
- STU382
- STU385
- STU398
- STU560
- STU564
- SYS998
- TRQ060
- TRQ061
- TRQ062
- TRQ063

Database table structure

Updated: March 25, 2022

Familiarize yourself with the tables used in Transfer Equivalency Admin.

DAP_APPLICNT_MST

Each student has a dap_applicant_mst record that stores their catalog year, school and degree goals in the dap_catalog_yr, dap_school and dap_degree fields.

DAP_APPDATA_DTL

A `dap_appdata_dtl` record is created for each goal item for the student. The goal type is stored in `dap_goal_code`. Valid values are CAMPUS, COLLEGE, CONC, MAJOR, MINOR, and PROGRAM. The selected goal value code is stored in `dap_goal_value`.

DAP_TRANSFER_DTL

A `dap_transfer_dtl` record is created for each transcript class or exam created for a student.

For classes, the `dap_school_id` is the `rad_ets_code` for the selected school while the `dap_tr_disc`, `dap_tr_crse_num`, `dap_tr_title`, `dap_tr_credits`, `dap_tr_cr_method`, and `dap_tr_grade` should be the values provided for the selected class.

For exams, the `dap_school_id` will be TESTS while the `dap_tr_disc` is the test code, the `dap_tr_title` is the test description, `dap_tr_start` is the test date (if entered) and `dap_tr_grade` is the score provided for the exam.

Localizations

Updated: March 25, 2022

The text, labels, and messages for Transfer Equivalency Admin can be localized for your institution and are defined in `TransferEquivalencyAdminMessages.properties`.

For more information on how to localize application properties, see the [Localize an application](#) topic.

Transfer Equivalency Admin deployment

Updated: September 30, 2022

The Transfer Equivalency Admin is designed to deploy equally well in on-premise and cloud-hosted scenarios on all supported platforms. It provides a mobile-ready, responsive user interface communicating with the servers using lightweight, stateless, restful API over HTTP SSL.

The Transfer Equivalency Admin deployment artifact is a JAR file that is executable using java and includes an embedded container.

Prerequisites

Updated: March 25, 2022

Java is a prerequisite, and SSL is a best practice and recommended for Transfer Equivalency Admin.

Java

Java version 11 or above is the only third-party dependency required to run this application. Oracle Java and OpenJDK are supported.

SSL

SSL is in general a best practice and is recommended for Transfer Equivalency Admin because the security implementation uses stateless tokens.

Using signed certificates is recommended for all deployment scenarios, even test or development environments. Self- signed certificates, even for internal test deployments, can be problematic because of browser requirements and warnings. The SSL certificate needs to be configured in a keystore and that keystore will need to be accessible in the deployment environment.

There are various options for how to do this, for example using Java's keytool command.

Transfer Equivalency Admin environment settings

Updated: September 29, 2023

Several environment variables are required to be configured before running the application.

Depending on the desired deployment platform there are many options for how these environment variables can be configured. The only requirement is that they are available in the environment where the product's JAR file will be executed. For example, it may be desirable to configure them in a shell script, in a specific user's profile, a startup instruction like init.d, or in a continuous deployment tool like Jenkins.

The minimum required variables are documented here. But many optional variables are provided in the Spring Boot platform, for example tuning the embedded container. Please refer to Spring documentation for those: <http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>

Environment Variables are the recommended method to configure these properties. But there are various other ways these can be configured. Please refer to this documentation for more information: <http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html>

Required environment variables for Transfer Equivalency Admin

- **DW_DATASOURCE_URL:** A JDBC connection string for the Degree Works database. For example: `jdbc:oracle:thin:@{SERVER}:{PORT}/{SERVICE_NAME}` where {SERVER} is the address, {PORT} is the port of the listener, and {SERVICE_NAME} is the service name of the database. This is the same database and schema used by the classic Degree Works software.
- **DW_DATASOURCE_USERNAME:** The Degree Works database username.
- **DW_DATASOURCE_PASSWORD:** The Degree Works database password. This can be encoded. To get the encoded value, use the `showdbpasswords -- password` command in the classic environment. It will display the encoded value to place here (e.g. `ENC(skdfjldjs)`).

- **SERVER_PORT:** The desired port for the Transfer Equivalency Admin. This must be different from the ports used by other applications on this server.
- **SERVER_SERVLET_CONTEXT_PATH:** An optional context path. If this is blank, the application will be deployed at the root URL like `https://myserver.edu:NNNN` (where NNNN is the SERVER_PORT). If a value is specified like `/my-context-path`, the application would be accessible at a URL like `https://myserver.edu:NNNN/my-context-path` (where NNNN is the SERVER_PORT).
- **SERVER_SSL_KEY_STORE:** The path to a keystore file you created to contain your SSL certificate.
- **SERVER_SSL_KEY_STORE_PASSWORD:** The password for your keystore file.
- **SERVER_SSL_KEYSTORETYPE:** The type of your keystore. JKS is the default.
- **SERVER_SSL_KEY_ALIAS:** The alias of the key you created.

Optional environment variables

- **JAVA_OPTIONS:** The memory allocation for the application can be defined using this variable, and is recommended. The value defines the initial heap size and max heap size and is in the following format: `-Xms1536m -Xmx1536m`. The heap size values should be adjusted as appropriate for your server.
- **SERVER_MAX_HTTP_HEADER_SIZE:** This setting allows you to set the maximum size of the HTTP message header. By default, this is 8KB, and for some users with a large number of Shepherd access keys, this can be too small. It's recommended to use this variable and set it to at least 12KB. For example: `export SERVER_MAX_HTTP_HEADER_SIZE=12288`
- **SERVER_TOMCAT_REDIRECT_CONTEXT_ROOT:** Embedded Tomcat for Spring Boot has a default behavior to redirect a request without a trailing slash to one with a trailing slash. However, that redirect happens without evaluating forward headers like X-Forwarded-Proto. So, when SSL offloading is in place, the redirect happens to http instead of https. This can be an issue especially for load balanced environments. To disable the default behavior, set this variable to false.

Warning! You should do this only if the default behavior is not working. If you are offloading SSL at a proxy or load balancer, make sure to follow the instructions in the [Load balancing and proxies](#) topic. Then, if you have problems accessing the application in the browser when you do not include a trailing slash at the end of the request, try setting this variable to false.

- **DEBUG:** A boolean true or false which will enable or disable debug logging. The log file is by default named `transferEquivalency.log`. But, the location and name of that file can be customized by setting the `LOGGING_PATH` and `LOGGING_FILE` environment variables.
- **DEPLOY_LOCATION:** The location of the `TransferEquivalencyAdmin.jar` file. This location should be set then referenced in the `java -jar` command that starts the application.
- **LOGGING_FILE:** The name of the file including the path location. If this is specified, the `LOG_PATH` variable should not be used.

- **LOG_PATH:** The location of the log file. The default PATH location is the Java temporary java folder from the Java property `java.io.tmpdir`. This is usually `/tmp`. The file name will be `transferEquivalency.log` for Transfer Equivalency Admin.
- **LOG_DEFAULT_THRESHOLD:** If set to `DEBUG`, the application will output more verbose logging. Debug output will be sent to the log during the startup process and for all requests thereafter. Warning: this will result in very large log files. If not set, the debug threshold is `INFO`.
- **DW_ENABLE_MONITOR** - This environment variable can be set to enable `/health`, `/metrics`, `/prometheus`, and `/status` endpoints for health and monitoring metrics. By default, this monitoring functionality is not enabled. Adding this variable to an application's startup script and setting it to `true` would enable these endpoints for the application. See individual API documentation in the API Catalog for additional information.
- **DW_RABBITMQ_SSLALGORITHM:** Specify the SSL algorithm to use when connecting to the RabbitMQ broker using SSL. If not specified, it uses the current RabbitMQ driver's default (at least TLSv1.2). This is only considered if the Shepherd setting `core.amqp.useSsl` is set to `true`.

Database pool configuration for Microservice applications

For production deployments of java applications, it is important to be able to configure the size of the database connection pool. In Tomcat, this is done through the Resource definition in `server.xml`. For microservice applications like Transfer Equivalency Admin, there are environment variables that can be used for this configuration.

These environment variables start with `DW_DATASOURCE_`. They end with the name of the datasource pooling attribute you want to control. These are the most common attributes:

Attribute	Meaning
<code>initialSize</code>	The initial number of connections that are created when the pool is started.
<code>maxActive</code>	The maximum number of active connections that can be allocated from this pool at the same time.
<code>maxIdle</code>	The maximum number of connections that can remain idle in the pool, without extra ones being released.
<code>minIdle</code>	The minimum number of active connections that can remain idle in the pool, without extra ones being created.

To set the maximum number of active connections in the pool, for example, put the following line in your startup script:

```
export DW_DATAPOOL_MAX_ACTIVE=100
```

You can set any of the properties of a version 1.4 Apache Commons BasicDataSource object. These are documented at <http://commons.apache.org/proper/commons-dbcp/api-1.4/org/apache/commons/dbcp/BasicDataSource.html>.

The number of active connections in addition to the current configured values for the limits can be monitored using any JMX (Java Management Extensions) client, such as JConsole, attached to the microservice JVM. The values will appear under the mbean `net.hedtech.degreeworks/`

JmxBasicDataSource/datasource. The getState operation will return a formatted report of the current pool state.

SSL offloading

When SSL offloading is in use (SSL is terminated before reaching the application) with a load balancer, proxy server, and so on, some additional configuration needs to be done to make sure Degree Works applications behave as expected. For more information, see [Load balancing and proxies](#).

Deployment examples

Updated: March 24, 2023

When the prerequisites and configuration are in place, start the application using a "java-jar" command like this one.

```
java $JAVA_OPTIONS -jar TransferEquivalencyAdmin.jar
```

Example startup script for Transfer Equivalency Admin

If it is desirable to employ a shell script to start the application, here is an example. Note that the TransferEquivalencyAdmin.jar file must be staged at the DEPLOY_LOCATION:

```
#!/bin/ksh
export
DW_DATASOURCE_URL="jdbc:oracle:thin:@MyServer:1521/service_name"
export DW_DATASOURCE_USERNAME="my-dw-user"
export DW_DATASOURCE_PASSWORD="my-dw-password"
export SERVER_PORT="8531"
export SERVER_SERVLET_CONTEXT_PATH=""
export SERVER_SSL_KEY_STORE="/usr/local/my-keystore.jks"
export SERVER_SSL_KEY_STORE_PASSWORD="my-key-password"
export SERVER_SSL_KEYSTORETYPE="jks"
export SERVER_SSL_KEY_ALIAS="my-keystore-alias"
export DEPLOY_LOCATION="/path/to/jarfile"
export JAVA_OPTIONS="-Xms1536m -Xmx1536m";
export DEBUG="false"
# To enable full-time debug log output, remove the # from the line below
#export LOG_DEFAULT_THRESHOLD=DEBUG
java -jar $JAVA_OPTIONS DEPLOY_LOCATION/TransferEquivalencyAdmin.jar
> startup-
log- file.log &
```

Example stop script for Transfer Equivalency Admin

If using a start script like the example above, here is a corresponding example stop script:

```
#!/bin/ksh
pkill -f TransferEquivalencyAdmin
```

Classic server requirements

Updated: March 25, 2022

On the classic server you must have web07 running.

Run webrestart to start up the web07 daemons. You mainly need this up and running while performing a roll to banner or while running an audit. View the \$ADMIN_HOME/logdebug/web.log for issues web07 may encounter while either starting up or processing these requests.

Navigation

Updated: March 25, 2022

The header bar at the top of Transfer Equivalency Admin helps users navigate through the application.

The user can access the functionality to which they have been given access by clicking the link for that item in the Transfer Equivalency Admin menu.

Configuration

Access is granted to the following if the user has the specified Shepherd key:

- Mappings - TEAMAP
- Transcript - TEATRANS
- Articulate - TEAARTIC
- Audit - TEAAUDIT
- Roll - TEAROLL
- Debug - DEBUG

Mega Menu

Users can also navigate throughout the application using the Mega Menu on the right of the header bar. Note that for users on a mobile device, only the mega menu will be available for navigation.

Help

To open help in Transfer Equivalency Admin, select Help from the persona drop-down button on the right of the header bar.

Logout

To log out of Transfer Equivalency Admin, select Sign Out from the persona drop-down button on the right of the header bar.

Student search

Updated: March 25, 2022

Persistent on all pages that focus on a student (that is, Transcript, Articulation, Audit, and Roll) is the student search component. Existing student records can be searched for by student ID, name, status, or email address.

To search for student records, first select a category then type a keyword string and hit Enter. A paginated list of matching records will be returned. The list is initially sorted alphabetically ascending by the name, but the grid can be sorted by any column by clicking on the column header. An arrow will appear next to the column header to indicate that it is being used for the sort, and the direction of the arrow indicates whether it is ascending or descending.

Search criterion can be entered one time per category. Every search criterion entered will be created as a chicklet under the search bar. To clear a search criteria, click the x in the chicklet.

For example, if the name category is selected and the string “Sam” is entered, a chicklet will appear with “Name: Sam” inside of it. Another category can be selected to refine the search. If the status category is selected and the string “Transcript Entry” is entered, then another chicklet will appear as “Status: Transcript Entry”. A paginated list of matching students will be returned who have “Sam” in their names and who also have a “Transcript Entry” status.

The pagination details at the bottom of the search results grid allow the user to page to additional matching records, control how many records per page should display and how many total matching records were found.

Note that only students with a valid status for the function will be selectable. For example, you cannot select students in the Audit tab if their status is not “Articulated – Resolved”. To select a student, click on the Student ID link.

Admin Mappings

Updated: March 25, 2022

Mappings allows you to view how course equivalents from transfer institutions are equated to your institution’s courses.

Mapping data is typically bridged into the Degree Works database from the student system but can be created or modified in Mappings. The following are examples of course mappings:

- Your institution has a basic accounting course titled “Accounting Elective.” You can map all equivalent basic accounting courses from various transfer institutes to the “Accounting Elective” course.
- You can map different specialized anthropology electives offered by different transfer institutes to one anthropology elective course that your institution offers.

Configuration

- Review TRQ060 to define the transfer institutions for which mappings are most frequently maintained, and from which many transcripts are received by your institution.
- Access to manage mappings is granted if the user has the TEAMAP Shepherd key.

Transfer institution selection

Updated: March 25, 2022

You can select a transfer institution from a list schools to add or maintain course mappings.

The default view will be a list of your favorite schools. Unselect the check box **Show favorite schools only** to display full list of schools from RAD_EST_MST.

Transfer institutions can be filtered by School ID, name, city, and state. To search for transfer institutions, select a category and (for example, Name) type a keyword string and hit Enter. A paginated list of matching records will be returned. The list is initially sorted alphabetically ascending by the name, but the grid can be sorted by any column by clicking on the column header. An arrow will appear next to the column header to indicate that it is being used for the sort, and the direction of the arrow indicates whether it is ascending or descending.

Transfer institutions may also be searched by selecting a category and then entering a few characters allowing a suggestive set of records to display below the search bar. Clicking a particular student will populate the respective details into the search results grid.

Search criteria can be entered one time per category. For every search criteria entered a chicklet is created under the search bar. To clear a search criteria, click the x on the chicklet.

For example, If category name is selected and the string High is entered, a chicklet will appear with Name: High inside of it. Another category can be selected to refine the search. If the category state selected and the string entered is NY, then another chicklet will appear State: NY. A paginated list of matching institutions will be returned who have High in their names and who are in the State NY.

The pagination details at the bottom of the search results grid allow the user to page to additional matching records, control how many records per page should display and how many total matching records were found.

School mappings

Updated: March 25, 2022

Course equivalents, or mappings, that have been defined for a school are displayed on the Select a Mappings page.

This page also includes the transfer institution's name, and the list of courses from the transfer institution with equivalent courses from your curricular offerings. It also allows you to view edit or add new transfer conditions.

To view or print a report of the mappings, click **View Report**.

You can filter the list of mappings using Search, using the transfer, local discipline, and course boxes. If you filter this list and then generate a mappings report, the report will only include that subset of mappings.

Note: Currently only 1000 records can be printed at one once.

Mapping details

Updated: March 25, 2022

The Mapping page displays the details of a particular course that has been mapped from the transfer institute to a course that exists in your school's curriculum.

If there is more than one transfer course or more than one equivalent course in a mapping, the credits for each equivalent course at your institution must be specified. If the Credits field is left blank on a one-to-one mapping, Transfer Equivalency Admin will calculate the credits earned for a transfer class based on the credits transferred and based on whether the transfer school follows a quarter or semester system.

The page has the following sections:

Transfer course(s)

The Transfer course(s) grid displays the following details about the course from the transfer institute that you want to map to an equivalent course that your institution offers:

Transfer course field	Description
Transfer discipline	Enter the course category. For example, Chemistry or Chem.
Transfer course#	Enter transfer school's course catalog number.
Title	Enter the full name of the course.
Min grade	Enter the minimum grade for the course.
Max grade	Enter the maximum grade for the course.
Min credits	Enter the minimum required credits to take the course.
Max credits	Enter the maximum accepted credits to take the course.
Years ago	Enter the maximum number of valid years for the grade after the class is complete

Transfer course field	Description
When taken between catalog	The catalog year range within which a student must have completed a course for this mapping to be valid.

Mapping to local course(s)

The Mapping to local course(s) grid displays details of the equivalent course from your institution to which the transfer course or courses are mapped. The following course details are displayed: Local Discipline, Local Course num, Title, and the Credits for which that course can be taken. If there is more than one transfer course or more than one equivalent course in a mapping, the credits for each equivalent course at your institution must be specified. If the Credits field is left blank on a one-to-one mapping, Transfer Equivalency Admin will calculate the credits earned for a transfer class based on the credits transferred and based on whether the transfer school follows a quarter or semester system.

Local course	Description
Local discipline	Select the course category. For example, Chemistry or Chem.
Local course#	Select the local school's course catalog number.
Title	Full title will be auto-populated from the course number selected.
Valid for Catalog Years	The catalog year range within which a student must have completed a course for this mapping to be valid.
Credits	Enter the credits for the course selected.

Note: The catalog year associated with the local courses grid has a relationship with the catalog year on the applicant detail of the Transcript function.

Transfer Conditions

The transfer conditions for a particular mapping are special conditions that can exist for that mapping to be valid for a student. For example, if a mapping is valid only if a student's major is Chemistry, you can specify Major=Chemistry as a transfer condition.

History

Click on the **History** icon to display the login ID of the user who last accessed or modified the mapping record, the date the mapping was created, and the date the mapping details were last modified.

Add a mapping in Transfer Equivalency Admin

Updated: March 25, 2022

You can add a course mapping in Transfer Equivalency Admin.

About this task

New mappings will be saved in the Degree Works database.

Procedure

1. Click **Create new school mapping** on the School Mappings page.
The New Mapping page is displayed. This page has two sections, the Transfer course(s) grid, where you can enter details of the course to be mapped, and the Mapping to local course(s) grid, where you can enter details of the equivalent course in your curriculum.
2. Click + in the Transfer course(s) grid.
A new row is added.
3. Enter the discipline and number of the course, and the title in the **Transfer course** fields.
4. Specify the minimum and maximum grade for the course.
5. Specify the minimum and maximum credits for which the class can be taken.
6. Specify the maximum number of years that can have passed from when the student completed the class.
7. Specify the Catalog Years during which a student must have completed a course for this mapping to be valid.

Note: If you want the course to be valid indefinitely, select the earliest term and the latest term from the drop-down lists. These values are usually associated with term codes such as "000000" and "999999".
8. Click + in the Mapping to local course(s) grid.
A new row is added.
9. Select the discipline from the **Local Discipline** drop-down list.
Transfer Equivalency Admin attempts to find the courses in its database of existing courses for that discipline. The matching courses are displayed, and you can select the required course.
10. Specify the credits for the equivalent course at your institution.
11. Specify the catalog years under which this mapping is valid.

Note: If you want the course to be valid indefinitely, select the earliest term and the latest term from the drop-down lists. These values are usually associated with term codes such as "000000" and "999999".
12. Both "Authorized by" and "Comment" sections are optional.
13. Click **Save**.

Modify an existing mapping in Transfer Equivalency Admin

Updated: March 25, 2022

You can modify an existing course mapping in Transfer Equivalency Admin.

About this task

The course mapping details are updated in the Degree Works database.

Procedure

1. On the School Mappings page that displays the list of mappings for that school, click the edit icon alongside the mapping want to modify.
The mapping record is displayed on the Mapping <mapping number> page, with the details of the equivalent course mapping.
2. Add, edit, or delete a transfer or local course.
 - To add a course mapping, click and Save.
 - To edit a course mapping, click and Save after the modifications are done.
 - To delete a course mapping, select one or more rows. Click Remove and Save.

Note:

- You need to have at least one transfer course and one local course.
 - Access information can be found by clicking the **History** icon. The details of who has created, when it was created, when it was last modified, and who has modified it will be displayed.
 - Refer to the *Transfer Equivalency Admin Mapping details* section for more information about the details of the equivalent course mappings.
3. Click **Save**.

Test mappings

Updated: March 25, 2022

Test score mappings are handled by creating mappings for a transfer school with an ID of TESTS. Tests can be added under the section of Test Mappings.

Test score	Description
Test	Enter the test score ID.
Title	Enter the title of the test.
Min score	Enter the minimum grade for the test.
Max score	Enter the maximum grade for the test.
Years ago	Enter the maximum number of valid years for the grade after the test has been completed.
When taken between catalog years	Make the Beginning of Time and End of Time selections.
Mapping to local course	Description
Local discipline	Select the course category. For example, Chemistry or Chem.

Mapping to local course	Description
Local course#	Select the local school's course catalog number.
Title	Full title will be auto populated from the course number selected.
Valid for Catalog Years	The catalog year range within which a student must have completed a course for this mapping to be valid.
Credits	Enter the credits for the course selected.

Add a test mapping in Transfer Equivalency Admin

Updated: March 25, 2022

You can add a test mapping in Transfer Equivalency Admin.

About this task

New mappings will be saved in the Degree Works database.

Procedure

1. Click Create new test mapping on the Test Mappings page.
The New Mapping page is displayed. This page has two sections, the Test Score grid, where you can enter details of the course to be mapped, and the Mapping to local course(s) grid where you can enter details of the equivalent course in your curriculum.
2. Click + in the Tests score grid.
A new row is added.
3. Enter the Test ID, Title.
4. Specify the minimum and maximum Score for the test.
5. Specify the maximum number of years that can have passed from when the student completed the class.
6. Specify the Catalog Years during which a student must have completed this Test for this mapping to be valid.
If you want the course to be valid indefinitely, select the earliest term and the latest term from the drop-down lists. These values are usually associated with term codes such as "000000" and "999999".
7. Click + in the Mapping to local course(s) grid.
A new row is added.
8. Select the discipline from the Local Discipline drop-down list.
Transfer Equivalency Admin attempts to find the courses in its database of existing courses for that discipline. The matching courses are displayed, and you can select the required course.
9. Specify the credits for the equivalent course at your institution.
10. Specify the catalog years under which this mapping is valid.

Note: If you want the course to be valid indefinitely, select the earliest term and the latest term from the drop-down lists. These values are usually associated with term codes such as “000000” and “999999”.

11. Both “Authorized by” and “Comment” sections are optional.
12. Click **Save**.

Modify a test mapping in Transfer Equivalency Admin

Updated: March 25, 2022

You can modify an existing test mapping in Transfer Equivalency Admin.

About this task

The course mapping details are updated in the Degree Works database.

Procedure

1. On the Test Mappings page that displays the list of mappings for that school, click the edit icon alongside the mapping want to modify.
The mapping record is displayed on the Mapping <mapping number> page, with the details of the equivalent course mapping.
2. Modify the details as required.
Add, edit, or delete Test score:

- To add a Test mapping, click and Save.
- To edit a Test mapping, click and Save after the modifications are done.
- To delete a Test mapping, select one or more rows. Click Remove and Save.

Add, edit, or delete Local Course:

- To add a course mapping, click and Save.
- To edit a course mapping, click and Save after the modifications are done.
- To delete a course mapping, select one or more rows. Click Remove and Save.

Note:

- You need to have at least one transfer course and one local course.
- Access information can be found by clicking the **History** icon. The details of who has created, when it was created, when it was last modified, and who has modified it will be displayed.
- Refer to the *Transfer Equivalency Admin Mapping details* section for more information about the details of the equivalent course mappings.

3. Click **Save**.

Define transfer conditions in Transfer Equivalency Admin

Updated: March 25, 2022

The transfer conditions for a particular mapping are special conditions that must exist for that mapping or a test score mapping to be valid for a student.

About this task

For example, if a mapping is valid only if a student's major is Chemistry, you can specify Major=Chemistry as a transfer condition. You can add, modify, or delete a transfer conditions from the transfer condition section on the Mappings page for a particular mapping.

Transfer conditions can be added either from the home page of either School Mappings/Test mappings or inside an individual mapping ID.

Procedure

1. Click **Add** under the conditions column on any mapping.
2. Select the condition from the **Condition** drop-down list. For example, select Major.
3. Select the condition operator from the **Operator** drop-down list. For example, select Is Equal To.
4. Enter the values for the mapping condition. For example, type Chemistry.
5. Click **Save**.
The new condition is added.
6. To modify an existing mapping condition, select the condition in the Edit section and click the edit icon.
7. To delete an existing transfer condition, select the check box next to the condition and click the delete icon.

Transcript

Updated: March 25, 2022

Use the Transcript module to enter transcript information for students.

You can add transcripts from multiple schools and maintain test scores for each student. You can also update student records to add details like a student's major in each school that they attended and can set the articulation conditions for courses that the student has completed in those schools.

Transcript details for a student are maintained through the following tabs.

Student Information - The student details page displays basic student information such as student ID and degree. You can also add additional student details and specify articulation conditions from this page.

Test Scores - The tests page displays the list of tests that a student has completed. You can add test scores for articulation to course credit at your institution from this page.

Transcripts - You can view transcripts from different schools that the student has attended using this page. After adding a transcript, school name is added on to the drop down and the respective transcript is displayed upon selection.

Add School Transcripts - You can add transcripts from different schools that the student has attended using this page.

Configuration

- Review the following Shepherd settings to ensure they are configured correctly:
`treq.treqadmin.applicant.show.articulationConditions`
`treq.goal.scr004.codesToShow`
- Access to manage transcript data is granted if the user has the TEATRANS Shepherd key.

Edit student information

Updated: March 25, 2022

The student information page displays basic student information such as student ID and degree.

About this task

You can also add additional student details and specify articulation conditions from this page.

Click the “Edit student detail” icon to edit the student’s Degree, Level and Catalog Year. The Student ID and Status are read only information and cannot be edited.

Click the “Add student goal” icon to add additional student goal data. The options in the Goal type dropdown are defined by the `treq.goal.scr004.codesToShow` setting.

Student goals can be deleted by selecting the check box to the left of the item and clicking “Remove”.

Click “Save” to save your changes. Click “Cancel” to discard your changes.

Manage test scores

Updated: March 25, 2022

The test scores page displays the list of placement tests that a student has completed.

You can create mappings to articulate placement tests for course credit at your institution.

To add a placement test, select the Test name from the dropdown. The Test name dropdown is populated with all values from STU314. The Test title will default from the test code description. Enter the Score, Test date and any specific conditions for this placement test.

Click the “Add test score” icon to add additional placement test data.

To edit details of a placement test that has already been entered, select the record and click the “Edit test score” icon.

Placement test scores can be deleted by selecting the check box to the left of the item and clicking “Remove”.

Click “Save” to save your changes. Click “Cancel” to discard your changes.

Manage transcripts

Updated: March 25, 2022

You can add, edit, or delete class details from each school transcript for a student.

About this task

First a transfer school must be added – see the *Add school transcripts in Transfer Equivalency Admin* section.

Select the school transcript that you want to modify. Existing transcript classes for that school will be displayed.

Click the “Add Transcript Class” button to add a class. Enter the Discipline, Course number, Title, Grade, Credits, Credit type, Term, Calendar and any applicable Conditions. Click Save.

To edit a class that has already been entered, select the record and click the “Edit Transcript Class” icon.

Transcript classes can be deleted by selecting the check box to the left of the item and clicking “Remove”.

Click “Save” to save your changes. Click “Cancel” to discard your changes.

Add school transcripts

Updated: March 25, 2022

Before entering transcript classes, you must first select the transfer institution.

About this task

On the Add School Transcripts tab, select the school for which the student has attended. By default, schools designated as a favorite school will display. To view other schools, clear the Show favorite schools only check box.

The list of schools can be filtered by selecting a category, typing a keyword string and hitting Enter. A paginated list of matching records will be returned. The list is initially sorted alphabetically ascending by the name, but the grid can be sorted by any column by clicking on the column header. An arrow will appear next to the column header to indicate that it is being used for the sort, and the direction of the arrow indicates whether it is ascending or descending.

Search criterion can be entered one time per category. Every search criterion entered will be created as a chicklet under the search bar. To clear a search criteria, click the x in the chicklet.

For example, if the state category is selected and the string "WA" is entered, a chicklet will appear with "State: WA" inside of it. Another category can be selected to refine the search. If the name category is selected and the string "University" is entered, then another chicklet will appear as "Name: University". A paginated list of matching schools will be returned who have "WA" as the state and who also have "University" in their name.

Click "Add transcript" for the school for which you want to add transcript details. The school transcript is added to the Transcript classes dropdown list in the Transcripts Tab. Select the required school from the dropdown to add details of transcript classes for that school to the student's record.

Articulation

Updated: March 25, 2022

Use the Articulate module to view and edit articulated transfer classes for students.

You can process the student's transcript against defined articulation rules and resolve classes with multiple, duplicate, or missing mappings.

Configuration

- Review the following Shepherd settings to ensure they are configured correctly:
 - `treq.treqadmin.applicant.show.articulationConditions`
 - `treq.treqadmin.default.gradeAssignMode`
 - `treq.treqadmin.default.manyPassGrade`
 - `treq.treqadmin.testscores.default.gradeType`
 - `treq.treqadmin.testscores.default.schoolId`
- Access to articulate mappings is granted if the user has the TEAARTIC Shepherd key.

Articulation summary

Updated: March 25, 2022

When you select a student, the results of the last articulation for that student are displayed.

An articulation will be performed if it was not done previously for that student.

The date and time of the last articulation are displayed. The articulated classes are listed with the class details.

The Summary section shows a numeric representation of the articulated, undecided, leftover, and duplicate classes expressed as a percentage of the total number of transfer classes for the student. When all the mappings are resolved, the articulated section will indicate 100%.

- Undecided classes are those transfer classes for which multiple mappings were found.
- Leftover classes are classes for which no mappings were found.
- Duplicate classes are classes where more than one transfer course maps to the same class at the receiving institution.

Re-articulate

You can perform a new articulation, for example, if you alter mappings for a particular institution and want to re-articulate. When a student is re-articulated, changes made to a previous articulation are lost and the articulation results will be based on the mapping rules that are in effect at the time the re-articulation is performed.

Click the **Re-articulate** button to perform a re-articulation. The new articulation is performed, and the articulation summary is displayed along with the classes.

Articulated classes

Updated: March 25, 2022

You can edit already articulated classes to change the local course to which the transfer course is mapped.

Click the “Edit articulate class” button to enter edit mode. In the Local Course section, select the course at your institution to which you want to map the transfer course, from the Course drop-down lists. The title of the course is filled in based on your selection. Specify the equivalent credits for the course at your school, in the Credits field. Specify the equivalent Grade and the Grade Points. Click “Save” to save your changes.

Resolve undecided classes

Undecided classes are those transfer classes for which multiple mappings were found. For example, there could be a transfer class, MATH 1A from another institution that can map to either MATH 101 or MATH 1001 at your institution.

To resolve undecided classes from the Unresolved classes section, either select Undecided in the dropdown if there are multiple types of unresolved classes or click “Resolve” for the class which has status as Undecided. The transfer classes are displayed to the left of the arrow and the local classes, to which each transfer class maps to, are displayed to the right of the arrow.

To resolve a class for the student, select the box next to the class that you would like to use in this student's articulation.

For example, if you want to map transfer class MATH 1A to MATH 101 at your institution, select the check box next to MATH 1A > MATH 101.

Click “Save” to save your changes.

Resolve undecided classes for Many to One and Many to Many mappings

There is a special case when there are two or more mappings for a school that are not the same but have a transfer class in common. For example, a student has taken MATH1A, MATH1B and MATH1C at a transfer institution and your institution has the following mappings for that institution:

MATH1A, MATH1B > CALC101 MATH1A, MATH1C > CALC102

Because MATH1A is in both mappings this is an Undecided case, and you have to be careful to choose the option that best fits your transfer policies. You cannot choose the same class more than one time – the system will prevent you from doing so by showing an error message. You must be sure to clear the mapping option that you do not want to select.

In the example above, because MATH1A is in both options you can articulate MATH1A and MATH1B OR MATH1A and MATH1C. The class(es) not chosen will be articulated as “Zero Credits” which means that they will be in the articulated list, but the transfer credits and grades will not be counted. In the example above, if we choose the first mapping – MATH1A and MATH1B– then CALC101 will be articulated for your school, but the CALC102 will not, and in the articulated list the transfer class MATH1C will be pointing to nothing.

Resolve leftover classes

Leftover classes are those transfer classes for which no mappings were found. You can resolve these classes in either of the following ways:

- By articulating them as zero credits
In this case, the class is articulated and transferred to your institution without a credit or grade value.
- By articulating them through a special mapping
Using this option, you can define special or advanced mappings to articulate the class to your institution.
 - Simple mapping - one-to-one mappings that articulate the transfer course to a single course at your institution. For example, you can define a mapping where a transfer class, English 100, articulates to a single class, Eng 101 at your institution.
 - Advanced mapping - one-to-many mappings that articulate the transfer course to more than one course at your institution. For example, the ENGLISH 200 class articulates to two

classes, ENG 101 and ENG 102 at your institution, with credits and grades transferred to each class.

You can also apply a special mapping to all future articulations by selecting Global mapping or use it for a particular student articulation alone by selecting Student mapping.

Resolve leftover classes through zero credits

You can resolve leftover classes and transfer them to your institution without a credit value. To resolve leftover classes from the unresolved classes section, either select Leftover in the dropdown if there are multiple types of unresolved classes or click “Resolve” for the class which has status as Leftover. Click “Articulate as Zero Credits”. The class will be moved to the Articulated classes grid with Zero Credits.

Resolve leftover classes through special mappings

You can resolve leftover classes and transfer them to your institution through special mappings. To resolve leftover classes from the unresolved classes section, either select Leftover in the dropdown, if there are multiple types of unresolved classes or click “Resolve” for the class which has status as Leftover. Select the class and click “Special Mapping”.

To make this mapping applicable for all future articulations, select the Global Mapping option, and specify the applicable period. This will be selected by default.

To make this mapping applicable this student alone, select the Student Mapping option.

Enter the local course information that this transfer class is equivalent to.

Click “Save” to save your changes.

Resolve duplicate classes

Duplicate classes are classes where more than one transfer course maps to the same class at your institution.

For example, there may be two separate transfer classes, MATH 1A and MATH 1B that map to the same class, MATH 101, at your institution. To resolve duplicate classes from the unresolved classes section, either select Duplicate in the dropdown if there are multiple types of unresolved classes or click “Resolve” for the class which has status as Duplicate. The transfer classes are displayed to the left of the arrow, and the local classes to which each transfer class maps to, are displayed to the right of the arrow.

To resolve a class for the student, select the check box next to the class at your institution to which you would like the transfer class to be mapped.

For example, to map transfer class MATH 1A to MATH 101 at your institution, select the check box next to MATH 1A > MATH 101.

Note: The courses that you do not select will articulate to zero credits.

To give the student duplicate credit for the coursework, select more than one check box.

For example, to give the student transfer credit from two transfer classes, MATH 1A and MATH 1B, select the check boxes next to MATH 1A > MATH 101 and next to MATH 1B > MATH 101.

Click “Save” to save your changes.

Audit

Updated: March 25, 2022

You can run a degree audit on the results of a student's articulation.

The audit report displays the transfer classes for that student, how they equate to courses at your institute, and the extent to which they satisfy the requirements for a particular degree.

Degree audits can be run only after a student's classes have been successfully articulated. If the student's data has been articulated successfully but has not yet been rolled to the student system, Transfer Equivalency Admin allows a degree audit to be performed on the results of the last articulation. You cannot run audits for students with unresolved classes.

Before running the transfer audit, review the student's details and goal data. See the Student Information section for information on editing this data.

Click the “Process new audit” button to generate the transfer audit.

Configuration

- Review the following Shepherd settings to ensure they are configured correctly:
`treq.treqadmin.audit.report`
- Access to generate transfer audits is granted if the user has the TEAAUDIT Shepherd key.

Roll

Updated: March 25, 2022

Use the Roll function to update the Banner student system with articulated student data from Transfer Equivalency Admin.

If the student's data has been articulated successfully but has not yet been rolled to the student system, Transfer Equivalency Admin allows the roll of only the articulated transfer classes and not the test scores.

All the classes for all schools attended by a student are rolled when using this process. When the transcript classes are rolled, they will be deleted from Transfer Equivalency Admin.

To roll a student to Banner, select the student and click “Roll”.

Configuration

Access to roll student data to Banner is granted if the user has the TEAROLL Shepherd key.

Roll process

1. Load ETS data for Transfer Equivalency (bannerextract ets).
2. Transfer mappings must exist in Transfer Equivalency Admin:
 - a. Import Banner Mappings to Transfer Equivalency Admin (bannerextract mappings).
 - b. Create transfer class mappings in Transfer Equivalency Admin.
3. Configure the UCX CFG020 BANNER flag, "Write Transfers to Banner", to "Y".
4. Allow the Degree Works users write access to the SHRTRIT, SHRTRAM, and SHTRTK Banner database tables.
5. Enter transfer classes in Transfer Equivalency Admin. Articulate, and Roll.
6. Articulate and Roll the data.
7. The SHRTRIT, SHRTRAM, and SHTRTK tables are updated in Banner. Transfer records are deleted from Transfer Equivalency Admin.
8. The user must use the SHATAEQ screen to roll classes to academic history, creating SHRTRCE and SHRTRCR records.
9. New transfer classes will be included when the student is re-bridged in Degree Works.

Roll results to Banner

Updated: March 25, 2022

The Transfer Equivalency Admin to Banner feature rolls a student's dap_transfer_dtl records to Banner's SHTRTK table, which is used in Banner's transfer articulation process.

In Banner, a user must roll the SHTRTK records to "academic history" to create the transfer class records SHRTRCE and SHRTRCR, which are pulled into Degree Works by the bannerextract process. After the Banner user rolls SHTRTK (with SHATAEQ) to academic history, SHTRTK is emptied.

Before Transfer Equivalency Admin writes records to SHTRTK, certain configurations are required on the Banner side. The SHTRTK table contains a link to a required table SHRTRAM, which in turn requires a link to SHRTRIT.

Transfer Equivalency Admin looks for the required SHRTRIT and SHRTRAM tables and creates them if required. Degree Works has the basic information, student's level and transfer term, required to create these records. So, if the transfer college tables do not exist, SHRTRIT and SHRTRAM will be created in Banner by Transfer Equivalency Admin.

Note: This activity will not have to be done using the SHATRNS screen. Transfer Equivalency Admin will NOT write to the Banner transfer class tables SHRTRCE and SHRTRCR, which exist in Banner's academic history. Instead, Transfer Equivalency Admin will roll to the Banner "Transfer Institution Transfer Course Taken" table (SHRTRTK) which temporarily holds transfer classes before they are rolled to academic history in Banner.

Banner's internal logic prevents a class from existing in both SHRTRTK and SHRTRCE, so Transfer Equivalency Admin must check both tables first. Transfer Equivalency Admin will check Academic History before rolling. If ANY classes from any school being rolled exist in SHRTRCE, Transfer Equivalency will not roll any classes. Transfer Equivalency Admin will check SHRTRTK for each class being rolled. If the same class exists, it will not be overwritten.

Debug

Updated: March 25, 2022

Transfer Equivalency Admin allows users to enable debugging to troubleshoot issues in the application for their session.

Users with access to this functionality will be able to enable and disable debug from the Debug page, which is located under the person dropdown menu. All debugging output for the application is aggregated into one file, even if multiple applications or APIs are involved. Access to the application server is not needed to enable debugging. However, access to the server is required to access the generated log files. Debug files for the user interface can be found in the logs directory of the application server, while debug files for functionality like generating audits can be found in the logdebug directory of the classic server.

Enable debug

Updated: March 25, 2022

On the Debug page, toggle the Enable Debug switch to turn debugging on for the application.

A random Debug Tag will be generated, and the user can copy this to their clipboard by clicking on the page icon. This Debug Tag will be appended to each line of debug in the log file so that the user can easily identify which lines are from their session, as opposed to other users using the application at the same time.

After enabling debug, the user can then resume using Transfer Equivalency Admin and execute the steps to duplicate the issue they are experiencing. Debug should be turned off by going back to the Debug page and toggling the Enable Debug switch after the issue has been duplicated. Debug for this user's session will also be turned off when they log out of the application.

Access debug files

Updated: March 25, 2022

Debug will appear in the Transfer Equivalency Admin log file on the application server. This is typically defined by the \$LOGGING_FILE environment variable.

Searching for the user's Debug Tag will isolate the lines of debug specific to their actions. For example:

```
2017-10-24 16:22:23,144 68b4e19d-3fb4-4975-9815-83eea01266ff DEBUG [
41-exec-17]
h.d.s.s.      elessPreAuthenticatedFilter : Checking secure context
token: null
```

The packdebug script can be used to collect the log files generated for a user session's Debug Tag. For more information, on using this script, see the [packdebug script](#) topic.

Configuration

Access to Debug is granted if the user has the DEBUG Shepherd key.

Transfer Equivalency Self-Service Administration

Updated: March 25, 2022

Transfer Equivalency Self-Service is a tool used outside of the standard Degree Works interface to allow students who are considering transferring to your school to see how their transfer classes might articulate and apply to a degree at your institution.

To see how classes articulate, prospective students should:

- Choose an intended degree and major and enter academic year.
- Choose a transfer school and enter the grade, credits, and term for each transfer class.
- Enter placement exam scores.
- Process to see the articulation and degree audit.

Initial Transfer Equivalency Self-Service setup and configuration

Updated: March 25, 2022

For students to use Transfer Equivalency Self-Service, you must set up and configure the tool.

Transfer Equivalency Self-Service user access

Updated: September 29, 2022

Transfer Equivalency Self-Service is an anonymous application, meaning users do not need to have an access ID to use it.

However, the application requires that the TREQER user exists. This user is typically delivered when Degree Works is initially installed. The TREQER user must have the DWTESELF, RSAUDIT, and SDWORKS Shepherd keys. Use the Users function of Controller to verify and add these keys if necessary.

When a user creates an account, they will be assigned the Shepherd keys defined in the `core.security.newUser.roles` settings and the user class defined in the `core.security.newUser.userClass` setting. These settings are delivered with the standard default values. Note that to edit a user in Controller Users, the user class defined in `core.security.newUser.userClass` must be valid in AUD012.

The `core.security.referenceUser.password` must be the same as the TREQER user's password on the `shp_user_mst`.

Transfer Equivalency Self-Service allows only SHP authentication. If you are using an authentication method like SAML for other Degree Works applications, you will need to add a specification to the `core.security.authenticationType` Shepherd setting for Transfer Equivalency Self-Service:

- `core.security.authenticationType = SHP`
- `specification = treqss`

If your environment is configured to use SHP as your default authentication method, you should remove the `treqss` specification for the `core.security.authenticationType` Shepherd setting so Transfer Equivalency Self-Service will also use that default specification.

Shepherd settings

Updated: September 29, 2023

Most configurations to manage the behavior of Transfer Equivalency Self-Service can be maintained using the Configuration tab of Controller.

Review all entries with the `treq.selfService` prefix and ensure that they are configured correctly for your environment.

You must review the following Shepherd settings to ensure they are configured correctly for your environment:

- `classicConnector.serverNameOrIp`
- `core.credit.format`
- `core.security.newUser.roles`
- `core.security.newUser.userClass`
- `core.security.referenceUser.password`
- `core.security.referenceUser.username`
- `core.security.stateless.secretkey`
- `core.site.displayName`
- `core.treq.selfService.auditArticulate.report.default`
- `treq.selfService.*`

UCX tables

Updated: March 25, 2022

Several existing UCX tables are used for Goals, Classes and Exams.

These should be reviewed to ensure they contain the necessary entries and the descriptions are clear and accurate. After updating one of these UCX tables, refresh the page in the browser to load the changes – it is not necessary to restart the application.

Review the following UCX tables to ensure that they are configured correctly for your Transit Jobs environment:

- Show in Transit Jobs in STU016, STU023, STU024, STU307, STU316, STU346, STU350, STU398, STU560, STU563
- Transit Jobs Degree Filter in STU023
- Credit By Exam in STU314
- Description in TRQ065

Database table structure

Updated: March 25, 2022

Transfer Equivalency Self-Service uses several tables in its database table structure.

SHP_USER_MST

When a user creates an account, a *shp_user_mst* record is created for that user with the email stored as the *shp_alt_id*, the password in the *shp_access_code* and the name in *shp_name*. The value in *core.security.newUser.roles* will be stored in *shp_key_list* and the value in *core.security.newUser.userClass* will be stored in *shp_filter*.

DAP_APPLICNT_MST

A *dap_applicant_mst* record is added when a user saves goal data. The selected school and degree goals are stored in the *dap_school* and *dap_degree* fields, while the selected term is converted to the correct catalog year using the STU016 Catalog Year and is stored in the *dap_catalog_yr*.

DAP_APPDATA_DTL

A *dap_appdata_dtl* record is created for each goal item selected when a user saves goal data. The goal type is stored in *dap_goal_code*. Valid values are CAMPUS, COLLEGE, CONC, MAJOR, MINOR and PROGRAM. The selected goal value code is stored in *dap_goal_value*.

DAP_TRANSFER_DTL

A *dap_transfer_dtl* record should be created for each class or exam the user enters.

For classes, the `dap_school_id` is the `rad_ets_code` for the selected school while the `dap_tr_disc`, `dap_tr_crse_num`, `dap_tr_title`, `dap_tr_credits`, `dap_tr_cr_method` and `dap_tr_grade` should be the values provided for the selected class.

For exams, the `dap_school_id` will be TESTS while the `dap_tr_disc` is the test code, the `dap_tr_title` is the test description, `dap_tr_start` is the test date (if entered) and `dap_tr_grade` is the score provided for the exam.

RAD_PRIMARY_MST

When a user creates an account, a `rad_primary_mst` record is created for that user with the email stored in `rad_email` and the name in `rad_name`.

Transfer Equivalency Self-Service Deployment

Updated: September 30, 2022

Transfer Equivalency Self-Service is designed to deploy equally well in on-premise and cloud-hosted scenarios on all supported platforms.

It provides a mobile-ready, responsive user interface communicating with the servers using lightweight, stateless, restful API over HTTPS SSL.

The TransferEquivalencyUI deployment artifact is a JAR file that is executable using java and includes an embedded container.

Prerequisites

Updated: September 30, 2022

Transfer Equivalency Self-Service requires several prerequisites in its setup.

Java

Java version 11 or above is the only third-party dependency required to run this application. Oracle Java and OpenJDK are supported.

SSL

SSL is a best practice and is required for Transfer Equivalency Self-Service because the security implementation uses stateless tokens. Using signed certificates is recommended for all deployment scenarios, even test or development environments. Self-signed certificates, even for internal test deployments, can be problematic because of browser requirements and warnings.

The SSL certificate needs to be configured in a keystore and that keystore will need to be accessible in the deployment environment. There are various options for how to do this, for example using Java's keytool command.

Firewalls

Transfer Equivalency Self-Service can produce PDFs of its audits. It requires access to an external content delivery network address to retrieve fonts and other assets. You must allow access from the Transfer Equivalency Self-Service server to `cdn.elluciancloud.com` on port 443 for this to work.

Transfer Equivalency Self-Service environment settings

Updated: September 29, 2023

Several environment variables are required to be configured before running the application. Several Shepherd settings are also required but they are only used at runtime so they can be configured or changed while the application is running without restarting.

Depending on the desired deployment platform there are many options for how these environment variables can be configured. The only requirement is that they are available in the environment where the product's JAR files will be executed. For example, it may be desirable to configure them in a shell script, in a specific user's profile, a startup instruction like `init.d`, or in a continuous deployment tool like Jenkins.

The minimum required variables are documented here. But many optional variables are provided in the Spring Boot platform, for example tuning the embedded container. Please refer to Spring documentation for those: <http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>.

Environment Variables are the recommended method to configure these properties. But there are various other ways these can be configured. Please refer to this documentation for more information: <http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html>.

Required environment variables for TransferEquivalencyUI

- `SPRING_DATASOURCE_URL`: A JDBC connection string for the Degree Works database. For example `jdbc:oracle:thin:@{SERVER}:{PORT}/{SERVICE_NAME}` where `{SERVER}` is the address, `{PORT}` is the port of the listener, and `{SERVICE_NAME}` is the service name of the database.
- `SPRING_DATASOURCE_USERNAME`: The database username.
- `SPRING_DATASOURCE_PASSWORD`: The database password. If you want to hide this password using an encrypted string, see Database Credentials Changes.
- `SERVER_PORT`: The desired port for TransferEquivalencyUI. This must be different from the ports used by other applications on this server.
- `SERVER_SERVLET_CONTEXT_PATH`: An optional context path. If this is blank, the application will be deployed at the root URL like `https://myserver.edu:NNNN` (where NNNN is the `SERVER_PORT`). If a value is specified like `/my-context-path`, the application would be accessible at a URL like `https://myserver.edu:NNNN/my-context-path` (where NNNN is the `SERVER_PORT`).

- `SERVER_SSL_KEY_STORE`: The path to a keystore file you created to contain your SSL certificate.
- `SERVER_SSL_KEY_STORE_PASSWORD`: The password for your keystore file.
- `SERVER_SSL_KEYSTORETYPE`: The type of your keystore. JKS is the default.
- `SERVER_SSL_KEY_ALIAS`: The alias of the key you created.

Optional environment variables

- `JAVA_OPTIONS`: The memory allocation for the application can be defined using this variable, and is recommended. The value defines the initial heap size and max heap size and is in the following format: `-Xms1024m -Xmx1024m`. The heap size values should be adjusted as appropriate for your server.
- `SERVER_MAX_HTTP_HEADER_SIZE`: This setting allows you to set the maximum size of the HTTP message header. By default, this is 8KB, and for some users with a large number of Shepherd access keys, this can be too small. It's recommended to use this variable and set it to at least 12KB. For example:

```
export SERVER_MAX_HTTP_HEADER_SIZE=12288
```

- `SERVER_TOMCAT_REDIRECT_CONTEXT_ROOT`: Embedded Tomcat for Spring Boot has a default behavior to redirect a request without a trailing slash to one with a trailing slash. However, that redirect happens without evaluating forward headers like X-Forwarded-Proto. So, when SSL offloading is in place, the redirect happens to `http` instead of `https`. This can be an issue especially for load balanced environments. To disable the default behavior, set this variable to false.

Warning! You should do this only if the default behavior is not working. If you are offloading SSL at a proxy or load balancer, see the [Load balancing and proxies](#) topic. Then, if you have problems accessing the application in the browser when you do not include a trailing slash at the end of the request, try setting this variable to false.

- `DEBUG`: A boolean `true` or `false` which will enable or disable debug logging. The log file is by default named `treqss.log`. But, the location and name of that file can be customized by setting an environment variable `LOGGING_FILE`. The log can also be accessible through an API with administrative authorization. Please refer to logfile in the Monitoring section of this documentation, below.
- `DEPLOY_LOCATION`: the location of the `TransferEquivalencyUI.jar` file. This location should be set then referenced in the `java -jar` command that starts the application.
- `LOGGING_FILE`: the name of the file including the path location. If this is specified, the `LOG_PATH` should not be used.
- `LOG_PATH`: the location of the log file. The default PATH location is the Java temporary java folder from the Java property `java.io.tmpdir`. This is usually `/tmp`. The file name will be `treqss.log` for `TransferEquivalencyUI`.

- **LOG_DEFAULT_THRESHOLD:** If set to **DEBUG**, the application will output more verbose logging. Debug output will be sent to the log during the startup process and for all requests thereafter. Warning: this will result in very large log files. If not set, the debug threshold is **INFO**.
- **DW_ENABLE_MONITOR** - This environment variable can be set to enable **/health**, **/metrics**, **/prometheus**, and **/status** endpoints for health and monitoring metrics. By default, this monitoring functionality is not enabled. Adding this variable to an application's startup script and setting it to **true** would enable these endpoints for the application. See individual API documentation in the API Catalog for additional information.
- **DW_RABBITMQ_SSLALGORITHM:** Specify the SSL algorithm to use when connecting to the RabbitMQ broker using SSL. If not specified, it uses the current RabbitMQ driver's default (at least TLSv1.2). This is only considered if the Shepherd setting **core.amqp.useSsl** is set to **true**.

Database pool configuration for microservice applications

For production deployments of java applications, it is important to be able to configure the size of the database connection pool. In Tomcat, this is done through the Resource definition in **server.xml**. For microservice applications like **TransferEquivalencyUI**, there are environment variables that can be used for this configuration.

These environment variables start with **DW_DATASOURCE_**. They end with the name of the datasource pooling attribute you want to control. These are the most common attributes:

Attribute	Meaning
initialSize	The initial number of connections that are created when the pool is started.
maxActive	The maximum number of active connections that can be allocated from this pool at the same time.
maxIdle	The maximum number of connections that can remain idle in the pool, without extra ones being released.
minIdle	The minimum number of active connections that can remain idle in the pool, without extra ones being created.

The name of the attribute should be given in all uppercase, with an underscore separating the words (denoted by a capital letter in the table above). To set the maximum number of active connections in the pool, for example, put the following line in your startup script:

```
export DW_DATASOURCE_MAX_ACTIVE=100
```

You can set any of the properties of a version 1.4 Apache Commons BasicDataSource object. These are documented at <http://commons.apache.org/proper/commons-dbcp/api-1.4/org/apache/commons/dbcp/BasicDataSource.html>.

The number of active connections and the current configured values for the limits can be monitored using any JMX (Java Management Extensions) client, such as JConsole, attached to the microservice JVM. The values will appear under the mbean **net.hedtech.degreeworks/JmxBasicDataSource/datasource**. The **getState** operation will return a formatted report of the current pool state.

Deployment examples

Updated: September 30, 2022

When the prerequisites and configuration are in place, start the application using a `java -jar` command.

```
java $JAVA_OPTIONS -jar TransferEquivalencyUI.jar
```

Example startup script for TransferEquivalencyUI

Updated: March 24, 2023

You may want to employ a shell script to start the application.

Note that the `TransferEquivalencyUI.jar` file must be staged at the `DEPLOY_LOCATION`:

```
#!/bin/ksh
export SPRING_DATASOURCE_URL="jdbc:oracle:thin:@MyServer:1521/servic
e_name"
export SPRING_DATASOURCE_USERNAME="my-user"
export SPRING_DATASOURCE_PASSWORD="my-password"
export SERVER_PORT="8451"
export SERVER_SERVLET_CONTEXT_PATH=""
export SERVER_SSL_KEY_STORE="/usr/local/my-keystore.jks"
export SERVER_SSL_KEY_STORE_PASSWORD="my-key-password"
export SERVER_SSL_KEYSTORETYPE="jks"
export SERVER_SSL_KEY_ALIAS="my-keystore-alias"
export DEPLOY_LOCATION="/path/to/jarfile"
export JAVA_OPTIONS="-Xms1024m -Xmx1024m";
export DEBUG="false"
# To enable full-time debug log output, remove the # from the line b
elow
#export LOG_DEFAULT_THRESHOLD=DEBUG
java -jar $JAVA_OPTIONS DEPLOY_LOCATION/TransferEquivalencyUI.jar >
startup-log-file.log &
```

Example stop script for TransferEquivalencyUI

If using a start script like the example above, here is a corresponding example stop script:

```
#!/bin/ksh
pkill -f TransferEquivalencyUI
```

Monitoring

Updated: March 25, 2022

There are several built-in monitoring API which are accessible to users who are assigned the DWTEADMN Shepherd key.

These endpoints are enabled by default but they can be disabled by configuring the environment variable `endpoints_actuator_enabled=false`. Please see Spring documentation for more information about each: <http://docs.spring.io/spring-boot/docs/current/reference/html/production-ready-endpoints.html>.

- /health
- /actuator
- /dump
- /info
- /beans
- /autoconfig
- /env
- /mappings
- /configprops
- /trace
- /logfile
- /metrics

Transfer Equivalency Self-Service navigation

Updated: March 25, 2022

A navigation bar at the top of Transfer Equivalency Self-Service helps users navigate through the application.

The user can access online help and log out of the application from the navigation bar.

The primary header on the main pages of the application allows the user to refresh the session data, save their data or access the Goal and Transfer page side panels.

Log out

Updated: March 25, 2022

The Log Out button will only display if the user logs on to an existing account, or creates an account either from the landing page or while saving data from within the application.

When the application is used anonymously, the Log Out button will not display.

Start over

Updated: March 25, 2022

The Start Over button takes the user back to the Basics page and resets their input data.

For a user who hasn't saved any data, all their selected goals and entered classes/exams will be cleared. For a user who has saved data and logged on, all their selected goals and entered classes/exams will be reset to the values from their last save and any changes to selected goals and entered classes/exams will be lost.

Save

Updated: March 25, 2022

The Save button will be inactive until the user makes a change to their goals or classes/exams.

If the user is logged on, clicking the button will update the saved goals, classes and exams in the database.

If the user is not logged on, they will be prompted to create a new account to save their data. The user will not be able to log on to an existing account from within the application, only from the Landing page.

Process waypoint bar

Updated: March 25, 2022

On the main pages of the application, there is a process waypoint bar that indicates the user's progress.

This always present waypoint system will both inform the user of their overall progress, and allow for navigation by clicking on the various steps.

A navigation waypoint is bold for the step at which they are currently, active but not bold for a previous stage, and greyed out for a future stage.

When a user is on the Goals page, the Basics process waypoint will be bold while Transfer and Results will be greyed out and inactive. When a user selects all of the required goal data, the Transfer and Results process waypoints will become active and the user can click on them to navigate to other steps in the process.

Landing page

Updated: March 25, 2022

On the Landing page, the prospective student can select how they want to use the application.

The user will have the option to continue anonymously without signing in, logging in with a previously created account, creating a new account, or logging on with an existing Facebook account.

If the `treq.selfService.enableAnonymous` setting is false, the **Continue without signing in** button will not display.

If the `treq.selfService.persistence.enableShpAccount` setting is false, the **Have an account? Log in** and **Create an account** buttons and the Save icon on other pages will not display.

If the `treq.selfService.persistence.enableSocial.facebook` setting is false, the **Sign in with Facebook** buttons will not display.

Note: All of these settings cannot be false, at least one should be true.

You must review the following Shepherd settings to ensure they are configured correctly for your environment:

- `treq.selfService.enableAnonymous`
- `treq.selfService.persistence.enableShpAccount`
- `treq.selfService.persistence.enableSocial.facebook`

Create an account

Updated: March 25, 2022

When the user clicks on the **Create an account** button, they will be prompted to provide an email address, name and password.

These are required fields and cannot be left blank. A `shp_user_mst` record will be created for the user and the data provided by the user will be stored as the `shp_alt_id`, `shp_name` and `shp_access_code`, respectively. This SHP user will be able to save their work and log on again with these credentials. The values for `core.security.newUser.roles` and `core.security.newUser.userClass` will be stored as `shp_key_list` and `shp_filter`, respectively.

You must review the following Shepherd settings to ensure they are configured correctly for your environment:

- `core.security.newUser.roles`
- `core.security.newUser.userClass`

Log in with an existing account

Updated: March 25, 2022

When the user clicks on the **Have an account? Log in** button, they will be prompted for a previously saved email address and password. These are required fields and cannot be left blank.

Goals page

Updated: March 25, 2022

To get started, users provide some information about their degree goals by answering questions about their intended enrollment term, degree, major, level, and other information, making selections from the drop-down lists.

Questions that are mandatory are designated with an * and the user will not be allowed to proceed until they select an answer from the drop-down list.

When they have completed their selections, the wizard will take them on to the Transfer page. If any required fields have been skipped the user will not be able to move forward until they are answered.

As a user responds, their answers will load in the My Transfer Work panel.

Transfer Equivalency Self-Service can be configured to use curriculum rules to filter the goal data drop down lists. Additionally, which goal data can be used for filtering, required goal data and other behavior can be configured. If Transfer Equivalency Self-Service is not configured to use curriculum rules for filtering, the goal data drop down lists will contain all values from the associated UCX table.

Your answers panel

Updated: March 25, 2022

The user can see all of the questions they will be asked by the goal wizard in the Your Answers side panel.

As selections are made, the response will show under the question in Your Answers.

The user can jump to the different questions in Your Answers if they want to make a change to their response. However, they will not be able to jump to optional questions until all required questions have been answered.

Use the standard UCX filter

Updated: March 25, 2022

When Transfer Equivalency Self-Service is configured to not use curriculum rules, standard UCX filtering will be used in the drop-down lists on the Goals page.

Which goals display and the order of display is defined in the `treq.selfService.goalPresentation` setting. The value of this setting is a comma-delimited list of goal prompts in order of visibility. Valid goal prompt names are term, level, degree, major, campus, program, college, concentration and minor. Term, level, degree and major are required but their order can be changed.

The contents of the Term drop-down will be all terms in STU016 with Show in Transfer Equivalency Self-Service = Q or B. The contents of all other goal drop-downs will be all values in the corresponding UCX validation table with Show in Transfer Equivalency Self-Service = Y.

If configured, the Major drop-down will be filtered by the Degree selected by the user.

Review the following to ensure they are configured correctly for your Transfer Equivalency Self-Service environment:

- *Show in Transfer Equivalency Self-Service* in STU016, STU023, STU024, STU035, STU307, STU316, STU350, STU563, STU560, STU576
- *Transfer Equivalency Self-Service Degree Filter* in STU023

Review the following Shepherd settings to ensure they are configured correctly for your Transfer Equivalency Self-Service environment:

- `core.whatIf.enableCurriculumRules`
- `treq.selfService.goalPresentation`
- `treq.selfService.degreeMajorFilter`

Use the curriculum rule filter

Updated: March 25, 2022

When Transfer Equivalency Self-Service is configured to use curriculum rules, the values that appear in the drop-down lists will come from the curriculum rules loaded in `rad_currule_dtl`.

The Term and Degree drop-downs will always display, while Campus, Program, Level, College, Major, Concentration and Minor can be configured to display.

The **Term** will always be displayed and is a required field. The Terms that are in the dropdown come from the codes in STU016 with *Show in Transfer Equivalency Self-Service* flag = Q or B.

If configured to display, the **Campus** will be a required field. The values in the Campus drop down list will only be those from STU576 that are on the curriculum rule.

If configured to display, the **School** will be a required field. The values in the School drop down list will only be those from STU350 with Show in What-If flag = Y or blank that are also on the selected curriculum rule.

If configured to display, the **College** will be a required field. The values in the College drop down list will only be those from STU560 with Show in What-If flag = Y or blank, that are also on the selected curriculum rule. If the College is configured to display before the Degree, College will drive the Degree in the curriculum rule filtering.

The **Degree** will always display and is a required field. The values in the Degree drop down list will only be those from STU307 with Show in What-If flag = Y or blank, that are also on the selected

curriculum rule. If the Degree is configured to display before the College, Degree will drive the College in the curriculum rule filtering.

If the **Program** is configured to display, it will be a required field and the Level, Degree, and College (if displayed) will be inactive. The values in the Program drop down list will only be those from STU316 with Show in What-If flag = Y or blank, that are also on the selected curriculum rule.

The **Major** may be configured to display, and can also be configured to be a required item. The values in the Major drop down list will only be those from AUD027 that are also on the curriculum rule.

The **Concentration** may be configured to display, and will be required if the selected major has AUD027 Concentration Required = Y. The values in the Concentration drop down list will only be those from STU563 with Show in What-If flag = Y or blank, that are also on the curriculum rule. Additional configuration will require that the Concentration be tied to a specific major. If showing all concentrations on a rule or major is enabled, all STU563 values with Show in What-If flag = Y or blank will be displayed.

The **Minor** may be configured to display and will be an optional field. The values in the Minor drop down list will only be those from AUD029 that are on the curriculum rule. If showing all minors on a rule is enabled, all AUD029 values will be displayed.

Review the following to ensure they are configured correctly for your Transfer Equivalency Self-Service environment: *Show in What-If* in STU016, STU035, STU307, STU316, STU350, STU563, STU560, STU576

Review the following Shepherd settings to ensure they are configured correctly for your Transfer Equivalency Self-Service environment:

- core.whatIf.showCampus
- core.whatIf.showProgramPrimary
- core.whatIf.showSchool
- core.whatIf.showCollege
- core.whatIf.degreeBeforeCollege
- core.whatIf.showConcentration
- core.whatIf.concTiedToMajor
- core.whatIf.showAllConcsInPrimary
- core.whatIf.showMinor
- core.whatIf.showAllMinors

Note: When core.whatIf.enableCurriculumRules is true then treq.selfService.goalPresentation is ignored

Transfer page

Updated: March 25, 2022

On the Transfer page, users may enter transfer class or placement exam information that could be articulated and applied to their intended goal.

Entering transfer information is optional; the user may proceed to the Results page without adding transfer classes or exams by clicking the **I'm all done!** button

Add a transfer class

Updated: March 25, 2022

Clicking the **Class** button will open the Select a School page. To add a transfer class, the user must first select a transfer school, then select the transfer class and then enter in the class information.

Select a school

Updated: March 25, 2022

The list of schools is populated with the schools for which you have mappings in *dap_mapping_dtl*.

The City and State comes from the *rad_ets_mst*, if this record exists. Schools are listed in alphabetical order by school, and the list can be filtered by School, City or State.

Closing the Select a School window will take you back to the Transfer page.

Selecting a school will load a list of all the mappings for that school.

Select a class

Updated: March 25, 2022

The list of classes is populated from the mappings for the selected school.

Classes are listed in alphabetical order by course discipline and number, and the list can be filtered by Course Title, Course Discipline or Course Number.

Closing the Select a Class window will take you back to the Transfer page.

Clicking on the Select a School link will take you back to the Select a School page.

Selecting a class will open the Class Information edit page.

Class information

Updated: March 25, 2022

On the Class Information edit page, the user will select the Term in which they took the class, how many Credits they received, the Grade they earned, and select the Credit Type.

The contents of the **Term** dropdown will be all terms in STU016 with Show in Transfer Equivalency Self-Service = Y or B.

The **Credits** field allows numeric values only and follows the format defined in `core.credit.format`.

The contents of the **Grade** drop-down is all grades in STU398 with Show in Transfer Equivalency Self-Service = Y.

The contents of the **Credit Type** drop-down is all values in STU346 with Show in Transfer Equivalency Self-Service = Y. If the `rad_calendar` field on the selected school's `rad_ets_mst` is a valid value in STU346, this value will load as the initial default.

If the **Add another class from this school** box is checked, clicking on the **Proceed** button will add the entered class data to the My Transfer Work panel and return the user to the Class Information page. If the **Add another class from this school** box is not checked, clicking on the **Proceed** button will add the entered class data to the My Transfer Work panel and return the user to the Transfer page. Classes are not saved to the database until the Save button is clicked.

Closing the Class Information window will take you back to the Transfer page.

Clicking on the Select a Class link will take you back to the Select a Class page.

Review the following UCX tables to ensure that they are configured correctly for your Transfer Equivalency Self-Service environment:

- *Show in Transfer Equivalency Self-Service* in STU016, STU346, STU398
- *Convert Number* in STU346

Review the `core.credit.format` Shepherd setting to ensure it is configured correctly for your Transfer Equivalency Self-Service environment.

Add an exam

Updated: March 25, 2022

Clicking the Exam button will open the Exam Information page. The test name and score are required but the date the test was taken is optional.

Exam information

Updated: March 25, 2022

On the Exam Information edit page, the user will select the Test Name, enter the Test Score they received, and select the Test Date.

The contents of the **Test Name** dropdown will be all tests in STU314 with Credit by Exam = Y.

The **Test Score** field allows alphanumeric values up to 4 digits long.

The **Test Date** is optional.

Clicking on the **Done** button will add the entered exam data to the My Transfer Work panel. Exams are not saved to the database until the Save button is clicked.

Closing the Class Information window will take you back to the Transfer page.

Review the following UCX table to ensure that they are configured correctly for your Transfer Equivalency Self-Service environment:

Credit by Exam in STU314

My transfer work panel

Updated: September 30, 2022

Users can see all of the classes and exams they entered in the My Transfer Work side panel, and can edit or delete any of their classes or exams in My Transfer Work.

Transfer Work is ordered by exams and the classes. Exams are ordered by the test name, and classes are ordered by school, and then by course discipline and number.

To edit a class, users can click the **Edit** icon to the right of the class. The class will load in the Class Information page, and users can make edits and click **Proceed** when finished.

To edit an exam, users can click the **Edit** icon to the right of the exam. The exam will load in the Exam Information page, and users can make edits and click **Done** when finished.

To delete one or more classes or exams, users can select the check box next to the item and click the **Delete** icon.

Classes and exams are not modified in or deleted from the database until the **Save** button is clicked.

Results page

Updated: March 25, 2022

On the Results page, the user can view a transfer audit showing how the classes and exams they added might articulate and apply to their selected program goal.

The user can also have the option to generate and download a PDF file of the audit and articulation results, and access external links for more information about transferring to your institution.

The user can access the Results page when the required goal questions have been answered by either clicking on the **I'm all done!** Button on the Transfer page, or by clicking **Results** on the process waypoint bar.

Transfer audit

Updated: September 29, 2023

When the user accesses the Results page, a request for a transfer audit will be generated.

The classes and exams the student entered and their selected goal data will be passed to the articulation and auditor engines, and an audit worksheet will be returned that includes both the transfer audit and the articulation results.

The report format and audit xsl to be used for the transfer audit are defined in Shepherd settings. The standard RPT036 report format is TRQ31 and the standard xsl for this audit is SelfServiceAudit.xsl. Note that the RPT036 Title and XSL Stylesheet fields are not used by Transfer Equivalency Self-Service, but the other flags in this table are used to configure the transfer audit.

The Course Equivalencies section is organized by each sending school, and the transfer course number, title, grade, credits, and term taken display on the left while the equivalent course number, title, grade and credits at the partner school display on the right.

Many-to-One (MTO1), One-to-Many (1TOM), and Many-to-Many (MTOM) equivalencies will be grouped together by mapping, to visually demonstrate the relationship between the classes.

You must review the following Shepherd settings to ensure they are configured correctly for your environment:

- classicConnector.serverNameOrIp
- core.site.displayName
- core.treq.selfService.auditArticulate.report.default

Auto resolution

Updated: March 25, 2022

Transfer Equivalency can be configured to resolve duplicate and undecided situations automatically based on a set of defined rules.

Rather than returning an unresolved articulation, if more than one matching mapping is found (an undecided scenario) or if the student has more than one transfer class that maps to the same course(s) at the transfer school (a duplicate scenario), a hierarchy of resolution rules can be defined and if a match still cannot be determined after working through the hierarchy, it will check the defined "rule of last resort".

Transfer classes for which no mapping can be found can be configured to map to a generic course. If this setting is used, then the course will be counted in the overall degree credits and can apply to rules on the transfer audit.

Review the following Shepherd settings to ensure they are configured correctly for your Transfer Equivalency Self-Service environment:

- articulation.leftoverCourse.courseDiscipline
- articulation.leftoverCourse.courseNumber
- articulation.resolutionRules.duplicate
- articulation.resolutionRules.undecided
- articulation.ruleOfLastResort

Resolution options

Updated: March 25, 2022

These resolution options apply to the Duplicate and Undecided settings.

You can specify multiple options using a comma-delimited list of these options. These options should be applied in list order until a final mapping is found or until all resolution options have been tried.

The sub sections that follow provide an explanation of each resolution option followed by a list of configuration settings and their supported resolution options.

CREDITMAX

For all the potential mappings, add up the total amount of credits that would be awarded in each. The mapping that would result in the most transfer credits wins.

For example, you can have the following Undecided situations:

- a. MATH 101, MATH 102 → MATH 200, MATH 300, MATH 400
- b. MATH 101, MATH 103 → MATH 51, MATH 22

Or

- c. ENG 35, ENG 36, ENG 37 → HUMN 104
- d. ENG 35, ENG 38 → ENG 200

In Case A, MATH 101 is part of a pair of transfer classes that maps to three local classes. In Case B, the transfer class pair maps to a pair of local classes. But all that matters is the total number of credits awarded by the local classes. If the total transfer credits in Case B is 10 and Case A is 8, then CREDITMAX says the articulation engine would choose Case B.

Similarly, in Case C and D, even though ENG 35 is part of a transfer class pair in one mapping and a trio in another, all we care about is if HUMN 104 awards more or less credits than ENG 200.

LOCALHIGH

Use the mapping with the highest numbered local course (course at the Target School). This can be used any time transfer classes can be mapped to more than local course.

- a. MATH 101, MATH 102 → MATH 200, MATH 300, MATH 400
- b. MATH 101, MATH 103 → MATH 51, MATH 22

Or

- c. ENG 101 → ENG 200
- d. ENG 101 → ENG 500

Or

- e. ENG 101 → ENG 200
- f. ENG 102 → ENG 200

In Case A and Case B, MATH 400 is the highest numbered local course in any of the resulting mapping options, so that is the option the articulation engine would choose.

Similarly, in Case C and Case D, ENG 500 is the highest numbered local course, so the articulation engine will choose it if using LOCALHIGH.

In Case E and F, both local courses are the same. This is a Duplicate situation. In this case, LOCALHIGH would have no effect in resolving the mapping and the auditor engine would move to the next resolution option, so it really should not be used at all. However, if it is used, it will not cause any errors.

Course numbers will be compared numerically to determine which is highest (any alpha characters will be stripped out and the resulting integer will be used). That means that MATH 30L, CALC 30, and STAT 3H0 are the same number.

TRANSFERHIGH

Use the mapping with the highest numbered source class (the classes taken by the student).

- a. MATH 101, MATH 102 → MATH 200, MATH 300, MATH 400
- b. MATH 101, MATH 103 → MATH 51, MATH 22

Or

- c. ENG 101 → ENG 500
- d. ENG 102 → ENG 200

Or

- e. ENG 101 → ENG 200
- f. ENG 101 → ENG 500

In Case A and Case B, MATH 103 is the highest numbered transfer class in any of the resulting mapping options, so that is the option the articulation engine would choose.

In Case C and Case D, ENG 102 is the highest numbered transfer class, so it is the option the articulation engine would choose.

In Case E and Case F, the same transfer class is used in both mapping pairs, so TRANSFERHIGH will not lead to a resolution and the articulation engine will need to move to the next option.

As with LOCALHIGH, class numbers will be compared numerically.

RECENT

Use the class that was taken most recently. This only applies to transfer classes. The term would determine which class was most recent (for example, Spring 2013 is more recent than Fall 2012).

RANDOM AND NONE

RANDOM allows the auditor to pick a mapping arbitrarily. This resolution option could lead to different results each time. NONE tells the auditor do not resolve this mapping.

- a. MATH 101, MATH 102 → MATH 200, MATH 300, MATH 400
- b. MATH 101, MATH 103 → MATH 51, MATH 22

In Case A and Case B, RANDOM means the auditor may choose A one time and B the next.

It is recommended that RANDOM is used as the Rule of Last Resort rather than NONE.

REPEAT POLICY AND RESOLUTION OPTIONS

The RAD_COURSE_MST.RAD_REPEAT_MAX for the target school class can affect the resolution option. That field can be set to a value between 1 and 99 or Y, which indicates that a class can be repeated an infinite amount of times.

For duplicate situations, this means that multiple source school classes can articulate to the same target class. For example:

MATH 101 → CALC 100
MATH 102 → CALC 100

If CALC 100 has RAD_REPEAT_MAX set to 1-99 or Y, then both classes will articulate to CALC 100 in a transfer audit.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- articulation.resolutionRules.duplicate
- articulation.resolutionRules.undecided
- articulation.ruleOfLastResort

Download PDF

Updated: March 25, 2022

If configured, the user may download a PDF of the transfer audit and articulation results.

Depending on browser configurations, the user will either be prompted for a location where the file is to be saved or the file will be saved in the default location configured in the browser. The default filename is a combination of the level, degree and catalog year.

Review the `treq.selfService.createAuditPDF` Shepherd setting to ensure it is configured correctly for your Transfer Equivalency Self-Service environment.

Other options

Updated: March 25, 2022

An unlimited number of external links can be configured to display on the Results page. These links will appear after the Download PDF button.

To add an external link, you need to add the label for the button in a properties file and add a Shepherd setting through Controller with the URL.

The Shepherd setting and properties file label should be in the format `treq.selfService.nextSteps.##`, where # is a sequence number, and must match. The sequence numbers do not need to be consecutive, but the links will display in sequence order. A good practice would be to use sequence numbers with a gap that leave some room to insert additional links at a future date, positioned as desired, without having to modify existing settings. For example, sequence numbers of 10, 20, 30 would be a good pattern because then you can decide later that you want to insert a new one at the second position by giving it a key of 15.

For an example, add the following settings in Controller with a Specification of `treqss`:

Key	Value
<code>treq.selfService.nextSteps.10</code>	<code>http://www.yourschool.edu/application</code>
<code>treq.selfService.nextSteps.20</code>	<code>http://www.yourschool.edu/moreinformation</code>

In `TransferEquivalencyMessages.properties`, add the following labels:

```
treq.selfService.nextSteps.10=Apply for Admission  
treq.selfService.nextSteps.20=More Information
```

Transfer Equivalency Self-Service localization

Updated: March 25, 2022

All the text displayed in Transfer Equivalency Self-Service can be localized by either modifying UCX tables or a properties file.

The UCX Tables section of this document describes which UCX tables are used in Transfer Equivalency Self-Service. Modifying code descriptions in these tables will alter the contents of drop-down lists.

Application text, labels, and messages

Updated: March 25, 2022

The text, labels, and messages that appear in the application come from a properties file.

For information on how to modify application properties, see the [Localize an application](#) topic.

Graphics and styles

Updated: March 25, 2022

The graphics and styles used in Transfer Equivalency Self-Service can be modified by overriding the default style sheet (CSS).

For information on how to modify CSS, see the [Localize an application](#) topic.

Debug Transfer Equivalency Self-Service

Updated: March 25, 2022

Transfer Equivalency Self-Service users can enable debugging to troubleshoot issues for their session from the application.

Users with access to this functionality will be able to enable and disable debug from a page in Transfer Equivalency Self-Service. Most debugging output for the application is aggregated into one file, even if multiple applications or APIs are involved. Access to the application server is not needed and the application does not need to be restarted. However, access to the server is required to access the generated log file, which will be in the logs directory of the application server.

Debug access

Updated: March 25, 2022

To access the debugging functionality, users should be assigned the DEBUG key.

For Transfer Equivalency Self-Service, this can be done several ways, depending on how the user is using the application.

To debug issues for anonymous users, the DEBUG key should be assigned to the user defined in the `core.security.referenceUser.username` setting, which is typically TREQER. Use the Users function of Controller to assign `core.security.referenceUser.username` access. All anonymous users will then see the Debug link in the Navigation Bar. Note that if `core.security.referenceUser.username` has the DEBUG key and a user signs in with an existing account, logs on with Facebook or creates an account, they may see the Debug link while on the Landing page, but it will no longer appear after entering their credentials, unless the DEBUG key has been assigned as described below.

To debug issues for users with an existing account or signing in with Facebook, the DEBUG key should be assigned to that user through the Users function of Controller. After logging in, they will then see the Debug link in the Navigation Bar.

To debug issues for users creating an account, the DEBUG key should be assigned in the `core.security.newUser.roles` setting. After creating an account, they will then see the Debug link in the Navigation Bar.

Clicking the Debug link will take the user to the Debug page.

Debug enabling

Updated: March 25, 2022

On the Debug page, toggle the Enable Debug switch to turn debugging on for the Transfer Equivalency Self-Service application.

A random Debug Tag will be generated, and the user can copy this to their clipboard by clicking on the page icon. This Debug Tag will be appended to each line of debug in the log file so that the user can easily identify which lines are from their session, as opposed to other users using the application at the same time.

After enabling debug, the user can resume using Transfer Equivalency Self-Service by clicking on the Basics link and then execute the steps to duplicate the issue they are experiencing. Debug should be turned off by going back to the Debug page and toggling the Enable Debug switch when the issue has been duplicated. Debug for this user's session will also be turned off when they log out of the application.

Debug file access

Updated: March 25, 2022

Debug will appear in the Transfer Equivalency Self-Service log file on the application server.

This is typically defined by the `$LOGGING_FILE` environment variable.

Searching for the user's Debug Tag will isolate the lines of debug specific to their actions. For example:

```
2017-10-24 16:22:23,144 68b4e19d-3fb4-4975-9815-83eea01266ff DEBUG [
41-exec-17] .h.d.s.s.StatelessPreAuthenticatedFilter : Checking secu
re context token: null
```

The *packdebug* script can be used to collect the log files generated for a user session's Debug Tag. For more information on using this script, see the [packdebug script](#) topic.

Transfer Finder Administration

Updated: March 25, 2022

Transfer Finder allows students to apply their classwork at their home institution to the requirements at a partner school and generate a transfer audit. The goal is to let students see their transfer potential at the partner school.

Transfer Finder requires a Degree Works installation at each partner school and an agreement between the partner schools to share data. Additionally, a central server is required to help translate the unique grade, calendar terms, and school codes at each school in addition to managing the communication between Degree Works installations.

While Transfer Finder is provided as a baseline feature, a services engagement is highly recommended to facilitate implementation.

You can set up and configure Transfer Finder in the Responsive Dashboard application.

Initial Transfer Finder Setup and Configuration

Updated: March 25, 2022

When configuring Transfer Finder in Responsive Dashboard, you must have already set up and deployed Responsive Dashboard.

In addition to each partner school having its own Degree Works installation, a central Degree Works installation is needed to manage centralized configuration settings and services. Additionally, a set of global configuration resources with XML output must be provided.

For partner schools, the Transfer Finder components are installed through a standard Degree Works update or new installation package. The Responsive Dashboard and API Services JAR files also need to be deployed.

The central server should be set up through a standard Degree Works new installation package. In the installation process, respond “Y” to the Transfer Finder central server installation prompt. You also need to deploy the Controller and API Services JAR files for the central server, but the Responsive Dashboard JAR file does not need to be deployed on the central server.

Central server

Updated: March 25, 2022

The central server is used to translate codes for terms, grades and academic calendars between the partner schools. There are also several configurations that manage the overall behavior of Transfer Finder that reside in the central server.

Central services user

Updated: March 25, 2022

A default standard user for central services, TFCENTRL, was created during the software installation.

This user is used to connect to the central server from local environments when running the etssync and ucxsync scripts. You must set the password in the central environment for this user to something only known to your team through the Users tab of Controller. Make note of these values so you will have them when you configure the local settings (transferFinder.service.central.password and transferFinder.service.central.username).

This central services user should be assigned Shepherd keys:

```
RSSETTING
RSVALID
SHENCRPT
```

Central server Shepherd settings

Updated: March 25, 2022

Most configurations to manage the behavior of Transfer Finder can be maintained using the Configuration tab of Controller.

The central server requires certain Shepherd settings for the configuration of Transfer Finder, including the URLs for the global directories, the logon credentials for these global directories, and the logon credentials for services run from the central server. Review all entries with the transferFinder.central.* and articulation.* prefixes and ensure that they are configured correctly for your Transfer Finder environments.

These settings should be the values used when setting up the Global Configuration Resources:

```
transferFinder.central.directory.password
transferFinder.central.directory.username
transferFinder.central.globalDirectory.uri
transferFinder.central.schoolDirectory.uri
transferFinder.central.transferDirectory.uri
```

These settings should be the values used when setting up the Global Services User in the local environments:

```
transferFinder.central.partnerService.password
transferFinder.central.partnerService.username
```

These settings define the general education category values that may display in the transfer audit:

```
transferFinder.central.worksheet.category.all.complete.minimum
transferFinder.central.worksheet.category.all.exclude.list
transferFinder.central.worksheet.category.block.type
transferFinder.central.worksheet.category.block.value
transferFinder.central.worksheet.category.classes.minimum
```



```
transferFinder.central.worksheet.category.credits.minimum
transferFinder.central.worksheet.category.discrete.list
transferFinder.central.worksheet.category.special.complete.minimum
transferFinder.central.worksheet.category.special.list
```

These settings define credentials for a Banner API to retrieve class schedule information:

```
transferFinder.central.classSchedule.banner.password
transferFinder.central.classSchedule.banner.username
transferFinder.central.classSchedule.banner.termEndOffset.numberOfDay
ys
transferFinder.central.classSchedule.banner.termStartOffset.numberOfDay
Days
```

The above settings should only exist on the central server.

Transfer Finder has several settings to define the expected behavior for the articulation of classes in the transfer audit. The `articulation.*` settings can reside on the central and the local servers. However, if these settings reside on the central server, the central server settings will override the local server settings in Transfer Finder. See [Transfer Audit Detail](#), *Resolution Options* section for more information on configuring these settings.

```
articulation.leftoverCourse.courseDiscipline
articulation.leftoverCourse.courseNumber
articulation.resolutionRules.duplicate
articulation.resolutionRules.undecided
articulation.ruleOfLastResort
```

Central server UCX tables

Updated: March 25, 2022

When classes are manually added in the Classwork History section or classes are sent from one school to another for articulation in a transfer audit, there needs to be a way to translate the values for the source school code for Grade, Term, and Academic Calendar Type (for example, Semester, Quarter) into values that the receiving school understands.

These global codes are maintained in the central server database using Controller. The `ucxsync` script updates global codes in the local Degree Works instance. The global codes do not need to contain every value from every partner school. For example, you don't need to put in terms that are no longer supported for transfer. However, the global codes should contain all the values needed to manage class articulation between source classes from all partner/non-partner schools and target classes at partner schools in a transfer audit.

Create the global term, academic calendar, and transfer grade codes in TRF100, STU346, and STU398 on the central server. These should be the common values that will be used throughout the system and local school codes will be mapped to these.

TRF100 – Global Term Codes

TRF100 stores the common term codes used by the system and is used to map the global codes used system-wide to the local school's codes. Create the global term codes in this table on the central server. These entries will be copied to every local server through the `ucxsync` script, and

then the local school will need to map each global term code to the corresponding STU016 term code for that school.

STU346 – Global Academic Calendar Codes

The types of academic calendars used by partner and non-partner schools should be defined in STU346. These codes will be used when articulating courses to calculate the correct number of credits for a course. Define these global academic calendar codes in STU346 on the central server. These entries will be copied to every local server through the ucxsync script, and then the local school will need to review the STU346 Convert Number to ensure it is accurate for their academic calendar.

STU398 – Global Transfer Grades

STU398 stores the common transfer grades used by the system and is used to map the global codes used system-wide to the local school's transfer grade codes. Define the global transfer grades in this table on the central server. These entries will be copied to every local server through the ucxsync script, and then the local school will need to map each global transfer grade code to the corresponding entry in STU385 at the local school.

The Numeric Grade field on STU398 should be populated as it used by the interface to know how to sort the grades appearing in drop-downs.

Global configuration resources

Updated: March 25, 2022

Transfer Finder requires three configuration resources for the system – the Global Directory, School Directory, and Transfer Directory.

The choice of how to provide these resources is up to the school system. They would typically reside on the central server and can be a simple XML file hosted on a web server, or a web service backed by a database that provides XML in response to a request. An easy way to host the Transfer Finder directory files is to use a simple web server such as Apache. Configure a directory and then place the files in the directory.

The location of each of these resources and the username and password needed to access them are stored as central server Shepherd settings. Make note of these values so you will have them when you configure the central server.

There are three types of configuration resources that must be created for your school system:

Configuration resource	Description
Global Directory	A directory of other resources that can be used by the student to get general information on transferring between schools, to contact an advisor, or to apply for a transfer.
School Directory	A directory of schools both in and out of your partner network.

Configuration resource	Description
Transfer Directory	A list of all programs available to a transfer student for all the schools in your system.

Global directory

Updated: March 25, 2022

This resource provides the location of three other global resources used by Transfer Finder.

The location of this directory is defined in the `transferFinder.central.globalDirectory.uri` Shepherd setting on the central server. An example of the expected XML is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<GlobalDirectory xmlns="urn:com:ellucian:degreeworks:gen:directory:GlobalDirectory_v1.0.0">
  <GlobalResources>
    <TransferHelpUrl>http://dwcentral.system.edu/transfer<Transfer
HelpUrl>
    <ContactUrl>http://dwcentral.system.edu/contact<ContactUrl>
    <ApplyUrl>http://dwcentral.system.edu/apply<ApplyUrl>
  </GlobalResources>
</GlobalDirectory>
```

TransferHelpUrl

This URL will be used when the student requests more information about a transfer program. The actual request sent will contain the local school code (SourceSchool) from the school directory and the area of interest code (from the Transfer Directory) chosen by the student (AreaOfInterest) in the following format:

```
{url}?
mode=transferHelp&SourceSchool={SourceSchool}&AreaOfInterest={AreaOfInterest}
```

For example:

```
http://dwcentral.system.edu/transfer?
mode=transferHelp&SourceSchool=2011&AreaOfInterest=BIOLGY
```

ContactUrl

This URL will be used when the student clicks on the "Find a transfer advisor" button when viewing the results for a particular school. It is intended to provide a link to counselor information, for example- email, phone, etc. The same link is used for any school, however the school ID and other data are provided to the resource so that it can be redirected or customized for the target school.

The actual request will contain the school ID {TargetSchool} and Major {TargetMajor} for the program being considered. The format of the request is as follows:

```
{url}?
mode=contact&TargetMajor={TargetMajor}&TargetSchool={TargetSchool}
```

For example:

```
http://dwcentral.system.edu/transfer?
mode=contact&TargetMajor=MATH&TargetSchool=2011
```

ApplyUrl

This URL will be used when the student clicks on the “Apply now” button when viewing the results for a particular school. It is intended to provide a link to an application for the school. The same link is used for any school, however the school ID and other data are provided to the resource so that it can be redirected or customized for the school.

The actual request will contain the school ID {TargetSchool} and Major {TargetMajor} for the program being considered. The format of the request is as follows:

```
{url}?mode=apply&TargetMajor={TargetMajor}&TargetSchool={TargetSchool}
```

For example:

```
http://dwcentral.system.edu/transfer?
mode=apply&TargetMajor=MATH&TargetSchool=2011
```

School directory

Updated: March 25, 2022

This resource lists all the schools in the system and any school for which the student may manually enter classes.

The location of this directory is defined in the transferFinder.central.schoolDirectory.uri Shepherd setting on the central server. Among other things, this resource defines a set of codes used to identify schools and provides a way to translate this global ID to local values for any school that does not use the global code natively. It serves as a searchable list when the student elects to add classes manually. Finally, it provides a universal resource locator (URL) for Transfer Finder to use when it wants to request classwork and audits from that school. An example of the expected XML is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<SchoolDirectory codeType="CEEB" xmlns="urn:com:ellucian:degreeworks
:gen:directory:SchoolDirectory_v1.0.0">
  <Schools>
    <School Id="2017" Partner="true">
      <Name>Local Community College</Name>
      <City>Anytown</City>
      <State>PA</State>
      <Country>US</Country>
      <ServiceUrl>http://localhost:8200/degreeworks-services/</Ser
viceUrl>
      <ClassServiceUrl>http://localhost:8400/mocksis/</ClassServic
eUrl>
      <ClassServiceAdapterBeanName>ClassAdapter</ClassServiceAdapt
erBeanName>
      <CalendarType>SE</CalendarType>
```

```

    <AlternateId type="OPEID">11-29388<AlternateId>
    <AlternateId type="SCH00L2">S011<AlternateId>
  <School>
    <School Id="1111" Partner="false">
      <Name>University of Washington<Name>
      <City>Seattle<City>
      <State>WA<State>
      <Country>US<Country>
      <CalendarType>QU<CalendarType>
      <AlternateId type="OPEID">21-93822<AlternateId>
      <AlternateId type="SCH00L2">S020<AlternateId>
    <School>
  <Schools>
<SchoolDirectory>

```

Element	Description
School	<p>The primary element in the directory. There must be one and only one of these elements for any particular school. The Id attribute is required and should be set to the global code associated with the school. The Partner attribute is also required and should be set to "true" for any school that belongs to your system.</p> <p>Transfer Finder expects to be able to contact partner schools through the required (for partner schools) ServiceUrl element.</p>
Name	The name of the school. This is searchable and will be displayed to the student.
City	The city where the school is located. This is searchable and will be displayed to the student.
State	The state where the school is located. This is searchable and will be displayed to the student.
ServiceUrl	This is the URL used to contact Degree Works services at this school. It is required for partner schools. It should be set to the context of the Web Services installation at the school.
ClassServiceUrl	<p>This is the URL used to retrieve class information when the student looks for classes at another school.</p> <p>This is not a resource provided by Degree Works. It is optional, and should only be provided if the student information system provides such a resource.</p>

Element	Description
ClassServiceAdapterBeanName	<p>The name of the adapter used by Degree Works to access the ClassServiceUrl.</p> <p>This is optional and depends on the source of the Class Service resource. For Banner, this should be bannerClassScheduleAdapter.</p>
CalendarType	This indicates the type of academic calendar in use by the school for example, semester vs. quarter. This value must be valid on the global UCX table STU346.
AlternateId	<p>This is another type of ID code for the school. If any of the schools in the system use a different ID type from the global code, then those codes can be entered in this directory as an alternate ID. The "type" attribute identifies the type of code being used. This can be any value.</p> <p>The schools using this type must have this code placed in the core.site.idType Shepherd setting. A utility called etssync is available that can read this resource and update the local RAD_ETS_MST, setting the RAD_ETS_USER to the global code for that school.</p> <p>This is required for any school not using the global set of ID codes.</p>

Transfer directory

Updated: March 25, 2022

This resource provides a list of all transfer programs for all schools.

It is used primarily when the student is searching for a transfer option to another school. The student can search through this resource by area of interest, major, and school, among other things. The data in this tag is passed along with the student's academic record to the target school and specifies the degree, major, and other program-oriented data necessary for that school to do an articulation of the student's record and an audit. The location of this directory is defined in the transferFinder.central.transferDirectory.uri Shepherd setting on the central server.

It is important to note that the values in a Program entry are used to generate the transfer audit and must have corresponding Scribe blocks at that school. At a minimum, a degree and major must be provided. In addition, if the program has a minor, concentration, or other goal data that make it unique, they must also be provided here. An example of the expected XML:

```
<?xml version="1.0" encoding="UTF-8"?>
<TransferDirectory xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xmlns="urn:com:ellucian:degreeworks:gen:directory:transfer:Tran
```

```

sferDirectory_v1.0.0">
  <AreasOfInterest>
    <AreaOfInterest>
      <Code>BIOL0GY<Code>
      <Description>Biology<Description>
    </AreaOfInterest>
    <AreaOfInterest>
      <Code>CHEM<Code>
      <Description>Chemistry<Description>
    </AreaOfInterest>
  </AreasOfInterest>
  <Programs>
    <Program schoolId="2017" schoolType="UG" areaOfInterest="BIOL0
GY">
      <Degree>
        <Code>BS<Code>
        <Description>Bachelor of Science<Description>
      </Degree>
      <Major>
        <Code>10883<Code><!--This is the code at the local school
-->
        <CategoryCode>01.0101<CategoryCode><!-- This is a CIP le
vel 6 code. -->
        <Description>Bioengineering<Description>
      </Major>
    </Program>
    <Program schoolId="2214" schoolType="UG" areaOfInterest="BIOL0
GY">
      <Degree>
        <Code>BA<Code>
        <Description>Bachelor of Arts<Description>
      </Degree>
      <Major>
        <Code>11995<Code>
        <CategoryCode>01.0101<CategoryCode>
        <Description>Biological Sciences<Description>
      </Major>
      <Minor>
        <Code>CS<Code>
        <Description>Computer Science<Description>
      </Minor>
      <Concentration>
        <Code>JS<Code>
        <Description>Javascript<Description>
      </Concentration>
    </Program>
  </Programs>
  <MajorCategories>
    <Category>
      <Code>01.01<Code>
      <Description>Agricultural Business and Management.<Descript
ion>

```

```

    <Category>
    <Category>
      <Code>01.0101<Code>
      <Description>Agricultural Business and Management, General.<
/Description>
    <Category>
    <MajorCategories>
  <TransferDirectory>

```

There are three main sections to this resource: AreasOfInterest, MajorCategories, and Programs.

AreasOfInterest

AreasOfInterest define codes that allow the system to group programs into fewer areas that enhance the student's ability to search through the large number of programs. Not all programs require an AreaOfInterest, so the system may decide to provide a limited number of common or otherwise convenient set of areas that facilitate the student's experience.

AreaOfInterest elements:

- Code - the code used in the areaOfInterest attribute of the Program tag.
- Description - a descriptive literal for the code. This is displayed to the student when selecting by area.

MajorCategories

MajorCategories define standard major codes that facilitate searching by major. Trying to provide a picklist of majors based on all the varieties available at all the schools would result in an inconstant and confusing list. Instead, a set of standard set of major categories is defined and the student searches through this list. Each of the programs is marked with the standard category so it can be retrieved based on the student's category selection.

MajorCategories elements:

- Code - the code used in the CategoryCode element in the Major tag under Program.
- Description - a descriptive literal for the code. This is displayed to the student when selecting by major.

Programs

Programs is the main section and lists all the academic programs open to students for transfer.

Program elements:

- Program - the only child of Programs. This represents one choice of an academic program that can be chosen by the student in Transfer Finder. It has up to 3 attributes:
 - schoolId – the Id of the school providing this program, from SchoolDirectory resource. Required.

- schoolType – the Degree Works school code associated in Scribe for this program. Required.
- areaOfInterest – the area of interest code associated with this program. Defined in the AreasOfInterest section of this resource. Optional.
- Degree - contains the Code and Description for the degree offered by this program. Required. The Code and Description elements, as defined below, are required children of this element.
- Major - contains the Code and Description for the major offered by the program. Required. The Code, Description, and CategoryCode elements, as defined below, are required children of this element.
- Minor - contains the Code and Description for the minor offered by this program. Optional. However, if the audit requires a minor for this program, then it must be specified here. The Code and Description elements, as defined below, are required children of this element.
- Concentration - contains the Code and Description for the concentration offered by this program. Optional. However, if the audit requires a concentration for this program, then it must be specified here. The Code and Description elements, as defined below, are required children of this element.
 - Code - child of Degree, Major, Minor, and Concentration. This is the code specified in the Scribe requirements that define this program. Required.
 - Description - child of Degree, Major, Minor, and Concentration, and always associated with the Code element. This is a descriptive literal for Code. In particular, the Description under Major is used as the description for the entire program, so it must be unique, at least among all the programs for a school. If there are two or more programs, for example, that use the same major code, perhaps being differentiated by concentration, then the description for those two programs should be altered so that the student can tell them apart. Required.
- CategoryCode - child of Major. Required. This is the major category associated with this program, and is defined in the MajorCategories section. This is used to standardize majors for the purpose of searches and selections.

Local schools

Updated: March 25, 2022

Each local or partner school environment must have the Responsive Dashboard and API Services JAR files. Additionally, there are also several configurations that must be done for each local school.

Global services user

Updated: September 30, 2022

A default standard user for global services, TFGLOBAL, was created during the software installation.

This user is used by Transfer Finder to connect to other institutions to get student data and perform articulations. You must set the password for this user through the Users tab of Controller in each local environment to the same value and something only known to your team. Make note of these values so you will have them when you configure the central settings (`transferFinder.central.partnerService.password` and `transferFinder.central.partnerService.username`).

This global services user should be assigned by Shepherd keys:

```
NOREFER
RSAUDIT
RSCCLASS
RSCRSINF
RSMAPPING
RSSETTING
RSVALID
SDSTUANY
TFCATGRY
TFTRAUDT
```

Local school Shepherd settings

Updated: March 25, 2022

Use the Settings tab of Controller in each local school environment to configure the Shepherd Settings required by Transfer Finder.

In addition to the central server configuration information, there are several settings that control the appearance and behavior of Transfer Finder in the local environment. Review the following entries with the `transferFinder.*` and `core.*` prefixes in each local school environment and ensure that they are configured correctly.

These settings should be the values you used when setting up the Central Services User on the central server:

```
transferFinder.service.central.password
transferFinder.service.central.username
```

This setting should be the URL for API Services in the central server:

```
transferFinder.service.central.uri
```

These settings define the appearance and behavior of Transfer Finder in the local environment:

```
transferFinder.acknowledgement.show
transferFinder.courseSearch.enableSchoolSelection
transferFinder.courseSearch.limitResultsToPartners
transferFinder.creditType
transferFinder.snapshot.audit.process.includeTransferWork.codeList
transferFinder.snapshot.audit.process.includeTransferWork.defaultValue
transferFinder.snapshot.audit.process.includeTransferWork.enabled
transferFinder.snapshot.audit.process.inprogress.defaultValue
```

```
transferFinder.snapshot.audit.process.inprogress.enabled
transferFinder.snapshot.audit.process.preregistered.defaultValue
transferFinder.snapshot.audit.process.preregistered.enabled
transferFinder.snapshot.enableDisciplines
transferFinder.snapshot.enablePaths
```

These settings define environment variables related to Transfer Finder in the local environment:

```
core.institution.calendarType
core.site.displayName
core.site.id
core.site.idType
```

Transfer Finder has several settings to define the expected behavior for the articulation of classes in the transfer audit. The `articulation.*` settings can reside on the central and the local servers. However, if these settings reside on the central server, the central server settings will override the local server settings in Transfer Finder. See [Transfer Audit Detail](#), Resolution Options section for more information on configuring these settings. The `core.articulation.*` settings only reside on the local server.

```
articulation.leftoverCourse.courseDiscipline
articulation.leftoverCourse.courseNumber
articulation.resolutionRules.duplicate
articulation.resolutionRules.undecided
articulation.ruleOfLastResort
core.articulation.default.catalogYear
core.articulation.default.gradeType
core.articulation.default.school
core.articulation.gradeType
core.articulation.translateGlobalSchoolIds
```

Local environments synchronization with local codes

Updated: March 25, 2022

In each local school environment, use `ucxsync` to load the global code values for TRF100, STU346, and STU398.

The `ucxsync` script will not overwrite existing entries; it will only add the values defined on the central server.

```
$ucxsync --globaldir TRF100
$ucxsync --globaldir STU346
$ucxsync --globaldir STU398
```

Then use Controller to map the TRF100 codes to STU016 terms, cross-map the STU016 terms to TRF100 codes, and update STU385 with the appropriate STU398 values.

TRF100 and STU016 – Global Term Codes

The entries in TRF100 at each local school need to be updated with a valid Local Term from STU016 at the local school. This will map the school's value to a common term code used throughout the system.

Additionally, the STU016 terms that were mapped in TRF100 need to be updated with that TRF100 code in the Transfer Finder Period.

STU346 – Global Academic Calendar Codes

Review the STU346 Convert Number in each local school environment to ensure it is accurate for their academic calendar.

When a local class is articulated during a transfer audit, the local school's `core.institution.calendarType` setting is looked up in the partner school's STU346 to determine how the credits should be adjusted. However, if a mapping exists for the local class at the partner school and the number of credits is specified on the mapping, that credit value will be used for the articulated class instead of the calculated value.

STU385 – Grade Table

Review the entries in STU385 at each local school and update them with the corresponding Resident Grade value from STU398.

STU398 – Global Transfer Grades

Review the entries in STU398 at each local school and update them with the corresponding Transfer Finder Grade value from STU385.

SCR002 and SCR044 - CATEGORY

To use the special CATEGORY RuleTag for scribing system wide GENED requirements, make sure that an entry for CATEGORY exists in both SCR002 and SCR044 in all local Degree Works deployments and the central server deployment.

SCR002 Key	Description	Data Element
CATEGORY	transfer audit Category	0000

SCR044 Key	Description	Element	Offset	Length
CATEGORY	transfer audit Category	ATTR	00	00

Local rad_ets_mst synchronization with global translations

Updated: March 25, 2022

ETS codes are stored in the rad_ets_mst and are extracted from the institution's student information system.

In Transfer Finder, not all schools may be using the same type of school codes (that is, CEEB, FICE). In that case, translations are required. The global school directory specifies the code set that is being used globally and will contain the global codes for all schools. For schools that use the same set of codes in their Student Information System (and therefore in their rad_ets_mst), no translation is necessary. For the other schools, you must translate the global set to their local codes. This will be done by looking up the rad_ets_mst using the global code as a search item for the rad_ets_user column.

Because many schools may commonly use a code set that is different from the global code set, and there may be thousands of schools in the global directory, a means of updating the `rad_ets_mst` from a common crosswalk table is needed. The `etssync` script will update the `rad_ets_user` field with the global school code in each local environment:

```
$etssync
```

It uses the `alternateld` specified in the school directory to find the `rad_ets_mst` by `rad_ets_code`, and then updates the `rad_ets_user` with the global code. This may be the same as the `rad_ets_code` if the school locally uses the same code as the global value.

If all schools use the same code set, then `core.articulation.translateGlobalSchoolIds` should be set to “false”, and you do not need to run `etssync`. Otherwise, set this to “true” and run `etssync`.

User access to Transfer Finder

Updated: March 25, 2022

Access to the features of Transfer Finder is managed by assigning Shepherd keys and groups to users.

This can either be done globally based on a user’s role in `core.security.rules.shpcfg`, or individually on a user’s `shp_user_mst`.

Several standard Shepherd Groups have been delivered to help you organize the Transfer Finder keys for each of the primary user classes. Add these groups on an individual user basis using Controller or add them to a user class in `core.security.rules.shpcfg`. Standard Shepherd Groups that give access to Transfer Finder functionality are TFADV, TFREG, and TFSTU.

For more information, please see the [Access control \(authorization\)](#) topic.

Student extracts for student system ID

Updated: March 25, 2022

If different student IDs are assigned for each school that the student attends in the system, a common system ID will be required to retrieve coursework from partner schools.

This common system ID should be stored in the `shp_user_mst.shp_finder_id`. If the student IDs are the same throughout the system, that ID needs to be stored in the `shp_user_mst.shp_finder_id` as well.

This field can be loaded from the data in the student system into Degree Works through the student data extract. For Banner sites, the student ID must be stored in `GORADID_ADDITIONAL_ID` and will be loaded through the standard Banner extract.

Class schedule adapter

Updated: March 25, 2022

The Course Equivalent functionality in Transfer Finder supports retrieval of schedule information for individual mapped classes from remote systems.

This is accomplished through an adapter that is capable of retrieving data in any format and transforming it to the data structure supported by the software. This document describes two included adapters along with how they are configured. It will also define how other adapters can be implemented to support arbitrary external systems.

classScheduleAdapter configuration

Updated: March 25, 2022

There are two included adapter implementations for Transfer Finder: bannerClassScheduleAdapter for Banner clients and classScheduleAdapter for non-Banner clients.

This is configured on a per school basis in the School Directory and calls out to RESTful APIs (web services) located at predefined URLs. Any student information system that supports or can be modified to support these APIs can be configured to use the classScheduleAdapter.

Two nodes in the School Directory XML are relevant to this configuration: ClassServiceUrl and ClassServiceAdapterBeanName. For the included Banner adapter the ClassServiceAdapterBeanName must be set to the case sensitive name "bannerClassScheduleAdapter". The ClassServiceUrl is the base URL from which all applicable APIs are served. It can be set uniquely for each school. Theoretically, if there were a single system capable of producing class schedule data for many individual schools, then the ClassServiceUrl could be set to the same URL for all of the applicable schools.

For the integration with a Banner SIS, four central server settings must be configured. These define a username and password that gives access to the relevant APIs in addition to term start and end configurations. Each configured Banner system must support the same set of credentials defined in these settings. In addition, the Banner API requires a range of terms to be queried. That range is defined by two settings, the number of days before the current date (termStartOffset) and number of days after the current date (termEndOffset).

```
transferFinder.central.classSchedule.banner.password
transferFinder.central.classSchedule.banner.username
transferFinder.central.classSchedule.banner.termEndOffset.numberOfDay
s
transferFinder.central.classSchedule.banner.termStartOffset.numberOf
Days
```

Adapter implementation for other systems

Updated: March 25, 2022

For student systems for which the RESTful APIs above are not available or cannot be implemented, one can provide any number of implementations of the adapter.

This is supported only in the Dashboard version, but not in Responsive Dashboard. This documentation will describe the interface and Spring configuration necessary to configure new implementations. The audience for this section of documentation is a software developer who is

familiar with coding in the JVM ecosystem, Spring configuration, producing a jar package and installing it on their server container's classpath.

The interface is named:

```
net.hedtech.degreeworks.webclient.sis.service.ClassScheduleAdapterInterface.
```

It can be found in interface defined in degreeworks-webclients-{version}.jar.

This simple interface defines two methods and one data structure to provide enough flexibility so that implementations can be coded to the limitations of unknown, external systems. For example, there is nothing in the interface that forces one to provide an implementation using web service APIs.

Method **getClassScheduleList**

This method is the most important one to implement. Using whatever data access method is appropriate, the implementation of this method should retrieve sections in a student system that match the input `courseDiscipline/courseNumber` parameters and transform it to the data structure known as `ScheduledClassInterface` in Degree Works. That data structure is described below.

```
public List<ScheduledClassInterface> getClassScheduleList(String
    courseDiscipline, String courseNumber);
```

Method **setServiceUrl**

This method must be implemented but it doesn't have to do anything, so the implementation can be "empty". This method will be called with the purpose of informing an implementation of the `ClassServiceUrl` set in School Directory for the school selected by the end user. This could be useful, for example, to have a unique URL, configured per school for the purpose of calling a web service in the `getClassScheduleList` implementation. But there's nothing forcing an implementation to use this for the intended purpose as a URL; it could be any arbitrary String set in the relevant `ClassServiceUrl` node of School Directory.

```
public void setServiceUrl(String url);
```

Data Structure: **ScheduledClassInterface**

These interfaces define the data structure that must be returned as a List in implementations of `getClassScheduleList`. The `ScheduledClassInterface` defines an instance of an individual class or section. The fields that are called for display are highlighted bold and blue.

```
public interface com.ellucian.degreeworks.domain.classes.ScheduledClassInterface
{
    public String getId();
    public void setId(String id);
    public String get_sTerm();
    public void set_sTerm(String termId);
    public String get_sNumber();
    public void set_sNumber(String courseNumber);
    public String get_sDiscipline();
    public void set_sDiscipline(String courseDiscipline);
    public String get_sTitle();
    public void set_sTitle(String title);
    public int get_iAvailable();
```

```

    public void set_iAvailable(int available);
    public int get_iCapacity();
    public void set_iCapacity(int capacity);
    public String get_sTermDescription();
    public void set_sTermDescription(String termName);

    public List<com.ellucian.degreeworks.domain.classes.Meeting> getMeetings();

    public void setMeetings(List<com.ellucian.degreeworks.domain.classes.Meeting> meetings);
}

```

Data Structure: MeetingInterface

Each ScheduledClassInterface contains a collection of MeetingInterface objects:

```

public interface com.ellucian.degreeworks.domain.classes.MeetingInterface
{
    public String getInstructionalMethodCode();
    public void setInstructionalMethodCode(String InstructionalMethodCode);
    public MeetingInterface withInstructionalMethodCode(String InstructionalMethodCode);
    public String getStartTime();
    public void setStartTime(String StartTime);
    public MeetingInterface withStartTime(String StartTime);
    public String getEndTime();
    public void setEndTime(String EndTime);
    public MeetingInterface withEndTime(String EndTime);
    public List<Day> getDays();
    public void setDays(List<Day> Days);
    public MeetingInterface withDays(List<Day> Days);
    public String getRoom();
    public void setRoom(String Room);
    public MeetingInterface withRoom(String Room);
    public String getStartDate();
    public void setStartDate(String StartDate);
    public MeetingInterface withStartDate(String StartDate);
    public String getEndDate();
    public void setEndDate(String EndDate);
    public MeetingInterface withEndDate(String EndDate);
    public String getFrequency();
    public void setFrequency(String Frequency);
    public MeetingInterface withFrequency(String Frequency);
    public Map<String, Object> getAdditionalProperties();
    public void setAdditionalProperties(String name, Object value);
    public String getLocation();
    public void setLocation(String location);
}

```


Data Structure: Day

Each MeetingInterface contains a collection of Day objects where each day of the week is represented and associated with a property name in TransferFinderMessages.properties for each one to have a label that can be localized:

```
public enum com.ellucian.degreeworks.domain.classes.Day
{
    _0("dw.app.calendar.day.short.sunday"),
    _1("dw.app.calendar.day.short.monday"),
    _2("dw.app.calendar.day.short.tuesday"),
    _3("dw.app.calendar.day.short.wednesday"),
    _4("dw.app.calendar.day.short.thursday"),
    _5("dw.app.calendar.day.short.friday"),
    _6("dw.app.calendar.day.short.saturday");
}
```

Configuring Implementations in Transfer Finder

For an implementation to be available to the Transfer Finder application, it must be compiled, probably packaged in a jar, and placed on the classpath of the chosen server/container. Then, a spring configuration file named applicationContext-sis.xml and the file must be placed in the container's classpath as well. The purpose of the spring configuration is to make the implementation class available as a bean in the application's spring context. The bean is identified by a unique ID which is also the name used in School Directory ClassServiceAdapterBeanName nodes for the purpose of associating each school with an adapter.

For example, if we consider a hypothetical adapter implementation named edu.example.MyScheduleAdapter, here is how a sample spring configuration file might look:

```
<?xml version="1.0" encoding="UTF-8"?>
<beans xmlns="http://www.springframework.org/schema/beans"
    xmlns:aop="http://www.springframework.org/schema/aop"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
    default-lazy-init="true"
    xsi:schemaLocation="http://www.springframework.org/schema/beans
        http://www.springframework.org/schema/beans/spring-beans-2.0.xsd

        http://www.springframework.org/schema/aop
        http://www.springframework.org/schema/aop/spring-aop-2.5.xsd">

    <bean id="myChosenNameScheduleAdapter"
        class="edu.example.MyScheduleAdapter">
    </bean>

</beans>
```

To continue the example, if we were to associate this adapter with School Id="2017", it would look like the following. After this is configured and all prerequisite classpath resources are in place, Transfer Finder will attempt to call the getClassScheduleList method of class edu.example.MyScheduleAdapter.

```
<?xml version="1.0" encoding="UTF-8"?>
<SchoolDirectory codeType="CEEB" xmlns="urn:com:ellucian:degreeworks
```

```
:gen:directory:SchoolDirectory_v1.0.0">
<Schools>
<School Id="2017" Partner="true">

<!-- Several required fields are omitted since this focuses on the ad
apter configuration only. -->

<ClassServiceUrl>An optional String value<ClassServiceUrl>
<ClassServiceAdapterBeanName>myChosenNameScheduleAdapter<ClassServic
eAdapterBeanName>

<School>
<Schools>
<SchoolDirectory>
```

Transfer Finder scribing

Updated: March 25, 2022

There are several special Scribe options that allow certain rules and qualifiers in the partner school's blocks to be used only in the transfer audit.

AuditAction=TRANSFER

Updated: March 25, 2022

Using the AuditAction=TRANSFER statement allows certain rules and qualifiers to only apply to the transfer audit.

For example, if you have a set of general requirements that can be completed at any school in the system but is different from the general requirements needed at the local school, you might want to call a different block in the transfer audit rather than add additional and perhaps complex scribing to the existing local block. Adding the following statement to the DEGREE block might be an option:

```
If (AuditAction=TRANSFER) then
  1 Block (OTHER=SYSGENED)
  Label SYSGENED "Ellucian System General Education Requirements"
else
  1 Block (OTHER=GENED)
  Label GENEDBLOCK "General Education Requirement";
```

When a local audit for this DEGREE is generated, it will call in the OTHER=GENED block, but when a transfer audit is generated, the OTHER=SYSGENED block will be called.

The AuditAction=TRANSFER statement can be used on header qualifiers:

```
If (AuditAction=TRANSFER) then
  MinGPA 3.2
else
  MinGPA 2.0
```

and rules:

```
If (AuditAction=TRANSFER) then
  1 Class in ACCT 3a
    Label 8a "300-level Accounting Elective for Transfers"
Else
  1 Class in ACCT 390:399
    Label 8b "300-level Accounting Elective";
```

CATEGORY RuleTag

Updated: March 25, 2022

If a school system has a minimum set of requirements that must be completed for graduation and completing the requirement at one school automatically completes the requirement at all schools, the CATEGORY RuleTag can be used to identify those completed requirements at the local school and apply them to requirements at the transfer school.

The CATEGORY values can be defined by the school system, but it is important that they are consistent throughout the system. Valid CATEGORY values should be agreed upon by all schools in the system and then the applicable blocks should be updated with these RuleTag statements.

Local block scribing

Updated: March 25, 2022

The local block should simply use the CATEGORY RuleTag to assign the category to the requirement.

For example:

```
1 Class in ENGL 100
  RuleTag Category = ENGLISH
  Label A "Basic English";
```

Or

```
2 Classes in ENGL 100, ENGL 101
  RuleTag Category = ENGLISH
  Label A "Basic English";
```

It is important to note that only one class is required to complete a category, so while a requirement may be incomplete because additional classes are still needed, the category would be complete if just one class applied to the rule. In the example above, if a student completes ENGL 100, the local audit will show that ENGL 101 is still needed to complete Basic English.

However, ENGL 100 will be tagged as satisfying the Basic English category, and that category will show as completed on the transfer audit.

IF-Statements

A school may use IF-Statements to require Math Majors to take two Mathematics classes. For example, all majors must take MATH 101 but Math majors must also take MATH 102. The school considers MATH 101 to be the system general requirement class and MATH 102 to be the local graduation requirement class for Math majors. For example:

```
If (Major = Math) Then
  2 Classes in MATH 101, 102
  RuleTag Category = MATHEMATICS
  Label "Math requirement for Math majors"
```

Else

```
  1 Class in MATH 101
  RuleTag Category = MATHEMATICS
  Label "Math requirement for non-math majors"
```

With the CATEGORY RuleTag for MATHEMATICS added to the existing scribed requirements, a student can not only take MATH 102, but also be given credit for satisfying the MATHEMATICS category in a transfer audit.

If the school does not want MATH 102 to satisfy system general requirements in a transfer audit, the local graduation and system general requirements must be separated:

```
  1 Class in MATH 101
  RuleTag Category = MATHEMATICS
  Label M1 "Math requirement";
If (Major = MATH) then
  1 Class in MATH 102
  Label M2 "Additional Math requirement for Math majors";
```

Using Attributes for Category Completion

Rather than defining a specific set of classes, attributes can be used to satisfy a category:

```
If (ATTRIB=MAT4) Then
  RuleComplete
  RuleTag Category = MATHEMATICS
  Label TransferClass "Mathematics category waived"
```

Else

```
  1 Class in MATH 1a
  RuleTag Category = MATHEMATICS
  Label MathGE "Math class met requirements"
```

Using Placeholders to Represent a Transfer Class

Schools may use a specific test score, non-course, etc. to indicate that a transfer class was used to meet a system general requirement category. For example, TEST SCORE = 45 may actually mean a transfer class.

```

If (M2H4 = 45) Then
#This is the test score we give to students who completed the category with a transfer class
  RuleComplete
    RuleTag Category = MATHEMATICS
    Label TransferClass "Transfer Class met requirements"

Else

  1 Class in MATH 1a
    RuleTag Category = MATHEMATICS
    Label MathGE "Math class met requirements"

```

This will cause no issues in a regular degree audit. However, a transfer audit will attempt to retrieve classes from each partner school that the student has attended. If a student with a placeholder runs a transfer audit, there is a strong possibility that both the placeholder and the original transfer class will be sent to the transfer audit, thereby double-counting them. The only way to avoid the double-counting would be to map the transfer class to a local class instead of the test score and change the scribe code referring to that placeholder test score to refer to the mapped class.

Waiving a Category Within a Major

If a category's requirements are waived within the major, the implication is that when all the major requirements are complete, the category is complete, but with no specific course completing the requirement. For example, it might be assumed that an English major will complete the Basic English category by completing the major requirements.

```

If (Major = ENGL) then
  RuleComplete
    RuleTag Category= ENGLISH
    Label "Basic English category waived for English majors";
Else #Major = ENGL
  1 class in ENGL 101
    RuleTag Category= ENGLISH
    Label "Basic English category for non-English majors"

```

Group Rules

Group rules are used to manage requirements that are more than just a course list. Group rules may not transcribe easily to the transfer block and while Group rules can work with IF statements, any Group rule should be examined to make sure that it doesn't require modifications in the OTHER=GETA block. In particular, Group rules that are used to manage system general requirements and local graduation requirements together in the local block must be examined when system general requirements and local graduation requirements are split in the transfer block. For example, if the system general requirements and local graduation requirements for Foreign Language are combined in the local block, it might be scribed as follows:

```

1 Group in
  (3 Classes in SPAN 101 + 102 + 103
    RuleTag Category = FOREIGNLANG
    Label "Three Spanish classes") OR
  (3 Classes in CHIN 101 + 102 + 103
    RuleTag Category = FOREIGNLANG

```

```

    Label "Three Chinese classes")
Label "LANGUAGE REQUIREMENT";

```

In the transfer block, this Group rule would have to be separated into system general requirements and local graduation requirements sections:

```

If (Category=FOREIGNLANG) then
  1 group in
    (RuleComplete
      Label "Foreign Language Category - Non-Class") or
  # Tests, waivers, etc. (non-classes) will complete this rule
    (1 Class in @ (With Category = Foreign Language)
  # Classes will complete this part of the rule
    HighPriority
      Label "Foreign Language Category - Class Used")
    RuleTag Category=FOREIGNLANG
    Label GROUPLANG "Foreign Language Category"
Else
  1 Group in
    (1 Class in SPAN 101,102,103
      RuleTag Category = FOREIGNLANG
      Label "General System Spanish class") OR
    (1 Class in CHIN 101,102,103
      RuleTag Category = FOREIGNLANG
      Label "General System Chinese class")
    Label "GENERAL SYSTEM LANGUAGE REQUIREMENT";

```

Note that this part of the Group rule would be at the bottom of the transfer block after all of the system general requirements categories.

```

1 Group in
  (2 Classes in SPAN 101,102, 103
    Label "Two Spanish classes") OR
  (2 Classes in CHIN 101, 102, 103
    RuleTag Category = FOREIGNLAN
    Label "Two Chinese classes")
  Label "LOCAL GRADUATION LANGUAGE REQUIREMENT";

```

Exceptions

If a Force Complete exception is applied to a local requirement with a CATEGORY RuleTag, the CATEGORY will be passed to the transfer audit as complete and can be applied to the transfer requirements if a RuleComplete is in the rule. The transfer audit will show at which school the CATEGORY was completed and include the exception description.

```

If (Category=WORLDCIV) then
  1 group in
    (RuleComplete
      RuleTag Category=WORLDCIV
      Label 37 "World Civilization Category - Non-Class") or
    (1 Class in @ (With Category=WORLDCIV)
      HighPriority
      RuleTag Category=WORLDCIV
      Label 38 "World Civilization Category - Class Used")

```

```

Label 39 "OTHER WORLD CIVILIZATIONS"
Else
  3 Credits in WRLD @
  RuleTag Category=WORLDCIV
  Label 40 "Other World Civilizations";

```

Non-Course

Non-Course requirements can also satisfy a category if the local requirement has a CATEGORY RuleTag. Like Exceptions, the Non-Course CATEGORY will be passed to the transfer audit as complete and can be applied to the transfer requirements if a RuleComplete is in the rule. The transfer audit will show at which school and term the CATEGORY was completed. See the transfer block scribing example above.

MAXPASSFAIL

Some schools specify a maximum number of Pass/Fail classes that can be used by classes to satisfy system general requirements and local graduation requirements. However, MaxPassFail relies on the Pass/Fail flag, which is not passed as part of the transfer audit.

To achieve the same result, use MaxCredits instead. The following example allows a maximum of 12 credits taken Pass/Fail with a passing grade:

```
MaxCredits 12 in @ (with DWGrade=P)
```

Transfer block scribing

Updated: March 25, 2022

The transfer block will contain a modified version of the local requirements, but using guidelines.

- The system general requirements categories must be spelled correctly so they can be identified in the transfer audit. Be sure to use UPPERCASE values for all your category names.
- You can only require one class per system general requirements category within the transfer block.
 - Any additional local graduation requirements must be separated from the system general requirements and placed at the end of the transfer block or in a separate block.
 - For example, if the local block requires 2 classes for the ENGLISH category, the transfer block should be modified to have 1 class for ENGLISH and 1 class for local graduation requirements in English.
- Within the transfer block, each system general requirements category must begin with Scribe code that supports identifying classes, non-classes, exceptions, etc. that satisfied the system general requirements category at the student's home school or any other partner schools the student has attended.
- Student attributes, class attributes, and test scores are not passed to the transfer audit, so any local requirements that use these must be modified appropriately in the transfer block.

- There must be a CATEGORY RuleTag on the IF and ELSE portion of each category in the transfer block.

An example of this transfer scribe code for a system general requirements category section is shown below.

Transfer Requirement Template

```
If (Category=[GENED CATEGORY CODE]) then
  1 group in
    (RuleComplete
      Label "[GENED CATEGORY CODE] Category - Non-Class") or
  # Tests, waivers, etc. (non-classes) will complete this rule
    (1 Class in @ (With Category = [GENED CATEGORY CODE])
  # Classes will complete this part of the rule
      HighPriority
      Label "[GENED CATEGORY CODE] Category - Class Used")
  RuleTag Category=[GENED CATEGORY CODE]
  Label SOMEGROUP "[GENED CATEGORY CODE] Category"
Else
  [Insert your system general requirements here]
```

Transfer Requirement Example

```
If (Category=MATHEMATICS) then
  1 group in
    (RuleComplete
      Label "Mathematics Category - Non-Class") or
    (1 Class in @ (With Category = MATHEMATICS)
      HighPriority
      Label "Mathematics Category - Class Used")
  RuleTag Category=MATHEMATICS
  Label GROUPMATH "Mathematics Category"
Else
  1 Class in MATH 220
  RuleTag Category = MATHEMATICS
  Label Calc1 "Calculus I"
```

The changes needed to the system general requirements and local graduation requirements for the transfer block are not complicated overall but do require some planning when system general requirements and local graduation requirements are combined in the local block. However, if system general requirements and local graduation requirements are already separated in the local block, then the changes needed in the transfer block should be minimal.

Transfer block header

Updated: March 25, 2022

A special block header section that does calculation based on completed categories can be included.

The display of this block header and its values is defined in several Shepherd settings on the central server.

```
transferFinder.central.worksheet.category.all.complete.minimum  
transferFinder.central.worksheet.category.all.exclude.list  
transferFinder.central.worksheet.category.block.type  
transferFinder.central.worksheet.category.block.value  
transferFinder.central.worksheet.category.classes.minimum  
transferFinder.central.worksheet.category.credits.minimum  
transferFinder.central.worksheet.category.discrete.list  
transferFinder.central.worksheet.category.special.complete.minimum  
transferFinder.central.worksheet.category.special.list
```

Transfer Finder use

Updated: March 25, 2022

On the Transfer tab, a student can verify their classwork history at all schools in the system, add any missing schools or classes, find equivalent classes offered at partner schools, generate a transfer audit for programs at partner schools, and get program and advisor contact information.

Acknowledgment

Updated: March 25, 2022

When accessing the Transfer tab for the first time, a user may be prompted to acknowledge the release of their academic history to other institutions.

If the user authorizes this release, the `dap_tf_ack_date` on their `dap_student_mst` will be updated with the current date. The user will not be prompted again for this acknowledgment if this field is not blank.

Access to bypass the Acknowledgment page is granted if the user has the TFNOACK key.

This key is included in the standard TFADV and TFREG groups.

Configuration

Review the `transferFinder.acknowledgement.show` Shepherd setting to ensure it is configured correctly:

Classwork History

Updated: March 25, 2022

Classwork History allows the student to verify their classwork at their home school, retrieve coursework from partner institutions, and add missing schools and classes for inclusion on the transfer audit.

Initially, just the student's classwork at their home school will load in Classwork History. This will be the classwork that has been bridged over into the Degree Works database from the SIS. After the student selects partner schools or manually adds other schools and classes, this data will be displayed in Classwork History in subsequent sessions.

Local classwork

Updated: March 25, 2022

All of a student's completed and in-progress classwork at their local school will be displayed.

Pre-registered and most transfer classes will not display. Classes are sorted alphabetically by descending term and course code.

The Grade and Term values that display are the globally translated values for those fields. This means that what the student sees as a Grade or Term in Classwork History may be different from what they see in their academic audit or transcript because the local school values have been equated to the common values used for the school system.

If there is classwork that is missing, the student can manually add it by clicking Add New Class so it can be included in the transfer audit.

Local classwork that is stored as a transfer class – that is, AP and CLEP test scores – can be configured to display. To allow for this classwork to be included, the ETS codes for these transfer records need to be defined. See the configuration information below.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

```
transferFinder.snapshot.audit.process.includeTransferWork.codeList
transferFinder.snapshot.audit.process.includeTransferWork.defaultValue
transferFinder.snapshot.audit.process.includeTransferWork.enabled
```

Partner school courses

Updated: March 25, 2022

If a student attended other schools within the system, they can retrieve their classwork at those schools for inclusion on the transfer audit.

To do so, they need to select those partner schools from Partner School Courses in the side panel and click the add button for the other schools in the system they have attended or are attending.

A header for each of the selected partner schools will display in the main Classwork History grid. Expand the school card to see all of the classwork retrieved from that school. If the web service was unable to connect to the remote school or if classes for that student could not be found at the partner school, an error message will display.

Partner schools can be removed by clicking the delete icon. This school will then no longer display in Classwork History, and classes from this school will not be retrieved for the transfer audit.

Partner schools are defined in the global School Directory and are configured as Partner = true.

Student system ID

Updated: March 25, 2022

Retrieving classes from a partner school requires that there is a common system ID for the student.

This ID should be stored in the shp_finder_id field on the student's shp_user_mst and can be loaded from the SIS through the standard Degree Works student data extracts. If the system ID is the same as the local school ID, the local school ID should be loaded in this field. If the student's shp_finder_id is blank, a warning message will display in the Partner School Courses side panel.

Non-Partner schools

Updated: March 25, 2022

If a student attended other schools outside the system, they can manually add these schools and their classwork for inclusion on the transfer audit.

To do so, they need to select partner schools from Other School Courses in the side panel and click the add button for the other schools they have attended or are attending.

A header for each of the added non-partner schools will display in the main Classwork History grid. After adding the non-partner school, click on the add button to manually add the classes taken at the school.

Non-partner schools can be removed by clicking the delete icon. This will delete the school and all of the associated manually added classes from the database. This school will then no longer display in Classwork History, and classes from this school will not be included on the transfer audit.

Non-partner schools are defined in the global School Directory and are configured as Partner = false.

Manually added classes

Updated: March 25, 2022

Classes can be manually added to the local school, partner schools and non-partner schools by clicking on the add button in the corresponding school header.

A Course subject and Course number are required, can be up to 12 characters each. The Course Code and Number are not validated.

A Class title up to 50 characters may be entered.

The number of Credits received should be entered.

The Type of credits used at the school should be selected. This is also known as the school calendar. The values in the drop-down list come from STU346 Description.

A Grade received for the class must be entered. The values in the drop-down list come from STU385 Transfer Finder Grade and are the globally translated values for grades. This means that what the student sees as a Grade in this drop-down list may be different from what they see in their academic audit or transcript because the local school values have been equated to the common values used for the school system.

The Term the class was taken should be selected. The values in the drop-down list come from TRF100 Description and are the globally translated values for terms. This means that what the student sees as a Term in this drop-down list may be different from what they see in their academic audit or transcript because the local school values have been equated to the common values used for the school system.

After entering the class data, click **Save** to add the record or click **Cancel** to exit without saving. When the class is saved it will appear in the class grid.

To edit a manually added class, click the edit icon on the right of the course card in the school grid.

To delete a single manually added class, click the delete icon on the right of the course card in the school grid. To delete all manually added classes for a school, click the delete icon on the right of the school header card.

Transfer Snapshot

Updated: March 25, 2022

In Transfer Snapshot, the student can search for transfer programs at partner schools and generate transfer audits to compare degree progress across programs.

Program search

Updated: March 25, 2022

The student can use criteria such as areas of interest, program categories and school to filter the list of programs they are interested in comparing.

When the student clicks on the filter link, they will get a list of the options in that category. For example, clicking **School** will generate a list of all the schools in the system. The student should check all of the schools for which they would like to see program data. The selected criteria will display at the top of the side panel while a list of matching programs will load in the lower grid on the screen. The student can filter the results even more by adding additional criteria such as Transfer path or Academic discipline. The filter categories are limited by any previous filter selections. That is, if you have selected several Transfer paths, the options you will see in Academic discipline or School will only be those that are associated with the Transfer paths you selected. Schools will always display, but transfer path and academic discipline are optional.

The values that appear in the lists for each filter category are defined in the global Transfer Directory. Transfer path values are defined in Area Of Interest. Academic discipline values are

defined in Category. School uses the Program School ID and gets the literal from the global School Directory.

To remove a filter item, click the **X** on the corresponding criteria chip. To remove all filter criteria, click **Clear all**.

The results will initially load sorted by School, but the grid can be sorted by any of the columns by clicking on the column header.

From the list of results, the student can select up to three programs to compare by running a transfer audit. After selecting the desired programs, the student should click Compare to generate a transfer audit for each selection.

If configured, check boxes to select whether in-progress and preregistered classes from the local school should be included on the transfer audit. Note that if the in-progress and preregistered check boxes are configured to not display, the auditor will follow the default value settings to determine whether in-progress and preregistered classes should be included.

If configured, a check box to select whether local classwork stored as a transfer class – that is, AP and CLEP test scores – should be included on the transfer audit.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

```
transferFinder.snapshot.audit.process.includeTransferWork.codeList
transferFinder.snapshot.audit.process.includeTransferWork.defaultValue
transferFinder.snapshot.audit.process.includeTransferWork.enabled
transferFinder.snapshot.audit.process.inprogress.defaultValue
transferFinder.snapshot.audit.process.inprogress.enabled
transferFinder.snapshot.audit.process.preregistered.defaultValue
transferFinder.snapshot.audit.process.preregistered.enabled
transferFinder.snapshot.enableDisciplines
transferFinder.snapshot.enablePaths
```

Transfer Audit Summary

Updated: March 25, 2022

For each of the programs selected, a transfer audit at the partner school will be run using the student's local classwork, classwork for any partner school selected in Classwork History, and any non-partner school or other classwork added in Classwork History.

The degree progress results will be returned to the transfer audit Summary page. For each program selection there will be a header showing the school, degree, and major, and if configured to display, the Requirements and Credits progress bars. Displaying the requirements and credits progress is configurable.

To return to the Program Search page, click the **Back to program search** link above the results.

To see the complete transfer audit, click **View transfer detail**.

Configuration

Review the Show Progress Bar Requirements and Show Progress Bar Credits flags in RPT036 for TRF31 to ensure they are configured correctly for each report.

Transfer Audit Detail

Updated: March 25, 2022

The transfer audit detail page shows a complete Degree Works audit which has been run against the partner school's requirements for the selected program using the student's local classwork, classwork for any partner school selected in Classwork History, and any non-partner school or other classwork added in Classwork History.

Course Equivalencies

Updated: March 25, 2022

The first section of the transfer audit shows the Course Equivalencies – how the student's classwork articulated to requirements at the partner school.

The articulation engine uses the `dap_mapping_dtl` records at the partner school and tries to find defined articulation agreements for each of the student's schools that best fit the student's course history.

If more than one matching mapping is found (an undecided scenario) or if the student has more than one course that maps to the same course(s) at the partner school (a duplicate scenario), the articulation needs guidance on how to best to resolve these situations. A hierarchy of resolution rules can be defined for both undecided and duplicate scenarios, and if a match still cannot be determined after working through the hierarchy, it will check the defined "rule of last resort".

Transfer classes for which no mapping can be found can be configured to map to a generic course. If this setting is used, then the course will be counted in the overall degree credits and can apply to rules on the transfer audit.

The type of Academic Calendar and the default Grade Type and School, with the correct values for the articulation engine to look up grades in STU385, must be configured for each school. Additionally, a standard Grade Type and default Catalog Year to be used when generating the transfer audit must be specified.

The Course Equivalencies section is organized by each sending school, and the transfer course number, title, grade, credits, and term taken display on the left while the equivalent course number, title, grade and credits at the partner school display on the right.

Many-to-One (MTO1), One-to-Many (1TOM), and Many-to-Many (MTOM) equivalencies will be grouped together by mapping, to visually demonstrate the relationship between the classes.

If different types of school codes are used throughout the system (such as CEEB, FICE, and so on), it will be necessary to translate the different types of school codes using the Alternate ID Type in the global School Directory.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

```
articulation.leftoverCourse.courseDiscipline
articulation.leftoverCourse.courseNumber
core.articulation.default.catalogYear
core.articulation.default.gradeType
core.articulation.default.school
core.articulation.gradeType
core.institution.calendarType
```

Resolution options

Updated: March 25, 2022

These resolution options apply to the Duplicate and Undecided settings.

You can specify multiple options using a comma-delimited list of these options. These options should be applied in list order until a final mapping is found or until all resolution options have been tried.

The sub sections that follow provide an explanation of each resolution option followed by a list of configuration settings and their supported resolution options.

CREDITMAX

For all the potential mappings, add up the total amount of credits that would be awarded in each. The mapping that would result in the most transfer credits wins.

For example, you can have the following Undecided situations:

- a. MATH 101, MATH 102 → MATH 200, MATH 300, MATH 400
- b. MATH 101, MATH 103 → MATH 51, MATH 22

Or

- c. ENG 35, ENG 36, ENG 37 → HUMN 104
- d. ENG 35, ENG 38 → ENG 200

In Case A, MATH 101 is part of a pair of transfer classes that maps to three local classes. In Case B, the transfer class pair maps to a pair of local classes. But all that matters is the total number of credits awarded by the local classes. If the total transfer credits in Case B is 10 and Case A is 8, then CREDITMAX says the articulation engine would choose Case B.

Similarly, in Case C and D, even though ENG 35 is part of a transfer class pair in one mapping and a trio in another, all we care about is if HUMN 104 awards more or less credits than ENG 200.

LOCALHIGH

Use the mapping with the highest numbered local course (course at the Target School). This can be used any time transfer classes can be mapped to more than local course.

- a. MATH 101, MATH 102 → MATH 200, MATH 300, MATH 400
- b. MATH 101, MATH 103 → MATH 51, MATH 22

Or

- c. ENG 101 → ENG 200
- d. ENG 101 → ENG 500

Or

- e. ENG 101 → ENG 200
- f. ENG 102 → ENG 200

In Case A and Case B, MATH 400 is the highest numbered local course in any of the resulting mapping options, so that is the option the articulation engine would choose.

Similarly, in Case C and Case D, ENG 500 is the highest numbered local course, so the articulation engine will choose it if using LOCALHIGH.

In Case E and F, both local courses are the same. This is a Duplicate situation. In this case, LOCALHIGH would have no effect in resolving the mapping and the auditor engine would move to the next resolution option, so it really should not be used at all. However, if it is used, it will not cause any errors.

Course numbers will be compared numerically to determine which is highest (any alpha characters will be stripped out and the resulting integer will be used). That means that MATH 30L, CALC 30, and STAT 3H0 are the same number.

TRANSFERHIGH

Use the mapping with the highest numbered source class (the classes taken by the student).

- a. MATH 101, MATH 102 → MATH 200, MATH 300, MATH 400
- b. MATH 101, MATH 103 → MATH 51, MATH 22

Or

- c. ENG 101 → ENG 500
- d. ENG 102 → ENG 200

Or

- e. ENG 101 → ENG 200
- f. ENG 101 → ENG 500

In Case A and Case B, MATH 103 is the highest numbered transfer class in any of the resulting mapping options, so that is the option the articulation engine would choose.

In Case C and Case D, ENG 102 is the highest numbered transfer class, so it is the option the articulation engine would choose.

In Case E and Case F, the same transfer class is used in both mapping pairs, so TRANSFERHIGH will not lead to a resolution and the articulation engine will need to move to the next option.

As with LOCALHIGH, class numbers will be compared numerically.

RECENT

Use the class that was taken most recently. This only applies to transfer classes. The term would determine which class was most recent (for example, Spring 2013 is more recent than Fall 2012).

RANDOM AND NONE

RANDOM allows the auditor to pick a mapping arbitrarily. This resolution option could lead to different results each time. NONE tells the auditor do not resolve this mapping.

- a. MATH 101, MATH 102 → MATH 200, MATH 300, MATH 400
- b. MATH 101, MATH 103 → MATH 51, MATH 22

In Case A and Case B, RANDOM means the auditor may choose A one time and B the next.

It is recommended that RANDOM is used as the Rule of Last Resort rather than NONE.

REPEAT POLICY AND RESOLUTION OPTIONS

The RAD_COURSE_MST.RAD_REPEAT_MAX for the target school class can affect the resolution option. That field can be set to a value between 1 and 99 or Y, which indicates that a class can be repeated an infinite amount of times.

For duplicate situations, this means that multiple source school classes can articulate to the same target class. For example:

MATH 101 → CALC 100
MATH 102 → CALC 100

If CALC 100 has RAD_REPEAT_MAX set to 1-99 or Y, then both classes will articulate to CALC 100 in a transfer audit.

Configuration

Review the following Shepherd settings to ensure they are configured correctly:

- articulation.resolutionRules.duplicate
- articulation.resolutionRules.undecided
- articulation.ruleOfLastResort

Transfer audit

Updated: March 25, 2022

The body of the transfer audit is built using the student's goal data and the corresponding scribe blocks at the partner school.

There are several special Scribe options that can be used for the transfer audit.

Rules and qualifiers within those blocks can be changed for the transfer audit only by using the AuditAction statement with a TRANSFER value. For example:

```
If (AuditAction=TRANSFER) then
  1 Block (OTHER=SYSGENED)
    Label SYSGENED "Ellucian System General Education Requirement"
```

The CATEGORY RuleTag can be used with transfer audits to identify groups of requirements that might be considered complete at all schools in the system if they have been completed at one of the schools. For example:

```
1 Class in MATH a (With Category=MATHEMATICS)
  RuleTag Category=MATHEMATICS
  Label MATH "General Mathematics Requirements")
```

Note that Transfer Finder will always use the most recent local audit as the source of CATEGORY requirements, and if a student has a double-major, this may result in the student seeing different categories completed in the transfer audit based on which major's audit is chosen.

A special block header section that does calculation based on completed categories can be included. The contents of this block header are defined in several Shepherd settings.

Configuration

Review the following Shepherd settings on the central server to ensure they are configured correctly:

```
transferFinder.central.worksheet.category.all.complete.minimum
transferFinder.central.worksheet.category.all.exclude.list
transferFinder.central.worksheet.category.block.type
transferFinder.central.worksheet.category.block.value
transferFinder.central.worksheet.category.classes.minimum
transferFinder.central.worksheet.category.credits.minimum
transferFinder.central.worksheet.category.discrete.list
transferFinder.central.worksheet.category.special.complete.minimum
transferFinder.central.worksheet.category.special.list
```

Apply now

Updated: March 25, 2022

A link to information on applying to the transfer school can be made available from a link on the transfer audit. This is defined in the Global Directory.

Find a transfer advisor

Updated: March 25, 2022

A link to information on contacting a transfer advisor at the transfer school can be made available from a link on the transfer audit. This is defined in the Global Directory.

Find Class Equivalent at Partner Schools

Updated: March 25, 2022

The Find Class Equivalent at Partner Schools option can help a student find an equivalent course at a partner school and also to retrieve schedule information for sections of that class.

For example, if the student is planning on transferring to a partner school and would like to complete the MATH 101 requirement they will need to take for their transfer program this summer, they can enter MATH 101 in the search field, select the transfer school from the list and search for equivalents of that course that are offered at other schools.

The Course must contain a course code and course number, separated by a space. The code and number can be up to 12 characters each. The course code and number are not validated.

If enabled, the student can select a school different than their local school. The values in the School drop-down list come from the schools defined in the global School Directory with Partner = true.

If enabled, the student will be limited to only seeing equivalent results for partner schools.

After clicking Submit, any matching dap_mapping_dtl records for the specified local course at the selected school will display in the Course Equivalents grid. If equivalent course cannot be found, a message will display in the Course Equivalents grid.

The Course Equivalents grid can be sorted by transfer Course or School by clicking the column header. The default sort is ascending by School, with a secondary sort on the Course.

If the mapping for the equivalent course is a many-to-one (MTO1) or many-to-many (MTOM), all of the transfer courses needed to earn credit for the specified course will appear in the Course column.

When special conditions for an equivalent course mapping exist, an icon to expand the mapping card will display in the Conditions column. Expanding the card will display conditions such as minimum/maximum credits, minimum/maximum grade, and maximum years ago taken, in addition to required major, degree, level, etc. that may be associated with the mapping. These conditions are stored as the dap_tr_cr_min, dap_tr_cr_max, dap_tr_gr_min, dap_tr_gr_max or dap_tr_yrs_ago on the dap_mapping_dtl, or as dap_map_cond_dtl records. Additionally, if the mapping for the equivalent course is a one-to-many (1TOM) or many-to-many (MTOM), the additional local classes that may be satisfied by the transfer course(s) will be displayed.

When a course is displayed as a link, schedule information for this class can be obtained from the transfer school. After clicking on the link, a web service will connect to the partner school's SIS and retrieve the schedule information. The web service adapter is defined in the global School Directory as ClassServiceAdapterBeanName. If this value is blank or doesn't exist, courses at this partner school will display as plain text.

The course schedule section will display the Term, Class ID, Title, Meeting Days and Times, and Seat Availability for the selected course sections.

Configuration

Review the following Shepherd setting to ensure it is configured correctly:

```
transferFinder.courseSearch.enableSchoolSelection  
transferFinder.courseSearch.limitResultsToPartners
```

Partnership Transfer Program Information

Updated: March 25, 2022

Additional information about transfer programs can be provided in Partnership Transfer Program Information.

A school system can create a web application that return a web page describing the transfer program at the institution identified in parameters sent as part of the web request.

The parameters sent will be the request type, Area of Interest, and global school code from the global Transfer Directory.

Transit Administration

Updated: September 29, 2023

Transit is a tool Degree Works administrators use to launch and manage batch processes.

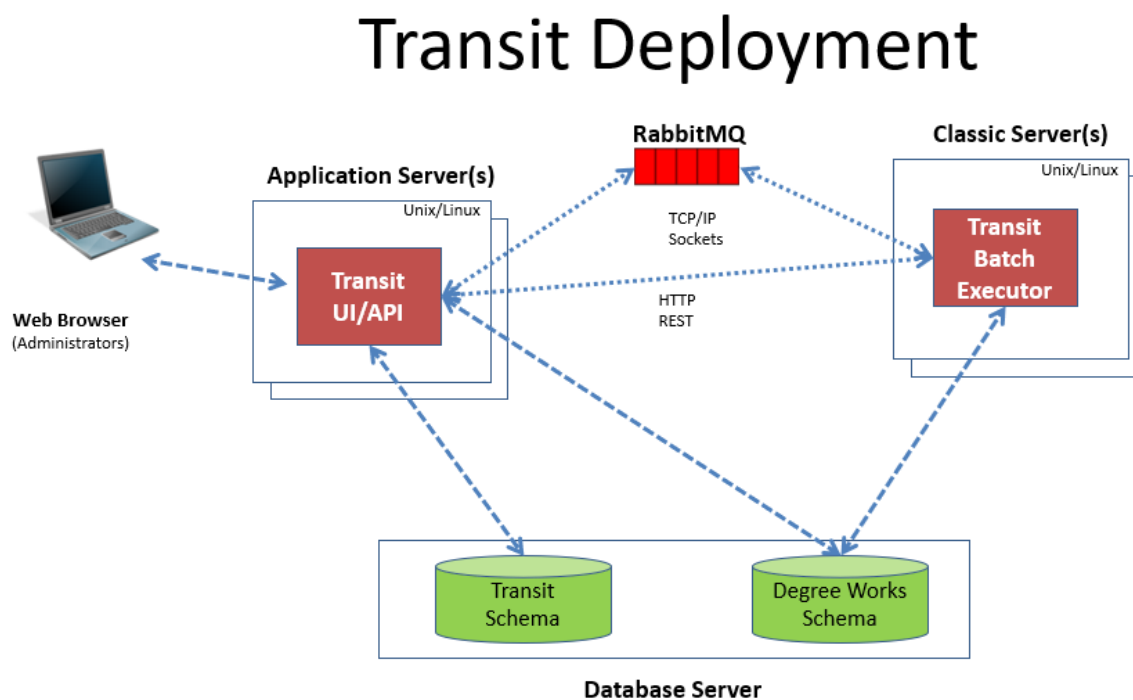
Transit consists of two primary elements: The Transit UI (User Interface) and a Transit Batch Executor. Both must be deployed and running to successfully launch Degree Works jobs.

The Transit UI contains both the frontend (browser-based) application and the backend API that is used by the client code. The API is also used by the batch executor. It is deployed as a jar on any supported Unix server and is independent of the classic software. It includes an embedded container. A separate Java container like Tomcat is not required.

The Transit Batch Executor is also deployed as a standalone jar file, but it must be deployed into the same environment as the classic software. It is automatically included when the installation script (dwdeploy) is run during the software installation and update process.

Transit requires its own schema separate from the primary Degree Works schema. The Degree Works install and update process creates and populates the schema into the same database as the existing primary schema. Only the Transit API, deployed in the Degree Works UI jar, accesses the database.

The following diagram shows the deployment and relation of the various Transit pieces.



Initial Transit setup and configuration

Authentication

Updated: March 25, 2022

The Transit user interface supports SHP, CAS, and SAML authentication methods, and external access managers such as Oracle Access Manager. For more information, please see the [Authentication security](#) topic.

Transit user access

Updated: September 29, 2023

User access to Transit and its features is managed with a set of Shepherd keys and groups.

The Shepherd keys required for normal Transit users are TRANSIT and TRANRUN. To run a specific job, a user needs either the Shepherd key for that job (for example, TRDAP22 to run DAP22 for generating batch audits) or TRANALL, which grants access to all reports and processors.

Users with the TRANSQL Shepherd key have the option to provide an SQL statement as selection criteria in jobs that support this feature. To delete completed jobs and their associated artifacts, users need the TRANDEL Shepherd key.

TRANPARM allows a user to save the selection criteria and responses to questions for frequently run jobs and to use these saved job parameters when running the job again in the future. TRANSCHD allows users to schedule jobs to run one time at a future date. TRANRCUR grants users access to saved job parameters, scheduling jobs to run one time in the future and creating recurring schedules for jobs.

The TRANREG and TRANBAN Shepherd groups can be used to assign job access to users. The Users function of Controller or the core.security.rules.shpcfg Shepherd setting can be used to assign keys or groups to users. The list of all Transit keys and groups is available in the [Keys and keyrings](#) topic.

The Transit Batch Executor requires credentials to access the API. A default user, TRANSITAPI, was created during the update or installation process, but you need to set a password for this user through the Users function in Controller or with the changepassword script. After setting the password for the TRANSITAPI user, use the Settings function of Controller to update the core.security.transit.username and core.security.transit.password settings. The TRANSITAPI user requires and has been delivered with the TRANSIT, TRANART, DEBUG, and NOREFER keys.

The launchjob script uses the TRANSITBAT user to access the API when launching jobs. This user was created during the update or installation process, but you need to set a password for this user through the Users function in Controller or with the changepassword script. After setting the

password for the TRANSITBAT user, use the Settings function of Controller to update the `core.security.transit.batch.username` and `core.security.transit.batch.password` settings. The TRANSITBAT user is delivered with the minimum keys required for Transit. You must assign appropriate keys to the TRANSITBAT user for the jobs you want to run. For example, if you want to run the Banner student extract using a list of IDs or the default SQL file `bannerstudents.sql`, you must add the TRRAD30 key to the TRANSITBAT user.

MEP clients should set the TRANSITAPI and TRANSITBAT passwords for each MEP entity.

Shepherd settings

Updated: September 29, 2023

Most configurations to manage the behavior of Transit can be maintained using the Settings function of Controller.

To make any setting specific to just Transit, use the spec value **transit**.

The following settings should be configured before you deploy Transit UI and the Transit executor:

```
transit.api.url

core.security.cas.callbackUrl spec=transit (if you use CAS authentication)

core.amqp.exchange.transit
core.security.transit.batch.password
core.security.transit.batch.username
core.security.transit.password
core.security.transit.username
core.week.startingDay
transit.cleanup.completed.maxDays
transit.cleanup.incomplete.maxDays
transit.dap16.workerCount
transit.dap22.workerCount
transit.dap27.workerCount
transit.dap28.workerCount
transit.rad11.workerCount
transit.rad30.workerCount
```

Transit user interface deployment

Updated: March 25, 2022

TransitUI is designed to deploy equally well in on-premise and cloud hosted scenarios on all supported platforms. It provides a mobile-ready, responsive user interface communicating with the servers using lightweight, stateless, restful API over HTTP SSL.

The deployment artifacts are two JAR files, which are executable using java and include an embedded container: TransitUI and Transit Batch Executor.

Prerequisites

Updated: March 25, 2022

Java is a required prerequisite for Transit. SSL is recommended.

Java

Java version 11 or above is the only third-party dependency required to run this application. Oracle Java and OpenJDK are supported.

SSL

SSL is in general a best practice and is recommended for Transit because the security implementation uses stateless tokens. Using signed certificates is recommended for all deployment scenarios, even test or development environments. Self-signed certificates, even for internal test deployments, can be problematic because of browser requirements and warnings. The SSL certificate needs to be configured in a keystore and that keystore will need to be accessible in the deployment environment. There are various options for how to do this, for example using Java's keytool command.

SSL offloading

When SSL offloading is in use (SSL is terminated before reaching the application) with a load balancer, proxy server, and so on, some additional configuration needs to be done to make sure Degree Works applications behave as expected. Please refer to the [Java application load balancing](#) topic for more information about this setup.

Transit environment settings

Updated: September 29, 2023

Several environment variables are required to be configured before running the application.

Depending on the desired deployment platform there are many options for how these environment variables can be configured. The only requirement is that they are available in the environment where the product's JAR file will be executed. For example, it may be desirable to configure them in a shell script, in a specific user's profile, a startup instruction like init.d, or in a continuous deployment tool like Jenkins.

The minimum required variables are documented here. But many optional variables are provided in the Spring Boot platform, for example tuning the embedded container. Please refer to Spring documentation for those: <http://docs.spring.io/spring-boot/docs/current/reference/html/common-application-properties.html>.

Environment Variables are the recommended method to configure these properties. But there are various other ways these can be configured. Please refer to this documentation for more information: <http://docs.spring.io/spring-boot/docs/current/reference/html/boot-features-external-config.html>.

Required environment variables for TransitUI

- **DW_DATASOURCE_URL:** A JDBC connection string for the Degree Works database. For example: `jdbc:oracle:thin:@{SERVER}:{PORT}/{SERVICE_NAME}` where {SERVER} is the address, {PORT} is the port of the listener, and {SERVICE_NAME} is the service name of the database. This is the same database and schema used by the classic Degree Works software.
- **DW_DATASOURCE_USERNAME:** The Degree Works database username.
- **DW_DATASOURCE_PASSWORD:** The Degree Works database password. This can be encoded. To get the encoded value, use the `showdbpasswords --password` command in the classic environment. It will display the encoded value to place here (e.g. `ENC(skdfjldjs)`).
- **TRANSIT_DATASOURCE_URL:** A JDBC connection string for the Transit schema in the Degree Works database. For example: `jdbc:oracle:thin:@{SERVER}:{PORT}/{SERVICE_NAME}` where {SERVER} is the address, {PORT} is the port of the listener, and {SERVICE_NAME} is the service name of the database.
- **TRANSIT_DATASOURCE_USERNAME:** The Transit schema username.
- **TRANSIT_DATASOURCE_PASSWORD:** The Transit schema password. This can be encoded. To get the encoded value, use the `showdbpasswords --transitpassword` command in the classic environment. It will display the encoded value to place here (e.g. `ENC(skdfjldjs)`).
- **SERVER_PORT:** The desired port for Transit. This must be different from the ports used by other applications on this server.
- **SERVER_SERVLET_CONTEXT_PATH:** An optional context path. If this is blank, the application will be deployed at the root URL like `https://myserver.edu:NNNN` (where NNNN is the SERVER_PORT). If a value is specified like `/my-context-path`, the application would be accessible at a URL like `https://myserver.edu:NNNN/my-context-path` (where NNNN is the SERVER_PORT).
- **SERVER_SSL_KEY_STORE:** The path to a keystore file you created to contain your SSL certificate.
- **SERVER_SSL_KEY_STORE_PASSWORD:** The password for your keystore file.
- **SERVER_SSL_KEYSTORETYPE:** The type of your keystore. JKS is the default.
- **SERVER_SSL_KEY_ALIAS:** The alias of the key you created.
- **MAX_ARTIFACT_SIZE:** The size, in bytes, of the largest job artifact that may be uploaded. This value may be set as high as 2 GB (2147483648 bytes). A size too small may prevent large reports from uploading to the database after a job has completed. A large value too large may cause an out-of-memory error when uploading large reports.
- **SPRING_SERVLET_MULTIPART_MAXFILESIZE, SPRING_SERVLET_MULTIPART_MAXREQUESTSIZE, and SERVER_TOMCAT_MAXHTTPFORMPOSTSIZE:** These variables are based on the MAX_ARTIFACT_SIZE value. See the example configuration below.

- `SPRING_SERVLET_MULTIPART_FILESIKETHRESHOLD`: The threshold beyond which the server will write file data out to disk as opposed to putting it all in memory. A somewhat slower operation, but saves on memory.

Optional environment variables

- `JAVA_OPTIONS`: The memory allocation for the application can be defined using this variable, and is recommended. The value defines the initial heap size and max heap size and is in the following format: `-Xms1536m -Xmx1536m`. The heap size values should be adjusted as appropriate for your server.
- `SERVER_MAX_HTTP_HEADER_SIZE`: This setting allows you to set the maximum size of the HTTP message header. By default, this is 8KB, and for some users with a large number of Shepherd access keys, this can be too small. It's recommended to use this variable and set it to at least 12KB. For example: `export SERVER_MAX_HTTP_HEADER_SIZE=12288`
- `SERVER_TOMCAT_REDIRECT_CONTEXT_ROOT`: Embedded Tomcat for Spring Boot has a default behavior to redirect a request without a trailing slash to one with a trailing slash. However, that redirect happens without evaluating forward headers like X-Forwarded-Proto. So, when SSL offloading is in place, the redirect happens to http instead of https. This can be an issue especially for load balanced environments. To disable the default behavior, set this variable to false.

Warning! You should do this only if the default behavior is not working. If you are offloading SSL at a proxy or load balancer, make sure to follow the instructions in the [Java application load balancing](#) topic. Then, if you have problems accessing the application in the browser when you do not include a trailing slash at the end of the request, try setting this variable to **false**.

- `DEBUG`: A boolean true or false which will enable or disable debug logging. The log file is by default named Transit.log. But, the location and name of that file can be customized by setting the `LOGGING_PATH` and `LOGGING_FILE` environment variables.
- `DEPLOY_LOCATION`: the location of the TransitUI.jar file. This location should be set then referenced in the java -jar command that starts the application.
- `LOGGING_FILE`: the name of the file including the path location. If this is specified, `LOG_PATH` variable should not be used.
- `LOG_PATH`: the location of the log file. The default PATH location is the Java temporary java folder from the Java property `java.io.tmpdir`. This is usually /tmp. The file name will consist of this value plus transitAPI.log.
- `LOG_DEFAULT_THRESHOLD`: If set to `DEBUG`, the application will output more verbose logging. Debug output will be sent to the log during the startup process and for all requests thereafter. Warning: this will result in very large log files. If not set, the debug threshold is `INFO`.
- `DW_ENABLE_MONITOR` - This environment variable can be set to enable /health, /metrics, /prometheus, and /status endpoints for health and monitoring metrics. By default, this monitoring functionality is not enabled. Adding this variable to an application's startup script and setting it to true would enable these endpoints for the application. See individual API documentation in the API Catalog for additional information.

- **DW_RABBITMQ_SSLALGORITHM:** Specify the SSL algorithm to use when connecting to the RabbitMQ broker using SSL. If not specified, it uses the current RabbitMQ driver's default (at least TLSv1.2). This is only considered if the Shepherd setting `core.amqp.useSsl` is set to true.
- **TRANSIT_CLEANUP_INPUT_ENABLED:** This environment variable can be set to disable the cleanup of orphaned input files, Transit input files that have been uploaded but were never associated with a Transit job. This will rarely, if ever, happen. It should only be set upon instruction by Ellucian to troubleshoot an issue with the functionality. Setting this variable to 0 (zero) would disable the cleanup.
- **TRANSIT_CLEANUP_SCHEDULER:** By default, the Transit job cleanup process runs every day at noon and at midnight. This schedule can be changed by setting this variable. The value is a crontab expression following the format defined in [https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/scheduling/support/CronExpression.html#parse\(java.lang.String\)](https://docs.spring.io/spring-framework/docs/current/javadoc-api/org/springframework/scheduling/support/CronExpression.html#parse(java.lang.String)). For example, `export TRANSIT_CLEANUP_SCHEDULER=0 15 10 * * *` would set the job cleanup process to run every day at 10:15 am. A dash or negative sign (-) would totally disable the cleanup.

Database pool configuration for Microservice applications

For production deployments of java applications, it is important to be able to configure the size of the database connection pool. In Tomcat this is done through the Resource definition in `server.xml`. For microservice applications like TransitUI, there are environment variables that can be used for this configuration.

These environment variables start with `DW_DATAPOOL_` for the Degree Works database, and `TRANSIT_DATAPOOL` for the Transit database. They end with the name of the datasource pooling attribute you want to control. These are the most common ones:

Attribute	Meaning
<code>initialSize</code>	The initial number of connections that are created when the pool is started.
<code>maxActive</code>	The maximum number of active connections that can be allocated from this pool at the same time.
<code>maxIdle</code>	The maximum number of connections that can remain idle in the pool, without extra ones being released.
<code>minIdle</code>	The minimum number of active connections that can remain idle in the pool, without extra ones being created.

The name of the attribute should be given in all uppercase, with an underscore separating the words (denoted by a capital letter in the table above). To set the maximum number of active connections in the pool, for example, put the following line in your startup script: `export DW_DATAPOOL_MAX_ACTIVE=100`

You can set any of the properties of a version 1.4 Apache Commons BasicDataSource object. These are documented at <http://commons.apache.org/proper/commons-dbcp/api-1.4/org/apache/commons/dbcp/BasicDataSource.html>.

The number of active connections, and the current configured values for the limits can be monitored using any JMX (Java Management Extensions) client, such as JConsole, attached to the microservice JVM. The values will appear under the mbean `net.hedtech.degreeworks/`

JmxBasicDataSource/datasource. The getState operation will return a formatted report of the current pool state.

SSL offloading

When SSL offloading is in use (SSL is terminated before reaching the application) with a load balancer, proxy server, and so on, some additional configuration needs to be done to make sure Degree Works applications behave as expected. See the [Java application load balancing](#) topic for more information about this setup.

Deployment

When the prerequisites and configuration are in place, start up the application using a java -jar command like this one:

```
java $JAVA_OPTIONS -jar TransitUI.jar
```

Example startup script for TransitUI

If it is desirable to employ a shell script to start the application, here is an example. Note that the TransitUI.jar file must be staged at the DEPLOY_LOCATION:

```
#!/bin/ksh
export DW_DATASOURCE_URL="jdbc:oracle:thin:@MyServer:1521/service_name"
export DW_DATASOURCE_USERNAME="my-dw-user"
export DW_DATASOURCE_PASSWORD="my-dw-password"
export TRANSIT_DATASOURCE_URL="jdbc:oracle:thin:@MyServer:1521/service_name"
export TRANSIT_DATASOURCE_USERNAME="my-transit-user"
export TRANSIT_DATASOURCE_PASSWORD="my-transit-password"
export SERVER_PORT="8481"
export SERVER_SERVLET_CONTEXT_PATH=""
export SERVER_SSL_KEY_STORE="/usr/local/my-keystore.jks"
export SERVER_SSL_KEY_STORE_PASSWORD="my-key-password"
export SERVER_SSL_KEYSTORETYPE="jks"
export SERVER_SSL_KEY_ALIAS="my-keystore-alias"
#-----
# Set the MAX_ARTIFACT_SIZE to the size, in bytes, of the largest
# artifact that may be uploaded.
export MAX_ARTIFACT_SIZE=2147483648 # 2048 MB, 2 GB
# Threshold beyond which the server will write file data out to disk,
# saving memory.
export SPRING_SERVLET_MULTIPART_FILESIZETHRESHOLD=1000
# The next three settings are based on MAX_ARTIFACT_SIZE and should
# not
# be set independently
export SPRING_SERVLET_MULTIPART_MAXFILESIZE=$MAX_ARTIFACT_SIZE
export SPRING_SERVLET_MULTIPART_MAXREQUESTSIZE=$((($MAX_ARTIFACT_SIZE
+10000))
export SERVER_TOMCAT_MAXHTTPFORMPOSTSIZE=$((($MAX_ARTIFACT_SIZE+10000
))
export DEPLOY_LOCATION="/path/to/jarfile"
```

```
export JAVA_OPTIONS="-Xms1536m -Xmx1536m";
export DEBUG="false"
# To enable full-time debug log output, remove the # from the line below
#export LOG_DEFAULT_THRESHOLD=DEBUG
java -jar $JAVA_OPTIONS DEPLOY_LOCATION/TransitUi.jar > startup-log-file.log &
```

Example stop script for TransitUI

If using a start script like the example above, here is a corresponding example stop script:

```
#!/bin/ksh
killall -f TransitUI
```

Transit Batch Executor

Updated: March 25, 2022

The Transit Batch Executor is delivered to your Classic server environment as \$DWJAVA_HOME/transitexecutor.jar. The executor can be managed with the tbestart and tbestop commands. These commands use the dwinit.tbe script located in \$DGWHOME/initd.

Transit executes the appropriate job script located in the \$DGWHOME/jobs directory. Artifacts (logs files, pdf or xml audits, and reports) are initially written to the \$ADMIN_HOME/jobdata directory and are uploaded to the Transit database when the job completes, and then the files are removed. When the Transit Executor is started, all prior job artifacts are removed from the jobdata directory, as they are now accessible for viewing or download from the Transit User Interface.

When the Transit Executor is started, a log file is created in the logdebug directory with today's date appended. For example, transitexecutor.log_2019-05-31. Start and stop job events are noted there, and any errors. This log is automatically rolled over to a new file each day.

Note that TransitUI and RabbitMQ must be started and running before starting the Transit Batch Executor.

Transit Batch Executor configuration

Updated: March 25, 2022

Transit Batch Executor requires specific configuration.

The \$DGWBASE/dwenv.config file must be configured with the following variables:

```
DB_LOGIN_TRANSIT
TRANSIT_EXECUTOR_ID
TRANSIT_EXECUTOR_PORT
SERVER_SSL_KEY_STORE
SERVER_SSL_KEY_STORE_PASSWORD
```

SERVER_SSL_KEYALIAS
SERVER_SSL_KEYSTORETYPE

The DB_LOGIN_TRANSIT variable contains the Transit schema name and service. This is used by the tbedelete script and the update process.

The TRANSIT_EXECUTOR_ID contains a value that will be recorded as the MODIFY_WHAT column on the TRANSIT_JOB_INSTANCE table. In a load-balanced environment, this value should be configured with a different value for each instance, for example, EXEC-PROD001. This value has a maximum length of 14 characters.

The TRANSIT_EXECUTOR_PORT contains the TCP/IP port to be used by the Executor's REST API. It should not conflict with another service on its server.

The SERVER_SSL_KEY_STORE variable contains the path to a keystore file you created to contain your SSL certificate.

The SERVER_SSL_KEY_STORE_PASSWORD contains the password for your keystore file.

The SERVER_SSL_KEY_ALIAS variable contains the alias of the key you created.

The SERVER_SSL_KEYSTORETYPE variable contains the type of your keystore. JKS is the default.

The above SSL variables are like the variables created for the other Degree Works Java applications in their startup scripts.

The Transit Executor will require memory, up to as much as 2 GB. The memory for Transit Executor defaults to 1536MB but can be overridden with the optional environment variable TBE_MEMORY.

The batch executor uses credentials stored in the `core.security.transit.username` and `core.security.transit.password` Shepherd settings to contact the Transit UI. This user must be granted the TRANSIT key to access the API, the TRANART key to upload artifacts, and the NOREFER key to avoid the referrer check in the API. See [Transit user access](#) for more information.

The Transit Executor receives job requests from the RabbitMQ broker and contacts the Transit UI while executing jobs on your Classic server. Access through firewalls to these deployments is required. The URL for the Transit UI application must be found in the Shepherd setting `transit.api.url`.

After completing the Batch Executor configuration, start the Batch Executor:

```
$ tbestart
```

Job parameters

Updated: September 29, 2023

The selection criteria and responses to processing questions for frequently run jobs can be saved for reuse.

If users have the TRANPARM or TRANRCUR Shepherd key, they will see a saved job parameters drop-down on the Run Jobs tab. After selecting a report or processor, users have the option of not using saved job parameters, creating a new set of saved job parameters, or using an existing set of saved job parameters.

Selecting **None** will allow the user to enter a new set of selection criteria and responses to processing questions. These will not be saved, and if this specific set of job parameters is needed for a future job, they will need to be manually entered again.

Selecting **New** will prompt users for a description of these saved job parameters, and when launching or scheduling the job, this set of job parameters will be saved for reuse. Users can also just save this set of job parameters without launching or scheduling the job. If the **Enable debugging?** option is selected, this will not be included in the saved job parameter. This is a runtime-only option.

Any previously created sets of parameters for a specific job will display in the Saved job parameters drop-down. Selecting one of these saved job parameter sets will preload the selection criteria and responses to processing questions. Users can then launch or schedule this job using these parameters. If users makes modifications to these job parameters, they will have the option to update the existing job parameters or save these changes as a new set of job parameters.

All saved job parameter sets created by any user can be viewed in the Saved Jobs tab. Clicking on the **Description** will load that set of job parameters in the Run Jobs tab. Users can then edit the job parameter set, or launch, or schedule a job using these parameters.

Users with the TRANRCUR Shepherd key will see an **In Use** column on the Saved Jobs tab. This shows the number of scheduled jobs using that set of saved job parameters, and clicking on the link shows the next time a job using these job parameters will run.

Saved job parameters can be deleted from the saved jobs by clicking the **Delete** icon. If saved job parameters are being used by a scheduled or recurring job, they cannot be deleted. After the scheduled job runs or all recurring instances complete, the saved job parameters can be deleted.

Some functionality, such as launching, editing, and deleting saved job parameters, may be limited if users are not authorized to run that specific job, either because they do not have the TRANALL or job-specific Shepherd key, like TRDAP22.

Configuration

Access to saving job parameters is granted if the user has the TRANPARM or TRANRCUR Shepherd key.

Job scheduling

Updated: September 29, 2023

Jobs can be scheduled to run one time at a future date and time, or on a recurring daily, weekly, monthly, or yearly basis.

If users have the TRANSCHD or TRANRCUR Shepherd key, they will see the **Schedule** button on the Run Jobs tab. Clicking on this button opens the Schedule a job dialog.

To schedule a job to run one time in the future, users select the desired **Schedule date** and **Schedule time** in the Schedule a job dialog and click the **Schedule** button to schedule this job.

To create a recurring job schedule, users click the **Schedule recurring job** link in the Schedule a job dialog. Only users with the TRANRCUR Shepherd key will see this link. They can select the desired **Start date** and **Start time** for the recurrence, the frequency (daily, weekly, monthly, or yearly), and when the recurrence should end, and then click the **Schedule** button to create the recurrence.

The **Enable debugging?** option cannot be selected when scheduling jobs. This is a runtime-only option.

There are some jobs such as those that require an input file (for example, DAP41) that cannot be scheduled to run on a recurring basis. For these jobs, the **Schedule recurring job** link in the Schedule a job dialog will not display, even if the user has TRANRCUR.

The schedule and start dates and times are displayed in Transit in the user's time zone, while the next scheduled time for scheduled and recurring jobs is stored in UTC (GMT) in the database. The time zone of users who create or last modified the recurring schedule is also stored in the database. This is necessary to ensure that date and time calculations are done according to the user's time zone.

Scheduled and recurring jobs can be viewed in the Scheduled Jobs tab. Clicking **Type** will open the Edit a job schedule dialog. The user can then edit or delete the job schedule, or save the changes as a new job schedule.

Scheduled and recurring jobs can be deleted from the Scheduled Jobs by clicking the **Delete** icon.

Some functionality, such as editing and deleting scheduled and recurring jobs, may be limited if users are not authorized to run that specific job, either because they do not have the TRANALL or job-specific Shepherd key, like TRDAP22.

Configuration

Review the core.week.startingDay Shepherd setting to ensure it is configured correctly.

Access to scheduling jobs is granted if the user has the TRANSCHD or TRANRCUR Shepherd key.

The tbeshow script

Updated: March 25, 2022

This script shows the transitexecutor process.

The tbestart script

Updated: March 25, 2022

This script starts the Transit Batch Executor and is needed to run any Transit jobs. The Transit UI application must be running before starting the executor, as should the RabbitMQ broker.

The tbestop script

Updated: March 25, 2022

This script stops the Transit Batch Executor by sending a message to the executor using its API.

After the executor receives the request to stop, it will normally wait for all jobs to complete before shutting down. It may take the executor some time to stop after this command has finished. You can use the `tbeshow` command to see if the executor has stopped. Use the `--wait` option to force the `tbestop` command to pause until the executor has completely shut down. To force the executor to shut down immediately, use the `--immediate` flag on the `tbestop` command. This will terminate all running jobs before the executor shuts down and updates the job statuses to Terminated.

The `--wait` and `--immediate` options are independent and may both be used together. The `--wait` option applies to the `tbestop` command itself, and the `--immediate` option applies to the executor. If both are used, for example, it causes the executor to terminate all jobs and shut down while the `tbestop` command waits for this to finish.

The tberestart script

Updated: March 25, 2022

This script restarts the Transit Batch Executor by first stopping the currently running executor and then starting a new one.

Normally, it will wait for existing jobs to stop before it starts the new executor. You can use the `--immediate` option to force the script to terminate all running jobs immediately before starting the new executor. This is the same as executing a `tbestop --wait` followed by a `tbestart`.

The launchjob script

Updated: March 24, 2023

There are times when you may want to launch a job in a scripted or automated setting that makes using the user interface impossible.

This includes scheduling jobs through cron. The launchjob script is available for these purposes. It submits a job to the Transit queue using parameters provided by a parameter file. The job is not run by the script but is run by one of the executors that also run jobs submitted by the Transit UI. This may be on the same server as the launchjob script, but it may also be on another server if you have Transit executors running on multiple machines.

The format of the command is:

```
launchjob --help
launchjob [--debug | --verbose] [--wait] parameterFile
```

The first form of the command (with --help) just displays a brief help message. The second form submits the job and displays the job number. More progress information is displayed if the optional --verbose option is used. The --debug option also displays additional information and producing a debug log file that may be used by Ellucian support to resolve issues.

The --wait parameter instructs the command to wait until the newly launched job has completed. When used, the command will not return immediately, but will poll the Transit API continuously for the job's status and will only exit when the status changes to DONE or FAILED. By default, the command will poll the API every 5 seconds. This can be changed by setting the environment variable JOBREQUEST_WAIT_MILLIS to the desired number of milliseconds. The default value is 5000 (5 seconds). By default, the launchjob command does not wait for the job to finish.

This command must be run in an environment that has been initialized by the dwenv command (a normal Degree Works installed environment). If run by cron, the environment must be initialized before the launchjob command is used.

The parameter file is a JSON data file. Some example parameter files can be found in the \$DGWHOME/samples subdirectory on the classic server. Please copy any of the sample files into your own directory and modify as needed. It is recommended that you create a new myjobs directory under \$ADMIN_HOME and manage your JSON files there. Before running launchjob, you should first cd into the myjobs directory so launchjob can find your json file, or you should supply the full pathname to the json file.

The Transit user interface uses the same JSON format to submit jobs to the queue. Another way to get the JSON parameter data is to use the developer tools provided by your browser. Using Chrome, by inspecting the network submissions to the API, you can find the POST request sent to the API to launch the job: The name of the API call will be instances. Go to the Headers tab, Request Payload section, click **view source** to see the JSON that the API sent for the job. This JSON data can be copied and pasted into the file used by the launchjob command.

In case launchjob encounters an error executing, be sure to redirect launchjob output to a log. Also be sure that stderr is redirected to stdout, either on the same line or elsewhere. For example:

```
launchjob --wait $ADMIN_HOME/myjobs/rad30.currentSql.json >> $LOGFILE
E 2>&1
```

The Transit UI application (which contains the API) must be running to run launchjob.

The script uses the TRANSITBAT user, which must be configured before use. See the [Transit user access](#) topic.

The jobstatus script

Updated: March 25, 2022

The jobstatus script gets the status of a previously launched job. It might be used in a shell script to wait for a job to complete, or to find out if it was successful.

The format of the command is:

```
jobstatus --help
jobstatus [--debug | --verbose] [--wait] job
```

The first form of the command (with --help) just displays a brief help message. The second form checks the job and displays the status. More progress information is displayed if the optional --verbose option is used. The --debug option also displays additional information in addition to producing a debug log file that may be used by Ellucian support to resolve issues.

This command must be run in an environment that has been initialized by the dwenv command (a normal Degree Works installed environment). If run by cron, the environment must be initialized before the jobstatus command is used.

The job parameter may be the job number as shown in Transit or displayed by the launchjob command. It may also be the internal job ID.

Normally, the command does not wait for the job to complete, it simply tells you the current status. If run with the --wait option, it waits for the job to reach either the “DONE” or “FAILED” status before it returns. The wait option polls the Transit API every 5 seconds to check on the job status. The polling interval can be changed by setting the environment variable JOBREQUEST_WAIT_MILLIS to the desired number of milliseconds. The default value is 5000 (5 seconds).

The jobstatus command will normally exit with a zero status if successful. If there was a problem getting the job status, the command will return a status of 1. If it was able to get the job status and the --wait option was used and the job finished with a “FAILED” status, the command will return a status of 2.

The tbedelete script

Updated: March 24, 2023

Transit job metadata (start and stop times, and so on) are stored in the Transit database along with job artifacts (reports, action list, and so on).

By default, this job data will be deleted from the database on a schedule defined in the transit.cleanup.completed.maxDays and transit.cleanup.incomplete.maxDays Shepherd settings. Optionally, the tbedelete script could be used by an administrator to delete transit job data in batch.

The `tbedelete` script removes job data based on age. It is available in the classic Degree Works environment. The `DB_LOGIN_TRANSIT` must be set correctly in order for this script to operate. It may be run through cron so long as the classic environment is initialized.

You pass a number to the `tbedelete` script that represents how old, in days, a job must be to be deleted. It will only delete jobs that are in a `DONE` or `FAILED` status.

It will list the jobs deleted, and the jobs that were not deleted because they are not `DONE` or `FAILED`.

```
$ tbedelete 30
tbedelete - Tue Jun 25 11:52:33 EDT 2019
Deleting jobs that are 30 days old with status DONE or FAILED
JOB_NUMBER JOB_NAME
-----
2 dap27
3 dap27
4 dap27
5 rad30
6 rad30
7 rad30
8 scr92
9 scr91
8 rows selected.
Deleting artifacts that are 30 days old
12 rows deleted.
Deleting job instances that are 30 days old
8 rows deleted.
The following jobs will not be deleted due to their STATUS
JOB_NUMBER JOB_NAME STATUS
-----
1 dap27 PENDING
```

Job request flow through Transit

Updated: March 25, 2022

The Transit UI runs on the Java application server.

The Transit API is in the `transit.jar` along with the Transit UI. It is not deployed separately. The Transit Batch Executor is in the `transitexecutor.jar` that runs on the classic server. All three are required to successfully launch and run a Transit job.

Process:

1. User clicks Launch on a job in the Transit UI.
2. Transit UI sends the request to the Transit API.
3. Transit API processes the request, creates a job instance in the Transit database with a status of `PENDING`, and places a new request on the Rabbit MQ message exchange. If the job is

scheduled to be run at a specified time, it creates a job instance with a status of SCHEDULED first. The system then checks after every interval that is specified (default 5 seconds) and then changes the job to PENDING status at the scheduled time.

4. Transit Batch Executor reads the request from the queue and retrieves the parameters from the API. The status of the job is changed to "RUNNING".
5. The job is executed by a shell script from the \$DGWHOME/jobs directory.
6. When the job is complete the executor contacts the Transit API to attach the job's artifacts to the job instance and to update its status.
7. At this point the Transit UI is able to view the job's artifacts (reports).

Debug Transit

Updated: March 25, 2022

There are two distinct types of debugging available in the Transit user interface, the Transit application itself and the jobs run by Transit.

There are two different ways to enable these debugging modes and they can be used separately or together. Transit users can enable either mode to troubleshoot issues for their session from the user interface.

Additionally, debugging for the Transit Batch Executor can be enabled on the classic server, if needed.

Debug the Transit user interface

Updated: September 29, 2023

With the applicable permissions, users can debug the Transit application as a whole and individual Transit jobs.

Transit debug access

To access the debugging functionality, users should be assigned the DEBUG key either through the Users function of Controller or in the core.security.rules.shpcfg Shepherd setting.

Debug the Transit application

Users with the DEBUG key will see the Debug option under the User Profile image in the upper right corner of the screen. Clicking on Debug image will take the user to the Debug page. There they can toggle the Enable Debug switch to turn debugging on for the Transit application. A random Debug Tag will be generated, and the user can copy this to their clipboard by clicking on the page icon. This Debug Tag will be appended to each line of debug in the log file so that the user can easily identify which lines are from their session, as opposed to other users using the application at the same time.

After enabling debug, the user can then resume using Transit and execute the steps to duplicate the issue they are experiencing. Debug should be turned off by going back to the Debug page and toggling the Enable Debug switch when the issue has been duplicated. Debug for this user's session will also be turned off when they log out of the application.

Debug will appear in the TransitUI log file on the application server. This is typically defined by the \$LOGGING_FILE environment variable in the TransitUI start script.

Searching for the user's Debug Tag will isolate the lines of debug specific to their actions. For example:

```
2017-10-24 16:22:23,144 68b4e19d-3fb4-4975-9815-83eea01266ff DEBUG [
41-exec-17] .h.d.s.s.StatelessPreAuthenticatedFilter : Checking secu
re context token: null
```

The packdebug script can be used to collect the log files generated for a user session's Debug Tag. For more information on using this script, refer to the [packdebug script](#) topic.

Debug the Transit jobs

Users who have the DEBUG key can run any job or processor for which they have access in debug mode. In the Transit UI, an Enable debugging? check box will display as the final question for users with the DEBUG key. When debug has been enabled, jobs such as dap22 or rad30 will generate debug files which are available as job artifacts.

Debugging cannot be enabled when scheduling jobs. To schedule a job, deselect the **Enable debugging?** check box and click the **Schedule** button.

Debug the Transit Executor

Updated: March 25, 2022

The Transit Batch Executor can also be run with debugging enabled.

When the Transit Executor is started, a log file is created in the logdebug directory with today's date appended, for example, transitexecutor.log_2019-05-31. In addition, if a java exception should occur, the stack trace will be written to the \$DW_LOGDEBUG_DIR/transitexecutorError.log.

Debug can be enabled when starting the Transit Executor on the classic server by passing the --debug option:

```
tbestart --debug
```

In addition, JMX Monitoring can be enabled with the --jmx option:

```
tbestart --jmx
```

You can also increase the memory allocated to the executor by passing the value you want to use:

```
tbestart -memory 1536m
```

If you choose to permanently change the memory allocated, you can set the following in `dwenv.config`:

```
export TBE_MEMORY=1536m (substituting the value you want to use with 1536m)
```