



# Ellucian Degree Works

Use, 5.1.2

September 5, 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](http://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

<b>Ellucian Degree Works</b> .....	<b>14</b>
<b>Composer</b> .....	<b>15</b>
Composer editor .....	15
Composer Image editor shortcuts .....	15
CSS .....	16
Shepherd Scripts .....	16
XSL .....	17
Images .....	17
Get started .....	17
Log in .....	18
Sign Out .....	18
Home .....	18
Language Properties .....	19
Language Properties Details .....	19
Localize a Language Property .....	20
Edit a localized Language Property .....	20
Delete a localized Language Property .....	21
Localize Language Property in an Application .....	22
CSS Resources .....	23
CSS Resource Details .....	23
Localize a CSS Resource .....	24
Edit a localized CSS Resource .....	25
Delete a localized CSS Resource .....	25
Localize CSS in an Application .....	26
Scripts .....	27
Shepherd Script Details .....	27
Localize a Shepherd Script .....	27
Edit a localized Shepherd Script .....	28
Delete a localized Shepherd Script .....	29
XSL Resources .....	29
XSL Resource Details .....	30
Localize an XSL Resource .....	30
Edit a Localized XSL Resource .....	31

---

Delete a localized XSL Resource . . . . .	32
Localize XSL in an Application . . . . .	32
Images . . . . .	33
Image Details . . . . .	33
Localize an Image . . . . .	34
Revert a localized Image . . . . .	34
<b>Controller . . . . .</b>	<b>36</b>
User and group authorization . . . . .	36
Create or modify a user access record . . . . .	36
Remove a user . . . . .	38
Create or modify a Shepherd group record . . . . .	39
Remove a Shepherd group record . . . . .	40
Configuration . . . . .	41
Add a Shepherd setting . . . . .	41
Add to or edit existing Shepherd settings specifications . . . . .	41
Delete Shepherd settings specifications . . . . .	42
Create, edit, or copy UCX entries . . . . .	43
Delete UCX entries . . . . .	44
Modify UCX entries in bulk . . . . .	44
Language Properties . . . . .	45
Language Properties Details . . . . .	45
Delete a localized Language Property . . . . .	46
Edit a localized Language Property . . . . .	46
Localize Language Property in an Application . . . . .	47
Localize a Language Property . . . . .	48
Scribe . . . . .	49
Scribe shortcuts . . . . .	49
Scribe blocks . . . . .	50
Requirement ID . . . . .	51
Block type . . . . .	51
Block tags . . . . .	51
Primary tags . . . . .	51
Secondary tags . . . . .	52
Parsing . . . . .	52
Search for blocks . . . . .	52
Search using the requirement ID . . . . .	53

---

---

Search using block tags . . . . .	53
Search results . . . . .	54
Create a block . . . . .	54
Saving the block as a file . . . . .	55
Create a block based on an existing block . . . . .	55
Edit a block . . . . .	56
Find and replace . . . . .	57
Open Local Block . . . . .	57
Parse a block . . . . .	58
Delete a block . . . . .	58
Print a block . . . . .	58
<b>Scribe Language . . . . .</b>	<b>60</b>
Basic Course Rules . . . . .	60
Block . . . . .	61
Block header . . . . .	62
Block type . . . . .	62
Comment . . . . .	62
Course list . . . . .	63
Custom block . . . . .	63
Database tag . . . . .	63
Linked block . . . . .	64
Non-course . . . . .	64
Remark . . . . .	65
Rule qualifier . . . . .	65
Qualifiers . . . . .	65
Header Qualifiers . . . . .	65
Rule Qualifiers . . . . .	66
AdviceJump . . . . .	68
RemarkJump . . . . .	69
Block types and multi-block programs . . . . .	69
Comments . . . . .	71
Proxy-Advice . . . . .	72
Remarks . . . . .	73
Variable to Text Addition . . . . .	73
Group Rule . . . . .	75
Label Tags . . . . .	76

---

---

Recommended Tag Values . . . . .	77
Qualifier Tags . . . . .	78
Change of Catalog Year . . . . .	79
Change of Major . . . . .	81
Course Title as the Label . . . . .	81
Max Header Qualifiers . . . . .	83
Max Zero . . . . .	83
Sharing . . . . .	85
Example 1: Exclusive Block with no Sharing Rules . . . . .	89
Example 2: ShareWith Block with no DontShare Rules . . . . .	90
Example 3: ShareWith Block with a DontShare Rule . . . . .	91
Example 4: Exclusive Block with a ShareWith Rule . . . . .	92
Example 5: Exclusive Block with a ShareWith (CONC). . . . .	93
Example 6: ShareWith on a Subset or Group . . . . .	94
StandAloneBlock . . . . .	95
Subset Rule Types . . . . .	95
Course List Order . . . . .	96
Ordering for Readability . . . . .	96
Ordering for Maintainability . . . . .	97
Ordering for Performance . . . . .	97
Complex Scribing . . . . .	98
Elective Credits Allowed (ECA) Calculations in Degree Works . . . . .	100
Scribing Considerations for Banner Student Course Program of Study, Banner Satisfactory Progress Financial Aid, and Athletic Audits . . . . .	101
Scribing Considerations for ECA Calculations. . . . .	102
Credit Totals . . . . .	102
Degree Block (Starting Block). . . . .	103
Majors . . . . .	103
Minors . . . . .	104
Concentrations . . . . .	105
UCX-CFG020 DAP14 Fall-through courses count in the overall GPA . . . . .	105
Undecided/Undeclared or Missing Minors, Majors and Concentrations . . . . .	105
Scribe Rules and ECA required Block Calculation examples . . . . .	106
Reserved Word Definitions. . . . .	107
1stConc, 1stMajor, 1stMinor, etc. . . . .	108
Accept . . . . .	109
AllBlocks (header) . . . . .	109

---

---

AllBlocks (rule) . . . . .	109
Allow (rule) . . . . .	110
AlwaysShowInAdvice (rule) . . . . .	111
And (header) . . . . .	112
And (rule) . . . . .	112
At (header) . . . . .	113
At (rule) . . . . .	114
Begin . . . . .	115
BeginElse . . . . .	116
BeginIf . . . . .	116
BeginSub (rule) . . . . .	116
Block (rule) . . . . .	117
Blocktype (rule) . . . . .	118
CheckElectiveCreditsAllowed (header) . . . . .	119
Classes (header) . . . . .	121
Classes (rule) . . . . .	122
College (header) . . . . .	123
College (rule) . . . . .	123
CompletedTermCount . . . . .	124
Conc (header) . . . . .	124
Conc (rule) . . . . .	125
CopyHeaderFrom (header) . . . . .	126
CopyRulesFrom (rule) . . . . .	127
Courses (rule) . . . . .	128
Credits (header) . . . . .	128
Credits (rule) . . . . .	129
CreditsAppliedTowardsDegree . . . . .	131
CreditsAttemptedThisAidYear . . . . .	132
CreditsAttemptedThisTerm . . . . .	133
CreditsEarnedThisAidYear . . . . .	134
CreditsEarnedThisTerm . . . . .	135
Current . . . . .	135
Decide (header) . . . . .	136
Decide (rule) . . . . .	137
Degree (header) . . . . .	138
Degree (rule) . . . . .	138
DegreeCreditsRequired . . . . .	139

---

---

Display (header) . . . . .	140
DontShare (header). . . . .	141
DontShare (rule) . . . . .	142
Else . . . . .	143
End. . . . .	144
EndSub . . . . .	144
Except . . . . .	144
Exclusive . . . . .	145
FirstYearEarnedCredits . . . . .	145
From. . . . .	146
Group (rule) . . . . .	146
HeaderTag . . . . .	148
Hide . . . . .	150
HideRule . . . . .	151
HighPriority . . . . .	152
If-Then . . . . .	153
In . . . . .	160
IncludeBlocksWith . . . . .	161
Including . . . . .	162
Label . . . . .	163
LastRes (header). . . . .	164
Libl (header). . . . .	166
Libl (rule) . . . . .	166
LowestPriority . . . . .	167
LowPriority . . . . .	168
Major (header) . . . . .	170
Major (rule). . . . .	170
MaxPassfail (header). . . . .	171
MaxPassfail (rule) . . . . .	172
MaxClasses (header). . . . .	173
MaxCredits (header) . . . . .	174
MaxPerDisc (header). . . . .	176
MaxPerDisc (rule) . . . . .	177
MaxSpread (rule). . . . .	178
MaxTerm (header) . . . . .	179
MaxTerm (rule). . . . .	180
MaxTransfer (header) . . . . .	181



---

MaxTransfer (rule) . . . . .	182
MinAreas (rule) . . . . .	183
MinClasses (header) . . . . .	184
MinClasses (rule) . . . . .	185
MinCredits (header) . . . . .	186
MinCredits (rule) . . . . .	188
MinGPA (header) . . . . .	189
MinGPA (rule) . . . . .	190
MinGrade (header) . . . . .	191
MinGrade (rule) . . . . .	192
Minor (header) . . . . .	193
Minor (rule) . . . . .	193
MinPerDisc (header) . . . . .	194
MinPerDisc (rule) . . . . .	195
MinRes (header) . . . . .	196
MinSpread (rule) . . . . .	197
MinTerm (header) . . . . .	198
MinTerm (rule) . . . . .	199
NoCount . . . . .	200
NonCourse (rule) . . . . .	201
NonExclusive (header) . . . . .	202
NonExclusive (rule) . . . . .	202
NotGpa (rule) . . . . .	202
NumConcs . . . . .	204
NumMajors . . . . .	204
NumMinors . . . . .	205
Optional (header) . . . . .	205
Or (header) . . . . .	206
Or (rule) . . . . .	207
Other (header) . . . . .	208
Other (rule) . . . . .	208
Previous . . . . .	209
Previous2TermsEarnedCredits . . . . .	210
Previous3TermsEarnedCredits . . . . .	210
PreviousAcademicYearEarnedCredits . . . . .	211
PreviousFullYearEarnedCredits . . . . .	211
PreviousTermEarnedCredits . . . . .	212

---

PreviousTermEarnedCredits-Fall . . . . .	212
Program (header) . . . . .	213
Program (rule) . . . . .	213
ProxyAdvice (header) . . . . .	214
ProxyAdvice (rule) . . . . .	216
Pseudo . . . . .	217
Regterm . . . . .	217
Remark . . . . .	218
ResidenceCompletedTermCount . . . . .	219
ResidenceCreditsEarned . . . . .	220
RuleComplete (rule) . . . . .	221
RuleIncomplete (rule) . . . . .	221
RuleTag . . . . .	222
SameDisc (header) . . . . .	224
SameDisc (rule) . . . . .	224
School (header) . . . . .	225
School (rule) . . . . .	226
ShareWith (header) . . . . .	227
ShareWith (rule) . . . . .	230
Spec (header) . . . . .	233
Spec (rule) . . . . .	234
SpMaxCredits (header) . . . . .	235
SpMaxTerm (header) . . . . .	236
StandAloneBlock (header) . . . . .	237
Tag . . . . .	237
Then . . . . .	239
ThisBlock (header) . . . . .	239
ThisBlock (rule) . . . . .	239
TotalCreditsAttempted . . . . .	240
TotalCreditsEarned . . . . .	241
Under (header) . . . . .	241
With . . . . .	242
<b>Transfer Equivalency Admin . . . . .</b>	<b>247</b>
Mappings . . . . .	247
School Mappings . . . . .	247
Create a new school mapping . . . . .	247

---

---

Edit school mappings . . . . .	248
Test Mappings . . . . .	249
Create a new test mapping . . . . .	249
Edit test mappings . . . . .	250
Transcript . . . . .	251
Add or modify student information . . . . .	251
Add or modify test scores . . . . .	252
Add or modify transcript classes . . . . .	252
Add school transcripts . . . . .	253
Articulation . . . . .	254
Perform a re-articulation . . . . .	254
Edit articulated classes . . . . .	255
Resolve unresolved classes . . . . .	255
Resolve undecided classes . . . . .	256
Resolve leftover classes . . . . .	256
Resolve leftover classes through zero credits . . . . .	256
Resolve leftover classes through special mappings . . . . .	257
Resolve duplicate classes . . . . .	257
Audit . . . . .	258
Perform a degree audit . . . . .	258
Roll . . . . .	259
Roll articulation results . . . . .	259
Processes and tables . . . . .	260
View history . . . . .	260
Print report . . . . .	260
<b>Transit . . . . .</b>	<b>262</b>
Transit Run Jobs . . . . .	262
Run ADMIN - Administrative tasks . . . . .	262
Run AUD01 - List unhooked and unenforced exceptions . . . . .	263
Run AUD02 - Delete audits by freeze type and date . . . . .	264
Run BAN62 - Banner SAP Processor . . . . .	265
Run DAP16 - Parse Requirements Processor . . . . .	267
Run DAP21 - Extract Articulated Transfers . . . . .	268
Run DAP22 - Generate Audits . . . . .	270
Run DAP27 - Generate What-if Audits . . . . .	272
Run DAP28 - Generate Alternate What-if audits . . . . .	275

---

---

Run DAP40 - Unload Scribe Blocks .....	277
Run DAP41 - Load Scribe Blocks .....	278
Run DAP42 - Unload Mappings .....	279
Run DAP43 - Load Mappings .....	280
Run DAP44 - Unload UCX Tables .....	281
Run DAP45 - Load UCX Tables .....	282
Run DAP54 - Create Plan from Template Processor .....	282
Run DAP58 - Batch Tracking Processor .....	284
Run DAP59 - Batch Timetabling Processor .....	286
Run RAD11 - Radbridge Batch Processor .....	288
Run RAD30 - Banner Student Extract and Bridge .....	289
Run RAD32 - Banner Applicant Extract and Bridge .....	291
Run RAD33 - Banner Non-Student Extract and Bridge .....	293
Run RAD34 - Banner Course Extract .....	294
Run RAD35 - Banner Curriculum Rules Extract .....	295
Run RAD36 - Banner Validation table Extract (UCX) .....	296
Run RAD37 - Banner Transfer School Extract (ETS) .....	297
Run RAD38 - Banner Equivalencies Extract .....	298
Run RAD39 - Banner Transfer Equivalency Extract (Mappings) .....	299
Run RAD40 - Student Data Delete Processor .....	300
Run SCR02 - Find blocks where this course is referenced .....	302
Run SCR05 - List blocks changed by date range .....	303
Run SCR06 - List block primary and secondary tags .....	304
Run SCR07 - List block text from Scribe .....	305
Run SCR08 - List log entries from Scribe text .....	306
Run SCR09 - List todo entries from Scribe text .....	307
Run SCR10 - Find blocks where this text is referenced .....	308
Run SCR11 - Find-and-Replace block text .....	309
Run SCR91 - Test Banner Prerequisite Checker Service .....	310
Run SCR92 - Test Banner Prerequisite Description Service .....	312
Run SCR93 - Report by Catalog .....	313
Run SCR94 - Report by Schedule .....	314
Run SCR95 - Report by Requisite block .....	315
Run UCX01 - UCX Records Modified .....	316
Transit Completed Jobs .....	317
View completed jobs .....	317
Delete completed jobs .....	317

---

---

Transit Scheduled Jobs . . . . .	318
View or edit scheduled jobs . . . . .	318
Delete scheduled jobs . . . . .	319
Transit Saved Jobs . . . . .	319
View or edit saved job parameters. . . . .	320
Delete saved job parameters. . . . .	320

# Ellucian Degree Works

Updated: September 29, 2023

Ellucian Degree Works is an academic advising and degree audit solution to guide student success.

When you access the Ellucian Documentation Site, you can use the drop-down list in the Contents pane to select the type of content you want to view. If you do not log in to the Ellucian Documentation Site, you can view the Use section. If you use your Customer Account credentials to log in to the Ellucian Documentation Site, you can view all of the Degree Works documentation.

See the Documentation Site Help under the site's main menu for information about additional site features.

# Composer

Updated: March 25, 2022

Composer is a tool for managing localizations in Degree Works.

## Composer editor

Updated: March 25, 2022

The Composer editor offers a set of standard editing features.

Composer provides a built-in editor to create and maintain localized content used by Degree Works applications.

## Composer Image editor shortcuts

Updated: March 25, 2022

A set of shortcuts are offered with the Composer editor.

### Shortcuts

The following shortcuts are available:

Shortcut	Action
<b>Ctrl+/ </b>	Toggle comment; adds/removes # from front of selected lines, or on the current line if no lines are selected
<b>Ctrl+A</b>	Select all text in the editor
<b>Ctrl+C</b>	Copy to clipboard
<b>Ctrl+D</b>	Delete the current line, or the selected text
<b>Ctrl+Down-arrow</b>	Scroll line down
<b>Ctrl+End</b>	Go to the last line
<b>Ctrl+F</b>	Find
<b>Ctrl+H</b>	Replace
<b>Ctrl+Home</b>	Go to the first line
<b>Ctrl+L</b>	Go to line number
<b>Ctrl+P</b>	Print

---

Shortcut	Action
<b>Ctrl+Shift+U</b>	Change to lower case - either the current word or the selected text
<b>Ctrl+U</b>	Change to upper case - either the current word or the selected text
<b>Ctrl+Up-arrow</b>	Scroll line up
<b>Ctrl+V</b>	Paste from clipboard
<b>Ctrl+X</b>	Cut selected text
<b>Ctrl+Z</b>	undo last change

## Find and Replace

Place the cursor focus in the editor's window and use the **Find (Ctrl+F)** or **Replace (Ctrl+H)** shortcut to open the Find/Replace toolbar.

## CSS

Updated: March 25, 2022

CSS (Cascading Style Sheets) is used to configure the colors, styles and graphics in the user interface of Degree Works.

Composer can be used to manage localizations of the style sheets in the user interface of Degree Works.

At installation or update, baseline versions of the CSS style sheets are stored in the shp\_resource\_mst table. These baseline versions of the CSS style sheets are not editable. When a CSS style sheet is localized, its changes are stored in a different field on the shp\_resource\_mst table, with a specific locale defined. The baseline version of the style sheet remains unchanged and is always available for comparison and troubleshooting.

A localized CSS style sheet must be enabled before it is used by the Degree Works web interface.

## Shepherd Scripts

Updated: March 25, 2022

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

Shepherd Scripts contain HTML and JavaScript code with special commands in a propriety format that are interpreted by the Degree Works software.

**Note:** Shepherd Scripts control only the content in a localization. Colors, images, and styles used in the Degree Works web interface are set using CSS, XSL and properties files.



At installation or update, baseline versions of the Shepherd Scripts are stored in the `shp_composer_mst` table. These baseline versions of the Shepherd Scripts are not editable. When a Shepherd Script is localized, its changes are stored in a different field on the `shp_composer_mst` table. The baseline version of the script remains unchanged and is always available for comparison and troubleshooting.

A localized Shepherd Script must be enabled before it is used by the Degree Works web interface.

## XSL

Updated: March 25, 2022

XSL (Extensible Stylesheet Language) is used to configure the formatting of XML content in the user interface of Degree Works applications.

Composer can be used to manage localizations of XSL style sheets.

XSL style sheets contain template elements that are interpreted by the Degree Works software.

At installation or update, baseline versions of these XSL styles sheets are stored in the `shp_resource_mst` table. These baseline versions of the XSL style sheets are not editable. When an XSL style sheet is localized, its changes are stored in a separate `shp_resource_mst` table, with a specific locale defined. The baseline version of the style sheet remains unchanged and is always available for comparison and troubleshooting.

A localized XSL style sheet must be enabled before it is used by the Degree Works Works web interface.

## Images

Updated: March 25, 2022

Images are used to localize the user interface of Degree Works applications.

Composer can be used to manage localizations of images displayed in an application.

At installation or update, baseline versions of these images are stored in the `shp_resource_mst` table. These baseline versions of the images are not editable. When an image is localized, a new uploaded image is stored in a separate `shp_resource_mst` table, with a specific locale defined. The baseline version of the image remains unchanged and is always available for comparison and troubleshooting.

A localized image must be enabled before it is used by the Degree Works web interface.

## Get started

Updated: March 25, 2022

Account credentials are required to access Composer.

## Log in

Updated: March 25, 2022

Log into Composer with your account credentials.

### About this task

This option applies only to those users who have Composer account credentials.

### Procedure

1. At the **Composer Log in** screen, enter your account information:
  - User name
  - Password

**Tip:** Click the display/hide icon at the right of the **Password** field to either show or mask password entry.
2. Click **Sign In** to access and begin working with Composer.

## Sign Out

Updated: March 25, 2022

Use this option to disconnect from your current session.

If at any point during your session you want to stop working, click the **Persona** drop-down (top right) and choose the **Sign Out** option.

**Note:** If Composer detects any unsaved work, you will be prompted to save your data or risk its loss.

---

## Home

Updated: March 25, 2022

After credential entry, the Composer **Home** page is displayed.

Select the desired option:

- Language Properties
- CSS
- Scripts
- XSL
- Images

## Language Properties

Updated: March 25, 2022

The **Language Properties** page allows you to manage localized Composer Language Property content.

The **Language Properties** page lists current Composer Language Properties available for localization. Content is listed by Name and Description.

## Language Properties Details

Updated: March 25, 2022

The Details tab allows you to display various information related to the Language Property.

When selected, the Details tab shows the following information:

Field	Description
<b>Last Modified Date</b> and <b>Who</b>	Updated when changes are saved to the localized Language Property.  <b>Note:</b> These fields will not display unless a localized properties file for the selected locale exists in the database.
<b>Created Date</b> and <b>Who</b>	Reflects when the localized Language Property was initially created.  <b>Note:</b> These fields will not display unless a localized properties file for

---

---

Field	Description
	the selected locale exists in the database.
<b>Baseline Version</b>	Degree Works release in which that version of the baseline source was delivered.

---

## Localize a Language Property

Updated: March 25, 2022

The **Language Properties** page allows you to select, copy, localize, and enable a Composer Language Property.

### Procedure

1. Select a Composer Language Property from the displayed list.

**Note:** Use the **Search title/description** field to reduce the number of listed Language Properties. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of Language Properties indicates the number of records stored.

2. In Editor mode, click the **Copy** button to duplicate the selected Baseline Language Property.

**Result:** The duplicate resource is shown in the Localized Resource editor pane.

3. Using the editor's available tools, modify the Language Property, as needed.

**Note:** A valid Language Property may have an unlimited number of lines, and each line is of unlimited length.

4. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Edit a localized Language Property

Updated: March 25, 2022

The **Language Properties** page allows you to specify and edit a previously localized Composer Language Property.

### About this task

Use this option to edit a previously localized Composer Language Property.

### Procedure

1. Select a previously localized Composer Language Property from the displayed list.

**Note:** Use the **Search title/description** field to reduce the number of listed Language Properties. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

2. Using the editor's available tools, modify the Composer Language Property, as needed.

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

3. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Delete a localized Language Property

Updated: March 25, 2022

Remove a localized Composer Resource.

### About this task

Use these steps to remove a localized Resource.

### Procedure

1. Select a Composer Language Property from the displayed list.

**Note:**

- Use the **Search title/description** field to reduce the number of listed Language Properties. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.
  - A counter at the top of the list of Language Properties indicates how many records have been found.
2. In Editor mode, switch the **Localization** toggle to Disabled.
  3. Select and delete all text in the Localized editor pane.
  4. Click the **Save** button to save your changes.

**Result:** A confirmation message will be displayed in the Notification Center.

## Localize Language Property in an Application

Updated: March 25, 2022

The **Language Properties** page allows you to enable Language Properties localizations in an application.

### About this task

Text, labels, and messages displayed in the following Degree Works Java-based applications are generated from baseline properties files:

- Responsive Dashboard
- Dashboard Servlet
- Student Educational Planner
- Scribe
- Composer
- Transit
- Controller
- Transfer Equivalency Self-Service
- Transfer Finder
- All APIs

---

## Procedure

1. Select a baseline Language Property from the displayed list.

**Note:**

- Use the **Search title/description** field to reduce the number of listed Language Properties. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.
- A counter at the top of the list of Language Properties indicates how many records have been found.

2. In Editor mode, choose the desired locale from the drop-down list.
3. Using the editor's available tools, copy and paste the properties you want to localize from the Baseline Language Property editor pane to the Localized Language Property editor pane.

**Note:** Each line of the properties file has of two sections, the KEY and the TEXT. The KEY includes all characters to the left of the = symbol, on a given line. The TEXT includes all characters to the right of the = symbol, on a given line.

**Note:** Do not modify the KEY portion of the line as this content is used by the application. If the KEY is modified, the message in question will not be displayed.

4. Click the **Save** button or switch the **Localization** toggle to save your changes. A confirmation message will be displayed in the Notification Center.

## CSS Resources

Updated: March 25, 2022

The CSS Resources page allows you to manage localized Composer CSS content

The **CSS Resources** page lists current Composer CSS Resources available for localization. Content is listed by Name and Description.

## CSS Resource Details

Updated: March 25, 2022

The Details tab allows you to display various information related to the Composer CSS Resource.

When selected, the Details tab shows the following information:

Field	Description
<b>Last Modified Date and Who</b>	Updated when changes are saved to the localized CSS Resource.

Field	Description
<b>Note:</b> These fields will not display unless a localized CSS Resource for the selected locale exists in the database.	
<b>Created Date and Who</b>	Reflects when the CSS Resource was initially loaded into the database.  <b>Note:</b> These fields will not display unless a localized CSS Resource for the selected locale exists in the database.
<b>Baseline Version</b>	Degree Works release in which that version of the baseline source was delivered.

## Localize a CSS Resource

Updated: March 25, 2022

The **CSS Resource** page allows you to specify, copy, localize, and enable a Composer CSS Resource.

### About this task

Use this option to localize a Composer CSS Resource.

### Procedure

1. Select a Composer CSS Resource from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed CSS Resources. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of CSS Resources indicates how many records have been found.

2. In Editor mode, click the **Copy** button to duplicate the selected Baseline CSS Resource.

**Result:** The duplicate CSS Resource is shown in the Localized editor pane.

3. Using the editor's available tools, modify the CSS Resource, as needed.

**Note:** A valid CSS Resource may have an unlimited number of lines, and each line is of unlimited length.

4. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.



**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Edit a localized CSS Resource

Updated: March 25, 2022

The **CSS Resource** page allows you to specify and edit a previously localized Composer CSS Resource.

### Procedure

1. Select a previously localized Composer CSS Resource from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed CSS Resources. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

2. Using the Editor's available tools, modify the Composer CSS Resource, as needed.

**Note:** A valid CSS Resource may have an unlimited number of lines, and each line is of unlimited length.

3. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Delete a localized CSS Resource

Updated: March 25, 2022

Remove a localized CSS Resource.

### About this task

Use these steps to remove a localized CSS Resource.

### Procedure

1. Select a Composer CSS Resource from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed CSS Resources. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of CSS Resources indicates how many records have been found.

2. In Editor mode, switch the **Localization** toggle to Disabled.
3. Select and delete all text in the Localized editor pane.
4. Click the **Save** button to save your changes.

**Result:** A confirmation message will be displayed in the Notification Center.

## Localize CSS in an Application

Updated: March 25, 2022

Enable CSS Resource localizations in an application

### About this task

Colors, styles, and graphics used in the following Degree Works java-based applications can be configured through CSS:

- Responsive Dashboard
- Dashboard Servlet
- Student Educational Planner
- Scribe
- Composer
- Transit
- Controller
- Transfer Equivalency Self-Service
- Transfer Finder
- All APIs

### Procedure

1. Select a baseline CSS Resource from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed CSS Resources.

With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of CSS Resources indicates how many records have been found.

2. In Editor mode, choose the desired locale from the drop-down list.
3. Using the editor's available tools, copy and paste the classes that you want to localize from the Baseline CSS Resource editor pane to the Localized CSS Resource editor pane.

**Note:** You can use the editor's **Copy** function to duplicate the entire baseline CSS Resource into a localized CSS Resource. However, this is not recommended unless you are planning to localize the CSS file.

4. Using the editor's available tools, make the desired modification to the class.
5. Click the **Save** button or switch the **Localization** toggle to save your changes.

**Result:** A confirmation message will be displayed in the Notification Center.

## Scripts

Updated: March 25, 2022

The **Scripts** page allows you to manage localized Composer Shepherd Script content.

The **Scripts** page lists current Composer Shepherd Scripts available for localization. Content is listed by Name, Description, and Localization Status (Enabled vs. Disabled).

## Shepherd Script Details

Updated: March 25, 2022

The Details tab allows you to display various information related to the Composer Shepherd Script.

When selected, the Details tab shows the following information:

Field	Description
<b>Last Modified Date and Who</b>	Updated when changes are saved to the localized script.
<b>Created Date and Who</b>	Reflects when the script was initially loaded into the database.
<b>Baseline Version</b>	Degree Works release in which that version of the baseline source was delivered.

---

## Localize a Shepherd Script

Updated: March 25, 2022

The **Scripts** page allows you to select, copy, localize, and enable a Composer Shepherd Script.

### Procedure

1. Select a Composer Shepherd Script from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed Scripts. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** The **Localization** column includes a **non-editable** indicator to show whether the localized script is enabled. Additionally, a counter at the top of the list of Shepherd Scripts indicates how many records have been found.

2. In Editor mode, click the **Copy** button to duplicate the selected Baseline Script.

**Result:** The duplicate script is shown in the Localized editor pane.

3. Using the editor's available tools, modify the script, as needed.

**Note:** A valid script may have an unlimited number of lines, and each line is of unlimited length. However, due to limitations, Ellucian recommends a maximum line length of 200 characters.

4. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Edit a localized Shepherd Script

Updated: March 25, 2022

The **Scripts** page allows you to specify and edit a previously localized Composer Shepherd Script.

### Procedure

1. Select a previously localized Composer Shepherd Script. from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed Scripts. With each

character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** The **Localization** column includes a **non-editable** indicator to show whether the localized script is enabled. Additionally, a counter at the top of the list of Shepherd Scripts indicates how many records have been found.

2. Using the editor's available tools, modify the script, as needed.

**Note:** A valid script may have an unlimited number of lines, and each line is of unlimited length. However, due to limitations, Ellucian recommends a maximum line length of 200 characters.

3. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Delete a localized Shepherd Script

Updated: March 25, 2022

Remove a localized Composer Shepherd Script.

### Procedure

1. Select a Composer Shepherd Script from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed Scripts. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** The **Localization** column includes a **non-editable** indicator to show whether the localized script is enabled. Additionally, a counter at the top of the list of Shepherd Scripts indicates how many records have been found.

2. In Editor mode, switch the **Localization** toggle to Disabled.
3. Select and delete all text in the Localized editor pane.
4. Click the **Save** button to save your changes.

**Result:** A confirmation message will be displayed in the Notification Center.

---

## XSL Resources

Updated: March 25, 2022

The XSL Resources page allows you to manage localized Composer XSL Resource content.

The **XSL Resources** page lists current Composer XSL Resources available for localization. Content is listed by Name and Description.

## XSL Resource Details

Updated: March 25, 2022

Field	Description
<b>Last Modified Date and Who</b>	Updated when changes are saved to the localized XSL Resource.  <b>Note:</b> These fields will not display unless a localized XSL Resource for the selected locale exists in the database.
<b>Created Date and Who</b>	Reflects when the CSS Resource was initially loaded into the database.  <b>Note:</b> These fields will not display unless a localized XSL Resource for the selected locale exists in the database.
<b>Baseline Version</b>	Degree Works release in which that version of the baseline source was delivered.

## Localize an XSL Resource

Updated: March 25, 2022

The **XSL Resource** page allows you to specify, copy, localize

### About this task

Use this option to localize a Composer XSL Resource.

### Procedure

1. Select a Composer XSL Resource from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed XSL Resources. With each character entered, the displayed list is automatically sorted (and shortened).

---

Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of XSL Resources indicates the number of records stored.

2. In Editor mode, click the **Copy** button to duplicate the selected Baseline Resource.

**Result:** The duplicate CSS Resource is shown in the Localized editor pane.

3. Using the editor's available tools, modify the XSL Resource, as needed.

**Note:** A valid XSL Resource may have an unlimited number of lines, and each line is of unlimited length.

4. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Edit a Localized XSL Resource

Updated: March 25, 2022

**XSL Resource** page allows you to specify and edit a previously localized Composer XSL Resource.

### About this task

Use this option to edit a previously localized Composer XSL Resource.

### Procedure

1. Select a previously localized Composer XSL Resource from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed XSL Resources. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of XSL Resources indicates the number of records stored.

2. Using the Editor's available tools, modify the Composer XSL Resource, as needed.

**Note:** A valid XSL Resource may have an unlimited number of lines, and each line is of unlimited length.

3. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Delete a localized XSL Resource

Updated: March 25, 2022

Remove a localized XSL Resource.

### About this task

Use these steps to remove a localized XSL Resource.

### Procedure

1. Select a Composer XSL Resource from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed XSL Resources. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of XSL Resources indicates the number of records stored.

2. In Editor mode, switch the **Localization** toggle to Disabled.
3. Select and delete all text in the Localized editor pane.
4. Click the **Save** button to save your changes.

**Result:** A confirmation message will be displayed in the Notification Center.

## Localize XSL in an Application

Updated: March 25, 2022

Enable XSL Resource localizations in an application.

### About this task

The formatting of XML content in the Degree Works web interface can be configured through XSL.



---

## Procedure

1. Select a baseline XSL Resource from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed XSL Resources. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of XSL Resources indicates the number of records stored.

2. In Editor mode, choose the desired locale from the drop-down list.
3. Using the editor's available tools, copy and paste the classes that you want to localize from the Baseline XSL Resource editor pane to the Localized XSL Resource editor pane.

**Note:** You can use the editor's **Copy** function to duplicate the entire baseline XSL Resource into a localized XSL Resource. However, this is not recommended unless you are planning to localize the XSL file.

4. Using the editor's available tools, make the desired modification to the class.
5. Click the **Save** button or switch the **Localization** toggle to save your changes.

**Result:** A confirmation message will be displayed in the Notification Center.

## Images

Updated: March 25, 2022

The **Images** page allows you to manage localized Composer Images.

The **Images** page lists current Images used in Degree Works applications that are available for localization. Images are listed by Name, Description, and Localization Status (Enabled vs. Disabled).

## Image Details

Updated: March 25, 2022

Field	Description
<b>Last Modified Date and Who</b>	Updated when changes are saved to the localized Image.  <b>Note:</b> These fields will not display unless a localized Image for the selected locale exists in the database.
<b>Created Date and Who</b>	Reflects when the Image was initially loaded into the database.

---

Field	Description
<b>Note:</b> These fields will not display unless a localized Image for the selected locale exists in the database.	
<b>Baseline Version</b>	Degree Works release in which that version of the baseline source was delivered.

## Localize an Image

Updated: March 25, 2022

The **Images** page allows you to specify, localize, and enable a Composer Image.

### About this task

Use this option to localizes Composer Image.

### Procedure

1. Select a Composer Image from the displayed list.

**Tip:** Use the **Search title/description** field to reduce the number of listed Images. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of Images indicates the number of records stored.

2. In Editor mode, click the **Browse** button and navigate to the location of the image file that you want to use.
3. Select the image file.
4. Click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

## Revert a localized Image

Updated: March 25, 2022

Revert a localized Composer Image to the Baseline Image.

**Before you begin**

A previously localized Composer Image to revert to the Baseline Image.

**Note:** A localized Composer Image that has been uploaded cannot be deleted. The image can be reverted to the Baseline Image.

**Procedure**

In Editor mode, switch the **Localization** toggle to Disabled.

**Result:** A confirmation message will be displayed in the Notification Center.

# Controller

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.

## User and group authorization

Updated: March 25, 2022

You can create, modify, and remove user and group access records.

## Create or modify a user access record

Updated: September 29, 2023

You can create or modify a user record in Controller for users to log in to Degree Works.

### Before you begin

To create user access records in Controller, you must have the CTLUSRAD Shepherd key. To modify user access records in Controller, you must have the CTLUSRMD Shepherd key.

### Procedure

1. If necessary, click **Users**.
2. Create or modify a user.
  - Create - click **Add new user**.
  - Modify - in the **User name or Access/SSO/DW ID** field, search for a user and click **Edit**.
3. Add or change the identifying user information.
  - a. In the **User name** field, enter or change the name that will display in Degree Works applications.

Typically you will have loaded most users with the data bridge.
  - b. In the **Degree Works ID** field, click **Search** to search for and select the correct record that links the Degree Works ID to the user access record.

You can search by rad\_id or rad\_name on the rad\_primary\_mst.

- 
- c. In the **Access ID** field, enter the user's unique ID.

When creating a user access record, you must provide an Access ID. This is the ID the user uses to access Degree Works applications. It can be up to 14 alphanumeric characters in length and is unique. After you save a user access record, you cannot change the Access ID.

- d. In the **Password** field, enter a password for the Access ID.

Degree Works uses this password only if your environment uses SHP authentication. Other methods of authentication, such as CAS or SAML will not use this password.

This is a required field when you create a user. It is optional if you are modifying a user.

- e. If you are using single sign-on authentication, enter the user ID that needs to be mapped to Degree Works in the **Single sign-on ID** field. This is the ID used in CAS/SAML authentication.
- f. If you are using Transfer Finder, enter the Transfer Finder ID that stores the common system ID used by Transfer Finder in the **Transfer Finder ID** field.
- g. If you are using Transfer Equivalency Self-Service, enter the alternate username value that stores the email address the prospective student will log in with in the **Alternate username** field.

4. Add or change the user access information.

- a. In the **User class** field, select the class that defines the service that the user can access in Degree Works.

This is a required field when you create a user. It is optional if you are modifying a user.

- b. If necessary, click **Add Filter** and select a Department, School, or College advisee filter type.

You can add up to 10 filters codes for the selected filter type. You can delete the filters individually or en masse by clicking **Clear all**.

Advisee filters prefilter the list of students in Responsive Dashboard for the user based on the filter type and filter codes defined for the user.

- c. **Optional:** Select **Deny** if you need to prevent a user from being able to access any service on their next logon.
- d. **Optional:** In the **Access expire** field, set a date for when this user's access privileges are to lapse.
- e. **Optional:** In the **User expire** field, set a date for when you want the account to expire.
- f. **Optional:** In the **Timeout increment** field, specify the time after which a user's session should automatically end if the user is inactive.
- g. **Optional:** In the **Timeout maximum** field, specify the maximum time for which a user session can be inactive before ending.

- ## Remove a user

2. Select one or more users.
  - One user - to the right of the applicable user, click **Delete**.
  - Multiple users - select the applicable users. Above the list, click **Delete**
3. At the prompt, click **Delete**.

## Create or modify a Shepherd group record

Updated: March 25, 2022

Groups are collections of Shepherd keys that you can assign to users to grant them access to that collection of Degree Works services.

### Before you begin

To create group records in Controller, you must have the CTLGRPAD Shepherd key. To modify group records in Controller, you must have the CTLGRPMD Shepherd key.

### About this task

Controller stores groups in the shp-group-mst record, which has a limit of 50 keys. You can, however, 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. For example, SRNREG-2, SRNREG-3, and so on. Note that you must have assigned 50 keys to a group before you can add a new sequence record. If one of the group sequence records has fewer than 50, Controller interprets that as the last sequence record and does not assign keys in the following sequence records 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, and so on.

### Procedure

1. Click **Groups**.
2. Create or modify a group.
  - Create - click **Add new group**.
  - Modify - in the **Filter by group or description** field, enter a keyword string containing the group name or description, and then click **Edit** to the right of the applicable group.
3. Add or modify the group details.
  - a. If you are adding a group, in the **Group** field, enter a unique name of your choosing.

The group name can be up to eight alphanumeric characters. After you save a group

name, you cannot change it.

b. **Optional:** In the **Expire date** field, enter an expiration date for the group.

c. **Optional:** In the **Description** field, enter a description.

The description can be 30 alphanumeric characters.

d. **Optional:** In the **Timeout increment** field, specify the time after which a session of a user with this group should automatically end if the user is inactive.

e. **Optional:** In the **Timeout maximum** field, specify the maximum time for which a session of a user with this group can be inactive before ending.

f. **Optional:** Select **Never timeout** if you want to keep the user with this group logged in regardless of the inactive time.

4. Add or modify a group key.

a. Click **Add key**, or to the right of an existing group key, click **Edit**.

**Tip:** If you want to delete a group key, to the right of any existing group key, click **Delete**.

b. If you are adding a group key, in the **Select a key** field, select the applicable key.

c. In the **Operator** field, select **Allow permission** to grant access or **Remove permission** to deny access to this key for this user.

d. **Optional:** In the **Expire date** field, set a date after which Controller will remove the key from the group.

5. Click **Save** or **Update**.

## Remove a Shepherd group record

Updated: March 25, 2022

You can remove a Shepherd group record from Controller.

### Before you begin

To remove Shepherd group records from Controller, you must have the CTLGRPDL Shepherd key.

### Procedure

1. Search for the applicable group or groups.
2. Select one or more groups.
  - One group - to the right of the applicable group, click **Delete**.



- Multiple groups - select the applicable groups. Above the list, click **Delete**.
3. At the prompt, click **Delete**.

## Configuration

Updated: March 25, 2022

You can add, edit, and delete Shepherd settings and create, modify, copy, and delete UCX entries.

### Add a Shepherd setting

Updated: September 29, 2023

You can add a Shepherd setting if absolutely necessary. For example, you may need to add a Shepherd setting to be able to localize the next steps displayed in Transfer Equivalency Self-Service.

#### Before you begin

To add a Shepherd setting, you must have the CTLSETAD Shepherd key.

#### About this task

When you add a Shepherd setting, Controller adds it with a default specification. You can define a description for this setting by adding a localized property to ShpSettingsMessages.properties in Controller. This property should be in the format of shpSetting.description. + Shepherd setting key. For example, if your new setting is treq.selfService.nextSteps.10, your description property should be shpSetting.description.treq.selfService.nextSteps.10.

#### Procedure

1. Click **Add a Shepherd setting**.
2. Enter the key and value.
3. Click **Save**.

### Add to or edit existing Shepherd settings specifications

Updated: March 25, 2022

In certain situations, you may want to specify a different value for general settings to different applications.

---

## Before you begin

To add to existing Shepherd setting specifications, you must have the CTLSETAD Shepherd key.  
To edit existing Shepherd setting specifications, you must have the CTLSETMD Shepherd key.

## About this task

While most settings are specific to a particular application, there are many that generally apply to all applications (for example, core settings). The Specification field allows you to create or modify configurations based on the application. Each application has a unique specification value

## Procedure

1. Click **Configuration**.
2. In the **Shepherd settings keys or UCX tables** field, enter a keyword string.
3. To the right of the search results row, click **View**.
4. Add to or edit the specification.

Choice	Instruction
Add specification	<ol style="list-style-type: none"><li>a. Click <b>Add specification</b>.</li><li>b. In the <b>Specification</b> field, select the applicable specification.</li><li>c. In the <b>Value</b> field, enter or select the applicable value.</li><li>d. Click <b>Save</b>.</li></ol>
Edit specification	<ol style="list-style-type: none"><li>a. To the right of the specification, click <b>Edit</b>.</li><li>b. In the <b>Value</b> field, change the value.</li><li>c. Click <b>Save</b>.</li></ol>

## Delete Shepherd settings specifications

Updated: March 25, 2022

You can delete Shepherd settings specifications when necessary.

## Before you begin

To delete Shepherd settings specifications, you must have the CTLSETDL Shepherd key.

---

## About this task

You cannot delete default specifications.

## Procedure

1. Click **Configuration**.
2. In the **Shepherd settings keys or UCX tables** field, enter a keyword string.
3. To the right of the search results row, click **View**.
4. To the right of the applicable specification, click **Delete**.

## Create, edit, or copy UCX entries

Updated: March 25, 2022

You can create, edit, or copy UCX table entries.

## Before you begin

To create UCX table entries, you must have the CTLUCXAD Shepherd key. To edit UCX table entries, you must have the CTLUCXMD Shepherd key.

## Procedure

1. Click **Configuration**.
2. In the **Shepherd settings keys or UCX tables** field, enter a keyword string.
3. To the right of the search results row, click **View**.
4. Create, edit, or copy a UCX entry.

Choice	Instruction
Create UCX entry	<ol style="list-style-type: none"><li>a. Click <b>Add record</b>.</li><li>b. Enter the applicable information.</li><li>c. Click <b>Save</b>.</li></ol>
Edit UCX entry	<ol style="list-style-type: none"><li>a. To the right of the specification, click <b>Edit</b>.</li><li>b. Make the applicable changes.</li><li>c. Click <b>Save</b>.</li></ol>

---

Choice	Instruction
Copy UCX entry	<ul style="list-style-type: none"><li>a. To the right of the specification, click <b>Copy</b>.</li><li>b. Make the applicable changes.</li><li>c. Click <b>Save</b>.</li></ul>

## Delete UCX entries

Updated: March 25, 2022

You can remove UCX table entries when necessary.

### Before you begin

To delete UCX table entries, you must have the CTLUCXDL Shepherd key.

### Procedure

1. Click **Configuration**.
2. In the **Shepherd settings keys or UCX tables** field, enter a keyword string.
3. To the right of the search results row, click **View**.
4. To the right of the applicable UCX entry, click **Delete**.

## Modify UCX entries in bulk

Updated: March 25, 2022

You can use the import and export functionality in Controller to modify UCX entries in bulk.

### Before you begin

To import and export UCX tables, you must have the CTLUCXBK Shepherd key.

### About this task

Controller also validates the file for the correct UCX table, duplicate entries, tab characters, and so on. It will not import until you correct any errors. Be mindful that Controller overwrites all of the records in the UCX table on import.

---

## Procedure

1. Click **Configuration**.
2. In the **Shepherd settings keys or UCX tables** field, enter a keyword string.
3. To the right of the search results row, click **View**.
4. Click **Export**.
5. Open the .txt file, make your changes, and save it to your computer.
6. In Controller, on the applicable UCX entry, click **Import**.
7. Navigate to and select the applicable .txt file, and click **Open**.
8. Click **Yes** at the prompt.

## Language Properties

Updated: March 25, 2022

Language Properties define most of the text, labels, and messages used in the user interface of Degree Works applications. The default values for an application can be changed to create a custom experience.

At installation or update, baseline versions of these resources are stored in the shp\_resource\_mst table. These baseline versions of the resources are not editable. When a resource is localized, its changes are stored in a separate shp\_resource\_mst table, with a specific locale defined. The baseline version of the resource remains unchanged and is always available for comparison and troubleshooting.

A localized Language Property must to be enabled before it is used by the Degree Works web interface.

## Language Properties Details

Updated: March 25, 2022

The Details tab allows you to display various information related to the Language Property.

When selected, the Details tab shows the following information:

Field	Description
<b>Last Modified Date and Who</b>	Updated when changes are saved to the localized Language Property.

**Note:** These fields will not display unless a localized properties file for

---

---

Field	Description
	the selected locale exists in the database.
<b>Created Date</b> and <b>Who</b>	Reflects when the localized Language Property was initially created.  <b>Note:</b> These fields will not display unless a localized properties file for the selected locale exists in the database.
<b>Baseline Version</b>	Degree Works release in which that version of the baseline source was delivered.

---

## Delete a localized Language Property

Updated: March 25, 2022

Remove a localized Composer Resource.

### About this task

Use these steps to remove a localized Resource.

### Procedure

1. Select a Composer Language Property from the displayed list.

**Note:**

- Use the **Search title/description** field to reduce the number of listed Language Properties. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.
- A counter at the top of the list of Language Properties indicates how many records have been found.

2. In Editor mode, switch the **Localization** toggle to Disabled.
3. Select and delete all text in the Localized editor pane.
4. Click the **Save** button to save your changes.

**Result:** A confirmation message will be displayed in the Notification Center.

---

## Edit a localized Language Property

Updated: March 25, 2022

The **Language Properties** page allows you to specify and edit a previously localized Composer Language Property.

### About this task

Use this option to edit a previously localized Composer Language Property.

### Procedure

1. Select a previously localized Composer Language Property from the displayed list.

**Note:** Use the **Search title/description** field to reduce the number of listed Language Properties. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

2. Using the editor's available tools, modify the Composer Language Property, as needed.

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

3. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## Localize Language Property in an Application

Updated: March 25, 2022

The **Language Properties** page allows you to enable Language Properties localizations in an application.

### About this task

Text, labels, and messages displayed in the following Degree Works Java-based applications are generated from baseline properties files:

- Responsive Dashboard

- Dashboard Servlet
- Student Educational Planner
- Scribe
- Composer
- Transit
- Controller
- Transfer Equivalency Self-Service
- Transfer Finder
- All APIs

## Procedure

1. Select a baseline Language Property from the displayed list.

### Note:

- Use the **Search title/description** field to reduce the number of listed Language Properties. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.
- A counter at the top of the list of Language Properties indicates how many records have been found.

2. In Editor mode, choose the desired locale from the drop-down list.
3. Using the editor's available tools, copy and paste the properties you want to localize from the Baseline Language Property editor pane to the Localized Language Property editor pane.

**Note:** Each line of the properties file has of two sections, the KEY and the TEXT. The KEY includes all characters to the left of the = symbol, on a given line. The TEXT includes all characters to the right of the = symbol, on a given line.

**Note:** Do not modify the KEY portion of the line as this content is used by the application. If the KEY is modified, the message in question will not be displayed.

4. Click the **Save** button or switch the **Localization** toggle to save your changes. A confirmation message will be displayed in the Notification Center.

## Localize a Language Property

Updated: March 25, 2022

The **Language Properties** page allows you to select, copy, localize, and enable a Composer Language Property.



## Procedure

1. Select a Composer Language Property from the displayed list.

**Note:** Use the **Search title/description** field to reduce the number of listed Language Properties. With each character entered, the displayed list is automatically sorted (and shortened). Matching names rise to the top of the list. Non-matching names are removed from the list.

**Note:** A counter at the top of the list of Language Properties indicates the number of records stored.

2. In Editor mode, click the **Copy** button to duplicate the selected Baseline Language Property.

**Result:** The duplicate resource is shown in the Localized Resource editor pane.

3. Using the editor's available tools, modify the Language Property, as needed.

**Note:** A valid Language Property may have an unlimited number of lines, and each line is of unlimited length.

4. When all modifications have been applied, click the **Save** button or switch the Localization toggle to save your changes.

**Note:** Clicking the **Save** button (alone), only updates the database record. The localized content is not enabled unless the **Localization** toggle was already enabled.

**Result:** A confirmation message will be displayed in the Notification Center.

## 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.

The Scribe application supports a simple programming language for encoding degree requirements. Use the Scribe Editor to encode degree requirements and save them as blocks. You can copy text and data from documents or spreadsheets into the Scribe Editor. Before a requirement block can be saved, it must be checked for syntax. In Scribe, syntax is checked using a parser utility.

See [Scribe Language](#) for information about the Scribe language used to code blocks.

---

## Scribe shortcuts

Updated: March 25, 2022

Keyboard 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.

The following keyboard 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
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

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

## Scribe blocks

Updated: March 25, 2022

A block is a set of requirements written in the Scribe language. Requirements for a specific degree or program may be written entirely in one block, or they may be divided among several blocks.

Every block consists of two sections: the Header and the Body. The block Header consists of conditions that apply to the block as a whole. The block Header section starts with the BEGIN command and ends at the first semicolon in the block. The Body section starts after the first semicolon and continues to the END command. The Body may consist of course requirements, noncourse requirements, block requirements, rule qualifiers that may apply to a particular rule,

proxy-advice, and remarks.

## Requirement ID

Updated: March 25, 2022

A Requirement ID for a block is an 8-character unique identifier for the block.

The Requirement ID is an eight-character identifier comprised of the prefix *RA* or *RB*, followed by six numbers. For example, *RA000001* or *RB000025*. Each block is assigned a unique, sequential requirement ID when it is created.

## Block type

Updated: March 25, 2022

A block type defines the kind of block that it is.

Blocks are categorized by block type. There are a number of possible block types. Each block type may contain requirements for specific sections of a student's transcript.

## Block tags

Updated: March 25, 2022

Block tags are the characteristics that Degree Works uses to match blocks to students.

When a block is saved, it is given a set of tags. These tags are the characteristics that Degree Works uses to match blocks to students. There are primary and secondary tags.

### *Primary tags*

Updated: March 25, 2022

Primary block tags are the required tags for each scribe block.

The following primary tags are defined in Scribe.

- **Block type**  
Block types are predefined in Scribe. There are a number of possible block types. Each block type may contain requirements for specific sections of a student's transcript.
- **Block value**  
The value is determined by the block type. For example, if the block type is Degree, then possible values could be degree codes such as Bachelor of Arts or Master of Business Administration.
- **Start catalog year**  
The start catalog year is the earliest catalog date for which the block requirements are effective. In the case of a Requisite block, this will be Start term.

- Stop catalog year  
The stop catalog year is the last catalog date for which the block requirements are effective. In the case of a Requisite block, this will be Stop term.

### *Secondary tags*

*Updated: March 25, 2022*

Secondary block tags are optional tags that can be assigned to scribe blocks.

The following secondary tags are defined in Scribe:

- Student ID
- School
- Degree
- College
- Major 1
- Major 2
- Concentration
- Minor
- Liberal Learning
- Specialization
- Program

### **Parsing**

Updated: March 25, 2022

Before a scribe block can be saved, it must be parsed.

The parser utility of Scribe scans the contents of the block to ensure that the block is formatted in a way that the auditor program can read it when producing student worksheets. The parser checks the syntax and Rule structure of a scribe block and checks if the information in a block is consistent with the corresponding information in the database. For example, the parser will check if all the course disciplines referred in the block are valid course disciplines defined in the database.

## **Search for blocks**

Updated: March 25, 2022

You can search for a block using the block's requirement ID or by specifying a combination of

---

primary and secondary database tags.

## Search using the requirement ID

Updated: March 25, 2022

If you know the Requirement ID of the block, you can search for it in the Open by Requirement ID section of the Block Search page.

### About this task

#### Procedure

1. If you know the Requirement ID of the block, select the **RA** or **RB** prefix from the drop-down list in the **Open by Requirement ID** section.
2. Type the requirement ID in the box.
3. Click **Open block requirement ID**.

#### Results

The block is opened in the editor.

## Search using block tags

Updated: March 25, 2022

You can search for blocks using a combination of primary and secondary block tags in the Block Selection Criteria section of the Block Search page.

### About this task

#### Procedure

1. Select values for **Primary Tags** and **Secondary Tags** from the drop-down lists.

#### Result:

For example, if you want to search for blocks that apply to a Bachelor of Arts degree, with a major in Art, select **Degree** and **BA - Bachelor of Arts** from the **Block type** and **Block value** drop-down lists. Select **ART - Art** from the **Major 1** drop-down list.

2. If you want to locate all the blocks that contain errors, select the **Only blocks with errors** check box.
3. Click **Search**.

## Results

The list of all blocks that match the search criteria is displayed.

### Search results

Updated: March 25, 2022

The block search results card lists the blocks that match the search criteria in a table.

Navigate the results list using the pagination controls at the bottom of the list. By default, the search results are sorted by block titles in ascending order. To change the default sort order, click any of the table column headings that represent the block tags, which sorts the search results by that block tag.

To open a block from the search results, click on the title link of the block. You can also delete the selected block.

## Create a block

Updated: September 29, 2023

Create a block by specifying the primary tags for the block and entering block requirements.

### Procedure

1. Click **New Block** on the Scribe home page.

#### Result:

The basic block frame is displayed in the block editor with ## placeholders for comments, the BEGIN and END tags for the block header, and with placeholders for the change log.

**Note:** The template for a new block is defined in the scribe.template.newBlock Shepherd setting.

2. Enter the block header qualifiers and block requirements.

A working knowledge of the Scribe language, rules, and syntax is required to create the block. See the [Scribe Language](#) for more information. See [Reserved Word Definitions](#) for templates, examples, and additional information about the Scribe language. The Reserved Word templates and examples can be highlighted and then dragged and dropped into the editor in most browsers.

3. Enter a title for the block.

#### Result:

This is the title that will be displayed in search results.

4. Click **Block details** and select the primary and secondary tags for the block from the drop-down lists.

These are the identifiers that Degree Works uses to match blocks to students. The **Block type** and **Block value**, **Start catalog year**, and **Stop catalog year** are mandatory tags and you must select values for them.

5. Click **Done**.
6. Click **Save**.

**Result:**

The block is parsed. If there are any errors in the block text or the tags, a notification will be displayed. You must fix these errors before saving the block.

7. To save the block as a local file on your computer, click **Save Local File**.

## Results

After the block is saved, a requirement ID is assigned to the block.

## Saving the block as a file

Updated: March 25, 2022

Save a block on your system as a local file.

When you create a block and click **Save**, the block is saved in the Scribe database. You may want to save a local copy of the block requirements, or you may not want to save an incomplete block to the database. In such cases, you can save the block as a local file.

To save a block as a file on your computer, click **Save Local File** instead of **Save** or **Save As**. Depending on your browser configurations, you will be prompted for a location where the file is to be saved and the Scribe 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.

## Create a block based on an existing block

Updated: March 25, 2022

If an existing block has several elements that you would like to include while creating a new block, you can base the new block on that existing block.

## About this task

### Procedure

1. Search for and open the existing scribe block on which you want to base the new block.
2. Click **Block details** and click **Edit Details**.
3. Enter a new title for the block.

This is the title that will be displayed in search results.

4. If you want to change the primary and secondary tags, select the new tags for the block from the drop-down lists.
5. Click **Done**.
6. Modify the Scribe text for the block in the editor.

A working knowledge of the Scribe language, rules, and syntax is required to create the block. See [Scribe Language](#) for more information.

7. Click **Save** to update the existing block or **Save As** to create a new block.

#### Result:

The block is parsed. If there are any errors in the block text or the tags, a notification will be displayed. You must fix these errors before saving the block.

### Results

After the block is saved, a requirement ID is assigned to the block.

## Edit a block

Updated: March 25, 2022

Edit blocks to modify block tags and scribe text for the blocks.

### Procedure

1. Search for the block you want to edit.
2. Open the block by clicking the block title link in the search results list, or by clicking the block in the list and then clicking **Open block requirement ID**.
3. To edit block tags, click **Block details** then **Edit Details**. Select the new block tag values from the drop-down lists.



---

**Result:**

The **Block details** are displayed. The block details list the block title and the primary and secondary tags.

4. To view additional information about the block, click **History**. This will display information about when the block was created, when it was modified last, and if there are any parse errors in the block.
5. Modify the Scribe text in the editor

A working knowledge of the scribe language, rules and syntax is required to make updates to the block text. See [Scribe Language](#) for more information.

6. After you have made all the required updates to the block, click **Save**.
7. Click **Yes** to confirm the save.

**Results**

The block is parsed and if there are any errors, a notification will be displayed. After the errors have been resolved, the block will be saved.

**Find and replace**

Updated: March 25, 2022

The find and replace functionality in the Scribe editor can be used to quickly find a string of text and to replace it with another string.

For example, if the name of a course has changed, you can use find and replace to search for the old course name wherever it occurs in Scribe blocks and replace those instances with the new course name,

To find and replace text in a block, use the Ctrl-F shortcut to open the find toolbar in the Scribe editor, or Ctrl-H to open the find/replace toolbar. Enter a string of text in the **Search for** field and press Enter, or click the arrow buttons to find the first instance of that string. The string search is case-insensitive and all matching instances of that string will be found.

To replace a string of text, enter the string that you want to find in the **Search for** field and the string that you want to replace it with in the **Replace with** field. Click **Replace** one time to find the first instance of that string and then click **Replace** again to make the text substitution. Alternatively, select **Replace All** from the drop-down list and all instances of the find string will be replaced with the replace string.

**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.

## Parse a block

Updated: March 25, 2022

Parse a block to check for any errors in the block text.

Search for and open the block you want to modify in the editor. After you make the required modifications to the block text, click **Parse**. The block is parsed and if there are any errors, a notification will be displayed. An arrow icon will also be displayed alongside each line of scribe text that contains an error. Move your mouse pointer to hover over the arrow icon for a description of the error and the error number.

**Note:** The parser may not always display the exact nature or location of an error. In addition, errors in the block can cause the parser to indicate error-free parts of the block. These are called propagation errors.

## Delete a block

Updated: March 25, 2022

Delete a block using the Requirement ID or matching block tags.

### Procedure

1. Search for the block you want to delete.
2. If you know the block requirement ID, search for the block using the requirement ID, open the block, and select **Delete** from the More menu.
3. If you searched for the block using the block tags, select the block(s) you want to delete in the search results list and click **Delete**.
4. Click **Confirm** to delete the block.

## Print a block

Updated: March 25, 2022

Printing a block will provide a copy of the block text that can be used for review.

### Procedure

1. Search for the block you want to print.

2. If you know the block requirement ID, search for the block using the requirement ID, open the block and select **Print** from the More menu.
3. If you searched for the block using the block tags, open the block from the search results list and select **Print** from the More menu.

# Scribe Language

Updated: September 29, 2023

The Scribe Language is used to define degree and course requirement rules in Degree Works. The two main tools in this language are rules and qualifiers. The language has its own syntax and Reserved Words and each Reserved Word has rules about its use.

Institutional requirements are entered or scribed using the Scribe function in Controller, resulting in a series of requirement blocks. These blocks are then read by Degree Works.

Blocks are user-defined and are likely to include degree, major, minor, and concentration, but may also include sets of requirements that are unique to an institution. There is no limit to the number of blocks that may be used to construct the requirements for a degree program. Degree Works determines which blocks to use for a particular student by checking standard data on the student record for degree, major, minor, or concentration.

## Basic Course Rules

Updated: September 29, 2023

There are four different formats for writing course requirement rules in the Scribe language.

1. Credits
2. Classes
3. Credits and Classes
4. Credits or Classes

The syntax for the different formats is similar. The most common format is Credits. Some general comments about the course rule are as follows:

1. Each rule optionally ends with a semicolon. The semicolon helps give better messages when parse errors are found but otherwise they are ignored if found.
2. The label must be enclosed in quotation marks and is limited to 200 characters. The label will appear on the student worksheets. It is suggested that the label contain a course title or a description of the requirement, but the label can contain any information the user desires. Leading spaces in a label are not allowed. Although 200 characters are allowed, long labels may have adverse effects on the worksheet. A label tag should be on every label to help ensure exceptions do not get unhooked.
3. Carriage returns and spaces are encouraged to make the rule easier to read in Scribe. These have no effect on the appearance of the worksheets, however.

Examples of the course rule at work:

```
3 Credits in SPAN 101 or FRE 101
Label SPANFREN "Intro. Spanish or Intro. French";
```

The rule will be satisfied by taking either SPAN 101 or FRE 101. A comma [,] may be used in place of the "or".

```
2 Classes in GEOL 200 and 201
Label GEOCAT "Geologic Catastrophes I and II";
```

The rule will be satisfied by taking GEOL 200 and GEOL 201. A plus sign [+] can be used in place of the "and". The discipline code does not need to be repeated when the next course in the rule has the same discipline.

```
5:10 CREDITS IN ENGL 1@, 2@
Label ENGELECT "English Elective";
```

The rule will be satisfied by taking five to ten credits of English at the 100 or 200 level. The colon is used to show a range of credits. An "or" could have been used in place of the comma. The @ symbol is used as a wild card. For example, 1@ means any course number that begins with the numeral one. If the school has course numbers below 100, such as 10 or 11, these courses will also be included in the 1@ range. In this case, it is best to use a range of 100:199 so that courses numbered below 100 do not satisfy this rule.

The Classes and Credits formats will accept courses linked with an "or" or courses linked with an "and", but it is not possible to use both kinds of connectors in a single course rule. For example, this is not allowed:

```
2 Classes in ENGL 101 and (ENGL 102 or 103)
```

In a case like this, it would be necessary to use two Classes requirements, one for an ENGL 101 requirement and one for an ENGL 102 or ENGL 103 requirement:

```
1 Class in ENGL 101
Label ENGCOMP "English Composition";
1 Class in ENGL 102 or 103
Label INTRO "Intro to English Lit or Intro to Fiction";
```

## Block

Updated: March 25, 2022

A block is a set of degree requirements written in the Scribe language.

Requirement blocks are defined by the user. Most colleges will have blocks for degree, major and

minor. A block consists of a block header, followed optionally by one or more rules or remarks, followed by "END.". Example: General Education requirements consist of 6 credits of Math, 6 credits of English, 6 credits of a foreign language, and a maximum of 6 pass-fail credits. In the Scribe language, this requirement is:

```
Begin
MaxPassFail 6 Credits
;
6 Credits In MATH @ Label "Math requirement";
6 Credits In ENGL @ Label "English requirement";
6 Credits In FRE @, GER @, SPA @, RUS @, CHI @
  Label "Language requirement";
End.
```

## Block header

Updated: March 25, 2022

A block header contains the degree requirements that apply to all courses satisfying rules in the block.

Keywords in the block header describe the block as a whole and are applied to the entire block, not to specific rules. The block header is composed of the word BEGIN, followed by optional block qualifiers (such as minimum grade, maximum transfer credits), followed by a semicolon. Example:

```
Begin
MinGPA 2.0
MaxTransfer 30 Credits
;
```

## Block type

Updated: March 25, 2022

A block type is the primary database tag, a characteristic of the block.

It describes what kind of requirements are in the block. Examples:

```
degree=BA
major=MATH
minor=BUS
```

In these examples, the block types are DEGREE, MAJOR, and MINOR respectively. The block type values are BA, MATH, BUS respectively.

## Comment

Updated: March 25, 2022

A comment is a string of free-text following a pound sign or exclamation point.

Comments are entered into a requirements block as an annotation that explains something to the person maintaining the requirements block. Comments are never printed on audit output. They are for internal use only. Example:

```
3 CREDITS IN BIO @; #doublecheck with biology department
```

## Course list

Updated: March 25, 2022

A course list is a list of required classes, where each class is represented by a course key.

Each course key consists of an academic discipline and a course number. Either the discipline or the course number can include a wildcard symbol (@). Example:

```
BIO 100, CHE 115, PHY 1@;
```

## Custom block

Updated: March 25, 2022

A custom block is a requirements block that is defined for non-standard student data or for a specific student.

A custom block either has a database tag of ID for a particular student or has a block type of OTHER. Examples: custom requirements for ID=12938593, requirements block for community service (block type is OTHER and block type value is COMMSERV).

## Database tag

Updated: March 25, 2022

A database tag is a characteristic of the requirement block that is stored in the database.

These characteristics are used to match the requirements to the students. The database tags are: beginning catalog year, ending catalog year, college, concentration, degree, student ID, liberal

learning, major, second major, minor, other, program, school, and specialization. One of these database tags is designated by the user as the primary database tag, the block type. The database tags are entered when the block is saved. They are related to the Scribe language only because the language refers to BLOCKTYPE and BLOCK, which tell the Auditor Engine which primary database tag to use when finding a requirement block for a student.

## Linked block

Updated: March 25, 2022

A linked block is a requirements block referenced in another block (in a BLOCK rule).

The requirements in the linked block are later added by the Auditor Engine in place of the BLOCK rule. A linked block is useful when requirements need to be repeated in multiple blocks. For example, if both the Bachelor of Arts degree and the Bachelor of Science degree have the same foreign language requirements then three Scribe blocks could be created:

```
Begin
120 Credits;
# Bachelor of Arts degree requirements block
6 Credits In ART @;
1 Block (OTHER=FORLANG);
End.
```

```
Begin
126 Credits;
# Bachelor of Science degree requirements block
6 Credits In MATH @;
1 Block (OTHER=FORLANG);
End.
```

```
Begin
MinGrade 2.0;
# Foreign language requirements block
6 Credits In FRE @, GER @, SPA @, RUS @, CHI @;
End.
```

## Non-course

Updated: March 25, 2022

A non-course is a degree requirement, such as a recital, chapel, test, or thesis, that is not a course for which a student registers but whose completion is recorded in the student information system.



Example:

```
1 NonCourse (RECITAL) .
```

## Remark

Updated: March 25, 2022

A remark is a free-text string that describes a requirement using natural language similar to the language of the college catalog.

Example:

```
REMARK "Must take either 6 credits in a foreign language or pass a  
test."
```

## Rule qualifier

Updated: March 25, 2022

A rule qualifier is a Scribe keyword that describes a degree requirement that applies to all courses satisfying a particular rule.

Rule qualifiers are Scribe reserved words that indicate properties of the courses used to satisfy a rule, such as minimum grade or maximum number of transfer courses. Each rule can have zero or more rule qualifiers. Example: MinGrade 2.0.

## Qualifiers

Updated: September 29, 2023

Qualifiers are restrictions placed on a requirement or set of requirements. These restrictions must be met for the block to be completed.

There are two types of qualifiers:

- Header qualifiers that appear in the Header section of the block and pertain to all rules in the block body.
- Rule qualifiers that appear in the Body section of the block and pertain only to the rule with which they are associated.

## Header Qualifiers

Updated: September 29, 2023

Header qualifiers appear in the block Header, which is the area after the BEGIN command and before the first semicolon.

Here are some examples of commonly used Header qualifiers:

```
Share 12 Credits (MINOR)
```

This qualifier states that 12 credits of courses used within this block may also be counted against the rules in a MINOR block. This qualifier only functions if the MINOR block contains course rules that can accept the nonexclusive or shareable credits. For example, suppose a MAJOR block has this Share Header qualifier and also has a rule that states, 4 Credits in ENGL 201. If a MINOR block has a rule that can also use ENGL 201, that course will be applied to the rules in both blocks. Credits are not double counted in the overall degree however.

```
MinRes 30 Credits
ProxyAdvice "At least 30 credits must be taken in residence;"
ProxyAdvice "you have only taken <APPLIED> credits."
```

In the example above, at least 30 credits must be completed in residence for the block to be complete. Classes can be used in place of Credits for this qualifier. There is also a LastRes qualifier, which requires a certain number of last credits or classes to be completed in residence.

```
MaxCredits 3 in PE @
```

Here the MaxCredits qualifier is used to restrict the number of PE credits that will be accepted as satisfying course requirements in the block. If a student takes four 1-credit classes of PE, the fourth 1-credit class will appear in the Over-the-Limit section of the student's worksheet. Scribe also includes a MinCredits qualifier that works analogously.

```
MaxCredits 0 in @ 0@
```

Here the MaxCredits qualifier is used to disallow any credits in any discipline numbered below 100. The wildcard @ is used in two ways. The first wildcard stands for any course discipline. The second wildcard stands for any number.

```
MinGrade 2.5
```

In this example, any class with a grade less than 2.5 will not satisfy the course requirements in the block. Note that this is a stronger qualification than MinGPA because individual classes are disqualified from applying to requirements.

## Rule Qualifiers

Updated: September 29, 2023

Most rule qualifiers appear after the list of courses in a Course rule and before the label.

It is required that a single space separate the last course number in the course list from the first qualifier, and that single spaces separate the qualifiers. For ease of reading, however, it is suggested that scribes use the Enter key to separate the course list from the qualifiers and the qualifiers from each other. The following are some examples of commonly used Rule qualifiers in an easily readable format:

```
3 Credits in PE @
  Except PE 190, 192, 241, 285, 290
  Label "3 Credits of Physical Education Activity Classes";
```

The rule will be satisfied by taking three credits in any PE classes except those listed after the Except qualifier.

```
15 Credits in HIST @
  Including HIST 121
  Label "15 Credits of History";
```

The rule will be satisfied by taking 15 credits in any history classes, but HIST 121 must be included in the 15 credits.

```
15 Credits in ART 100:199,
              DRAMA 150, 160, 170,
              ENGL 161, 2@,
              MUS @,
              PHIL 101, 115, 121, 202,
              SPAN 101:103
  MinSpread 3
  Label "Humanities";
```

The rule will be satisfied by taking 15 credits from the list including at least one class from three different disciplines. A student who took 15 credits of 200-level English classes and no others from the list would not satisfy the requirement because of the MinSpread qualifier. (Note that the colon in the class list designates a range of classes, so 101:103 means 101, 102, or 103.)

```
1 Class in BIOL 101
  ShareWith (THISBLOCK)
  Label "Introductory Biology";
```

Biology 101 satisfies this course rule. Ordinarily, Degree Works uses a default setting of Exclusive (DontShare) for all rule statements. This means that a course can satisfy only one rule. Because of the ShareWith qualifier in this example, however, BIOL 101 can also be applied against another

course requirement within this block. For example, if there is another requirement in another part of the block that includes BIOL @ in the course list, then BIOL 101 will be used to satisfy that rule as well.

## AdviceJump

Updated: September 29, 2023

AdviceJump can be used to allow Degree Works users to link to a web page of information, such as a list or description of courses with specific attributes associated with them. AdviceJump has been set aside as a standard RuleTag value to be used in conjunction with Proxy-Advice.

When used as in the example below, the web audience will be presented with advice that is actually a link to another web page of information.

```
16 Credits in @ (WITH HONR=Y)
    RuleTag AdviceJump="http://myschool.edu/catalog/UpperHonors.htm
1"
    ProxyAdvice "16 Credits of upper-division Honors courses are re
quired."
    Label "Honors Requirement";
```

When your URL is too long for one line, you can break it up like this:

```
5 Credits in @ (WITH Attribute=WRIT)
    Ruletag AdviceJump="http://myschool.edu/catatlog/"
    Ruletag AdviceJump="englishdepartment/writing.html"
    ProxyAdvice "Click here to see classes that meet this requireme
nt."
    Label "Writing Requirement";
```

When the user clicks on the advice text in the audit, they will be transferred to the specified web page for more information. This other web page can provide the user with a list of the current set of courses that will satisfy the requirement along with any pre-requisite and descriptive information that can be used to help advise the student.

The token AdviceJump is case sensitive: ADVICEJUMP, advicejump, and Advicejump will not work. The name of the web page to the right of the equal sign in the scribed rule is usually also case sensitive, but the necessary format of the web page name ultimately depends on the web server your institution is using. Be sure to check with your IT staff to find out the exact name of the files to which you will direct your students.

If your .html files are located somewhere other than where your Degree Works web server files are located, your RuleTag values need to have the full path or relative path to the html page being referenced.

RuleTag can be used to associate any other piece of information with your rules, and the RuleTag code specified can then be trapped for in the stylesheet to give special meaning to certain rules.

Using AdviceJump with RuleTag here is merely one example of the power of RuleTag.

## RemarkJump

Updated: March 25, 2022

Use RemarkJump to allow the user to link to any other place on the internet.

Like AdviceJump, you can use RemarkJump to allow the user to link to any other place on the internet. In this case, however, the rule must have a Remark associated with the rule where the RuleTag is placed.

```
2 Classes in EVA 101 , RORY @, ELENA @
  RuleTag RemarkJump="http://some.place.edu/ontheinternet/anywherei
sfine/"
  RuleTag RemarkJump="support/getmoreinfo.html"
  RuleTag RemarkHint="More info on Gen Ed option-a"
  Label "Gen Ed option A";
Remark "You can click this link to find out more information";
Remark "about this requirement.";
```

In the example above, two RemarkJump RuleTags are used because the URL is so long. The values in quotes in each of the RuleTags are concatenated together.

You can optionally specify a RemarkHint RuleTag to give your user a small pop-up hint when the mouse is placed over the remark. Again, you can specify one or more RemarkHint RuleTags.

To add this same functionality to the header remarks you make use of the HeaderTag instead:

```
Begin
  30 Credits
  HeaderTag RemarkJump="http://some.place.edu/ontheinternet/anywh
ereisfine/"
  HeaderTag RemarkJump="support/getmoreinfo.html"
  HeaderTag RemarkHint="More info on Gen Ed requirements"
;
Remark "You can click this link to find out more information";
Remark "about the General Education requirements.";
```

The AdviceJump can only be used while the rule is not complete. When the rule is complete, the advice and the link go away. With RemarkJump, however, the link is always available because the Remark text appears on the worksheet regardless of whether the rule is complete.

---

## Block types and multi-block programs

Updated: September 29, 2023

There are several block types. The requirements for a degree can be broken up into more than one block using the Block rule.

The Available Block Types are ATHLETE, AWARD, COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, OTHER, PROGRAM, REQUISITE, SCHOOL, SPEC, and ID.

- DEGREE blocks usually specify degree requirements, such as total credits, residency, and GPA. DEGREE blocks usually call in MAJOR blocks and OTHER blocks.
- COLLEGE, CONC, LIBL, MAJOR, MINOR, PROGRAM, SCHOOL, SPEC, and ID blocks can be used to best fit your institution's requirements.
- OTHER blocks are generally used for General Education requirements and Core requirements that are common among several blocks.
- ATHLETE blocks are generally used for Athletic Eligibility requirements. These blocks are called in automatically into the Athletic Eligibility audit formats.
- AWARD blocks are generally used for Financial Aid requirements. These blocks are called in automatically based on the AWARD records found on the database.
- REQUISITE blocks define course prerequisites and co-requisites and are used by the prerequisite checking processes for the Student Educational Planner and Banner registration.

The requirements for a degree can be broken up into more than one block using the Block rule. For example, an Associate of Arts (AA) degree may consist of three sections of requirements: basic requirements, distribution requirements, and general electives. These three sections can be divided into three separate blocks. The main block will have a Block Type tag of DEGREE and will have a Value tag of AA (the Associate of Arts program code). The main (Degree) block, however, will not contain all the requirements necessary for completion of the AA degree. The distribution requirements and general elective requirements will each reside in unique blocks, which will then be called from within the main (Degree) block. These unique blocks will be Block Type OTHER and could have the Value tags of DISTRIB and ELECT respectively. The example below provides an illustration of this setup (you are viewing the Degree block):

```
BEGIN

90 Credits
MinGPA 2.0
;

1 Class in ENGL 111
  Label "English Composition I";

1 Class in ENGL 112
  Label "English Composition II";
```

```
1 Class in MATH 101:238
  Label "Math Requirement";

1 Block (OTHER = DISTRIB)
  Label "Distribution Requirements";

1 Block (OTHER = ELECT)
  Label "Elective Requirements";

END.
```

Notice that the main (Degree) block contains some of the requirements for the degree but not all. The other requirements are scribed in the DISTRIB and ELECT blocks, which have been created as separate blocks. These blocks are called from the main AA Degree block and, as such, are included as part of the AA degree requirements. Also note that there is a Header qualifier of minGPA 2.0 in the main (Degree) block. This qualifier also applies to requirements in the DISTRIB and ELECT blocks even though those blocks may or may not contain the minGPA qualifier.

What are the advantages of using more than one block for a degree? Individual blocks provide more structure for the worksheets. The audit reports include a heading for each block, then contains the credits applied and the GPA for the course requirements in that block. Use of multiple blocks for one degree allows the user to apply Header qualifiers to the contents of an OTHER block without affecting the rest of the degree requirements. For example, in the Associate of Arts degree described above, it may be the case that there are a maximum of five credits of performance and studio classes allowed in the distribution requirements section, but these classes may be allowed without restriction in the general electives section. This restriction for the distribution requirements section can be translated into the Scribe language by including a MaxCredits qualifier in the header of the DISTRIB block, which then restricts these requirements only in the DISTRIB block.

OTHER blocks are also useful in consolidating Scribe language. For example, it may be the case that many degree programs use the same set of elective requirements. These requirements may involve long lists of classes. If the list of classes changes, then all the blocks containing the elective requirements would need to be modified. An alternative solution would be to store the elective requirements in an OTHER block and include Block rules in the blocks that use the requirements. Now the list of courses exists in only one block, so the process of modifying the list is much less time consuming.

## Comments

Updated: March 25, 2022

Any line in a Scribe block that begins with the symbols # or ! is a Scribe comment. Scribe comments are ignored by the parser and the auditor. These comments are for scribes' eyes only.

By convention, many scribes include one or more comment lines at the beginning of a block, which list the database tags for the block. Another use of Scribe comments is as internal documentation. The person who scribes the block knows why the qualifiers and rules were entered in a particular way, but this logic is not always evident to a person reading the block for

the first time. Comments such as, **#This qualifier enforces the maximum three credits in PE rule**, or, **#At least one class must be a laboratory class**, make the block more readable to others.

## Proxy-Advice

Updated: March 25, 2022

Proxy-Advice was designed to remove the default advice on worksheets (what the auditor places based on Scribe code) and replace it with something more friendly and college-related, if needed.

An example could be a thesis requirement using a NonCourse. The advice the auditor gives to students and staff is **Still Needed: 1 NonCourse (Thesis)**. This advice, however, may not use terminology that is common to a particular campus. It would make more sense to a student for the auditor to provide **Still Needed: Your Completed Thesis Statement** advice instead. With Proxy-Advice, we can accomplish this modification. Here is an example of how to write Proxy-Advice.

```
1 Noncourse (Thesis)
  Proxy-Advice "Your Completed Thesis Statement"
  Label "Thesis Requirement";
```

If you need to display a large amount of advice, Proxy-Advice can be used more than one time. Within the quotes, you can include up to 200 bytes of text. If your desired advice exceeds this amount, simply scribe "Proxy-Advice" again on the next line of the block and continue scribing your advice. The text from each line will concatenate.

```
12 Credits in ENGL @ (With Attribute = HONR)
  Proxy-Advice "You need <REQUIRED> credits of English honors cours
ework. You have taken <APPLIED> credits "
  Proxy-Advice "and need <NEEDED> more."
  Label "English Honors Requirement";
```

The **<REQUIRED>**, **<APPLIED>**, and **<NEEDED>** can be used only on course rules and on header Min qualifiers, which are replaced by the number of credits or classes required, applied, and still needed. In the example above, you could use 12 instead of **<REQUIRED>**, but if an exception is placed on this rule to reduce the credits to 10, the advice still says 12. By using **<REQUIRED>**, the advice correctly shows 10 because that is the new number of required credits.

Some other examples of using Proxy-Advice:

```
BEGIN
  40 Credits
    ProxyAdvice "<REQUIRED> credits are required in this major - yo
u still need <NEEDED> credits more."
    LastRes 10 Credits
    ProxyAdvice "The last 10 credits in this major must be taken he
re."
```



```
MinGPA 2.5
  ProxyAdvice "You need a <REQUIRED> GPA in this major - your cur
rent major GPA is <APPLIED>."
;
If (THESIS <> PASSED) then
  RuleIncomplete
    ProxyAdvice "You need to submit a thesis in order to graduate."
    Label "Thesis Requirement";
1 NonCourse (RECITAL >= 4)
  ProxyAdvice "You need to get a score of 4 or more on your recita
1."
  Label "Recital Requirement";
END.
```

## Remarks

Updated: September 29, 2023

Remarks allow for additional narrative descriptions in some worksheet formats.

Worksheet reports include course information and advice for completing rules for a degree.

**Note:** Not all worksheet formats show remarks.

A line containing a remark starts with the reserved word Remark. The remark is enclosed in quotation marks, and a semicolon may optionally follow the closing quote. Each remark line can contain a maximum of 200 characters enclosed in the quotation marks. Successive remark lines will be concatenated in the worksheets. Remarks are not entered in the block Header. A remark entered immediately after the first semicolon of the block will appear in the Header section of the worksheets. Here is an example of how to scribe remarks:

```
1 Class in ENGL 101
  Label "Freshman English Composition I";
Remark "English 101 is a prerequisite for many classes at Ellucian
U "
Remark "Students are advised to take this class as early in their c
ollege "
Remark "careers as possible."
```

Remarks add information to the worksheets, but they come at a price. They make the worksheets more cluttered and harder to navigate. When designing Scribe blocks, this trade-off should be taken into account. Trial and error is a good method for adding remarks to a block: Add a remark to a block, save the block, and then run a worksheet with the new remark; evaluate the results and make the changes as necessary.

## Variable to Text Addition

Updated: September 29, 2023

You can use Degree Works variables to display custom data pulled from your student system into text that appears in the worksheet.

A variable is a tag you place in a Label, Remark, ProxyAdvice, or Display text. The variable you specify is replaced with the value found for the codes set up in UCX-SCR002 and UCX-RPT046 and any goal information you bridged, such as MAJOR, MINOR, and so on.

Label and ProxyAdvice example:

```
If (NumRecitals >= 3) then
  RuleComplete
    Label NUMREC3 "You have completed <NUMRECITALS>"
Else
  RuleIncomplete
    ProxyAdvice "You need at least 3 recitals but currently you"
    ProxyAdvice "have only <NUMRECITALS>"
    Label NUMREC3 "You need at least 3 recitals";
```

You can also add variables to your Remark. For example:

```
Remark "Your academic standing is <ACADSTANDING>"
```

You can also add variables to your Display. For example:

```
MinCredits 12 in @ (With DWAttribute=TROY)
  Display "Your special status is <SPECSTATUS>"
  ProxyAdvice "Some interesting advice"
```

You can have multiple variables in your text. For example:

```
Label "Your major is <MAJOR> and your standing is <ACADSTAND>"
```

Please note that the exact text replaces your variable. No lookup of the code will be done. For example, if your variable is ADMITTERM, the actual admit term code will be used. Your users will see a value of 201801, for example, and not Fall 2018. If you want the description of the code, you need to bridge that description and set up a special RPT046 entry. In this case, you might set up an ADMITTERMDES record that holds the description of the term.

If you specify a variable that does not exist for any particular student, the variable is left as it is. You either need to ensure all students get this code bridged or set up an IF-statement to check to see if they have the code and only then use the variable in the text as shown here using:

```
If (NumRecitals = NODATA)      then
    RuleIncomplete
    ProxyAdvice "You don't have any recitals on record"
    Label NUMREC3A "You need at least 3 recitals";
Else If (NumRecitals >= 3)      then
    RuleComplete
    Label NUMREC3B "You have completed <NUMRECITALS>"
Else
    RuleIncomplete
    ProxyAdvice "You need at least 3 recitals but currently you"
    ProxyAdvice "have only <NUMRECITALS>"
    Label NUMREC3C "You need at least 3 recitals";
```

A special tag, <DWECA>, can be used in your text to show the elective credits allowed in your worksheet. This variable does not have to be set up in SCR002. When you have CFG020 DAP14 **Calculate elective credits allowed** enabled, this variable will get replaced with the number of allowed elective credits the auditor has calculated. This value matches what you see in the Diagnostics Report under the CHECK-ELECTIVE-CREDITS-ALLOWED qualifier in the ECA value. For example, if the student needs to take 12 elective credits to meet the minimum number of credits required by the degree, you might add a Display line as part of your Credits qualifier in the degree block and the <DWECA> will be replaced with 12.

```
Display "You have taken <APPLIED> credits. You are able to take up
to <DWECA> elective credits."
```

## Group Rule

Updated: March 25, 2022

The Group rule allows the scribe to scribe more complicated degree requirements that cannot be handled with simple course lists.

For example, suppose that a degree requires a three-course sequence in either Spanish or Chinese. Review the following course rule:

```
3 Classes in SPAN 101 or 102 or 103 or CHIN 101 or 102 or 103
Label "Language Requirement";
```

This approach will not work because a student would be allowed to take two quarters of Spanish and then one quarter of Chinese or two quarters of Chinese and one quarter of Spanish. The degree requires three quarters in the same language.

The following is another example of the course rule:

```
3 Classes in (SPAN 101 and 102 and 103) or (CHIN 101 and 102 and 10
```

```
3)
Label "Language Requirement";
```

This approach will not parse because Scribe does not allow the Reserved Words "and" and "or" to be mixed within a course rule.

The Group rule is the easiest solution:

```
1 Group in
  (3 Classes in SPAN 101 + 102 + 103
   Label "Three Quarters of Spanish") OR
  (3 Classes in CHIN 101 + 102 + 103
   Label "Three Quarters of Chinese")
Label "LANGUAGE REQUIREMENT";      ##Master Labels are typic
ally scribed in upper case
```

In worksheets, the Group label appears above or to the left of the course requirements just as the BeginSub label does with the BeginSub rule. As with the BeginSub rule, it is suggested that you scribe Group labels in upper-case letters to distinguish them from the Rule labels that are contained within the Group.

## Label Tags

Updated: March 25, 2022

You should always use a label tag on a requirement label. The text in the tag uniquely identifies the rule and allows users to change the label text when needed without causing exceptions to become unhooked.

You can use alpha or numeric label tags. They have a maximum of 20 characters and cannot be duplicated within a block. Here are some examples of how you might use label tags:

```
5 Credits in ENGL 230, LIT 205
Label LITERATURE "English 230 or Literature 205";
```

```
5 Credits in ENGL 230, LIT 205
Label 1 "English 230 or Literature 205";
```

```
5 Credits in ENGL 230, LIT 205
Label A "English 230 or Literature 205";
```

The first example above shows where to place a label tag in reference to the label text of **English 230 or Literature 205**. The label tag of LITERATURE is placed after the reserved word LABEL and before the open quotes of the label text.

### Original Requirement

```
5 Credits in ENGL 230, LIT 205
Label LITERATURE "English 230 or Literature 205";
```

In the next example, the original rule above has changed so that ENGL 232 is now the class that can be taken to satisfy this requirement. The old label text is no longer an appropriate description of the requirement and needs to be changed to reflect the title of ENGL 232. By leaving the label tag as it was originally written, new text can be inserted or deleted from the label text within the quotes. Because the label tag remains the same, any exceptions that might be associated with this rule will not become unhooked. So as long as you never change the label tag exceptions should never become unhooked.

### Changed Requirement

```
5 Credits in ENGL 232
Label LITERATURE "Cross Cultural Connections";
```

Ellucian recommends that you put label tags on all existing requirements that have exceptions on them already; then, after saving the block you can change your label text and save the block again. With label tags in place you can change the rule contents and the label text without worrying about unhooked exceptions. Be sure not to alter the label tags after exceptions have been applied to these rules. However, technically you can change the label tag as long as you don't also change the label text at the same time – but it is safest if you never change the label tag.

## Recommended Tag Values

Updated: September 29, 2023

The best tags are those that reduce the chance of being changed in the future. Recommendation: use an alias or a random value; do not use numbers.

It is a common practice to use sequential numbers of 1, 2, 3, etc for each of your rules. Although you can use numbers as your label-tags it is not a good idea. The problem with this is that when a new rule is inserted or if the rules are reordered it is human nature that the scribe will want to renumber the label-tags so that they are again in sequential order. This, of course, will cause serious problems with your exceptions. The exceptions will either become unhooked or they will apply to incorrect rules.

Another option might be to use the course name for simple requirements.

```
1 Class in MATH 123
Label MATH123 "Algebra II";
```

The problem with this is if this course is renamed to MATH 134 then the scribe will most likely change the label-tag and that will cause all exceptions to be lost.

A better approach is to use a tag like ALGEBRA2 because that is what the requirement is regardless of the specific course on the requirement.

```
1 Class in MATH 123
  Label ALGEBRA2 "Algebra II";
```

This also works well when multiple courses are listed on the requirement such as on this example:

```
6 Credits in HIST 109, 114, 134, 135, 2@
  Label EUROHIST "European History";
```

The best approach is to use a value that you can just about guarantee nobody will change. It is important the tag is never changed regardless of how much the contents of the requirement changes. But of course, if the requirement changes drastically you will have to ask yourself if it is still the same requirement or a new one. If it is the same requirement then you need to keep the same label-tag. To ensure this will not be changed the best option is to use a random-like value that has no meaning whatsoever. For example:

```
1 Class in MATH 123
  Label 9E23J "Algebra II";
```

When the label text or the course list changes this label-tag should not be changed. Label-tags in groups or subsets do not need to have any similarity to the other tags in the same group or subset; each can be a random value that is quite different from the others in the group or subset.

## Qualifier Tags

Updated: March 25, 2022

You should always use a tag on a qualifier. The text in the tag uniquely identifies the qualifier and allows users to change the qualifier when needed without causing exceptions to become unhooked.

You can use alpha or numeric qualifier tags. They have a maximum of 20 characters and cannot be duplicated within a block. Here are some examples of how you might use qualifier tags:

```
Begin
  45 Credits
  MinCredits 12 in ENGL 2@, LIT @
    Tag=MINCRLIT
    ProxyAdvice "You need at least 12 credits in literature"
  MinCredits 6 in ENGL 24@, 25@
    Tag=MINCRWRITING
    ProxyAdvice "You need at least 6 credits in writing"
  MaxCredits 8 in ENGL 3@, 4@
    Except ENGL 343, 344
```

;

In the example above, no tag was placed on the "45 Credits" qualifier because there is only one of these qualifiers and only one such qualifier is allowed in a block so another will never be added. This example also does not have a tag on the MaxCredits qualifier because there is only one of them in this block. Because of this, we will always know which MaxCredits qualifier the exception belongs to – because there is only one of them. However, this is not recommended because the scribe may come along later and add another MaxCredits qualifier. For this reason it is best to add a tag just in case a second MaxCredits qualifier is added in the future. The two tags on the MinCredits qualifiers are essential because there are two of these qualifiers in the header. With the tag in place the scribe is able to change the credits and the list of courses without worrying about exceptions becoming unhooked. The software stores this tag with the exception in the database and will always locate the correct qualifier with the same tag.

In general, the safest thing to do is to add a tag on every header qualifier. If you already have pre-existing blocks without qualifier tags you should add the tags, save the block again, and then make any necessary changes to the qualifiers.

However, because labels are allowed on Min qualifiers you can actually make use of the label tag on your qualifier labels instead of using the qualifier tag.

```
MinCredits 12 in ENGL 2@, LIT @
  Tag=LIT
ProxyAdvice "You need at least 12 credits in literature"
Label LITERATURE "Literature requirement - 12 credits"
```

In this example, the tag of **lit** will be used to ensure the exception applies to the correct location. However, if the Tag=LIT was not scribed on this qualifier the software will locate and use the LITERATURE label tag in its place. This means that if you place a label on a header qualifier and it contains a label tag you don't have to add the qualifier tag. It is important that you have at least one of these tags, however, and having both of them does not hurt either.

You should also place tags on your rule qualifiers:

```
15 Credits in HIST 2@, 3@
  MinCredits 5 in HIST 2@ Tag=HIST2
  MinCredits 3 in HIST 3@ Tag=HIST3
Label HIST "History Requirement";
```

If you don't put a tag on your rule qualifiers the label tag will be used to locate the rule qualifier. However, if the rule has more than one of the same type of qualifiers (MinCredits, MinPerDisc, and so on) then a qualifier tag is essential. Without a qualifier tag in place the exception is assumed to be on the first qualifier of its type. In the example above, if the qualifier tags were missing the label tag of HIST would be used but all exceptions made to either of the MinCredits qualifiers would be placed on the first of the MinCredits qualifiers. If your rule contains just one qualifier of a particular type it is fine to rely on the label tag. But again, it is best to always add a qualifier tag.

## Change of Catalog Year

Updated: March 25, 2022

The auditor can attempt to find the corresponding requirement in the new block of the same type/value if you make sure like-requirements in the set of blocks have the same label-tag or qualifier tag.

Exceptions are tied to the block ID of the block on which the exception was originally placed. The auditor uses the block ID to find the correct block on which to place the exception. This is a problem when a student changes catalog years, however, because a new block with a different block ID will be pulled into the audit. For example, an exception may have been originally placed on a MAJOR=CHEM block with block ID RA000123. At some point later the student's catalog year changes. When a new audit is run the student may still get a MAJOR=CHEM block but the new one for the new catalog year may now be RA000456. The exceptions for the student were originally placed on block RA000123 so now the exceptions will be lost. Though, not necessarily.

The auditor can attempt to find the corresponding requirement in the new block of the same type/value if you make sure like-requirements in the set of blocks have the same label-tag or qualifier tag. For example, in these MAJOR=CHEM blocks there is a "Chemistry Elective" requirement. If you want any exception made to one requirement to apply to the corresponding requirement in the other block when a student changes catalog years then use the same label-tag on the requirements. The label-tag can be an alias like CHEMELECTIVE – like this:

```
9 Credits in CHEM 2@, 3@
  Label CHEMELECTIVE "Chemistry Elective";
```

By using the same label tag you are now tying the exception to a MAJOR=CHEM block with a label-tag of CHEM-ELECTIVE and not to a specific block ID.

You can use numbers or letters as your label tag as the example below shows. As long as the label tag in one block matches the label tag of the corresponding rule in the block of the same type and value the software will know that the exception can be applied to the matching rule. Label tags of "CHEMELECTIVE" or "15" are equally valid label tags. However, it is strongly recommended that you use text-based alias values like CHEMELECTIVE instead of numbers because it is very easy to lose track of which number is the alias for which rule and it is too easy for users to renumber the tags to keep them in order. Using an alias increases the likelihood that corresponding rules in like-blocks will have the same value.

```
9 Credits in CHEM 2@, 3@
  Label 15 "Chemistry Elective";
```

Be sure to set the UCX-CFG020 DAP14 **Apply exception to a similar block** flag to Y to use this feature or to N if you do not want the auditor to try to find like-requirements in new blocks. However, you may find that using a numeric label-tag like "15" above is not trustworthy because it is too easy for the wrong rule in another MAJOR=CHEM block to also have "15"; perhaps the corresponding rule in the other block instead has a label-tag of "13". If you do chose to use numeric label tags, you may want to set the UCX-CFG020 DAP14 flag to "A" instead. This allows exceptions to be placed on the corresponding rule in another block of the same type but only when the label-tag or qualifier tag contains at least one alphabetic character (A-Z). Using the "A"



option you are telling Degree Works to only do this when your rules and qualifiers have aliases.

## Change of Major

Updated: September 29, 2023

As with Change of Catalog Year, exceptions are tied to the block ID of the block on which the exception was originally placed.

It is common for students to change majors wherein the new major is quite similar to the old major. One example of this is when a student is assigned an intended major before they are accepted into the declared major. For example, the student may be assigned the pre-nursing major before they are assigned the real nursing major. Another example might be when a student moves from the marketing major to the business major wherein both share many of the same requirements. When this change of major occurs you may want the exceptions that were applied to the first major to apply to the similar second major.

For exceptions to apply to the new major the auditor needs to find a rule in the new block that has the same rule or qualifier tag as that on the original block. Simply using rule tag names that are identical between major blocks allows the exceptions to apply to the new block. The auditor knows that you consider two blocks to be similar based on the UCX-STU023 Similar major field. For more information, see the [UCX-STU023 Student Major Codes](#) topic.

Additionally, the UCX-CFG020 DAP14 **Apply exception to similar block** flag must be set to Y or A for the auditor to attempt to find a similar block in the student's audit. However, the auditor takes this step only if the same major with a different catalog year is not found.

## Course Title as the Label

Updated: March 25, 2022

For single course requirements, it is common to use the course title as the label text.

For example:

```
1 Class in ENGL 101
  Label KLJEA "College Composition I";
1 Class in ENGL 102
  Label AWFLK "College Composition II";
```

The issue arises when the title of the course is changed. You now need to find all of the places where this title is used and make the change. This often happens on many courses in each new catalog. This can be quite a chore.

To avoid the maintenance headache you may want to use the <COURSETITLE> tag inside of your label like this:

```
1 Class in ENGL 101
  Label KLJEA "<COURSESETITLE>";
1 Class in ENGL 102
  Label AWFLK "<COURSESETITLE>";
```

This tag tells the parser to use the title of the first course listed as the label. When course title changes occur in your student system you need to bridge the changed courses to Degree Works and reparse all of your blocks using DAP16 in Transit. After this point, any new audits run will show the new titles in the labels.

You may also choose to show the course credits as part of the label. For this you can use the `<COURSECREDITS>` tag in the label with or without the `<COURSESETITLE>` tag.

```
1 Class in ENGL 101
  Label KLJEA "<COURSESETITLE> - <COURSECREDITS> credits";
1 Class in ENGL 102
  Label AWFLK "<COURSESETITLE> - <COURSECREDITS> credits ";
```

The `<COURSECREDITS>` will not pull in both the starting and ending range values for a variable credit course; only the starting credit value is used. If your course might be a variable credits course then you may not want to use `<COURSECREDITS>`. If you scribe "`<COURSECREDITS> credits`" and it turns out that the course is only a one credit course then you will see "1 credits". For this reason, it is best not to use `<COURSECREDITS>` for courses that might be single credit courses. If the course is worth just a single credit, the text will come out as "1 credits" so you may want to consider that situation when setting up your labels.

You may list more than one course when using `<COURSESETITLE>` and `<COURSECREDITS>`, but only the title and credits from the first course will be used. This is a valid situation where you may want to list more than one course:

```
1 Class in MATH 122, {Hide BUSN 132}
  Label IOWUER "<COURSESETITLE>";
```

Here you are hiding the second course so there is no reason for you to want to show both course titles.

The parser does allow you to scribe multiple courses like this but again, only the title from the first course will be pulled into the title.

```
1 Class in HIST 209, 211
  Label OPQER "<COURSESETITLE>";
```

The parser also allows you to use a wildcard or range as the first course but when you do this, no title will exist.

```
1 Class in HIST 2@
  Label 34OPI "<COURSESETITLE>";
```

When this occurs, the label will appear as <COURSE TITLE> on the worksheet. This same label will appear in the worksheet if the first course listed is not found to be a valid course in the rad\_course\_mst.

When a substitution or any other exception is applied, the label will not get changed to reflect the new course; the title and credits associated with the course that was originally scribed is what will be used in the label.

When equivalences are processed by the parser the title of the new course will be used. For example, you may have scribed this:

```
1 Class in SCR 123
  Label LIV5MANU0 "<COURSE TITLE>";
```

If the equivalence table has SCR 123 equated to FUT 123 the parser will change the rule to use FUT 123 and the title that gets placed in the label will be for FUT 123 and not SCR 123.

This substitution of the label title and credits will occur only if your UCX-CFG020 DAP13 Validate Courses flag is enabled.

## Max Header Qualifiers

Updated: March 25, 2022

When the auditor attempts to enforce the maximum number of credits specified on a header qualifier one of the steps includes a check for marginal credits.

This is when the auditor finds that the removal of a class from the block will bring the credits applied to the qualifier below the maximum specified. The class is considered marginal because it is bordering the maximum: keeping the class exceeds the maximum while removal of the class brings us below the maximum.

It might be the case that this marginal check causes the auditor to not choose the right classes according to what you believe should happen. If you want the auditor to skip this marginal check, you can add a DECIDE operator as follows:

```
MaxCredits 9 (Decide = SOMEKEY) in ENV 115, MATH 198, @ @ (With Attribute=SCI)
```

If the DECIDE key is in UCX-SCR045 then the auditor will perform the checks following the settings in the UCX-SCR045 record. If the DECIDE key is not in the UCX then the marginal step is still skipped but no decisions will be made based on those flags. However, what will happen instead is that the auditor will first start comparing the match levels (aka priority) of each class on the rules in the block. If no decision is made, because the classes being compared have the same match level, then the auditor will continue on with other comparisons such as credits and total fits etc. The SOMEKEY, as shown above, can be anything in this second scenario but we recommend using something meaningful like PRIORITY so that it means something to you.

---

## Max Zero

Updated: September 29, 2023

Often you want to completely exclude certain classes from the audit. To do this you can specify zero credits or classes in a Max qualifier in the degree block.

Classes matching the qualifier will be placed in the over-the-limit section and will not count. For example, to exclude all remedial classes below 100-level you can do this:

```
MaxClasses 0 in @ 0@ # throw out classes with course numbers starting with a 0
```

You can do the same thing for classes with certain attributes:

```
MaxCredits 0 in @ (With Attribute=DEVL) # throw out developmental classes
```

However, if there are some classes you want to keep you can use Except list:

```
MaxClasses 0 in @ (With Attribute=DEVL) # throw out developmental classes
Except MATH @ # but keep the MATH developmental classes
```

Furthermore, specifying 0 in your degree qualifiers like this also ensures that failed classes will also not count in the GPA. Failed classes that match your zero qualifiers will be sent to over-the-limit and will not be placed in insufficient and therefore will not count in any GPA calculations.

However, if your policy is to only throw out the passed classes but to count the failed classes in the GPA you can certainly exclude those failed classes like this:

```
MaxClasses 0 in @ (With Attribute=DEVL) # throw out developmental classes
Except @ (With DWPassed=N) #ignore the failed devl classes
```

This actually applies to any class that was targeted to go to the insufficient section, not just failed classes.

### Warning!

If you have a Max 0 qualifier within an IF-statement that is testing a course, the auditor will skip the Max qualifier when it does this evaluation. This is because we have not yet resolved this IF-statement to TRUE or FALSE – it is a timing issue.

This qualifier will put all XYZ classes into over-the-limit but it will not put the failed XYZ classes into over-the-limit – these failed classes will end up in the insufficient list.

```
If (MATH 123 was Passed) then
  MaxClasses 0 in @ (With Attribute=XYZ)
```

However, this will work fine because the IF-statement has been resolved already. The WHO failed and passed classes will end up in over-the-limit:

```
If (Major = CHEM) then
  MaxClasses 0 in @ (With Attribute=WHO)
```

## Sharing

Updated: September 29, 2023

The Scribe language provides two reserved words, DontShare (aka Exclusive), and ShareWith (aka NonExclusive), that can be used to control how the auditor will apply courses to rules.

ShareWith can be a block header qualifier or a rule qualifier. DontShare can be a rule qualifier or a block qualifier in blocks that have ShareWith as a block qualifier.

The auditor applies courses taken by a student against requirements for the degree sought by the student. The requirements come from the requirement blocks created in Scribe using the Scribe language. When applying courses to rules, the auditor default is to use a course one time to satisfy only one rule. In other words, each course is applied to a rule in an exclusive fashion.

The Scribe language provides two reserved words, DontShare (aka Exclusive), and ShareWith (aka NonExclusive), that can be used to control how the auditor will apply courses to rules. ShareWith can be used to tell the auditor that a particular rule or block of rules should not be treated exclusively so that classes can be shared between requirements. Any course applied to a ShareWith block or rule can also be applied to additional blocks and rules. All rules in a block with a ShareWith block qualifier are nonexclusive unless qualified with the DontShare rule qualifier. ShareWith can be a block header qualifier or a rule qualifier.

DontShare indicates that the credits or classes applied by the auditor towards satisfying a rule cannot be used to satisfy exclusive requirements in other blocks or in other rules in the same block. A course applied to a DontShare rule is applied only to that rule (if it is the best fit) and to no other exclusive rule. All rules in an exclusive block are exclusive unless qualified with the ShareWith rule qualifier. DontShare can be a rule qualifier or a block qualifier in blocks that have ShareWith as a block qualifier.

Example of an exclusive block with all exclusive rules

```
BEGIN
  36 Credits                               #By default, all rules in this block are
exclusive                                  #unless the rule Qualifier ShareWith is
;                                          used
  12 Credits in SOC 300:499;              #this rule is exclu
```

```

sive
12 Credits in PSY 300:499;           #this rule is exclu
sive
12 Credits in HIST 300:499;         #this rule is exclu
sive
END.

```

#### Example of a sharing block with all sharing rules

```

BEGIN
  36 Credits
  ShareWith (ALLBLOCKS)      # ALL rules in this block can share unl
ess the
;                             # rule Qualifier DontShare is used
12 Credits in SOC 300:499;    #this rule can shar
e
12 Credits in PSY 300:499;    #this rule can shar
e
12 Credits in HIST 300:499;   #this rule can shar
e
END.

```

#### Example of a sharing block with one exclusive rule

```

BEGIN
  27 Credits
  ShareWith (ALLBLOCKS)      # All rules in this block can share unl
ess the
;                             # rule qualifier DontShare is used
9 Credits in RIO 100:299;    #this rule can sha
re
9 Credits in CHE 100:299;    #this rule can sha
re
9 Credits in PHY 100:299 DontShare ;    #this rule cannot
share
END.

```

#### Example of an exclusive block with one nonexclusive rule

```

BEGIN
  12 Credits                  # By default, all rules in this block ar
e exclusive
;                             # unless the rule qualifier ShareWith is
used
4 Credits in SOC 300:499;     #this rule cannot s
hare
4 Credits in PSY 300:499 ShareWith (MAJOR);    #this rule can shar
e

```

```
4 Credits in HIST 300:499;
hare
END.
```

```
#this rule cannot s
```

The ShareWith reserved word must be followed by a scope. The scope indicates to which blocks the courses can be applied again. The auditor will apply a course to as many ShareWith rules as possible and to ONE exclusive rule. The scope defines where the auditor looks for the rules to which the course can be applied. The scope can be one of the following:

Scope	Auditor applies course to
ALLBLOCKS	rules in this block and rules in all blocks
COLLEGE	this block or rule and rules in the COLLEGE block(s)
CONC	this block or rule and rules in the CONC block(s)
DEGREE	this block or rule and rules in the DEGREE block(s)
LIBL	this block or rule and rules in the LIBL block(s)
MAJOR	this block or rule and rules in the MAJOR block(s)
MINOR	this block or rule and rules in the MINOR block(s)
OTHER=value	this block or rule and rules in the OTHER=value block(s)
PROGRAM	this block or rule and rules in the PROGRAM block(s)
SCHOOL	this block or rule and rules in the SCHOOL block(s)
SPEC	this block or rule and rules in the SPEC block(s)
THISBLOCK	rules in this block

If the scope is COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, OTHER, PROGRAM, SCHOOL, or SPEC then the auditor determines that a block is within the scope by checking the block's primary database tag against the scope. For OTHER, the auditor also checks the primary database tag's value (e.g. OTHER=GENED). For ALLBLOCKS the primary database tag is not checked -- all the blocks being audited for the student are within the scope. For THISBLOCK the primary database tag is not checked -- only rules in the current block are within the scope.

The number of credits or classes optionally specified after the DontShare or ShareWith reserved word is used as an upper limit by the auditor. For example, if the rule qualifier is ShareWith 10 Credits (Major) then the auditor will apply the same courses to this rule and rules in the Major block up to the limit of 10 credits. If the number of credits/classes is not specified after DontShare or ShareWith then all of the credits/classes applied to the rule can be applied by the auditor to the scope. The courses applied to the rule that exceed the number of credits/classes associated with the DontShare rule qualifier are treated nonexclusively by the auditor. If ShareWith is followed by a number of credits/classes then the courses applied to the rule that exceed that number are treated exclusively by the auditor. In other words, the remaining credits/classes, are treated the opposite of the qualifier used.

Here are some examples:

```
BEGIN
  36 Credits                               #By default, all rules in this block are
```

```

exclusive
;                               #unless the rule qualifier ShareWith is used
REMARK "Example of an exclusive block with one partially sharing rule";
REMARK "and one totally sharing rule";
9 Credits in SOC 300:499
    Label "I don't like to share";                               # this rule is totally exclusive
9 Credits in PSY 300:499
    ShareWith 6 Credits (MAJOR)
    Label "I will share some credits";                             # sharing 6 credits, 3 credits exclusive
9 Credits in MAT 300:499
    ShareWith (MAJOR)                                             # this rule can share
    Label "I will share all of my credits";
9 Credits in HIST 300:499
    Label "I don't like to share";                               # this rule is totally exclusive
END.

```

```

BEGIN
    36 Credits
    ShareWith (ALLBLOCKS)    # All rules in this block share unless the
;                               # rule qualifier DontShare is used
REMARK "Example of a nonexclusive block with one partially exclusive rule";
REMARK "                               and one totally exclusive rule";
9 Credits in BIO 100:299;    #this rule is totally sharing
9 Credits in CHE 100:299;    #this rule is totally sharing
9 Credits in PHY 100:299
    DontShare 6 Credits;    #6 credits exclusive, 3 credits sharing
9 Credits in MAT 100:299 EXCLUSIVE;    #this rule is totally sharing
END.

```

The auditor applies the student's courses to the requirements in a series of passes through the rules. In the initial pass, the courses are applied to all the rules where they might fit. In the second pass, the auditor determines the "best fit" and removes courses that do not fit when rule qualifiers, block qualifiers, and exclusivity are applied. Unless ShareWith is specified, each course can be applied to only one rule.

If a course matches more than one rule, the auditor checks each rule to see if it can share. If both rules to which a course applies are exclusive, the auditor determines the best fit by examining the



options for each rule. A rule with multiple courses listed has more options than a rule with only one course listed. A rule with wildcards has more options than a rule with ranges. A rule with ranges has more options than a rule with specific course keys (course key = discipline + course number). The auditor will apply the course to the rule with the fewest options as long as the total number of credits/classes for the rule are not exceeded and rule qualifiers are not violated.

If a course matches more than one rule and one or both rules are sharing then the auditor checks the scope of the rules. If they are in the scope specified with ShareWith then the course is applied to both rules. If not, then both rules are treated as if they are exclusive and the course is removed from one of the rules according to the "best fit" algorithm described above.

If two courses match the same rule but only one of the courses can be applied in order not to exceed the number of credits/classes then the auditor determines which course to remove. First it will remove courses that fit another rule. If a tie still exists, it will remove the course that was listed with a wildcard or range, keeping the exact fit (course key with no wildcards or ranges). If a tie still exists, it will use the tiebreak configuration settings that indicate the institution's preferences (keep the oldest term, most recent term, best grade, most credits, highest course number, etc.) If there is still a tie then the auditor arbitrarily removes one of the courses.

The actions of the auditor can be explained best by examples.

## Example 1: Exclusive Block with no Sharing Rules

Updated: March 25, 2022

Exclusive Block with no Sharing Rules.

```
BEGIN
  24 Credits
;
2 Classes in HIST 100 + 115;           #this rule is EXCLUSIVE
6 Credits in HIST 200:299;           #this rule is EXCLUSIVE
3 Credits in HIST 310, 340;          #this rule is EXCLUSIVE
9 Credits in HIST @;                 #this rule is EXCLUSIVE
END.
```

The student being audited has taken (and passed) HIST 100, HIST 115, HIST 250, HIST 280, HIST 310, and HIST 340. Each course is worth 3 credits. In an initial pass through the student data and the requirements, the auditor temporarily assigns each course to every rule that includes the discipline and course number. The results of the initial pass are not seen by the end-user but are shown below as an example of how the auditor works.

Rule	Courses Applied in Initial Pass
2 Classes in HIST 100 + 115;	HIST 100, HIST 115
6 Credits in HIST 200:299;	HIST 250, HIST 280
3 Credits in HIST 310, 340;	HIST 310, HIST 340
9 Credits in HIST @;	HIST 100, HIST 115, HIST 250, HIST 280, HIST 310, HIST 340

The auditor then checks rule qualifiers, block qualifiers, and exclusivity. For exclusivity, the auditor determines if a course can be applied to more than one rule. This is done by checking if the rule is exclusive or sharing. In this example, all the rules are exclusive. Therefore, each course can be applied to only one rule.

The auditor removes courses that were initially applied to more than one rule. The determination of where to keep the course is based on a combination of factors: the number of credits/classes specified, rule qualifiers, block qualifiers, and best fit. The results of the second pass are shown below, and because this example has no rule or block qualifiers that could change the results, are the results the end-user would see on an audit report.

Rule	Courses Applied
2 Classes in HIST 100 + 115;	HIST 100, HIST 115
6 Credits in HIST 200:299;	HIST 250, HIST 280
3 Credits in HIST 310, 340;	HIST 310
9 Credits in HIST @;	HIST 340

The auditor did not apply any course to more than one rule. A choice was made between HIST 310 and HIST 340 for rule 3. In this example, it does not matter which course was chosen but assume that both courses were taken during the same term, HIST 310 had the better grade, and the tiebreak configuration is to keep the highest grade.

## Example 2: ShareWith Block with no DontShare Rules

Updated: March 25, 2022

ShareWith Block with no DontShare Rules.

```

BEGIN
  21 Credits
  ShareWith (ThisBlock)
;
1 Class in SOC 100;           #this rule is sharing
6 Credits in HIST @, SOC @;  #this rule is sharing
3 Credits in BIO 125;        #this rule is sharing
9 Credits in CHE @, MAT @, BIO @; #this rule is sharing
END.
```

The student being audited has taken (and passed) HIST 115, SOC 100, BIO 125, BIO 150, CHE 200, and MAT 250. Each course is worth 3 credits.

Rule	Courses Applied in Initial Pass
1 Class in SOC 100;	SOC 100
6 Credits in HIST @, SOC @;	HIST 115, SOC 100
3 Credits in B10 125;	BIO 125
9 Credits in CHE @, MAT @, BIO @;	BIO 125, BIO 150, CHE 200, MAT 250

After the auditor checks exclusivity and applies the best fit algorithm, the results of the audit are:

Rule	Courses Applied
1 Class in SOC 100;	SOC 100
6 Credits in HIST @, SOC @;	HIST 115, SOC 100
3 Credits in BIO 125;	BIO 125
9 Credits in CHE @, MAT @, BIO @;	MAT 250, BIO 150, CHE 200

SOC 100 is applied to both rule 1 and rule 2 because of the ShareWith block qualifier and both rules are within the scope of THISBLOCK. BIO 125 was not applied to rule 4 because there were sufficient courses to satisfy the rule without reusing BIO 125 and there was nowhere else for MAT 250, BIO 150, or CHE 200 to fit.

### Example 3: ShareWith Block with a DontShare Rule

Updated: March 25, 2022

ShareWith Block with a DontShare Rule.

```

BEGIN
  24 Credits
  ShareWith (ALLBLOCKS)
;
REMARK "General Education block";
1 Class in ENG 100 DontShare;           #this rule is exclusive
3 Credits in ENG 100:120 DontShare;     #this rule is exclusive
6 Credits in HIST @, ENG @;             #this rule is sharing
3 Credits in BIO 125;                   #this rule is sharing
9 Credits in CHE @, MAT @, BIO @;       #this rule is sharing
END.

BEGIN
  36 Credits
;
REMARK "English Major Block";
12 Credits in ENG 100:299;               #this rule is exclusive
24 Credits in ENG 300:499;               #this rule is exclusive
END.
```

The student being audited has taken (and passed) HIST 115, ENG 100, BIO 125, BIO 150, CHE 200, and MAT 250. Each course is worth 3 credits.

Rule	Courses Applied in Initial Pass
REMARK "General Education block";	
1 Class in ENG 100 DontShare;	ENG 100

Rule	Courses Applied in Initial Pass
3 Credits in ENG 100:120 DontShare;	ENG 100
6 Credits in HIST @, ENG @;	HIST 115, ENG 100
3 Credits in BIO 125;	BIO 125
9 Credits in CHE @, MAT @, BIO @;	BIO 125, BIO 150, CHE 200, MAT 250
REMARK "English Major Block";	
12 Credits in ENG 100:299;	ENG 100
24 Credits in ENG 300:499;	

After the Auditor checks exclusivity and applies the best fit algorithm, the results of the audit are:

Rule	Courses Applied
REMARK "General Education block";	
1 Class in ENG 100 DontShare;	ENG 100
3 Credits in ENG 100:120 DontShare;	
6 Credits in HIST @, ENG @;	HIST 115, ENG 100
3 Credits in BIO 125;	BIO 125
9 Credits in CHE @, MAT @, BIO @;	MAT 250, BIO 150, CHE 200
REMARK "English Major Block";	
12 Credits in ENG 100:299;	
24 Credits in ENG 300:499;	

Keeping in mind that each course can be applied one time to an exclusive rule and as many times as possible to sharing rules, ENG 100 was applied exclusively to rule 1 and also to rule 3.

## Example 4: Exclusive Block with a ShareWith Rule

Updated: March 25, 2022

Exclusive Block with a ShareWith Rule.

```

BEGIN
  24 Credits
;
2 Classes in HIST 100 + 115;           #this rule is exclusive
6 Credits in HIST 200:299;           #this rule is exclusive
3 Credits in HIST 310, 340           #this rule is sharing
  ShareWith (THISBLOCK);
9 Credits in HIST @;                 #this rule is exclusive
END.
```

The student being audited has taken (and passed) HIST 100, HIST 115, HIST 250. HIST 280,

HIST 310, and HIST 340. Each course is worth 3 credits.

Rule	Courses Applied in Initial Pass
2 Classes in HIST 100 + 115;	HIST 100, HIST 115
6 Credits in HIST 200:299;	HIST 250, HIST 280
3 Credits in HIST 310, 340	HIST 310, HIST 340
ShareWith (THISBLOCK);	
9 Credits in HIST @;	HIST 100, HIST 115, HIST 250, HIST 280, HIST 310, HIST 340

After the Auditor checks exclusivity and applies the best fit algorithm, the results of the audit are:

Rule	Courses Applied
2 Classes in HIST 100 + 115;	HIST 100, HIST 115
6 Credits in HIST 200:299;	HIST 250, HIST 280
3 Credits in HIST 310, 340	HIST 310
ShareWith (THISBLOCK);	
9 Credits in HIST @;	HIST 310, HIST 340

HIST 340 is applied to both rule 3 and rule 4 because of the ShareWith qualifier on rule 3.

## Example 5: Exclusive Block with a ShareWith (CONC)

Updated: March 25, 2022

Exclusive Block with a ShareWith (CONC).

```

BEGIN
  24 Credits
;
REMARK "History Major block"
2 Classes in HIST 100 + 115;           #this rule is exclusive
6 Credits in HIST 200:299;           #this rule is exclusive
3 Credits in HIST 310, 340           #this rule is sharing
  ShareWith (CONC);                  # with CONC
9 Credits in HIST @;                 #this rule is exclusive
END.

BEGIN
  12 Credits
;
REMARK "American History Concentration block"
6 Credits in HIST 250, 260, 280;       #this rule is exclusive
6 Credits in HIST 310, 315, 320:325;   #this rule is exclusive

```

END.

The student being audited has taken (and passed) HIST 100, HIST 115, HIST 250, HIST 280, HIST 310, and HIST 340. Each course is worth 3 credits.

Rule	Courses Applied in Initial Pass
REMARK "History Major block"	
2 Classes in HIST 100 + 115;	HIST 100, HIST 115
6 Credits in HIST 200:299;	HIST 250, HIST 280
3 Credits in HIST 310, 340	HIST 310, HIST 340
ShareWith (CONC);	
9 Credits in HIST @;	HIST 100, HIST 115, HIST 250, HIST 280, HIST 310, HIST 340
REMARK "American History Concentration block"	
6 Credits in HIST 250, 260, 280;	HIST 250, HIST 280
6 Credits in HIST 310, 315, 320:325;	HIST 310

After the auditor checks exclusivity and applies the best fit algorithm, the results of the audit are:

Rule	Courses Applied
REMARK "History Major block"	
2 Classes in HIST 100 + 115;	HIST 100, HIST 115
6 Credits in HIST 200:299;	
3 Credits in HIST 310, 340	HIST 310
ShareWith (CONC);	
9 Credits in HIST @;	HIST 340
REMARK "American History Concentration block"	
6 Credits in HIST 250, 260, 280;	HIST 250, HIST 280
6 Credits in HIST 310, 315, 320:325;	HIST 310

HIST 310 was applied to one sharing rule in the major block and to one exclusive rule in the concentration block. HIST 250 and HIST 280 were applied to the concentration block rather than the major block because both rules are exclusive and the concentration block had fewer options.

## Example 6: ShareWith on a Subset or Group

Updated: March 25, 2022

When you place ShareWith on a subset or group it is the same as placing the ShareWith on each of the rules in the subset or group.

```
3 Groups in
  (1 Class in @ (With Attribute=WA)
    Label WRITA "Writing A") OR
  (1 Class in @ (With Attribute=WB)
    Label WRITB "Writing B") OR
  (1 Class in @ (With Attribute=WC)
    Label WRITC "Writing C")
ShareWith (THISBLOCK)
  Label WRIT "WRITING";
```

When you place **ShareWith** on a subset or group it is the same as placing the **ShareWith** on each of the rules in the subset or group. The example above essentially acts like this:

```
3 Groups in
  (1 Class in @ (With Attribute=WA)
    ShareWith (THISBLOCK)
    Label WRITA "Writing A") OR
  (1 Class in @ (With Attribute=WB)
    ShareWith (THISBLOCK)
    Label WRITB "Writing B") OR
  (1 Class in @ (With Attribute=WC)
    ShareWith (THISBLOCK)
    Label WRITC "Writing C")
  Label WRIT "WRITING";
```

## StandAloneBlock

Updated: September 29, 2023

Using **StandAloneBlock** greatly simplifies the issue of sharing and dramatically reduces the work the auditor has to do to determine which classes can or cannot be shared.

When a block is sharing with all other blocks you should use **StandAloneBlock** as the header qualifier instead of listing a series of **ShareWith** qualifiers. **StandAloneBlock** is perfect for the special GPA or upper-division blocks that are used to create a special list of classes that are applying to all of the other blocks. Some schools allow the major, minor, concentration and gen ed blocks to share all classes with each other; using **StandAloneBlock** in each of these blocks is a better approach than using **ShareWith**.

## Subset Rule Types

Updated: March 25, 2022

Scribe uses two types of Subset rules: the **BeginSub** rule type and the **Group** rule type.

The **BeginSub** rule is useful for presenting a collection of course rules as a unit. For example,

some degrees divide course rules into categories such as, Core Requirements, Technical Specialty Courses, General Education Requirements, etc. The courses in each of these categories can be gathered together in a subset using the BeginSub and EndSub keywords. Each subset provides an additional label, which serves as a description of the subset category. It is also possible to attach qualifiers to a subset rule. These qualifiers apply to all the course rules listed in the subset. As a simple example, suppose the core requirements for a degree consist of English 111 and English 112, and the grade for each of these classes must be 2.5 or higher. The course rules for these classes can be gathered in a subset as follows:

```
BeginSub
  1 Class in ENGL 111
    Label "English Composition I";
  1 Class in ENGL 112
    Label "English Composition II";
EndSub
  MinGrade 2.5
  LABEL "COMMUNICATION SKILLS REQUIREMENTS"; #This label in upper
case intentionally
```

In this example we only had to use the MinGrade qualifier one time instead of two times (one time on each rule). The subset qualifier option can be very helpful when there are many rules that have to be qualified and many that do not.

Although the BeginSub label follows the course rule in the block layout, the label will be placed above the course rules in the worksheet layout. The individual rules that make up the subset will be indented underneath the subset label on the worksheet.

As a matter of style and to make worksheets easier to interpret, Ellucian suggests that BeginSub labels be written in upper-case letters. Labels for rules contained within subsets should be written in mixed case to distinguish them from the BeginSub label.

## Course List Order

Updated: March 25, 2022

It is a good idea to list the courses on your rules and qualifiers in alphabetic order by discipline and in numeric order for the course numbers.

The reasons for doing this are outlined in the sub-sections that follow.

## Ordering for Readability

Updated: March 25, 2022

When users view the rule advice on the worksheet it is best if the courses are listed in order by discipline and course number.



This is especially important when the rule contains a long list of courses. You want a user to quickly and easily see that a particular course is or is not in the list. Take the rule below as an example. Because the courses are listed in sorted order, it is very easy for the user to see that GEOG 2023 is not in this list simply by looking at the section of GEOG courses.

```
Still needed: 1 Classes in ARTS 2100, BIOS 2010, CSCI 1130, 1210, 1
301, 2150, 2610,
                                ENG 2000, GEOG 2011, 2300, MATH 1060, 11
13, 2110, 2250,
                                2260, 2300H, 2310H, 2400, 2400H, 241
0, 2410H, 2500, 2700,
                                PHIL 2500, 2500H, PHYS 1111, 1112, 1211,
1212, 1311,
                                STAT 2000, 2100H
```

## Ordering for Maintainability

Updated: March 25, 2022

Listing courses in sorted order makes maintaining requirements much easier.

When you are maintaining requirements, it makes your job much easier if you previously listed the courses in sorted order. Much like the user viewing the worksheet, you can quickly see what courses are or are not in the rule when the courses are sorted. Furthermore, some scribes like starting the courses for a new discipline on a new line. This does not affect the output on the worksheet but does make your job much easier as your eyes have to do much less work.

```
1 Classes in ARTS 2100,
              BIOS 2010,
              CSCI 1130, 1210, 1301, 2150, 2610,
              ENGG 2000,
              GEOG 2011, 2300,
              MATH 1060, 1113, 2110, {Hide 2200,} 2250, 2260, 2300H,
2310H,
              2400, 2400H, 2410, 2410H, 2500, 2700,
              PHIL 2500, 2500H,
              PHYS 1111, 1112, 1211, 1212, 1311,
              STAT 2000, 2100H
```

## Ordering for Performance

Updated: September 29, 2023

When the courses are in sorted order the auditor can save time when applying classes to rules and qualifiers.

```

1 Classes in ARTS 2100,
               BIOS 2010,
               CSCI 1130, 1210, 1301, 2150, 2610,
               ENGG 2000,
               GEOG 2011, 2300,
               MATH 1060, 1113, 2110, {Hide 2200,} 2250, 2260, 2300H,
2310H,
               2400, 2400H, 2410, 2410H, 2500, 2700,
               PHIL 2500, 2500H,
               PHYS 1111, 1112, 1211, 1212, 1311,
               STAT 2000, 2100H

```

The trick the software uses is to stop comparing when it knows it will not find any more matches. In the example above, if the class being applied is GEOG 2023 the auditor will now stop when it sees GEOG 2300 because 2300 is greater-than 2023. If the auditor was trying to apply CHEM 1104, it would stop when it encountered ENGG 2000 because ENGG is greater than CHEM. Generally speaking, the auditor will now attempt half as many matches when the rules are sorted – which is a great time savings.

Your course lists do not have to be sorted. Classes will still apply even if you do not sort your lists. The parser checks the rules/qualifiers when you save the block and it determines if the list is sorted. If the list is sorted, it communicates this information to the auditor so that it can make use of this fact.

## Complex Scribing

Updated: March 25, 2022

Sometimes you need to scribe complex requirements. This section provides some examples.

```

# No classes allowed over 10 years old
MaxClasses 0 in @ (With DWAge>10)

```

```

# No classes allowed over 10 years old but allow ANTH classes to be
older than 10 years
MaxClasses 0 in @ (With DWAge>10 and DWDiscipline<>ANTH)

```

```

# No classes allowed over 10 years old but allow ANTH 145 to be old
er than 10 years
# ANTH 145 does not fit either With statements so it will not be co
unted.
# ENGL 145 does fit into the first With so it will count.
# ANTH 123 does fit into the second With so it will count.
MaxClasses 0 in @ (With DWAge>10 and DWDiscipline <>ANTH),
                 @ (With DWAge>10 and DWCourseNumber <>145)

```

```
# In this GenEd block, CHEM students can only share 12 credits with
their minor
# All other students can share unlimited credits with the minor
If (Major = CHEM) then
    ShareWith 12 Credits (Minor)
Else
    ShareWith (Minor)

# We want the classes from both concentrations included in our
# our major GPA - if the student has 2 concentrations
If (NumConcs = 2) then
    2 BlockTypes (CONC)
    Label "Concentrations for this major"
else
    1 BlockType (CONC)
    Label "Concentration required";

# In the CHEM block, share all with the GenEd block only if
# chemistry is the student's first major;
# otherwise, only share 12 credits with the GenEd
If (1stMajor = "CHEM") then
    ShareWith (GenEd)
Else # must be the student's 2nd or 3rd major - only 12 credits all
owed
    Share 12 Credits (GenEd)

# Waive the humanity elective if the student has 2 majors or 2 mino
rs
If (NumberOfMajors >= 2 or NumberOfMinors >= 1) THEN
    RuleComplete
    Label "Gened elective waived because of double-major or additio
nal minor"
Else
    10 Credits in ENGL @, WRIT @, HUM @
    Label "Humanity elective";

# Allow students to take MATH 104 but do not show it in the advice
5 Credits in MATH 184, {Hide MATH 104 (WITH DWTerm<"1981") }
    Label "Math Requirement";
# Must be taken here but don't show that fact in the advice
5 Credits in HIST 184 (WITH Hide DWResident = Y)
    Label HIST184 "History Requirement";

# If the student passed a math class within the last 5 years then...
If (MATH 101:104 (With DWAge < 5) was Passed)
    <something>
# You can also use Taken, Failed, Inprogress, Transferred and Found
```

```
# 15 of last 30 must be taken in residence
LastRes 15 of 30 Credits

# We are defining "in residence" as DWResident or any class with SE
# - which are those transferred from our sister college
LastRes 15 of 30 Credits in @ (With DWResident=Y or Attribute=SE)

# Only 1 class should apply and we don't want a 2 credit (transfer
class usually) to apply
# Only a 3 or 4 credit class should apply here. We are hiding the W
ith to simplify the advice
1 Class in BIO 2@ (With Hide DWCredits >= 3)
  Label BIO2 "Biology Requirement";

# Passfail classes can apply or have a greater grade than 2.0
6 Credits in HIST 106 (With DWPassfail=Y or DWGrade>=2.0) +
  HIST 107 (With DWPassfail=Y or DWGrade>=2.0)

# Using BeginIf/EndIf allows you to scribe multiple rules and add r
emarks.
# You can also use BeginElse/EndElse on the ELSE portion.
If (Attribute = HONR) Then
  BeginIf
    5 Classes in HIST 100:199
    Label HIST100A "Honors: History 100 level classes";
    2 Classes in HIST 200:299
    Label HIST200A "Honors: History 200 level classes";
    Remark "As an honors student you need to take 5 classes"
    Remark " at the 100 level and 2 at the 200 level"
  EndIf
Else # not honors
  BeginElse
    3 Classes in HIST 100:199
    Label HIST100B "History 100 level classes";
    1 Classes in HIST 200:299
    Label HIST200B "History 200 level class";
  EndElse

# The course discipline is a reserved word; use quotes to allow it
to be parsed
6 Credits in "ELSE" 101, 102
```

# Elective Credits Allowed (ECA) Calculations in Degree Works

Updated: September 29, 2023

Use Degree Works to calculate and report on the credits applied and credits not applied to a degree program.

## Scribing Considerations for Banner Student Course Program of Study, Banner Satisfactory Progress Financial Aid, and Athletic Audits

Updated: September 29, 2023

Degree Works can provide information to other applications that must know the enrolled classes applied to a degree program.

Financial Aid eligibility calculations and athletic eligibility require that students make satisfactory academic progress and enroll in classes that apply towards their programs of study. In other words, the courses must apply on their degree audits to degree requirements including total credit requirements. The Banner Financial Aid Module can interact with Degree Works to gather information from student audits needed to determine financial aid eligibility. There are two different calculations.

Elective Credits Allowed Calculation	Process used to move information to Banner
Banner Student Course Program of Study	Uses the What-if API to move information from audits to Banner.
Banner Satisfactory Academic Progress	Uses BAN62 in Transit to move information from Degree Works to Banner.

Both financial aid processes and Athletic Eligibility Audits use Elective Credits Allowed calculations to help the auditor determine whether a course satisfies a degree requirement.

To enable Elective Credits Allowed, set **Calculate Elective Credits Allowed** to Y in UCX-CFG020 DAP14 using Controller.

When you use Degree Works to calculate ECA (assuming that **Fall Thru Count in Overall** is set to Y in UCX-CFG020 DAP14), the only change you see on the Student View is a change to the calculation used for the credits progress bar if **Fall Thru Count in Overall** is set to Y in UCX-CFG020 DAP14. With ECA turned on, credits identified as not needed for the degree are not included in credits applied to the credits bar. Additionally, the outcome of the calculation appears on the Diagnostics Report. In the Electives section, courses are marked as ECA-OK and ECA-Overflow and the ECA calculation and summary appears in the degree block.

Courses in non-required blocks are considered elective hours, even though they are not marked as such. Courses in non-required blocks may be treated the same as courses marked ECA-

Overflow. If a student's required blocks allow a student 20 elective credits and the student has a 30 credit non-required minor, 10 of the credits in the minor will not be considered in the student's program of study because the student does not have to complete the minor.

Scribing determines whether a block is required or non-required. If a block is optional, the block is non-required. A block may be optional because it requires 0 credits in the header, or because the block has an optional header qualifier. More importantly, the way the audit pulls in majors and minors determines whether a major or minor block is required or non-required. If there is no major or minor call (using Blocktype), all majors and minors are non-required. If the starting block (typically, the degree block) contains a rule of 1 Blocktype(Major), the first major is required and all other majors are non-required blocks. Any block called in from a non-required block is considered non-required blocks. If you want the majors to be required, you must explicitly call in the correct number of majors. For example, if a student is a double major and the starting block contains a rule of 2 Blocktypes(Major), both majors are required and all courses in the major block (or called in from the major block) are considered courses in the student's program of study. The courses in required blocks are not treated as electives. The ECA summary displays which blocks are required. Similar logic applies to minors and concentrations.

In addition to determining whether the program of study requires courses in both major blocks, Degree Works behaves differently if you scribe 1 Blocktype(Major) versus 2 Blocktypes(Major). In both cases, both majors are pulled into the student's audit. In the first case, the rule is marked complete if the student has completed one major and not the other. In the second case, the rule is not marked complete until the student has completed both majors. In both cases, the degree block is not marked complete until both majors are completed.

A special tag, <DWECA>, can be used in your scribing to show the elective credits allowed in your worksheet. For more information, see [Variable to Text Addition](#).

## Scribing Considerations for ECA Calculations

Updated: March 25, 2022

Review of specific Scribing strategies that help calculate credits required to complete a degree program.

### Credit Totals

Updated: March 25, 2022

All blocks called in from the degree block must have a Credits qualifier.

If you do not want credit totals to be evaluated as part of the regular audit, you can use Pseudo at the end of the Credits qualifier. By using Pseudo, no advice will be given in the worksheet for the Credits qualifier. If you want total classes evaluated, you must specify classes and credits. If you use classes, you cannot use Pseudo. If you do not include credit totals, you will see errors in the ECA section of the Diagnostics Report.

If you have a block that can be satisfied with 0 credits by an exam or non-course (see example) that has to be called in from the degree block, you still need a Credit Pseudo or you will have errors on the ECA calculation. Degree Works, however, does not allow 0 Credits Pseudo. You can,

however, use 0:1 Credits Pseudo which will make the calculations work correctly.

## Sample of a Proficiency Block with no credit check and pseudo credits

```
BEGIN
0:1 Credits Pseudo #A block with 0 Credits Pseudo will not parse
StandAloneBlock
;
  1 Group in
    (1 Noncourse (WPE=Y)
      Label 1 "Writing Proficiency Requirement - Met") OR
    (1 Class ENGL 2113
      Label 2 "Intermediate Composition and Grammar")
    LABEL 3 "WPE or INTERMEDIATE COMPOSITION";
END.
```

A better solution is to not call the block in from the Degree block, but to call it in from another block that is called into the Degree block. Doing it this way does not require a credit header on the proficiency block

## Degree Block (Starting Block)

Updated: March 25, 2022

The degree block or starting block for the audit contains the total number of required credits.

The degree block cannot contain any course rules because courses in the starting block are not included in the required credit calculations. To use ECA calculations correctly, an audit must consist of a minimum of two blocks where the actual required courses are included in the block called in from the degree or starting block. For example, a school using Program As Degree, requires 120 credits and puts all of its required courses with the exception of 9 elective credits into the degree block. The ECA calculation will not include these 9 required elective courses in the degree block.

## Majors

Updated: March 25, 2022

To count all courses in all majors in the student's program of study, you must call in the explicit number of majors in the degree block.

See the example below:

```
If (NumMajors=1) then
  1 Blocktype (Major)
  Label "Required Major"
```

```
Else If (NumMajors=2) then
  2 Blocktypes (Major)
  Label "Required Majors"
Else If (NumMajors=3) then
  3 Blocktypes (Major)
  Label "Required Majors"
```

To count courses in the first major towards the student's program of study and also towards the courses to satisfy other declared majors as elective courses, you must scribe the major in the degree block as shown in the example below:

```
1 Blocktype (Major)
Label "Required Major"
```

## Minors

Updated: March 24, 2023

If a degree requires a minor, you must scribe a rule explicitly for the minor to consider the minor as a required block.

If a minor is optional and the student's program of study considers courses that satisfy the minor requirements, you must scribe explicitly for the minor. If the program of study includes only one minor, scribe for one minor block. If the program of study can include multiple minors, scribe for the exact number of minors using NumMinors.

If minors are optional and should not be considered in the student's program of study, do not explicitly scribe for minors in a rule. Allow Degree Works to find the minor blocks so they are non-required blocks.

See the example below:

```
If (NumMinors=1) then
  1 Blocktype (Minor)
  Label "Required Minor"
Else If (NumMinors=2) then
  2 Blocktypes (Minor)
  Label "Required Minors"
Else If (NumMinors=3) then
  3 Blocktypes (Minor)
  Label "Required Minors"
```

To count courses in the first minor towards the student's program of study and also towards the courses to satisfy other declared minors as elective courses, you must scribe the minor in the degree block as shown in the example below:

```
1 Blocktype (Minor)
```



Label "Required Minor"

## Concentrations

Updated: March 25, 2022

Use the same scribing considerations for concentrations as you would for minors.

Concentrations work exactly like minors so the same scribing considerations apply for concentrations. If the scribe looks for explicit concentration codes and only one concentration counts in the student's program of study, use a nested if statement because Degree Works considers any concentration that you add to a rule as a required block.

## UCX-CFG020 DAP14 Fall-through courses count in the overall GPA

Updated: September 29, 2023

When using ECA, review the **Fall-through courses count in overall GPA** flag in CFG020 DAP14 using Controller. When set to Y Degree Works calculates the number of elective credits allowed for the degree.

If you set UCX-CFG020 DAP14 **Fall-through courses count in the overall GPA** to N, none of the fall-through credits count towards the overall GPA or credits applied towards the degree. Because of this, any classes ending up in fall-through will be marked ECA-OVERFLOW. When the CFG020 flag is set to Y, the auditor attempts to figure out how many credits are allowed to be in fall-through. The auditor uses sharing to help determine the number of elective credits allowed. If minors and second majors can be used as electives, make sure that elective blocks allow saving with the optional minors and majors. Be careful if some programs require minors that should be considered as courses in the program of study. For example, if a student's program of study counts only courses in the first major, include ShareWith (Major, Major<>1STMajor) in the elective block. You must also consider appropriate ShareWith qualifiers for minors and concentrations depending on the school's policies for including courses in the program of study. ECA calculations, however, only take sharing into account when courses are shared on the audit. Consequently, the ECA calculation changes as more shared courses appear on a student's audit.

Furthermore, when the CFG020 flag is N, you do not need to add Pseudo or add a Credits qualifier to any block that doesn't need one. The calculated ECA value is always 0 because no elective credits are allowed.

**Note:** It is okay if you have scribed elective rules. This information applies only to elective classes ending up in fall-through.

## Undecided/Undeclared or Missing Minors, Majors and Concentrations

Updated: March 25, 2022

When a student does not declare part of their program, their curriculum record does not include a major, minor, or concentration.

For ECA calculations to work correctly, the audit must include the total number of required credits

with no courses rules in the starting block. Until the student declares all elements of their program of study, the student is allowed more ECA-OK credits. The calculations are correct because the audit is less specific than it is after the student declares all the elements of their program of study. Clients should be aware of how the ECA calculations treat these students to ensure that the calculations are consistent with their financial aid policies or athletic eligibility rules.

## Scribe Rules and ECA required Block Calculation examples

Updated: March 25, 2022

Examples and explanations of Scribe rules and ECA calculations.

Table 1: Student Data for all examples

GoalCode	GoalValue	GoalSeq
Major	EDUC	0001
Major	BIOL	0002
Conc	PSCG	0001

Table 2: Scribe Rule examples

Example	Block	Value	Credit Qualifier	Block Calls
1	Degree	BS (143)	120 Credits	1 Block (Other=CORE)  1 Blocktype (Major)
	Other	CORE (589)	45 Credits Pseudo	(none)
	Major	EDUC (564)	75 Credits	(none)
		BIOL (673)	55 Credits	1 Block (CONC=PSCG)
	Conc	PSCG (1104)	(none)	(none)
2	Degree	BS (143)	120 Credits	1 Block (Other=CORE)
	Other	CORE (589)	45 Credits Pseudo	(none)
	Major	EDUC (564)	75 Credits	(none)
		BIOL (673)	55 Credits	1 Block (CONC=PSCG)
	Conc	PSCG (1104)	(none)	(none)
3	Degree	BS (143)	45 Credits Pseudo	1 Block (Other=CORE)  2 Blocktype (Major)
	Other	CORE (589)	75 Credits	(none)
	Major	EDUC (564)	55 Credits	(none)
		BIOL (673)	(none)	1 Block

Example	Block	Value	Credit Qualifier	Block Calls
				(CONC=PSCG)
	Conc	PSCG (1104)	45 Credits Pseudo	(none)

## Example 1 Output

```

CHECK-ELECTIVE-CREDITS-ALLOWED (Qualifier added by auditor per CFG020 flag)
Overall = 120 (Credits required by degree block)
Blocksum = 120 (Sum of credits required in all "required" blocks; excludes credits in blocks included in other required blocks)
Shared = 4 (Credits shared between required blocks)
ECA = 4 (Credits allowed/needed in fall-through and non-required blocks)
RequiredCreditsApplied = 7 (Credits applied to required blocks)
NonrequiredCreditsApplied = 3 (Credits applied to non-required blocks)
Overflow = 12 (Excess fall-through credits)
CreditsAppliedTowardsDegree = 13 (Credits applied minus overflow )
BlocksRequired = RA000564 RA000689
BlocksNotRequired = RA000673
BlocksRequiredIncluded = RA001104

```

Explanation: Biology is an optional block (non-required) because it is sequence 2 and requires only 1 blocktype (major) for the degree.

The concentration (#1104) is called in from a non-required block and is not required.

## Example 2 Output

```

CHECK-ELECTIVE-CREDITS-ALLOWED (Qualifier added by auditor per CFG020 flag)
Overall = 120 (Credits required by degree block)
Blocksum = 45 (Sum of credits required in all "required" blocks; excludes credits in blocks included in other required blocks)
Shared = 0 (Credits shared between required blocks)
ECA = 75 (Credits allowed/needed in fall-through and non-required blocks)
RequiredCreditsApplied = 4 (Credits applied to required blocks)
NonrequiredCreditsApplied = 6 (Credits applied to non-required blocks)
Overflow = 0 (Excess fall-through credits)
CreditsAppliedTowardsDegree = 25 (Credits applied minus overflow )
BlocksRequired = RA000689
BlocksNotRequired = RA000564 RA000673 RA001104
BlocksRequiredIncluded =

```

Explanation: No majors or minors are called into the starting block so all are considered non-required.

## Example 3 Output

```

CHECK-ELECTIVE-CREDITS-ALLOWED (Qualifier added by auditor per CFG020 flag)
Overall = 120 (Credits required by degree block)
Blocksum = 175 (Sum of credits required in all "required" blocks; excludes credits in blocks included in other required blocks)
Shared = 4 (Credits shared between required blocks)
ECA = 0 (Credits allowed/needed in fall-through and non-required blocks)
RequiredCreditsApplied = 10 (Credits applied to required blocks)
NonrequiredCreditsApplied = 0 (Credits applied to non-required blocks)
Overflow = 15 (Excess fall-through credits)
CreditsAppliedTowardsDegree = 10 (Credits applied minus overflow )
BlocksRequired = RA000564 RA000673 RA000689
BlocksNotRequired =
BlocksRequiredIncluded = RA001104

```

Explanation: Both majors are required because the scribe included 2 BlockType(Major) and the concentration is required because it is called from a required block.

## Reserved Word Definitions

Updated: March 25, 2022

The Scribe language consists of Reserved Words that have a special meaning for the degree advisory process. These keywords were chosen to be descriptive of their tasks.

Reserved Words are not case sensitive. Letters within square brackets are optional, but if you use the optional letters listed, use all of them. The number sign (#) and exclamation point (!) are used in the Scribe language to begin free-text comments that are never printed on the output as part of the requirements.

### 1stConc, 1stMajor, 1stMinor, etc.

Updated: March 25, 2022

These reserved words can be used in an if-statement. The 1st through the 9th of each can be used. The auditor checks the student's curriculum data or the data specified as part of a what-if audit.

1stConc, 2ndConc, ... 9thConc

1stMajor, 2ndMajor, ... 9thMajor

1stMinor, 2ndMinor, ... 9thMinor

1stProgram, 2ndProgram, ... 9thProgram

1stCollege, 2ndCollege, ... 9thCollege

1stLibl, 2ndLibl, ... 9thLibl

1stSpec, 2ndSpec, ... 9thSpec

### Template

```
If (1stMajor = XXXX) THEN
```

### Examples

```
# More credits are needed for biology majors
If (1stMajor = BIO) THEN
  15 Credits in BIO 3@
  Label BIO1 "Biology elective for Biology majors"
Else If (1stMajor = CHEM) THEN
```

```
12 Credits in BIO 3@
Label BIO2 "Biology elective for Chemistry majors"
```

### Notes

1. None of these needs to be setup in UCX-SCR002.
2. The `st`, `nd`, `rd`, and `th` are mostly ignored and can go with any number – but one of them must be present. That is, you can say `1ndMajor`, `2rdMinor` or `3stConc` – the parser will process these, as long as there is some suffix present there. An example such as `2xxMajor` would also be valid.
3. The `rad_goal_seq` field on the `rad_goaldata_dtl` determines the order of goal – and not the order of the blocks found in the audit.

## Accept

Updated: March 25, 2022

Same as [Allow \(rule\)](#).

## AllBlocks (header)

Updated: March 25, 2022

This indicates the scope for the `Share/ShareWith` qualifier. This scope signifies that all blocks are to be considered for sharing of classes with the current block.

### Examples

```
ShareWith (AllBlocks)
```

```
Share 10 Credits (AllBlocks)
```

```
ShareWith (AllBlocks)
```

### Notes

1. The sharing of each class will only occur with one of these blocks.
2. Class A may share with block X and class B may share with block Y.

---

## AllBlocks (rule)

Updated: March 25, 2022

This indicates the scope for the Share/ShareWith qualifier. This scope signifies that all blocks are to be considered for sharing of classes with this requirement.

### Template

```
ShareWith (AllBlocks)
```

### Examples

```
6 Credits in BUS 1@
Share 3 Credits (AllBlocks)
Label BUS "Business";
```

```
9 Credits in CHEM @, BIOL @, PHYS @
ShareWith (AllBlocks)
Label SCI "Science";
```

### Notes

1. The sharing of each class will only occur with one of these blocks.
2. Class A may share with block X and class B may share with block Y.

## Allow (rule)

Updated: March 25, 2022

Indicates that a fewer number of credits are allowed to satisfy the course rule. You can use Allow to deal with transfer classes that are not the same number of credits at your school.

### Template

```
n Credits (Allow n:n) in XXX 999
Label xxx "requirement";
```

## Examples

```
5 Credits (Allow 4:5) in HIST 134
Label HISTORY "History";
```

```
4 Credits (Allow 3.5:4) in ART 204
Label SCULPTURE "Sculpture";
```

## Notes

1. Allow and Accept are synonymous.
2. It is suggested that you use Allow because Accept sounds too much like Except, which is another Degree Works feature.
3. Allow is used within a course rule statement to indicate what range of credits the auditor engine will actually use to determine whether or not the rule has been satisfied.
4. The lower number in the range should be less than the credits scribed, but the upper bound on the range must be equal to the credits scribed; the upper bound on the range cannot be more than the credits scribed. For example: 4 Credits (Allow 3:5) is accepted by the parser, but the auditor will not obey this upper bound.
5. The audit advice always hides the allowable, lower-credit value telling the student that a higher number is required. This is useful when requiring a certain number of credits but allowing a lower number of credits if the class was taken at a transfer institution or if the number of credits for the class has changed over catalog years.

## AlwaysShowInAdvice (rule)

Updated: March 25, 2022

This keyword allows certain courses to always show in the course rule advice even if the student has taken the class. This is most useful for repeatable classes wherein the student needs to, or can, take more than one class towards the requirement.

## Template

```
n Credits in XXX 111, 222 AlwaysShow, 999
Label xxx "requirement";
```

## Examples

```
# Always show repeatable class ACCT 105 in the advice as long as
# more classes are needed
3 Classes in ACCT 103, 105 AlwaysShowInAdvice, 112
  Label ACCT "Accounting Requirement";

# RORY 104 can be repeated for credit; needs to be taken twice
6 Credits in RORY 104 AlwaysShow (WITH DWAge < 5)
  Label RORY "Rory Requirement";
```

## Notes

1. When using With, the AlwaysShow keyword comes first.
2. You can also use the lengthy AlwaysShowInAdvice or the shorter AlwaysShow.
3. AlwaysShow applies to the previous course only.
4. It makes no sense for AlwaysShow to be used with HideFromAdvice on any particular course. You can scribe both but the Hide operator takes precedence.

## And (header)

Updated: March 25, 2022

And is a Boolean operator used to connect the number of Classes and Credits and expressions in an IF-statement.

## Template

```
nn Credits and nn Classes
```

## Examples

```
36 Credits and 12 Classes
```

```
40:45 Classes and 120:135 Credits
```

```
If (Major=HIST and Attribute=UBEY) Then
  MinGPA 2.5
```



## And (rule)

Updated: March 25, 2022

And is a Boolean operator used to connect a list of courses or a link between Classes and Credits, can be used in an IF-statement and can combine With expressions.

### Template

```
nn Credits and nn Classes in XXX 998 and 999
Label xxxx "xxxx requirement";
```

### Examples

```
6 Credits and 2 Classes in BIO 100 and BIO 115
Label BIO "Biology";
```

```
3 Classes in CHE 100 + CHE 115 + 115L
Label CHEM "Chemistry";
```

```
12 Credits in BUSN @ (With Attribute=UB and Attribute=EYDA)
Label BUSINESS98 "Special Business Courses";
```

```
If (Major=HIST and Minor=EDU) Then
  1 Block (OTHER=EDUCATION)
  Label EDUC "Education Requirements";
```

### Notes

1. And can connect data conditions in an IF-statement.
2. And can connect Credits and Classes in a course statement.
3. And or plus sign (+) can be used between courses in a list in a course rule. The plus in a course list is equivalent to And. The plus cannot be used in place of And in an If condition or as a connector between Credits and Classes.
4. And cannot be intermingled with Or in a course list. The connectors in a list of courses must all be either And/plus or Or/comma. For example: "BIO 100, CHE 100 + PHY 100" is not valid.
5. If And is used in a course list, then the number of courses listed must match the number of Classes specified. For example, "4 Classes in BIO100 + CHE100 + PHY 100" is not valid. This syntax check is not done if any of the courses contain a wildcard or range. For example, "4 Classes in BIO100:199 + CHE1@ + PHY@" is valid.

## At (header)

Updated: March 25, 2022

At precedes a location code in a course list. It specifies that the course(s) applies to those taken at the specified location.

### Template

```
MaxClasses nn in MUS @ at XXXX
```

### Examples

```
# Max of 3 music classes taken at St. Mary's  
MaxClasses 3 in MUS @ at SM
```

```
# Max of 3 music classes at St. Mary's (preferred solution)  
MaxClasses 3 in MUS @ (With DWLocation=SM)
```

### Notes

1. Location codes must be valid in the institution's code list (UCX-STU576).
2. Only one location can be listed for a course. If multiple locations are needed, then repeat the course, for example, "ENG 100 at SM, ENG 100 at XX". If no location is given for a course, then the Auditor Engine does not check location when matching a student's course to the requirement.
3. You can instead use DWLocation instead of At. For example, MaxClasses 3 in MUS @ (With DWLocation=SM). DWLocation is preferred over using At.

## At (rule)

Updated: March 25, 2022

At precedes a location code in a course list. It specifies that the course(s) applies to those taken at the specified location.

### Template

```
1 Class in XXX 999 at XX}
```

```
Label xxx "requirement";
```

## Examples

```
# ENG 101 at St. Mary's is allowed
1 Class in ENG 100, {Hide ENG 101 at SM,} 103
Label ENGLISH "English";
```

```
# ENG 101 at St. Mary's is allowed (preferred solution)
1 Class in ENG 100, {Hide ENG 101 (With DWLocation=SM)}
Label ENGLISH "English";
```

## Notes

1. Location codes must be valid in the institution's code list (UCX-STU576).
2. Only one location can be listed for a course. If multiple locations are needed, then repeat the course, for example, "ENG 100 at SM, ENG 100 at XX". If no location is given for a course, then the Auditor Engine does not check location when matching a student's course to the requirement.
3. You can instead use DWLocation instead of At. For example, MaxClasses 3 in MUS @ (With DWLocation=SM). DWLocation is preferred over using At.

## Begin

Updated: March 25, 2022

Begin starts a block of requirements.

## Template

```
Begin
```

## Examples

```
Begin
  120 Credits
  MinGPA 2.0
;
```

## Notes

1. The first keyword of each block must be Begin. Only blank lines and comments beginning with "#" are allowed before Begin.
2. For each Begin there is a corresponding End at the end of the requirements block.

## BeginElse

Updated: March 25, 2022

See [If-Then](#).

## BeginIf

Updated: March 25, 2022

See [If-Then](#).

## BeginSub (rule)

Updated: March 25, 2022

BeginSub begins a subset of rules that go together under a parent rule.

## Template

```
BeginSub
  n Credits in XXX 999
    Label xxx "subrule 1";
  n Credits in XXX 999
    Label xxx "subrule 2";
EndSub
  Label xxx "SUBSET LABEL";
```

## Examples

```
BeginSub
  6 Credits in PSY 1@
    Label PSYCH100 "Psychology 100 Level";
  If (Attribute = SPCL) then
    3 Credits in SOC 1@, 2@
```

```
        Label SOC100200 "Sociology 100 or 200 Level"
    Else
        3 Credits in SOC 1@
        Label SOC100 "Sociology 100 Level";
    EndSub
    Label SOCPSY "100 LEVEL PSY AND SOC";

    If (Major = UBEYDA) Then
        BeginSub
            6 Credits in PSY 2@
            Label PSYCH200 "Psychology 200 Level";
            3 Credits in SOC 2@
            Label SOC200 "Sociology 200 Level";
        EndSub
        Label SOCPSY "200 LEVEL PSY AND SOC";
```

## Notes

1. Following BeginSub is one or more rule statements, each ending in a semicolon. Following the last rule in the subset is EndSub. Following EndSub is an optional list of rule qualifiers that will be applied by the auditor engine to the entire subset of rules as if the subset was one rule.
2. BeginSub and EndSub create a subset of rules against which rule qualifiers can be applied and allow a set of rules to be associated with a parent label on the worksheet to show a relationship between the sibling requirements.
3. BeginSub is allowed only at the beginning of a rule or within an If-statement. BeginSub cannot be used within a Group.
4. BeginSub and EndSub cannot be nested. That is, two BeginSub keywords cannot be in the same rule.
5. Allowable rule qualifiers: DontShare, HighPriority, LowPriority, LowestPriority, MaxPerDisc, MaxPassFail, MaxTransfer, MaxSpread, MinGrade, MinPerDisc, MinSpread, ShareWith, NotGPA, ProxyAdvice, SameDisc.

## Block (rule)

Updated: March 25, 2022

The Block rule pulls another requirements block into audit. The credits/classes in the other block are counted in the referencing block's totals. Furthermore, the completeness of the referenced block affects the completeness of the referencing block.

## Template

```
1 Block (Other = xxxxx)
  Label xx "xxxx xxxxx";
```

## Examples

```
1 Block (Other = GENED)
  Label GENED "General Education Requirements";
```

```
If (Attribute = ROTC) Then
  1 Block (Other = ROTC)
    HideRule
    Label ROTC "ROTC Requirement";
```

## Notes

1. The block referenced must already exist before this requirement can be scribed or the parser will give an error.
2. Other is the most commonly used block type used with this requirement. However, you can specify a Major or Minor, etc. block here also, but the block will only be pulled into the audit if that major or minor is already part of the student's curriculum. The Blocktype requirement is more appropriate for Major, Minor, etc. block types.
3. When HideRule is used with the Block rule, the label and advice do not appear in the referencing block on the worksheet but the referenced block itself does.
4. Allowable rule qualifiers: NotGPA, HideRule, ProxyAdvice, RuleTag.

## Blocktype (rule)

Updated: March 25, 2022

The Blocktype rule specifies that a block with the specified type must exist in the audit. The credits/classes in the referenced block are counted in the referencing block's totals. Furthermore, the completeness of the referenced block affects the completeness of the referencing block.

## Template

```
1 Blocktype (xxxx)
  Label xx "xxxx xxxxx";
```

## Examples

```

1 Blocktype (Major)
  Label MAJOR "Major Requirements";

# A minor is required but do not show a label/advice in this block
1 Blocktype (Minor)
  HideRule
  Label MINOR "Minor Requirements";

# If the student has 2 concentrations then include both in this
# block's credits and GPA calculations
If (NumConcs = 2) Then
  2 Blocktypes (Conc)
    Label CONC2 "Concentration Requirements"
Else
  1 Blocktype (Conc)
    Label CONC1 "Concentration Requirement";

```

## Notes

1. Typically, 1 Blocktype is specified, but if multiple blocks of the type are required then numbers other than 1 can be used.
2. BlockType must be followed by one of these block types: College, Conc, Libl, Degree, Major, Minor, Program, School, or Spec.
3. When HideRule is used with the Blocktype rule, the label and advice do not appear in the referencing block on the worksheet but the referenced block itself does appear.
4. When the audit includes multiple majors and multiple concentrations, for example, the auditor attempts to locate the concentration that is associated with the major that is requiring the concentration – through the Blocktype(CONC) requirement. The auditor first checks to see if either the major or the conc block has a secondary tag associating it with the other block. If this check fails to find a connection then the auditor will see if the concentration is attached to the major through the attach fields on the CONC rad\_goalData\_dtl record.
5. Allowable rule qualifiers: HideRule, ProxyAdvice, RuleTag.

## CheckElectiveCreditsAllowed (header)

Updated: September 29, 2023

Used in the starting block to tell the auditor to calculate how many elective credits the student is allowed to take based on the credits required in each of the other blocks. The recommended way to implement this functionality is to set the UCX-CFG020 DAP14 Calculate Elective Credits Allowed flag as it gives more functionality than this qualifier gives.

---

## Template

```
CheckElectiveCreditsAllowed
```

## Examples

```
CheckElectiveCreditsAllowed
```

## Notes

1. Only used in the starting block (usually Degree).
2. Each block must have a Credits qualifier; use Pseudo if the block does not have a real credit limit. Use "0:1 Credits Pseudo" if no credits are represented in the block's requirements.
3. Elective credits allowed (ECA) = Degree credits minus sum of block credits plus waived credits plus shared credits. Example: 120 Degree credits – 40 major credits – 30 minor credits – 30 gen ed credits + 3 waived credits + 9 shared credits = 32 ECA credits.
4. Classes with a WA (waived) rad-class-status are looked up on the rad-course-mst to get the amount of waived credits; these waived credits are added to the ECA total. Note that the Banner extract does not populate the rad-class-status with WA so waived credits will always be zero for Banner schools.
5. As students progress towards a degree their ECA may go up and down depending on how requirements share courses and what classes they takes.
6. The fall-through list is examined using the ECA credit value. If there are more credits in fall-through than allowed, the classes are processed and marked as ECA-OVERFLOW if they exceed the ECA limit. All other fall-through classes are marked at ECA-OK. Completed classes may also be marked as ECA-OVERFLOW, not just the in-progress classes. However, when CFG020 DAP14 **Fall-through courses count in the overall GPA** is N, all fall-through classes are marked as ECA-OVERFLOW because none of these classes count towards the total credits applied to the degrees. The ECA value is thus set to 0.
7. When the ECA limit is exceeded the auditor first removes (marks as OVERFLOW) classes starting at those in the most future term (in case there are preregistered classes) and then moves backwards in time term by term. The auditor will also make as efficient use of credits as possible while still preferring to keep older in-progress classes over more future in-progress classes. However, regardless of credits a completed class will be kept over an in-progress class.
8. The ECA overflow total credits are extracted into the CPA dap\_result\_dtl FALLCRCL record as rad-value4 and rad-number4.
9. The ECA-OK and ECA-OVERFLOW values are extracted onto the CPA dap\_result\_dtl table for FALLCLASSAPPLIED record as rad-value4.



10. This qualifier is created as a normal Qualifier xml node with the attributes and values for Overall credits allowed, Blocksum credits, Waived credits, Shared credits and Overflow credits.
11. The Fallthrough/Class xml nodes are created with a Code attribute containing the ECA-OK or ECA-OVERFLOW values.
12. The standard worksheets do not show any of these values except for the Diagnostics Report. If you want to make use or display any of these ECA values in your worksheet you must localize the xsl files. You may want to create two different fall-through sections, for example.
13. The Calculate Elective Credits Allowed flag in UCX-CFG020 DAP14 is trumped if this qualifier is in place. That is, if you are using this qualifier the CFG020 flag is ignored. The calculation performed when the CFG020 flag is enabled is different from the results of this calculation. When the CFG020 flag is used the auditor checks to see which blocks are required vs optional and makes appropriate calculations. This special check is not performed when this qualifier is used instead of the flag. For more information, see the [UCX-CFG020 DAP14](#) topic.

## Classes (header)

Updated: March 25, 2022

Indicates how many courses are required in the block.

### Template

```
nn Classes
ProxyAdvice "You have taken <APPLIED> classes but still need <NEE
DED> more."
```

### Examples

```
20 Classes
ProxyAdvice "You have taken <APPLIED> classes but still need <NEE
DED> more."
```

```
45 Credits and 15 Classes
ProxyAdvice "You have taken <APPLIED> credits but still need <NEE
DED> more."
```

### Notes

1. Classes cannot be followed by a course list when used in a block header. You may want to use MinClasses if you want to specify a course list.

2. The auditor treats the number of classes specified in the header as a minimum that must be satisfied. If the student takes at least the minimum number of courses, the qualifier will be met.
3. When specifying both a number of courses and a number of credits, connect them using "and" or "or". For example, "9 Classes and 25 Credits" means both conditions must be satisfied. But, "9 Classes or 26 Credits" means either condition can be satisfied to meet the requirement.

## Classes (rule)

Updated: March 25, 2022

Indicates how many courses are required.

### Template

```
3 Classes in XXX @  
Label xxxx "xxx Requirement";
```

### Examples

```
3 Classes in PE @  
Label PE "PE Requirement";
```

```
3 Classes and 9 Credits in MATH @  
Label MATH "Math Requirement";
```

```
4:6 Classes in COMM @  
Label COMM "Communication Requirement";
```

```
1 Class in MUS 101  
Label MUSIC2EARS "<COURSE TITLE> - <COURSE CREDITS> credits";
```

### Notes

1. Classes cannot be followed by a course list when used in a course rule.
2. Classes must follow an integer that indicates the number of courses required. The number of courses can be a range, specified as "integer colon integer". A range is used to indicate that the number of courses that satisfy the requirement varies between a lower and upper bound. The requirement will be met if the number of courses taken by the student is greater than or equal to the lower bound. For example, "2:4 Classes" is satisfied if the student takes 2, 3, or 4, courses. The upper bound of the range of classes indicates that no more than the upper bound number of courses will be used to satisfy the requirement. It is a strict cap.

3. See additional notes under [Credits \(rule\)](#).

## College (header)

Updated: March 25, 2022

When College is used in ShareWith it refers to the College block in the audit. When used in an If-statement it refers to the college on the student's curriculum.

### Examples

```
ShareWith (College)
```

```
If (College = SAFALI) Then  
  MinGPA 2.5
```

### Notes

1. College is used within a block header, only with an If-statement or ShareWith.

## College (rule)

Updated: March 25, 2022

When College is used in ShareWith or a Blocktype it refers to the College block. When used in an If-statement it refers to the college on the student's curriculum.

### Examples

```
1 Blocktype (College)  
  Label CLGBLOCK "College block is required";
```

```
If (College = ENGR) Then  
  3 Credits in MATH 400  
  Label MA400 "Math 400";
```

```
6 Credits in BUS 1@  
  ShareWith (College)  
  Label BUS "Business";
```

## Notes

1. College is followed by a college code except in a Blocktype rule.
2. The college code must be valid in UCX-STU560.
3. College can be used with BlockType, If-statement, or ShareWith.

## CompletedTermCount

Updated: March 25, 2022

This counts the terms where at least one class was taken.

## Template

```
if (CompletedTermCount > nn) then
```

## Examples

```
if (CompletedTermCount > 3) then
  BeginIf
    MinGPA 3.2
    ProxyAdvice "You need a GPA of 3.2 now that you have 3 terms co
mpleted"
  EndIf
```

## Notes

1. This can be used in any audit but is more appropriate to be used in the Athletic Eligibility Audit or the Financial Aid Audit.
2. For Athletic Eligibility Audits, summer terms are excluded and counting starts at the first full-time term.
3. See the [Athletic Eligibility Audit](#) or the [Financial Aid Audit](#) topics for more information on how this is used.

## Conc (header)

Updated: March 25, 2022

When Conc is used in ShareWith it refers to the Conc block in the audit. When used in an If-

statement it refers to the concentration on the student's curriculum.

## Examples

```
ShareWith (Conc)
```

```
If (Conc = SAFALI) Then  
  MinGPA 2.5
```

## Notes

1. When Conc is used within a block header, it is only used with an If-statement or ShareWith.

## Conc (rule)

Updated: March 25, 2022

When Conc is used in ShareWith, Blocktype, or Block, it refers to the Conc block. When used in an If-statement, it refers to the concentration on the student's curriculum.

## Examples

```
1 Blocktype (Conc)  
  Label CONCBLOCK "College block is required";  
  
# Only if the student has the SPAN concentration on their  
# curriculum will this block be used  
1 Block (Conc=SPAN)  
  Label CONCBLOCK "Spanish concentration requirements";  
  
If (Conc = ENGR) Then  
  3 Credits in MATH 400  
  Label MA400 "Math 400";  
  
6 Credits in BUS 1@  
  ShareWith (Conc)  
  Label BUS "Business";
```

## Notes

1. Conc is followed by a concentration code except in a Blocktype rule and is optional in

ShareWith.

2. The concentration code must be valid in UCX-STU563.
3. Conc can be used with BlockType, If-statement, or ShareWith.
4. When Conc is used in a Block rule, the specific concentration block specified will only be included in the audit if the student has that concentration on their curriculum. For this reason, it is rare for use a Block rule with Conc.

## CopyHeaderFrom (header)

Updated: March 25, 2022

CopyHeaderFrom is used to copy header qualifiers from another block. This is best used when you have a set of qualifiers that is used by many different blocks and is changed often. Using CopyHeaderFrom cuts down on block maintenance.

### Template

```
CopyHeaderFrom (RA000nnn)
```

### Examples

```
CopyHeaderFrom (RA000123)
```

```
If (Major = PSY) then
  BeginIf
    CopyHeaderFrom (RA000355)
  EndIf
```

### Notes

1. Rules in the referenced block are not copied – they are ignored. Please see [CopyRulesFrom \(rule\)](#) to copy the rules.
2. The referenced block must be parsed and saved before the calling blocks can be parsed and saved. It is best to parse and save the referenced block and then run DAP16 in Transit to be sure all of the blocks that use CopyHeaderFrom are reparsed.
3. You can use CopyHeaderFrom in an If-statement, but if the block you are copying has more than one qualifier, then you need to enclose the CopyHeaderFrom within BeginIf/EndIf. Always enclose your CopyHeaderFrom in BeginIf/EndIf to be safe.
4. Along with the qualifiers, the header Remarks are also copied. However, if CopyHeaderFrom

is in an if-statement in the header, then the remarks will not be copied.

## CopyRulesFrom (rule)

Updated: March 25, 2022

CopyRulesFrom is used to copy rules from another block. This is best used when you have a set of requirements that is used by many different blocks and is changed often. Using CopyRulesFrom cuts down on block maintenance.

### Template

```
CopyRulesFrom (RA000nnn);
```

### Examples

```
CopyRulesFrom (RA000123);
```

```
If (Major = PSY) then
  BeginIf
    CopyRulesFrom (RA000355)
  EndIf
```

```
BeginSub
  CopyRulesFrom (RA000355)
EndSub
Label COMM "COMMUNICATION REQUIREMENTS";
```

### Notes

1. Header qualifiers in the referenced block are not copied, they are ignored. To copy header qualifiers, see [CopyHeaderFrom \(header\)](#).
2. The referenced block must be parsed and saved before the calling blocks can be parsed and saved. It is best to parse and save the referenced block and then run DAP16 in Transit to be sure all of the blocks that use CopyRulesFrom are reparsed.
3. You cannot use CopyFromRules in a group, but you can use it in a subset. See the cautionary note below about remarks in subsets.
4. You can use CopyRulesFrom in an If-statement but if the block you are copying has more than one rule then you need to enclose the CopyRulesFrom within a subset or within BeginIf/EndIf. Basically, always enclose your CopyRulesFrom in a subset or BeginIf/EndIf to be safe.

However, if the block you are copying has remarks on the rules, you cannot place your CopyRulesFrom in a subset because remarks are not allowed in a subset. Instead use a BeginIf/EndIf.

5. Along with the rules and associated labels, the rule Remarks are also copied.

## Courses (rule)

Updated: March 25, 2022

See [Classes \(rule\)](#).

## Credits (header)

Updated: September 29, 2023

Indicates how many credits are required in the block.

### Template

```
nn Credits
ProxyAdvice "You have taken <APPLIED> credits but still need <NEE
DED> more."
```

### Examples

```
20 Credits
ProxyAdvice "You have taken <APPLIED> credits but still need <NEE
DED> more."
```

```
30.5 Credits
ProxyAdvice "You have taken <APPLIED> credits but still need <NEE
DED> more."
```

```
45 Credits and 15 Classes
ProxyAdvice "You have taken <APPLIED> credits but still need <NEE
DED> more."
```

```
30 Credits Pseudo
```

```
# Use this placeholder of 120 but the credits will really be based
```



```
on the sum of all
# required blocks. This is only valid to use in the starting block.
120 Credits
  HeaderTag DWCredits=DWBlockSum

# Use this placeholder of 30 but the credits will really be based o
n the sum of all
# the rules in this block. This is normally used in non-starting bl
ocks and blocks
# where the credits on the rules may vary based on if-statements.
30 Credits
  HeaderTag DWCredits=DWRuleSum

# Use this placeholder of 92 but the credits will be based on the T
OTALCREDITS
# value that is bridged for the student. TOTALCREDITS must be setup
in UCX-SCR002 but
# of course any name can be used
92 Credits
  HeaderTag DWCredits=TOTALCREDITS
```

## Notes

1. Credits cannot be followed by a course list when used in a block header. You may want to use MinCredits if you want to specify a course list.
2. The auditor treats the number of credits specified in the header as a minimum that must be satisfied. If the student takes at least the minimum number of credits, the qualifier will be met.
3. When specifying both a number of courses and a number of credits, connect them using "and" or "or". For example, "9 Classes and 25 Credits" means both conditions must be satisfied. But, "9 Classes or 26 Credits" means either condition can be satisfied to meet the requirement.
4. You can specify an integer or use a decimal number: Example: 23.5 Credits.
5. Pseudo is used in a block that does not have a strict credit limit. When Pseudo is specified, the credits qualifier will always be satisfied and thus no advice will appear. The auditor is told how many credits this block is worth when calculating the elective credits allowed. Use **0:1 Credits Pseudo** if no credits are represented in the block's requirements. However, when CFG020 DAP14 **Fall-through courses count in the overall GPA** is N, you don't need to add a Credits qualifier in any block. The elective credits allowed logic knows that no elective credits are allowed.
6. See the [HeaderTag](#) topic for more details on using DWCredits to calculate the credits for a block.

## Credits (rule)

Updated: March 25, 2022

Indicates how many credits are required.

### Template

```
3 Credits in XXX @  
Label xxxx "xxx Requirement";
```

### Examples

```
3 Credits in PE @  
Label PE "PE Requirement";
```

```
3.5 Credits in ACCT 2@  
Label ACCT2 "Accounting Requirement";
```

```
3 Classes and 9 Credits in MATH @  
Label MATH "Math Requirement";
```

```
4:6 Credits in COMM @  
Label COMM "Communication Requirement";
```

```
4.5:6.6 Credits in COMM 2@  
Label COMM2 "Communication Requirement";
```

```
1 Credit in MUS 101  
Label MUSIC2EARS "<COURSE TITLE> - <COURSE CREDITS> credits";
```

```
1 Class in MUS 201 (With Hide DWCredits=3)  
Label MUSIC "<COURSE TITLE>";
```

### Notes

1. Credits must follow a number that indicates the number of credits required. The number of credits can be a range, specified as "number colon number". A range is used to indicate that the number of credits that satisfy the requirement varies between a lower and upper bound. The requirement will be met if the number of credits taken by the student is greater than or equal to the lower bound. For example, "10:15 Credits" is satisfied if the student takes at least

10 credits. When advice is shown while the student has taken less than 10 credits the range of credits still needed will appear. For example, if the student has taken 3 credits so far, the advice will say "7:12 Credits". After 10 credits have been applied to this requirement the auditor may apply more credits if those additional credits are not needed elsewhere.

2. When specifying both a number of courses and a number of credits, connect Classes to Credits with And or Or. For example, "3 Classes and 6 Credits" means both conditions must be satisfied. However, both numbers are a minimum. It is possible six 1-credit classes could be used to satisfy this requirement. As long as at least 3 classes and at least 6 credits apply the requirement will be complete. Another example: 1 Class and 3 Credits. If you really want a single 3-credit class to apply then you should scribe it as 1 Class in MATH @ (With DWCredits=3). Note, "3 Classes or 6 Credits" means either condition can be satisfied to meet the requirement.
3. The wildcard (@) can be used as part of the discipline or course number. The wildcard signifies one or more of any alphanumeric characters.
4. A range of course numbers is indicated by separating two course numbers with a colon. The course numbers cannot contain any letters or wildcards. The lower bound (left side) must be less than or equal to the upper bound (right side). For example, "MATH 100:199" is valid; "BIO 100:199L" is not valid, and "SOC 3@:499" is also not valid.
5. The discipline is followed by one course number, many course numbers, or the wildcard. The list of courses is separated by either commas (or) or plus signs (and). Examples: "MATH 100, BIO 115, 120, 120L, PE 100:199, HE @"; "CHE 1@ + 1@L + PHY 100:115"; "FRE 100:299 or SPA 100:299 or GER 100:299"; "GRK 100 and 125 and LAT 100 and 115".
6. When listing multiple courses for the same discipline, the discipline is optionally repeated before the course number. For example, "MUS 100, 115" is equivalent to "MUS 100, MUS 115".
7. Courses in a list must be connected by a logical And or a logical Or. Do not intermingle And and Or in a course list. A logical OR is represented by the literal Or or a comma. A logical AND is represented by the literal And or a plus sign (+). A logical AND can only be used in the first course list that appears in a course rule, not with a course list associated with Including, Except, rule qualifiers, or block qualifiers.
8. Courses are validated against the list of courses bridged from the student system. Only courses that do not contain a wildcard or course number range are validated. The course is validated against the rad\_course\_mst but only if the UCX-CFG020 DAP13 validation flag is enabled.
9. The UCX-CFG020 DAP13 **Wildcard character match** flag controls whether the wildcard matches zero or more characters or one or more characters. When the flag is set to 1 (matching the normal behavior), ENGL 123 would not match the requirement; when the flag is set to 0 (stating that the wildcard matches 0 or more characters), ENGL 123 would match the requirement "1 Class in ENGL 123@". Regardless of the flag's setting, classes such as ENGL 123A, 123B, etc. would always match.
10. Allowable rule qualifiers: DontShare, Exclusive, Hide, HideRule, HighPriority, LowPriority, LowestPriority, MaxPassFail, MaxPerDisc, MaxSpread, MaxTerm, MaxTransfer, MinAreas, MinGrade, MinPerDisc, MinSpread, MinTerm, NotGPA, ProxyAdvice, RuleTag, SameDisc, ShareWith, With.

---

## CreditsAppliedTowardsDegree

Updated: September 29, 2023

Represents the number of credits the student has applied towards the degree. This takes into account the special elective credits allowed discarding the excess fall-through credits.

### Template

```
if (CreditsAppliedTowardsDegree >= nnn) then
```

### Examples

```
-- 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"

# Remind the student about the graduation application - during their senior year
if (CreditsAppliedTowardsDegree >= 100 and GRADAPP=N) then
    RuleIncomplete
    Label GRADAPP "You must apply for graduation"
```

### Notes

1. These credits are automatically calculated for Athletic Eligibility audits.
2. If it is not an athletic audit, these credits will be calculated only if the UCX-CFG020 DAP14 **Calculate Elective Credits Allowed** flag is Y (recommended) or if you have the CheckElectiveCreditsAllowed qualifier in your degree block.
3. CreditsAppliedTowardsDegree should not be used in conjunction with Scribe rules or qualifiers. Doing so can have an effect on the credits applied in the audit or on the ECA calculation, which is part of how CreditsAppliedTowardsDegree is calculated. It would be a circular dependency problem where a rule within an if-statement can affect whether the if-statement itself resolves to true. For instance, you should not use credits qualifiers, credits, or classes rules within an if-statement that evaluates CreditsAppliedTowardsDegree. Generally, it should be used only with things like RuleComplete and RuleIncomplete, as in the documented examples.

## CreditsAttemptedThisAidYear

Updated: March 25, 2022

These credits are taken from the classes on the terms in the aid-year specified. The attempted credits for these classes are summed. Typically, the credits for waived classes are excluded – unless they happen to be bridged with non-zero credits.

### Template

```
if (CreditsAttemptedThisAidYear > nn) then
```

### Examples

```
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";
```

### Notes

1. These credits are automatically calculated for Financial Aid audits.

## CreditsAttemptedThisTerm

Updated: March 25, 2022

These credits are taken from the classes on the terms in the aid-term specified. The attempted credits for these classes are summed. Typically the credits for waived classes are excluded – unless they happen to be bridged with non-zero credits.

### Template

```
if (CreditsAttemptedThisTerm > nn) then
```

## 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";
```

## Notes

1. These credits are automatically calculated for Financial Aid audits.

## CreditsEarnedThisAidYear

Updated: March 25, 2022

These credits are taken from the classes on the terms in the aid-year specified. The earned credits for these classes are summed up.

## Template

```
if (CreditsEarnedThisAidYear > nn) then
```

## Examples

```
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";
```

### Notes

1. These credits are automatically calculated for Financial Aid audits.

## CreditsEarnedThisTerm

Updated: March 25, 2022

These credits are taken from the classes on the aid term specified. The earned credits for these classes are summed up.

### Template

```
if (CreditsEarnedThisTerm > nn) then
```

### Examples

```
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";
```

### Notes

1. These credits are automatically calculated for Financial Aid audits.

## Current

Updated: March 24, 2023

CURRENT translates to the active term on the student's record. Typically this is used with the Financial Aid audit.

## Template

```
MinCredits nn in XXX @ (With DWTerm=CURRENT)
```

## Examples

```
MinCredits 12 in @ (With DWTerm=CURRENT)
```

## Notes

1. Maps to the rad\_term on the rad\_student\_mst.
2. Although CURRENT is typically used on the Financial Aid audit, it could be used on any type of audit in any WITH statement using DWTerm.
3. CURRENT must be all uppercase.

## Decide (header)

Updated: March 25, 2022

Indicates to the auditor how to decide which classes should be removed and which should be kept when the maximum has been exceeded on a block qualifier.

## Template

```
MaxXXXXX nn (Decide=XXXX) in XXX @
```

## Examples

```
MaxCredits 10 (Decide=NEWEST) MATH @, CHEM @
```

```
MaxPerDisc 2 Classes (Decide=HTRMHNUM) in (ENGL, HIST)
```

```
MaxClasses 1 (Decide=LOWTERM) in MATH 112, 133, 134
```

```
MaxTransfer 1 Class (Decide=HIGHNUM) from (UR, CO, OL)
```



```
MaxPassFail 4 Credits (Decide=HIGHGRADE)
```

## Notes

1. Decide is used within a block header only with Max or SpMax qualifiers to indicate how the Auditor Engine should decide which classes to keep when the maximum number of classes or credits has been exceeded.
2. The Decide code must be setup in UCX-SCR045. UCX-SCR045 is much like the UCX-CFG020 TIEBREAK record in that you have up to nine ways to tell the auditor how to distinguish one class as being more valuable than another.

## Decide (rule)

Updated: March 25, 2022

Indicates to the auditor how to decide which classes should be removed and which should be kept when the maximum has been exceeded on a course rule.

## Template

```
nn Credits (Decide=XXXXX) in XXXX @  
Label XXXX "xxx xxxx";
```

## Examples

```
10 Credits (Decide=NEWEST) in ARTH 200:300  
Label ARTHIS "Art History";
```

```
2 Classes (Decide=BESTGRADE) in PHIL 22@, 334  
Label PHIL "Philosophy";
```

```
1 CLASS (Decide=ORDER) in BUS 156, 159, 168, ECON 211, 215, 218  
Label BUSECON "Business or Economics";
```

```
3 Credits (Decide=ORDER) in MATH 123, 112, 145  
Label MATH "Math";
```

## Notes

1. Decide is used within a course rule statement to indicate how the auditor should decide which classes to keep when the maximum number of classes or credits has been exceeded.
2. The Decide code must be in UCX-SCR045. UCX-SCR045 is much like the UCX-CFG020 TIEBREAK record in that you have up to nine ways to tell the auditor how to distinguish one class as being more valuable than another.
3. The Decide keyword is only allowed on course rules; it is not allowed on Groups, Subsets, Blocks, or Blocktypes.
4. ORDER is a special reserved code within the Decide statement. ORDER does not have to exist in UCX-SCR045. It indicates that the auditor should decide which classes to remove and keep based on the order of the classes on the rule. The first classes listed have a higher precedence over those at the end. A class appearing earlier in the rule will be kept on the rule over one appearing later in the rule.

## Degree (header)

Updated: March 25, 2022

When Degree is used in ShareWith it refers to the Degree block in the audit. When used in an If-statement it refers to the degree on the student's curriculum.

## Examples

```
ShareWith (Degree)
```

```
If (Degree = BA) Then  
  MinGPA 2.5
```

## Notes

1. When Degree is used within a block header, it is only used with an If-statement or ShareWith.

## Degree (rule)

Updated: March 25, 2022

When Degree is used in ShareWith it refers to the Degree block in the audit. When used in an If-statement it refers to the degree on the student's curriculum.

## Examples

```
# You could do this but it would be very unlikely because the start  
ing block is usually the degree block  
1 Blocktype (Degree)  
  Label DEGBLOCK "Degree block is required";  
  
If (Degree = BS) Then  
  3 Credits in MATH 400  
  Label MA400 "Math 400";  
  
6 Credits in BUS 1@  
  ShareWith (Degree)  
  Label BUS "Business";
```

## Notes

1. Degree is followed by a degree code except in a Blocktype rule.
2. The degree code must be valid in UCX-STU307.
3. Degree can be used with Blocktype, If-statement, or ShareWith.

## DegreeCreditsRequired

Updated: March 25, 2022

Credits required from the starting block. This is allowed on right-hand-side of an IF-statement and the left-hand side.

## Template

```
if (DegreeCreditsRequired >= nnn) then
```

## Examples

```
if (CompletedTermCount >= 8 And  
  CreditsAppliedTowardsDegree >= 80% of DegreeCreditsRequired) th  
en  
  RuleComplete  
  Label 80a "80% complete by start of 5th year"
```

```
if (TotalCreditsEarned < 150% of DegreeCreditsRequired) then
  RuleComplete
  Label "Credits earned is less than 150% of required"
```

## Notes

1. This is synonymous with CreditsRequired.

## Display (header)

Updated: March 25, 2022

Display can be used to specify additional text to always show for any Min header qualifier.

## Template

```
MinXXXXXX nn Credits
  Display "You have taken <APPLIED> credits xxxx xxxx xx xxxx."
  Proxy-Advice "You still need <NEEDED> credits for xxxx xx xxxxx."
```

## Examples

```
MinGPA 2.5 in MATH @, BIOL @
  Display "You must have a <REQUIRED> GPA in your math and biology
classes."
  Display "Your GPA in these classes is currently <APPLIED>."
```

```
MinRes 30 Credits
  Display "You have taken <APPLIED> credits here at this great scho
ol."
  Proxy-Advice "You have not taken <REQUIRED> credits at State Coll
ege."
```

```
MinCredits 10 in PE @
  Display "You have taken <APPLIED> PE credits so far."
```

## Notes

1. Display is followed by up to 200 characters of text enclosed in quotes.
2. Display can be repeated as many times as needed; the text is appended together.

3. The text will always display, even when the header qualifier is complete.
4. The Display text will appear in the same section where the Remarks appear; the Display text does not appear next to the corresponding qualifier.
5. The <REQUIRED> tag is useful for when an exception is made to the qualifier the new required classes/credits appear in the text instead of the original value.
6. The <APPLIED> tag shows how many classes/credits apply to the qualifier.
7. The <NEEDED> tag shows that many classes/credits are still needed to satisfy the qualifier.
8. Display can be used by itself or in conjunction with ProxyAdvice.
9. Display must come before ProxyAdvice – not after it.
10. Display can be used on the following block header qualifiers: MinGPA, MinRes, LastRes, MinCredits, MinClasses, MinPerDisc, MinTerm, Under, Credits/Classes.

## DontShare (header)

Updated: March 25, 2022

DontShare indicates that sharing cannot exist between this block and those specified. This is equivalent to Exclusive.

### Template

```
DontShare (XXXX=XXXX)
```

### Examples

```
# DontShare can be used in conjunction with ShareWith to specify the specific
# blocks that are excluded from the ShareWith
# Note: this is equivalent to saying: ShareWith (Major, Major<>CHEM)
ShareWith (Major)
DontShare (Major=CHEM)
```

```
# DontShare can be used by itself in the header to prevent certain blocks
# from sharing with it. In this case we are preventing the minor from sharing
# with this block - in case the minor has ShareWith specified.
DontShare (Minor)
```

```
# This block does not want to allow sharing from any block whatsoever
DontShare (AllBlocks)
```

```
# You can prevent certain blocks from sharing with this block
DontShare (Minor=ART, Minor=HIST, Major=LANG)
```

```
# Don't share with the student's first major
DontShare (MAJOR=1st)
```

```
# Don't share with major that is associated with this conc block
DontShare (MAJOR=associated)
```

```
# Don't share with concentration that is associated with this major block
DontShare (CONC=associated)
```

## Notes

1. DontShare can be used in the header along with ShareWith. The DontShare has precedence. That is, if the ShareWith says to share with the Major and the DontShare says not to share then the block will not share.
2. DontShare can be used in the header without ShareWith to block sharing specified in other blocks. When used like this, DontShare serves as a limiter around the block.
3. DontShare may not work like you want it to. For example, in your major if you have DontShare (Conc=XYZ) and also ShareWith(Minor) and the minor and the concentration are sharing (because they have ShareWith) then a class may be applied to all three blocks. This is because the auditor is seeing that the minor and concentration are sharing and the major and minor are sharing; it does consider the major and concentration to be sharing.
4. DontShare (Thisblock) is not supported.
5. DontShare (Major<>CHEM) – the auditor will not be able to interpret this– don't use not-equal-to in DontShare.
6. You cannot specify a number of classes or credits on DontShare in the header.
7. 1st, 2nd, 3rd, 4th, etc. See the notes on this option under [ShareWith \(header\)](#).
8. ASSOCIATED. See the notes on this option under [ShareWith \(header\)](#).

## DontShare (rule)

Updated: March 25, 2022

DontShare indicates that the classes applied to this requirement cannot share with other requirements. This is equivalent to Exclusive.

### Template

```
nn Credits in XXXX @
DontShare
Label XXXX "xxx xxxx";
```

### Examples

```
6 Credits in POLI @
DontShare
Label POLYSCI "Political Science";
```

```
3 Classes in HIST @
DontShare 1 Class
Label HIST "History";
```

### Notes

1. DontShare indicates that the credits or classes applied by the auditor towards satisfying this rule cannot be used to satisfy requirements in other blocks and in other rules in this block. A course applied to a DontShare rule is applied only to that rule and to no other rule.
2. DontShare cannot be used in the same rule as ShareWith.
3. DontShare is needed as part of the rule only if the block has been declared ShareWith. DontShare is not needed on a rule if the ShareWith block qualifier is not used. By default, a requirement block does not share.
4. The number of classes or credits to be treated as DontShare optionally follows the keyword DontShare. To treat some classes/credits as DontShare and the rest as ShareWith, use ShareWith as a block qualifier and use DontShare with the number of classes or credits as a rule qualifier.

## Else

Updated: March 25, 2022

EndIf and EndElse are synonymous. Either can be used on the If, Elself or Else part. See [If-Then](#).

## End.

Updated: March 25, 2022

End marks the end of a block of requirements.

### Template

```
End.
```

### Examples

```
End.
```

### Notes

1. The last keyword of each block must be End followed by a period.
2. Everything after the End. is ignored by the parser.
3. For each End. there is a corresponding Begin at the start of the requirement block.

## EndSub

Updated: March 25, 2022

See [BeginSub \(rule\)](#).

## Except

Updated: March 25, 2022

Except specifies the courses that should not fill a requirement or be considered as part of a qualifier.



## Template

```
Except XXX 998, 999
```

## Examples

```
# A minimum of 10 classes in 100-level chemistry but do not count 1
20 and 121
MinClasses 10 in CHEM 1@
  Except CHEM 120, 121
  ProxyAdvice "You have taken <APPLIED> chemistry 100-level classes
but need "
  ProxyAdvice "<NEEDED> more. (CHEM 120, 121 are not allowed) "
```

```
# Do not allow any ABC classes - but do allow 200-level classes wit
h the SPCL attribute
MaxCredits 0 in ABC @
  Except ABC 2@ (With Attribute=SPCL)
```

```
6 Credits in ART @
  Except ART 120, 121
  Label ART "Art Studies";
```

## Notes

1. Except is allowed in a course rule and any header qualifier that allows a course list.
2. Except is appropriate only when the preceding course list contains wildcards or ranges of courses.
3. Use a comma or 'Or' to separate courses in the course list following Except. The plus sign or 'And' is not allowed. None of the courses following Except will be applied to the rule or qualifier if taken by the student.

## Exclusive

Updated: March 25, 2022

Exclusive indicates that the classes applied to this requirement cannot share with other requirements. See [DontShare \(header\)](#).

---

## FirstYearEarnedCredits

Updated: March 25, 2022

This represents the credits earned in the first full academic year. If first term is a summer term, it is counted. The summer ending term of the first year is also counted.

### Template

```
if (FirstYearEarnedCredits >= nn) then
```

### Examples

```
-- 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";
```

### Notes

1. These credits are automatically calculated for Athletic Eligibility audits.

## From

Updated: March 25, 2022

From precedes a list of courses, groups, disciplines, or transfer codes. See [In](#).

## Group (rule)

Updated: March 25, 2022

A Group defines a list of requirement choices, of which a specified number of rules must be satisfied.

## Template

```
1 Group in
  (N Credits in XXX 999, 998, 987
    Label XXXX "Option 2 xxxx") or
  (N Credits in XXX 999, 998, 987
    Label XXXX "Option 1 xxxx")
Label XXXX "XXXXX REQUIREMENT";
```

## Examples

```
1 Group in
  (1 Class in ENGL 101
    Label REMENGL "Remedial English") or
  (1 NonCourse (ENGTEST)
    Label ENGLENTREXAM "English Entrance Exam")
Label ENTRYENGL "ENTRY-LEVEL ENGLISH";
```

```
1 Group in
  (6 Credits in FRE @, GER @, SPA @
    Label GRPA "Group A") or
  (2 Groups in
    (1 Class in INS 142
      Label INS "INS Course") or
    (4 Credits in CCS @
      Label CCS "CCS Course")
    Label GRPB "Group B")
MaxPassfail 0 Classes Tag=MAXPF
Label TRANS "TRANSCULTURAL STUDIES";
```

## Notes

1. Group must be followed by a list of one or more rules. The list of rules following the Group keyword is referred to as the group list. Each rule in the group list is a group item. Each group item is enclosed in parentheses and does not end with a semicolon.
2. Each rule in the Group list is one of the following types of rules: Course, Block, BlockType, Group, RuleComplete, RuleIncomplete or NonCourse. A group item cannot be an If rule or a subset rule.
3. Each rule in the list is connected to the next rule by "or".
4. A Group statement can be nested within another Group statement. There is no limit to the number of times you can embed a Group within a Group. However, the worksheet display of a requirement with many depths may be difficult to understand.
5. Qualifiers that must be applied to all rules in the group list must occur after the last right

parenthesis and before the label at the end of the Group statement. Qualifiers that apply only to a specific rule in the group list must appear inside the parentheses for that group item rule.

6. Allowable rule qualifiers: DontShare, Hide, HideRule, HighPriority, LowPriority, LowestPriority, MaxPassFail, MaxPerDisc, MaxTransfer, MinGrade, MinPerDisc, NotGPA, ProxyAdvice, SameDisc, ShareWith, RuleTag.
7. Do not mix course rules with Block rules in a group. Although this will parse, the auditor may not handle this as expected. Putting Block rules into Groups is not a best practice.

## HeaderTag

Updated: March 25, 2022

HeaderTag is a qualifier you can place in the block header to give it special meaning when the audit worksheet is being displayed.

Any HeaderTag name-value pair you add to your header will be available for use within the xsl stylesheets used in the Dashboard to show the block in a different way—hide it, use a different color, etc. Both the name and the value following HeaderTag are limited to 200 characters each. The value must be enclosed in double-quotes if it contains non-alphanumeric characters.

## Template

```
HeaderTag XXXX=YYYY
```

## Examples

```
Begin
  40 Credits
  HeaderTag RemarkJump="http://some.place.edu/ontheinternet/anywhereisfine/"
  HeaderTag RemarkJump="support/getmemoreinfo.html"
  HeaderTag RemarkHint="More info on Gen Ed"
  HeaderTag CreditsApplied=Show # override UCX-SCR004 setting
;
Remark "You can click this link to find out more information";
Remark "about the General Education requirements.";

# Use this placeholder of 120 but the credits will really be based
on the sum of all
# required blocks. This is only valid to use in the starting block.
120 Credits
  HeaderTag DWCredits=DWBlockSum
```

```
# Use this placeholder of 30 but the credits will really be based on
the sum of all
# the rules in this block. This is normally used in non-starting blocks
and blocks
# where the credits on the rules may vary based on if-statements
30 Credits
  HeaderTag DWCCredits=DWRuleSum

# Use this placeholder of 92 but the credits will be based on the TOTALCREDITS
# value that is bridged for the student. TOTALCREDITS must be setup in UCX-SCR002
but
# of course any name can be used.
92 Credits
  HeaderTag DWCCredits=TOTALCREDITS

# When the max is exceeded the excess classes are placed in over-the-limit.
# This MaxReason text is used as the Reason text in the over-the-limit section.
MaxCredits 14 in @ (With attribute=ABCD)
  HeaderTag MaxReason="You exceeded the max of 14 credits in ABCD course work"
```

## Notes

1. HeaderTag names of RemarkJump and RemarkHint have defined meanings and are used by the standard Degree Works worksheets.
2. See the [AdviceJump](#), [RemarkJump](#), and [RuleTag](#) topics for more information.
3. HeaderTag name of DWRangeLimit has a defined meaning in the context of MinCredits. See the [MinCredits \(header\)](#) header qualifier for more details.
4. HeaderTag names of CreditsRequired, CreditsApplied, CatalogYear, and Gpa with values of Show and Hide can be used to override the UCX-SCR004 Show flags. For example, if Show GPA is N for the MINOR block type in UCX-SCR004, you can add "HeaderTag Gpa=Show" in any of the MINOR blocks to override this flag. This means that for most minors the GPA will not show but it will show for those blocks with this HeaderTag. These tags are only applicable for the Responsive Dashboard.
5. HeaderTag name of DWCCredits can be used on the credits qualifier to let the auditor calculate the credits required block on the blocks and rules within the audit. HeaderTag value of DWBlockSum means to sum the header credits of the required blocks in the audit. This can only be used on the starting block. HeaderTag value of DWRuleSum sums the credits of all rules in the block. It does not count the credits in blocks called from within the block. If a rule has Classes specified instead of Credits, the rule can be scribed with a DWCCredits RuleTag to specify how many credits should be used towards the DWRuleSum calculation. If no such RuleTag is found on a Classes rule, then no credits are used for the rule. A DWCCredits

RuleTag can be added to a group rule to represent the credits to use in the DWRuleSum calculation. If no DWCredits RuleTag is found on a group, then the auditor counts the credits of the rules in the group. If the group rule requires two rules to be satisfied, then the sum of the first two rules in the group are used, for example. If the HeaderTag value is not DWBlockSum and not DWRuleSum, then the value is looked up in the list of custom data for the student – based on setup of UCX-SCR002. This means that you may bridge a credit value to be used for the student. If that name is not found for the student, then the credits scribed on the credits qualifier are used.

6. HeaderTag name of MaxReason can be placed on Max qualifiers in the starting/degree block. This text will show as the Reason text explaining why classes were placed in over-the-limit.

## Hide

Updated: March 25, 2022

Hide allows certain courses to satisfy a requirement while hiding this fact from the audit advice. Hiding a course ensures it will never show in the advice on the worksheet but the course can be used to help satisfy the requirement.

## Template

```
N Credits in XXXX 999, {Hide XXX 992,} 998
Label XXXX "xxx xxx xxx";
```

## Examples

```
# Hide ECON 112 from the advice
3 Classes in ACCT 103, 105, {Hide ECON 112,} 114
Label ACCOUNTING "Accounting Requirements";

# Hide MATH 104 from the advice
3 Credits in MATH 184, {HideFromAdvice MATH 104 (WITH DWTerm < 2012
30)}
Label MATH "Math Requirement";

# Show HIST 184 but hide the WITH information
5 Credits in HIST 184 (WITH Hide DWResident = Y)
Label HIST "History Requirement";

# Do not allow HIST classes but don't mention that in the advice
5 Credits in @ (WITH Attribute=SPEC)
Except {Hide HIST @}
Label SPEC "Special Requirement";
```

## Notes

1. The Hide keyword is only allowed on courses specified in a course rule.
2. When you use Hide inside an AREA type rule with [ ], you need to place the Hide as the first course in the rule, inside the starting square bracket "[{. An example would be as follows: [{Hide PSY 154}, ANT 222, HIST 101] The curly brace "]" cannot be followed immediately by the square bracket "]".
3. The Hide keyword is not allowed with a course rule specified with a plus(+) list of classes. It makes no sense to require all classes but to hide some of them from the audit advice.
4. One or more classes can be listed within the Hide braces, and multiple occurrences of Hide are allowed within the same course rule.
5. The trick to knowing where to place the commas when using Hide is to be sure that everything within and including the braces can be removed leaving a valid, parseable course rule.

## HideRule

Updated: March 25, 2022

HideRule hides a rule and its advice on worksheets.

Typically this rule qualifier is used to hide one or more options within a Group rule or stand-alone Block and BlockType rules in the Degree, although the qualifier can also be used on the Noncourse rule.

## Template

```
HideRule
```

## Examples

```
# Pull in the GE block; hide if GE block exists;
# show label and advice if block not found in audit
1 Block (Other=GE)
  HideRule
  Label GENED "General Education Requirements";

# Require a major; hide if major exists;
# show label and advice if no major found
1 Blocktype (Major)
  HideRule
```

```
LABEL MAJOR "Major Requirements";

# The 3rd option is not recommended but is allowed - so hide it from the advice
# However, if the rule is partially complete then show the advice
1 Group in
  (2 Classes in MATH 101, 103, 105
    Label MATH1 "Math Option I - College Algebra") OR
  (2 Classes in MATH 121, 122, 123, 126
    Label MATH2 "Math Option II - Calculus Sequence") OR
  (2 Classes in BUSI 201, 221, 225
    HideRule
    LABEL MATH3 "Math Option III - Business Math")
Label MATHREQ "MATH REQUIREMENT";
```

## Notes

1. After a class is applied to a rule with HideRule, the rule will appear on the worksheet along with the classes that are applying to the requirement.
2. The Block and Blocktype rules will only appear if the referenced blocks are missing.

## HighPriority

Updated: March 25, 2022

HighPriority tells the auditor that a certain rule or block should be satisfied before others.

HighPriority is used mostly when classes fit multiple rules and you want to specify a preference. However, the auditor checks other criteria before examining the priority of a rule or block so HighPriority does not guarantee a rule or block will be satisfied before another rule or block; HighPriority simply increases the chances that a rule or block will be satisfied before others.

## Template

```
HighPriority
```

## Examples

```
# All courses listed on this rule have a higher priority
# than those on other rules
5 Credits in ACCT 103, 105
HighPriority
```



```
Label ACCT "Accounting";

# Accounting classes have twice the priority than those
# rules with just one HighPriority
5 Classes in ACCT 101:110
  HighPriority
  HighPriority
  Label ACCT1 "Accounting";

# Only HIST 112 has a higher priority
10 Credits in HIST 102, 106, HighPriority 112
  Label HIST1 "History requirement";

# Only HIST 102 has a higher priority
10 Credits in HighPriority HIST 102, 106, 112
  Label HISTORY "History requirement";

# Only HIST 106 has a higher priority
10 Credits in HIST 102, {HIDE HighPriority 106,} 112
  Label HIST2 "History requirement";

# Only the CHEM classes have a higher priority
10 Credits in BIOL @, PHYS @, HighPriority CHEM @
  Label SCIENCE "Science requirement";

BEGIN 40 Credits
  HighPriority # This major block has a higher priority than the mi
nor
;
```

## Notes

1. HighPriority can be repeated on a block or rule to give a higher preference over those rules or blocks with fewer HighPriority qualifiers.
2. HighPriority can be placed on a group or subset or directly on a course rule.
3. Each High- increases the match level of a class on a rule by 5 points.
4. HighPriority can be used at the block level and simultaneously on any rules within the block.
5. HighPriority may be combined with LowPriority as needed; each cancels out the other, however.
6. HighPriority, High-Priority, or HighPri can be used.

## If-Then

Updated: September 29, 2023

If begins a conditional statement. If statements may be used in the block header and in the body to control the qualifiers and rules used in the block based on the student's data.

### Template

```
If (XXXXX = YYYY) Then
  BeginIf
    # some requirement
  EndIf
```

### Examples

```
If (1stMajor = BIOL) THEN
  MaxCredits 15 in CHEM @ tag=MAX15
Else
  MaxCredits 10 in CHEM @ tag=MAX10
```

```
If (Major = HIST) Then
  BeginIf
    MaxCredits 15 in MATH @ tag=MAX15MATH
    MaxCredits 15 in CHEM @ tag=MAX15CHEM
  EndIf
Else
  BeginElse
    MaxCredits 10 in MATH @ tag=MAX10MATH
    MaxCredits 10 in CHEM @ tag=MAX10CHEM
  EndElse
```

```
If (Major = HIST) Then
  MaxCredits 15 in MATH @
Else If (Major = ACCT) Then
  MaxCredits 10 in MATH @
Else If (Major = ENGL) Then
  MaxCredits 12 in MATH @
Else
  MaxCredits 20 in MATH @
```

```
# In BIOL or BIOS or BIOX majors, require 45 credits
If (Major = "BIO@") Then
  MinCredits 45 in BIO @ tag=MIN45BIO
```

```
Else
  MinCredits 10 in BIO @ tag=MIN10BIO

If (1stMajor = BIO) THEN
  15 Credits in BIO 3@
  Label BIO1 "Biology elective for Biology majors"
Else If (1stMajor = CHEM) THEN
  15 Credits in CHEM 3@
  Label BIO2 "Biology elective for Chemistry majors"
Else
  10 Credits in BIO 3@
  Label BIO3 "Biology elective for non-Biology majors";

If ((Conc = AHST or Conc = EHST) and Major = COMM) Then
  2 Classes in POL 100:199
  Label POLISCI "Political Science";

If (Attribute = HONR) Then
  BeginSub
  5 Classes in HIST 100:199
  Label HIS100 "History 100 level classes"
  2 Classes in HIST 200:299
  Label HIS200 "History 200 level classes"
  EndSub
  Label HISHON "HISTORY FOR HONOR STUDENTS";

If (Attribute = HONR and NumberOfMajors >= 2) Then
  BeginIf
  5 Classes in HIST 100:199
  Label HIST100A "Honors: History 100 level classes";
  2 Classes in HIST 200:299
  Label HIST200A "Honors: History 200 level classes";
  Remark "As an honors student you need to take 5 classes"
  Remark " at the 100 level and 2 at the 200 level"
  EndIf
Else
  BeginElse
  3 Classes in HIST 100:199
  Label HIST100B "History 100 level classes";
  1 Classes in HIST 200:299
  Label HIST200B "History 200 level class";
  Remark "You need to take 3 classes"
  Remark " at the 100 level and 1 at the 200 level"
  EndElse

# Student System GPA; Banner sites may prefer to use BannerGPA
If (SSGPA > 2.0) Then
```

```

RuleComplete
  Label GPAGOOD "2.0 GPA requirement met"
Else #insufficient GPA
  RuleIncomplete
    ProxyAdvice "Your GPA is below the 2.0 required"
    Label GPABAD "2.0 GPA requirement not met";

# If a PLAN block is included then use that instead of the GenEd block
If (OTHER = PLAN) then
  1 Block (Other=PLAN)
    Label PLAN "Planned requirements"
Else
  1 Block (Other =GENED)
    Label GENED "GenEd requirements";

# If the student does not have the Ali Status value on her record;
# that is, no ALISTAT value was found in our student system for her.
# (or you can say If (ALISTAT = NODATA) then
If (ALISTAT = " ") then
  RuleIncomplete
    ProxyAdvice "See your advisor"
    Label "You have not completed the Ali Application "
Else
  RuleComplete
    Label "Thank you for completing the Ali Application - you are fantastic!";

# DOLLAR VARIABLE
# Previously you would have to scribe a long if-statement like this:
IF (SPCLMAJOR="ACCT" OR SPCLMAJOR="BIOL" OR SPCLMAJOR="CHEM" OR SPCLMAJOR="ENGL" OR

    SPCLMAJOR="HIST" OR SPCLMAJOR="MUSC" OR
    SPCLMAJOR="PHIL" OR SPCLMAJOR="POLS" OR
    SPCLMAJOR="PSYC" OR SPCLMAJOR="STEM")
THEN
  RuleComplete
    Label "You have the special major";
# But by creating a dollar-variable that points to a special value
pulled from the

# students's record you can simplify the if-statement like this:
# Note that SPCLMAJOR needs to be setup in UCX-SCR002
if (Major = $SPCLMAJOR) then
  RuleIncomplete
    Label "You have the special

```

```

        major";
# A simple way to see if the student changed their major; ADMITMAJOR must be in
    SCR002
if (Major <> $AdmitMajor) then
    RuleIncomplete
        Label "You changed your major";
# If you have both ActiveTerm and GradTerm setup in SCR002
# you can compare two custom values for the student
if (ActiveTerm = $GradTerm) then
    RuleComplete
        Label "Great - you are almost
            finished!";
# If the student does not have the attribute that
# that was pulled from her special SIS record; not-equals or any other operator can be
    used.
# Note that SOMECODE needs to be setup in UCX-SCR002
# The dollar variable can be uppercase or lowercase; it is always upshifted
if (Attribute <> $SomeCode) then
    RuleComplete
        Label "You have the special
            attribute";
# Don't allow the student to have the same concentration as their major
# A dollar variable of Major, Minor, Conc etc does not need to be in SCR002
if (Major = $Conc) then
    RuleIncomplete
        Label "Your major and concentration are the same
            - this is not allowed!";
# Check to see if the student has applied for graduation
# GradAppProg needs to be setup in UCX-SCR002 of course.
If (PROGRAM = $GradAppProg) then
    RuleComplete
        Label "You've applied for
            graduation!";

# If MATH 101 was completed (passed or failed) but is not in-progress
ss
If (MATH 101 was TAKEN) Then

# If one of these were passed (in-progress classes are considered passed also)
If (MATH 101:104 was Passed) Then

# If MATH 101 was passed (but is not in-progress)
If (MATH 101 was PASSED and MATH 101 isn't INPROGRESS) Then

```

```
# If MATH 101 was passed (but is not in-progress)
If (MATH 101 (With DWInprogress=N) was PASSED) Then

# If any 100-level math class was failed
If (MATH 1@ was Failed) Then

# If MATH 101 exists but is not in-progress (so it must be complete
d)
If (MATH 101 isnt INPROGRESS) Then

# If MATH 101 exists but was not transferred (so it must have been
taken natively)
If (MATH 101 wasnt TRANSFERRED) Then
```

## Notes

1. BeginIf/BeginElse and EndIf/EndElse are not required if you have one qualifier or rule. However, it is recommended that Begin-End always be used for consistency of scribing.
2. BeginIf and BeginElse are synonymous. Either can be used on the If, ElseIf or Else part.
3. EndIf and EndElse are synonymous. Either can be used on the If, ElseIf or Else part.
4. When you have an "Else If" you can use either the BeginIf/EndIf or the BeginElse/EndElse.
5. You can imbed Remarks within the Begin/End when scribing rules; you cannot imbed Remarks in the Begin/End in the header however.
6. When a wildcard is used in the value you must use double-quotes around the value. Example: Major = "BIO@".
7. Remember that semicolons are optional so if you cannot figure out where it goes you can leave it out. (Semicolons do help the parser give better error reporting however.)
8. SSGPA and BannerGPA are synonymous.
9. You can check for the nonexistence of a special value you have bridged to Degree Works by comparing to " " (a space surrounded by quotes) or to NODATA.
10. The keywords "is" and "was" are synonymous.
11. The keywords "isnt" and "wasnt" are synonymous.
12. COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, PROGRAM, SCHOOL and SPEC can be used on the left-hand-side of an expression. The value comes from the student's curriculum and must be valid in the corresponding UCX table.
13. Any custom code setup in UCX-SCR002 can also be used on the left-hand-side of an

expression.

14. These words are valid for Athletic Eligibility audits in an IF-statement: CompletedTermCount, TotalCreditsEarned, ResidenceCreditsEarned, TotalCreditsAttempted, ResidenceCompletedTermCount, DegreeCreditsRequired, CreditsAppliedTowardsDegree, PreviousTermEarnedCredits, PreviousTermEarnedCredits-Fall, Previous2TermsEarnedCredits, Previous3TermsEarnedCredits, PreviousAcademicYearEarnedCredits, PreviousFullYearEarnedCredits, FirstYearEarnedCredits. See the [Athletic Eligibility Audit](#) topic for more information.
15. These words are valid for Financial Aid audits in an IF-statement: CompletedTermType, CompletedTermCount, TotalCreditsEarned, ResidenceCreditsEarned, TotalCreditsAttempted, CreditsAttemptedThisTerm, CreditsEarnedThisTerm, CreditsAttemptedThisAidYear, CreditsEarnedThisAidYear, CompletedTermCount, ResidenceCompletedTermCount, LastCompletedTermType, DegreeCreditsRequired. See the [Financial Aid Audit](#) topic for more information.
16. **AuditAction** and AuditType can also be used on the left-hand-side of an expression. Valid values for AuditAction are NORMAL, WHATIF, LOOKAHEAD, EXCEPTIONS, PLANNER, and TRANSFER. The action of TRANSFER is used by both Transfer Finder and Transfer Equivalency. Valid values for **AuditType** are ACADEMIC, FINANCIALAID, ATHLETIC, and REQUISITE.
17. FOUND, TAKEN, PASSED, FAILED, INPROGRESS and TRANSFERRED can be used to check on the state of a class.
18. FOUND = historic or in-progress; passed or failed.
19. TAKEN = historic (not in-progress); passed or failed.
20. PASSED = passed grade; the Passed flag is Y; in-progress classes are usually bridged to Degree Works with Passed=Y.
21. FAILED = failed grade; the Passed flag is N.
22. INPROGRESS = the In-progress flag is Y.
23. TRANSFERRED = The class is on the rad\_transfer\_dtl and not the rad\_class\_dtl.
24. If multiple data conditions are used, then they must be joined by "and" or "or". Plus and comma are not allowed to connect conditions.
25. The rule or qualifier following Then or Else can be a single rule or qualifier or multiple rules bracketed by BeginSub/EndSub or can be one or more rules or qualifiers enclosed in BeginIf/EndIf or BeginElse/EndElse.
26. When using relation operator <> (not equal to) in a list of several data conditions, you must connect conditions with "and", otherwise the conditions will not be met. That is, "If (Major<>XYZ or Major<>ABC)" will always be false; use "and" instead.
27. If you are using an If-statement with a course specified you cannot use the MinGrade or MinGPA qualifiers in the header within such an IF-statement. The reason for this is that the auditor needs to know whether there is a MinGrade/MinGPA qualifier in the header before it

applies the classes – and the IF-statement is only evaluated after all classes have been applied. Although the parser allows you to do this the auditor will not give you expected results.

28. Semicolons – where do you put them? Historically, semicolons were required on all rules and scribing IF-statements was tricky. They are now optional on all rules and remarks. If you scribe a semicolon – that is fine. If you don't scribe a semicolon – that is fine too. (Though a semicolon is still required to end the header qualifiers and they do help the parser report parse errors when found.)
29. When OTHER is specified in the If expression, the auditor checks to see if that specific OTHER block is in the list of blocks that was sent to it. This is best used for particular OTHER blocks like those created by the planner.
30. You may specify a Dollar Variable on the right side of the equals sign in your if-statement. The variable name specified needs to be set up in UCX-SCR002 pointing to some custom value you bridged or to any other value in the Degree Works database. The Dollar Variable may also refer to Major, Minor, etc. Almost any code that you are allowed to place on the left side of the equals sign is allowed as a Dollar Variable on the right side. This Dollar Variable allows you to compare two pieces of student data in any way you like for any reason you need.

## In

Updated: March 25, 2022

In precedes a list of courses, groups, disciplines, or transfer codes.

### Template

```
N Classes in XXX @
```

### Examples

```
MaxCredits 12 in COMM 2@
```

```
MaxTransfer 36 Credits in (AP, CLEP)
```

```
MaxPerDisc 8 Credits in (BIO, CHE, PHYS)
```

```
3 Classes in INS @  
  Label INS "International Studies";
```

```
1 Group in
```



```
(1 Class in MATH 115,116
  Label GRPA "Group A") OR
(1 Class in MATH 125, MATH 126
  Label GRPB "Group B")
Label "FRESHMAN MATH";
```

```
6 Credits in ART @
Except ART 120, 121
Label ART "Art Studies";
```

## Notes

1. In is optional. It is used only for readability.
2. In is synonymous with From.

## IncludeBlocksWith

Updated: March 25, 2022

Specifies the blocks that should be included in this block's header qualifier calculations.

## Template

```
IncludeBlocksWith (XXX=YYY)
  Label IBW "xxxx xxxxxx";
```

## Examples

```
# Include all concentrations starting with XY
IncludeBlocksWith (CONC="XY@")
  Label IBW "Included Requirements";
```

```
# ART college blocks in the UG school are included
IncludeBlocksWith (College=ART and School=UG)
  Label IBW "Included Requirements";
```

## Notes

1. Use of wildcards within the code requires quotes.

2. And (or the plus) must separate multiple type-code combinations.
3. The type can be in upper, lower, or mixed case.
4. The rule is as complete as the average of the blocks that are included based on the type and code specified.
5. If no blocks of the specified type are found, the rule becomes 100 percent complete.
6. No rule qualifiers are allowed.

## Including

Updated: March 25, 2022

Including indicates mandated courses from a course list.

### Template

```
Including XXX 998, 999
```

### Examples

```
12 Credits in ART @  
Including ART 120, 121  
Label ART "Art Studies";
```

```
12 Credits in ENGR 103, 109, 123, 124  
Including ENGR 103  
Label ENGR "Engineering for Engineers";
```

### Notes

1. Including is allowed only within a course rule. It is preceded by a course list.
2. Including is followed by a list of courses that must be taken to fulfill the requirement. The list that follows Including must be a subset of the list that precedes Including.
3. Including is most appropriate when the preceding course list contains wildcards or ranges of courses but can be used when there are no ranges or wildcards also.
4. Use a comma or "or" to separate courses in the course list following Including. The plus sign or "and" is also allowed. The auditor does not care how the courses are separated following the Including. As the courses are part of an include list, they will always be audited as though the "and" operator was used. Therefore, using "and" (or the plus) has the same effect as using "or"

(or a comma) in that all courses on the list are required.

## Label

Updated: March 25, 2022

Label is a free-text comment used to identify the requirement.

## Template

```
Label XXXX "xxx xxx xxx"
```

## Examples

```
MinCredits 6 in FRE @, GER @, SPA @
  ProxyAdvice "You need 6 credits in French, German or Spanish"
  ProxyAdvice "but have only taken <APPLIED> credits; you need <NEE
DED> more."
  Label FORLANG "Foreign Language"
```

```
MinGPA 2.5 in ENGL @, LIT @
  ProxyAdvice "You need a GPA 2.5 but currently have a <APPLIED> GP
A"
  Label ENGLITGPA "Literature/English GPA 2.5 requirement"
```

```
If (Minor=EDU) Then
  MinClasses 3 in TEACH @
  Label STUTEACH "Student Teaching"
Else
  MinClasses 2 in TEACH @
  Label APPREC "Student Teaching"
```

```
12 Credits in ENGR 103, 109, 123, 124
  Label ENGR "Engineering for Engineers";
```

```
1 Block (OTHER=TEACH)
  Label STUTEACH "Student Teaching";
```

## Notes

1. Label is followed by a free-text comment enclosed in quotes. The text can be between 1 and 200 characters long but cannot include double quotation marks. The text cannot be empty, that

is, Label " " is not allowed.

2. All labels should have a label-tag – which is a code with a maximum length of 20 alphanumeric characters that follows the "Label" keyword. This is used by exception processing to ensure the exceptions apply to the correct requirements even when the contents of the block changes over time. When a Label is on a header qualifier the qualifier does not need a tag because the Label tag is used to identify the qualifier.
3. Labels are optional for header Min qualifiers. If found, the label appears with a check box and advice similar to how a rule appears.
4. Labels are only allowed on these header qualifiers: Classes, Credits, MinClasses, MinCredits, LastRes, MinRes, MinGPA, MinPerDisc, MinTerm, Under. That is, any qualifier specifying a minimum limit.
5. On rules, Labels must follow the rule qualifiers and precede a right parenthesis ending a rule in a Group or precede Else in an If rule or precede a semicolon ending a rule. One Label is allowed per rule statement. Each rule in a Group statement must have a label as must each rule in a subset.
6. Label is optional or required based on UCX-CFG020 DAP13 Require Labels flag. If the label is set to N, then the audit output will not display a description of the requirement.
7. The Label text only has to be unique if there are no label tags. However, it is strongly recommended that all labels have alphanumeric label tags. Numeric label tags like "4" or "8.3" are not recommended because it is too easy for someone to renumber the label tags and thus causing exceptions to apply to the wrong requirements. Label tags like "HISTREQ" or "CHEM200LEVEL" are preferred as they are much less likely to be changed by a novice user. You might even consider using a random (such as LP23ADF9X) so that it has no meaning whatsoever and therefore is much less likely to be changed accidentally.
8. See the [Label Tags](#) topic for more information.

## LastRes (header)

Updated: March 25, 2022

LastRes specifies the credits or classes that must be taken in residence as the last credits/classes taken before completing the degree.

### Template

```
LastRes x Credits Tag=LASTRES
ProxyAdvice "Your last x credits must be taken in residence"
ProxyAdvice "but only your last <APPLIED> credits were so you need"
ProxyAdvice "to take at least <NEEDED> more."
```

## Examples

```
# the last 12.5 credits must be taken in residence
LastRes 12.5 Credits Tag=LASTRES
  ProxyAdvice "Your last 12.5 credits must be taken in residence"
  ProxyAdvice "but only your last <APPLIED> credits were so you need"
  ProxyAdvice "to take at least <NEEDED> more."
```

```
# the last 4 classes must be taken in residence
LastRes 4 Classes Tag=LASTRES
  ProxyAdvice "You have taken <APPLIED> classes in residence"
  ProxyAdvice "but need to take at least <NEEDED> more."
```

```
# 15 of the last 30 credits must be taken in residence
LastRes 15 of 30 Credits Tag=LASTRES
  ProxyAdvice "You have taken <APPLIED> of your last 30 credits in residence"
  ProxyAdvice "but need to take at least <NEEDED> more."
```

```
# 15 of the last 30 credits must be taken in residence
# and SE courses are also considered in residence
# - but ignore all PE classes from counting as in-residence
LastRes 15 of 30 Credits in @ (With DWResident=Y or Attribute=SE)
  Except PE @
  Tag=LASTRES
  ProxyAdvice "You have taken <APPLIED> of your last 30 credits in residence"
  ProxyAdvice "but need to take at least <NEEDED> more;"
  ProxyAdvice "but note that PE classes do not count."
```

## Notes

1. LastRes can only be used if all of a student's courses and transfer credits have been linked to a term.
2. LastRes can also be used to specify a smaller number "of" a bigger number. For example, "15 of 30". The first number indicates how many credits/classes out of the second number of last credits/classes taken by the student must be earned in residence.
3. Only one LastRes qualifier is allowed in the block header.
4. LastRes allows a list of courses to specify what courses should be considered "in residence". Without a course list those courses that were not transferred in are considered in residence. When a course list is present only those courses that match the course list will be considered in residence. In doing this, special transfer classes with certain attributes can be lumped into the residency bucket and count towards the LastRes qualifier.

5. When LastRes is on a block other than the starting block only those courses in the block's scope are counted towards the LastRes credits or classes. For example, if LastRes is on the Major block only the transfer and resident courses in the Major block (and any blocks it references) are examined – those in the Gen Ed block or fall-through, for example, are ignored.
6. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## Libl (header)

Updated: March 25, 2022

When Libl is used in ShareWith it refers to the Libl block in the audit. When used in an If-statement it refers to the liberal learning on the student's curriculum.

### Examples

```
ShareWith (Libl)
```

```
If (Libl = XY) Then  
  MinGPA 2.5 tag=MINGPA  
  ProxyAdvice "You need a GPA 2.5 but currently have a <APPLIED> GP  
A"
```

### Notes

1. When Libl is used within a block header, it is only used with an If-statement or ShareWith.

## Libl (rule)

Updated: March 25, 2022

When Libl is used in ShareWith, Blocktype or Block it refers to the Libl block in the audit. When used in an If-statement it refers to the liberal learning on the student's curriculum.

### Examples

```
1 Blocktype (Libl)  
  Label LIBLBLOCK "Liberal Learning requirements";  
  
# Only if the student has the SPAN liberal learning on their
```

```
# curriculum will this block be used
1 Block (Libl=SPAN)
  Label LIBLBLOCK "Liberal learning requirements";

If (Libl = RPD) Then
  3 Credits in MATH 400
  Label MA400 "Math 400";

6 Credits in BUS 1@
  ShareWith (Libl)
  Label BUS "Business";
```

## Notes

1. Libl is followed by a liberal learning code except in a Blocktype rule and is optional in ShareWith.
2. The liberal learning code must be valid in UCX-STU324.
3. Libl can be used with Blocktype, Block, If-statement, or ShareWith.
4. When Libl is used in a Block rule, the specific liberal learning block specified will only be included in the audit if the student has that liberal learning on their curriculum. For this reason, it is rare to use a Block rule with Libl.

## LowestPriority

Updated: March 25, 2022

LowestPriority tells the auditor that a certain rule should be considered less important than others.

LowestPriority is used mostly when classes fit multiple rules and you want to specify a preference. LowestPriority is stronger than LowPriority because the classes on rules with LowestPriority are removed early on in the audit allowing the other rules where these classes apply to use these classes. However, during the redemption phase these classes are reapplied to these LowestPriority rules if the classes are in fall-through or if they can be shared (via ShareWith) with other rules.

## Template

```
LowestPriority
```

## Examples

```
15 Credits in ACCT @
  LowestPriority
  Label ACCTELECT "Accounting Electives";
```

## Notes

1. LowestPriority can be placed on a group or subset or directly on a course rule.
2. LowestPriority sets the match level for each class to -9876 – a very low value.
3. LowestPriority is synonymous with Lowest-Priority.
4. Unlike with LowPriority, LowestPriority cannot be placed on a specific course on a rule.
5. LowestPriority cannot be used as a block qualifier.
6. When LowestPriority is on a course rule that is a range (ex: 3:9 Credits in...) the auditor will most likely not apply up to the maximum number of credits the rule allows during redemption. The redemption algorithm will see that the rule is 100% complete and will not apply more credits even if more credits could be applied.

## LowPriority

Updated: March 25, 2022

LowPriority tells the auditor that a certain rule or block should be satisfied after others.

LowPriority is used mostly when classes fit multiple rules and you want to specify a preference. However, the auditor checks other criteria before examining the priority of a rule or block so LowPriority does not guarantee a rule or block will be not satisfied before another rule or block; LowPriority simply decreases the chances that a rule or block will be satisfied before others.

## Template

```
LowPriority
```

## Examples

```
# All courses listed on this rule have a lower priority
# than those on other rules
5 Credits in ACCT 103, 105
  LowPriority
```



```
Label ACCT "Accounting";

# Accounting classes have half the priority than those
# rules with just one LowPriority
5 Classes in ACCT 101:110
  LowPriority
  LowPriority
  Label ACCT1 "Accounting";

# Only HIST 112 has a lower priority
10 Credits in HIST 102, 106, LowPriority 112
  Label HIST1 "History requirement";

# Only HIST 102 has a lower priority
10 Credits in LowPriority HIST 102, 106, 112
  Label HISTORY "History requirement";

# Only HIST 106 has a lower priority
10 Credits in HIST 102, {HIDE LowPriority 106,} 112
  Label HIST2 "History requirement";

# Only the CHEM classes have a lower priority
10 Credits in BIOL @, PHYS @, LowPriority CHEM @
  Label SCIENCE "Science requirement";

BEGIN
  40 Credits
  LowPriority # This minor block has a lower priority than the major
r
;
```

## Notes

1. LowPriority can be repeated on a block or rule to give a lower preference over those rules or blocks with fewer LowPriority qualifiers.
2. LowPriority can be placed on a group or subset or directly on a course rule.
3. Each LowPriority decreases the match level of a class on a rule by 5 points.
4. LowPriority can be used at the block level and simultaneously on any rules within the block.
5. LowPriority may be combined with HighPriority as needed; each cancels out the other, however.

6. LowPriority, Low-Priority, and LowPri are synonymous.

## Major (header)

Updated: March 25, 2022

When Major is used in ShareWith it refers to the Major block in the audit. When used in an If-statement it refers to the major on the student's curriculum.

### Examples

```
ShareWith (Major)
```

```
If (Major = XY) Then
  MinGPA 2.5 tag=MINGPA
  ProxyAdvice "You need a GPA 2.5 but currently have a <APPLIED> GPA"
```

### Notes

1. When Major is used within a block header, it is only used with an If-statement or ShareWith.

## Major (rule)

Updated: March 25, 2022

When Major is used in ShareWith, Blocktype or Block it refers to the Major block in the audit. When used in an If-statement it refers to the major on the student's curriculum.

### Examples

```
1 Blocktype (Major)
  Label MJRBLOCK "Major requirements";
```

```
# Only if the student has the SPAN major on their curriculum will this block be used
1 Block (Major=SPAN)
  Label MJRBLOCK "Spanish major requirements";
```

```
If (Major = RPD) Then
  3 Credits in MATH 400
```

```
Label MA400 "Math 400";

6 Credits in BUS 1@
ShareWith (Major)
Label BUS "Business";
```

## Notes

1. Major is followed by a major code except in a Blocktype rule and is optional in ShareWith.
2. The major code must be valid in UCX-STU023.
3. Major can be used with Blocktype, Block, If-statement, or ShareWith.
4. When Major is used in a Block rule, the specific major block specified will only be included in the audit if the student has that major on their curriculum. For this reason, it is rare for use a Block rule with Major.

## MaxPassfail (header)

Updated: March 25, 2022

MaxPassfail indicates the maximum number of credits/classes that can be taken pass-fail in this block. If this is the starting block then this limit applies to all classes in all blocks including fall-through.

## Template

```
MaxPassfail x Credits Tag=MAXPF
```

## Examples

```
# A maximum of 12.5 pass-fail credits can be applied to the art min
or
MaxPassfail 12.5 Credits Tag=MAXPF
```

```
# A maximum of 4 pass-fail classes can be applied to the accounting
major
MaxPassfail 4 Classes Tag=MAXPF
```

```
# A maximum of 9 credits of pass-fail work can be taken in math and
```

```

chemistry
MaxCredits 9 in MATH @ (With DWPassfail=Y),
                CHEM @ (With DWPassfail=Y) Tag=MAXPFMATHCHEM

# A maximum of 4 pass-fail classes can be applied to the degree
MaxPassfail 4 Classes tag=MAXPF
HeaderTag MaxReason="You exceeded the limit of 4 pass-fail classes"

```

## Notes

1. The MaxPassfail qualifier is applied by the auditor against all the pass-fail classes used to fill the block. If MaxPassfail is in the starting block then it applies to the classes in all blocks and also those in the fall-through section. The number of classes and credits in MaxPassfail is a strict cap that cannot be exceeded. For example, if the maximum is 15 credits and four 4-credit classes match the course list, then one of them will be removed leaving only 12 credits applying.
2. If MaxPassfail is in the starting block and the number of credits or classes taken exceeds the maximum allowed the auditor will move the excess classes into the over-the-limit section.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
4. For all Max qualifiers in your starting/degree block you should add a HeaderTag for MaxReason to ensure the Reason showing in the over-the-limit section indicates exactly why the class cannot be used in the audit. You can scribe a MaxReason HeaderTag on any block but the auditor ignores it on any block that is not the starting block.

## MaxPassfail (rule)

Updated: March 25, 2022

MaxPassfail indicates the maximum number of credits/classes that can be taken pass-fail.

## Template

```
MaxPassfail x Credits Tag=MAXPF
```

## Examples

```

# Allow only 3.5 pass-fail credits
6 Credits in BIO @, CHE @

```

```
MaxPassFail 3.5 Credits tag=MPF35
Label SCI "Science";

# No pass-fail classes are allowed
3 Classes in ENG 300:499
MaxPassFail 0 Classes tag=MPF0
Label ENG "English";

# No pass-fail classes are allowed
3 Classes in ENG 300:499 (With DWPassfail=N)
Label ENG "English";
```

## Notes

1. You can use DWPassfail in the With qualifier to exclude pass-fail classes instead of using MaxPassfail.
2. The MaxPassfail qualifier is applied by the auditor against all the pass-fail classes used to satisfy the requirement. The number of classes and credits in MaxPassfail is a strict cap that cannot be exceeded. For example, if the maximum is 15 credits and four 4-credit classes match the course list, then one of them will be removed leaving only 12 credits applying.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## MaxClasses (header)

Updated: March 25, 2022

MaxClasses indicates the maximum number of classes that can be applied to a requirement block.

## Template

```
MaxClasses n in XXXX @ tag=MAXXXX
```

## Examples

```
MaxClasses 8 in MUS 1@ (With DWResident=Y) tag=MAXMUSIC
```

```
MaxClass 1 in GEOL 109, 145 tag=MAXGEOL
```

```
MaxClasses 3 in PSY 100:199
  Except PSY 108, 117 tag=MAXPSY

# Max of 8 in MUS and ART combined
MaxClasses 8 in MUS @, ART @ tag=MAXMUSART

# Max of 8 in MUS and max of 8 in ART
MaxClasses 8 in MUS @ tag=MAXMUS
MaxClasses 8 in ART @ tag=MAXART

# Disallow classes older than 10 years - but allow any ANTH to be older
MaxClasses 0 in @ (With DWAge>10)
  Except ANTH @ tag=MAX10

# A maximum of 4 classes can be applied to the requirement block
MaxClasses 4 in @ (With Attribute=WXYZ) tag=MAXCL
  HeaderTag MaxReason="You exceeded the limit of 4 classes"
```

## Notes

1. The connector in the course list following MaxClasses can be comma or "or". The plus sign or "and" is not allowed. If multiple courses are listed, then the sum of all classes that satisfy the list is compared against the maximum specified.
2. The MaxClasses qualifier is applied by the auditor against all the classes used to fill the block. If MaxClasses is in the starting block then it applies to the classes in all blocks and also those in the fall-through section. The number of classes in MaxClasses is a strict cap that cannot be exceeded.
3. If MaxClasses is in the starting block and the number of classes taken exceeds the maximum allowed the auditor will move the excess classes into the over-the-limit section.
4. The course list associated with MaxClasses allows Except but not Including.
5. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
6. For all Max qualifiers in your starting/degree block you should add a HeaderTag for MaxReason to ensure the Reason showing in the over-the-limit section indicates exactly why the class cannot be used in the audit. You can scribe a MaxReason HeaderTag on any block but the auditor ignores it on any block that is not the starting block.

---

## MaxCredits (header)

Updated: March 25, 2022

MaxCredits indicates the maximum number of credits that can be applied to a requirement block.

### Template

```
MaxCredits n in XXXX @ tag=MAXXXX
```

### Examples

```
MaxCredits 9 in MUS 1@ (With DWResident=Y) tag=MAXMUSIC
```

```
MaxCredits 3 in GEOL 109, 145 tag=MAXGEOL
```

```
MaxCredits 12 in PSY 100:199  
Except PSY 108, 117 tag=MAXPSY
```

```
# Max of 15 in MUS and ART combined  
MaxCredits 15 in MUS @, ART @ tag=MAXMUSART
```

```
# Max of 15 in MUS and max of 15 in ART  
MaxCredits 15 in MUS @ tag=MAXMUS  
MaxCredits 15 in ART @ tag=MAXART
```

```
# Disallow classes older than 10 years - but allow any ANTH to be o  
lder  
MaxCredits 0 in @ (With DWAge>10)  
Except ANTH @ tag=MAX10
```

```
# A maximum of 4 credits can be applied to the requirement block  
MaxCredits 4 in @ (With Attribute=WXYZ) tag=MAXCR  
HeaderTag MaxReason="You exceeded the limit of 4 credits"
```

### Notes

1. The connector in the course list following MaxCredits can be comma or "or". The plus sign or "and" is not allowed. If multiple courses are listed, then the sum of all credits for the classes that satisfy the list is compared against the maximum specified.

2. The MaxCredits qualifier is applied by the auditor against all the classes used to fill the block. If MaxCredits is in the starting block then it applies to the classes in all blocks and also those in the fall-through section. The number of credits in MaxCredits is a strict cap that cannot be exceeded. For example, if the maximum is 15 credits and four 4-credit classes match the course list, then one of them will be removed leaving only 12 credits applying. The SpMaxCredits qualifier also enforces a strict cap but it will split the last class moving 1 of its credits elsewhere allowing up to the maximum to apply.
3. If MaxCredits is in the starting block and the number of credits taken exceeds the maximum allowed the auditor will move the excess classes into the over-the-limit section.
4. The course list associated with MaxCredits allows Except but not Including.
5. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
6. For all Max qualifiers in your starting/degree block you should add a HeaderTag for MaxReason to ensure the Reason showing in the over-the-limit section indicates exactly why the class cannot be used in the audit. You can scribe a MaxReason HeaderTag on any block but the auditor ignores it on any block that is not the starting block.

## MaxPerDisc (header)

Updated: March 25, 2022

MaxPerDisc indicates the maximum number of credits/classes in each discipline listed that can be applied to a set of requirements.

### Template

```
MaxPerDisc x Credits (XXX, YYY, ZZZ) tag=MAXPD
```

### Examples

```
MaxPerDisc 3 Classes in (COMP) tag=MAXPDCOMP
```

```
# A maximum of 9 credits combined in chemistry, biology and physics
MaxPerDisc 9 Credits (CHEM, BIOL, PHYS) tag=MAXPDSCI
```

```
# A maximum of 9 credits in each area: chemistry, biology and physics
MaxPerDisc 9 Credits (CHEM) tag=MAXPDCHEM
MaxPerDisc 9 Credits (BIOL) tag=MAXPDBIOL
MaxPerDisc 9 Credits (PHYS) tag=MAXPDPHYS
```



```
# A maximum of 4 credits/classes can be applied to each discipline
listed
# that can be applied to a set of requirements
MaxPerDisc 4 Classes (ART) tag=MAXPD
HeaderTag MaxReason="You exceeded the limit of 4 ART classes"
```

## Notes

1. The MaxPerDisc qualifier is applied by the auditor against all the classes used to fill the block. If MaxPerDisc is in the starting block then it applies to the classes in all blocks and also those in the fall-through section. The number of classes and credits in MaxPerDisc is a strict cap that cannot be exceeded. For example, if the maximum is 15 credits and four 4-credit classes match the course list, then one of them will be removed leaving only 12 credits applying.
2. If MaxPerDisc is in the starting block and the number of credits or classes taken exceeds the maximum allowed the auditor will move the excess classes into the over-the-limit section.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
4. SameDisc may be combined with MaxPerDisc.
5. For all Max qualifiers in your starting/degree block you should add a HeaderTag for MaxReason to ensure the Reason showing in the over-the-limit section indicates exactly why the class cannot be used in the audit. You can scribe a MaxReason HeaderTag on any block but the auditor ignores it on any block that is not the starting block.

## MaxPerDisc (rule)

Updated: March 25, 2022

MaxPerDisc indicates the maximum number of credits/classes in each discipline listed that can be applied to a requirement.

## Template

```
MaxPerDisc x Credits (XXX, YYY, ZZZ) tag=MAXPD
```

## Examples

```
# Allow only 6 chemistry credits
16 Credits in BIO @, CHE @
MaxPerDisc 6 Credits in (CHE) tag=MAXPDCHE
Label SCI "Weird Science";
```

```
# Allow up to 3 art and music classes
19 Credits in @ (With Attribute=SAKI)
    MaxPerDisc 3 Classes (MUS, ART) tag=MPDMUSART
    Label SAKI "Sakinah Requirement";

# Allow up to 3 art classes and 3 music classes
19 Credits in @ (With Attribute=SAKI)
    MaxPerDisc 3 Classes (MUS) tag=MPDMUS
    MaxPerDisc 3 Classes (ART) tag=MPDART
    Label SAKI "Sakinah Requirement";

# Only 1 class can be used in any subject
BeginSub
    1 Class in @ @ (With ATTRIBUTE=AA) Label "AA requirement";
    1 Class in @ @ (With ATTRIBUTE=BB) Label "BB requirement";
    1 Class in @ @ (With ATTRIBUTE=CC) Label "CC requirement";
    1 Class in @ @ (With ATTRIBUTE=DD) Label "DD requirement";
    1 Class in @ @ (With ATTRIBUTE=EE) Label "EE requirement";
    1 Class in @ @ (With ATTRIBUTE=FF) Label "FF requirement";
EndSub
    MaxPerDisc 1 Class in (ALLDISCS)
    Label "Some requirement";
Remark "For some requirement, you are only allowed one class"
Remark "in each subject area."
```

## Notes

1. The MaxPerDisc qualifier is applied by the auditor against all the classes used to fill the requirement. The number of classes and credits in MaxPerDisc is a strict cap that cannot be exceeded. For example, if the maximum is 15 credits and four 4-credit classes match the course list, then one of them will be removed leaving only 12 credits applying.
2. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
3. SameDisc can be combined with MaxPerDisc.
4. Instead of creating a MaxPerDisc for each of the disciplines that could be used on the requirement, you can use the keyword **ALLDISCS** instead. This is equivalent to adding a MaxPerDisc qualifier for each of the disciplines. However, if your course rule or the rules in your subset or groups have wildcards specified, you may not know what disciplines may be used. In addition, if an exception is applied, then classes for additional disciplines may be entered into the mix. Using ALLDISCS, the auditor will find all of the classes currently on the rule and will apply a MaxPerDisc on each of the disciplines for the number of credits or classes specified. For example, if the student has taken one or more classes in the disciplines HIST, ACCT, and CHEM, the auditor will treat "MaxPerDisc 1 Class in (ALLDISCS)" as three separate qualifiers: MaxPerDisc 1 Class in (HIST), MaxPerDisc 1 Class in (ACCT), MaxPerDisc 1 Class in (CHEM). Whenever you have this maximum restriction on all of the disciplines in your rule, it is best to use ALLDISCS.

## MaxSpread (rule)

Updated: March 25, 2022

MaxSpread indicates the maximum number of disciplines from the course list in which courses can be taken.

### Template

```
MaxSpread N tag=MAXSPDXXX
```

### Examples

```
# Credits must be taken in at most two subjects
10 Credits in BIO @, CHE @, PHY @
  MaxSpread 2 tag=MAXSPREADSCI
  Label SCI "Weird Science";
```

### Notes

1. MaxSpread indicates the maximum number of disciplines from the course list that can be represented in the courses used to satisfy the requirement. If there are 3 disciplines listed and MaxSpread is 2, then the courses that satisfy the requirement can be from no more than 2 of the 3 disciplines.
2. MaxSpread can be used only within a course rule or subset.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
4. SameDisc may be combined with MaxSpread.

## MaxTerm (header)

Updated: March 25, 2022

MaxTerm indicates the maximum number of credits/classes that can be applied to a set of requirements each term.

### Template

```
MaxTerm N Credits in XXX @ tag=MAXTERMXXX
```

## Examples

```
MaxTerm 3 Credits in PE @ tag=MAXTERMPE
```

```
MaxTerm 2 Classes in ART @  
  Except ART 106    tag=MAXTERMART
```

```
# A maximum of 4 credits/classes can be applied to a set of require  
ments each term
```

```
MaxTerm 4 Classes in CHEM @ tag=MAXTM  
  HeaderTag MaxReason="You exceeded the limit of 4 CHEM classes"
```

## Notes

1. The number indicates the maximum number of classes or credits that can be taken per term. This number is a strict cap.
2. The course list associated with MaxTerm allows Except but not Including.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
4. For all Max qualifiers in your starting/degree block you should add a HeaderTag for MaxReason to ensure the Reason showing in the over-the-limit section indicates exactly why the class cannot be used in the audit. You can scribe a MaxReason HeaderTag on any block but the auditor ignores it on any block that is not the starting block.

## MaxTerm (rule)

Updated: March 25, 2022

MaxTerm indicates the maximum number of credits/classes that can be applied to a requirement each term.

## Template

```
MaxTerm N Credits tag=MAXTERMXXX
```

## Examples

```
12 Credits in MUS 299  
  MaxTerm 3 Credits tag=MAXTERMMUS299
```

```
Label MUS299 "Music Practice";
```

## Notes

1. The number indicates the maximum number of classes or credits that can be taken per term. This number is a strict cap.
2. MaxTerm can be used only within a course rule, group or subset.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## MaxTransfer (header)

Updated: March 25, 2022

MaxTransfer indicates the maximum number of transfer credits/classes that can be applied to a set of requirements.

## Template

```
MaxTransfer N Credits in (XXX, YYY) tag=MAXTRANXXXXYY
```

## Examples

```
MaxTransfer 36 Credits in (AP, CLEP) tag=MAXTRANAPCLEP
```

```
MaxTransfer 12 Classes tag=MAXTRAN
```

```
# Another way to scribe a transfer limit is to use MaxClasses or MaxCredits
```

```
MaxClasses 15 in PE @ (With DWTransfer=Y) tag=MAXPE15
```

```
# A maximum of 4 transfer credits/classes can be applied to a set of requirements
```

```
MaxTransfer 4 Classes tag=MAXTF
```

```
HeaderTag MaxReason="You exceeded the limit of 4 transfer classes"
```

## Notes

1. The number indicates the maximum number of classes or credits that transferred in from another institution. This number is a strict cap.
2. An optional list of transfer types can specify that the maximum applies to these transfer types only. Transfer type is also known as credit type.
3. The transfer type codes must be valid in UCX-STU355.
4. If MaxTransfer is in the starting block and the number of credits or classes taken exceeds the maximum allowed the auditor will move the excess classes into the over-the-limit section. When it is in the starting block it applies to all classes applying to all blocks and the fall-through section.
5. Instead of scribing MaxTransfer you might consider scribing MinRes.
6. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
7. For all Max qualifiers in your starting/degree block you should add a HeaderTag for MaxReason to ensure the Reason showing in the over-the-limit section indicates exactly why the class cannot be used in the audit. You can scribe a MaxReason HeaderTag on any block but the auditor ignores it on any block that is not the starting block.

## MaxTransfer (rule)

Updated: March 25, 2022

MaxTransfer indicates the maximum number of transfer credits/classes that can be applied to a requirement.

## Template

```
MaxTransfer N Credits in (XXX, YYY) tag=MAXTRANXXXXYY
```

## Examples

```
12 Credits in MUS 1@
MaxTransfer 3.5 Credits tag=MAXTRANMUS
Label MUS1 "Music 100-level";
```

```
20 Credits in MUS 1@
MaxTransfer 3 Classes in (SA, IO) tag=MAXTRANMUS
Label MUS1 "Music 100-level";
```

```
# Do not allow any transfer classes
15 Credits in CSCI @ (With DWTransfer=N)
Label CSCI "Computer Science";
```

## Notes

1. The number indicates the maximum number of classes or credits that transferred in from another institution. This number is a strict cap.
2. An optional list of transfer types can specify that the maximum applies to these transfer types only. Transfer type is also known as credit type.
3. The transfer type codes must be valid in UCX-STU355.
4. Instead of scribing MaxTransfer you might consider scribing DWTransfer.
5. MaxTransfer can be used only within a course rule, group or subset.
6. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## MinAreas (rule)

Updated: March 25, 2022

MinAreas indicates the minimum number of areas from a course list to which classes will be applied.

## Template

```
MinAreas N tag=MINAREASXXX
```

## Examples

```
# 12 credits needed across 2 areas; at least 1 class in each area
12 Credits in [PSY 130, 135, BIO 156,]
               [BIO 145, 177, CHE 205 ]
MinAreas 2 tag=MINAREASSCI
Label SCI1 "Sci-ence I";
```

```
# 10:15 credits needed across 3 areas; at least 1 class in at least
2 areas
10:15 Credits in
```

```
[{Hide MATH 103}, STAT 123, 124]
[CHEM 103, 104, 105,]
[BIO 103, 104, 105 ]
MinAreas 2 tag=MINAREASSCI2
Label SCI2 "Science II";
```

## Notes

1. MinAreas indicates the minimum number of areas from the list that can be represented in the courses used to satisfy the requirement. If there are 3 areas defined and MinAreas is 2, then the classes that satisfy the requirement must be from at least 2 of the 3 areas.
2. MinAreas is used in place of MinPerDisc whenever the areas cover more than one discipline. MinAreas works very much like MinPerDisc except that the brackets define the areas for the former while the discipline defines the areas for the latter.
3. When you use {Hide} inside an area-type rule with [ ], you need to place the Hide as the first course in the rule, inside the starting square bracket "[{". The curly brace "}" cannot be followed immediately by the square bracket "]".
4. MinAreas can be used only within a course rule.
5. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## MinClasses (header)

Updated: March 25, 2022

MinClasses indicates the minimum number of courses that must be earned to satisfy the requirement.

## Template

```
MinClasses N in XXXX @ tag=MINCLXXX
ProxyAdvice "You have taken <APPLIED> classes but"
ProxyAdvice "still need <NEEDED> more."
```

## Examples

```
MinClasses 8 in PE @ tag=MINPE
ProxyAdvice "You have taken <APPLIED> PE classes but"
ProxyAdvice "still need <NEEDED> more."
```



```
MinClasses 8 in POLI 1@ (With DWResident=Y)
  ProxyAdvice "You have taken <APPLIED> 100-level political science"
  ProxyAdvice "classes but still need <NEEDED> more."
  Label MINPOLI "8 Political Science classes"
```

```
MinClasses 3 from PSY @
  Except PSY 112      tag=MINPSY
  ProxyAdvice "You have taken <APPLIED> PSY classes but"
  ProxyAdvice "still need <NEEDED> more. (PSY 112 cannot count)"
```

```
MinClasses 8 in BUSN @, ACCT @ tag=MINBUSNACCT
  Display "You have taken <APPLIED> business and accounting credits."
  ProxyAdvice "You have taken <APPLIED> business and accounting"
  ProxyAdvice "classes but still need <NEEDED> more."
```

## Notes

1. The connector in the course list following MinClasses can be a comma or "or". The plus sign or "and" is not allowed. If multiple courses are listed, then the sum of all classes that satisfy the list is compared against the maximum specified.
2. The MinClasses qualifier is applied by the auditor against all the classes used to fill the block. If MinClasses is in the starting block then it applies to the classes in all blocks and also those in the fall-through section. The number of classes in MinClasses is a strict cap that cannot be exceeded.
3. If MinClasses is in the starting block and the number of classes taken exceeds the maximum allowed the auditor will move the excess classes into the over-the-limit section.
4. The course list associated with MinClasses allows Except but not Including.
5. ProxyAdvice should always be used because the default advice is much less helpful.
6. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made. Or if the qualifier has a label then a label-tag will suffice.

## MinClasses (rule)

Updated: March 25, 2022

MinClasses specifies the minimum number of classes that must be applied to the rule.

## Template

```
MinClasses N in XXXX 999, 998, 997 tag=MINCLXXX
```

## Examples

```
12 Classes in BUS 3@, ACCT 3@  
  MinClasses 1 in BUS 321, ACCT 306 tag=MINBUS321306  
  MinClasses 1 in BUS 327, ACCT 312 tag=MINBUS327312  
  Label ACCTELECT "Accounting electives";  
  
5 Classes in HIST 2@ (With Attribute=WRIT), ENGL 300:319  
  MinClasses 2 in HIST 206, ENGL 311, 316 tag=MINHISTENGL  
  Label WRIT "Writing intense option";
```

## Notes

1. If the minimum number of classes has not been taken the auditor will remove classes from the rule to make room for the classes that do need to be taken to satisfy the MinClasses that are still needed.
2. MinClasses can be used only within a course rule.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## MinCredits (header)

Updated: March 25, 2022

MinCredits indicates the minimum number of courses that must be earned to satisfy the requirement.

## Template

```
MinCredits N in XXXX @ tag=MINCRXXX  
  ProxyAdvice "You have taken <APPLIED> credits but"  
  ProxyAdvice "still need <NEEDED> more."
```

## Examples

```

MinCredits 8 in PE @ tag=MINPE
ProxyAdvice "You have taken <APPLIED> PE classes but"
ProxyAdvice "still need <NEEDED> more."

MinCredits 8 in POLI 1@ (With DWResident=Y)
ProxyAdvice "You have taken <APPLIED> 100-level political scienc
e"
ProxyAdvice "credits but still need <NEEDED> more." Label MINPOL
I "8 Political Science credits"

MinCredits 3 from PSY @
Except PSY 112 tag=MINPSY
ProxyAdvice "You have taken <APPLIED> PSY credits but"
ProxyAdvice "still need <NEEDED> more. (PSY 112 cannot count)"

MinCredits 8 in BUSN @, ACCT @ tag=MINBUSNACCT
Display "You have taken <APPLIED> business and accounting credit
s."
ProxyAdvice "You have taken <APPLIED> business and accounting"
ProxyAdvice "credits but still need <NEEDED> more."

MinCredits 52 in @
ProxyAdvice "You have taken <APPLIED> breadth"
ProxyAdvice "credits but still need <NEEDED> more."
Label MINBREADTH "52 breadth credits"
HeaderTag DWRangeLimit=Y

```

## Notes

1. The connector in the course list following MinCredits can be a comma or "or". The plus sign or "and" is not allowed. If multiple courses are listed, then the sum of all credits that satisfy the list is compared against the maximum specified.
2. The MinCredits qualifier is applied by the auditor against all the credits used to fill the block. If MinCredits is in the starting block then it applies to the credits in all blocks and also those in the fall-through section. The number of credits in MinCredits is a strict cap that cannot be exceeded.
3. If MinCredits is in the starting block and the number of credits taken exceeds the maximum allowed the auditor will move the excess classes into the over-the-limit section.
4. The course list associated with MinCredits allows Except but not Including.
5. ProxyAdvice should always be used because the default advice is much less helpful.

6. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made. Or if the qualifier has a label then a label-tag will suffice.
7. When a HeaderTag of DWRangeLimit (with any value on the right side of the equals) appears immediately after the MinCredits this special logic applies:
  - a. The credits in each rule in the same block (credits in included blocks are ignored) are summed with the credits exceeding the upper bound on the credit range being ignored. Example, a rule with 12:20 Credits in... will only count up to, but not exceeding, 20 credits toward the total for this qualifier.
  - b. Although you can scribe this HeaderTag after MinCredits in the starting/degree block you will not get the correct results. This should only be used in a non-starting block.
  - c. This qualifier either shows as complete (green check box) or incomplete (red check box). If the total of credits are met and in-progress classes are applying to rules, the qualifier will still be marked as complete; it will not be marked as in-progress complete.
  - d. It is advisable, but not required, that the UCX-CFG020 DAP14 **Show Advice on Range Rules** flag be set to Y so that your rules show the advice while this, or any header qualifier, has not been met.
  - e. When Allow is used to specify a different range of credits, the upper range value that is scribed at the start of the rule will be used and not the value in the Allow range.

## MinCredits (rule)

Updated: March 25, 2022

MinCredits specifies the minimum number of credits that must be applied to the rule.

### Template

```
MinCredits N in XXXX 999, 998, 997 tag=MINCRXXX
```

### Examples

```
12 Credits in BUS 3@, ACCT 3@
MinCredits 1 in BUS 321, ACCT 306 tag=MINBUS321306
MinCredits 1 in BUS 327, ACCT 312 tag=MINBUS327312
Label ACCTELECT "Accounting electives";

5 Credits in HIST 2@ (With Attribute=WRIT), ENGL 300:319
MinCredits 2 in HIST 206, ENGL 311, 316 tag=MINHISTENGL
Label WRIT "Writing intense option";
```

## Notes

1. If the minimum number of credits has not been taken the auditor will remove classes from the rule to make room for the classes that do need to be taken to satisfy the MinCredits that are still needed.
2. MinCredits can be used only within a course rule.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## MinGPA (header)

Updated: September 29, 2023

MinGPA indicates the minimum Grade Point Average for the requirement block.

## Template

```
MinGPA n.n @ tag=MINGPA
  ProxyAdvice "A <REQUIRED> GPA is required but your GPA is <APPLIED>."
```

## Examples

```
MinGPA 2.0 tag=GPA20
  ProxyAdvice "A <REQUIRED> GPA is required but your GPA is <APPLIED>."
```

```
MinGPA 2.0 tag=GPA20
  Display "Your major GPA is <APPLIED>."
  ProxyAdvice "A 2.0 GPA is required but your GPA is <APPLIED>."
```

```
MinGPA 3.0 in CHEM @, PHYS @, BIOL @
  Except CHEM 109, BIOL 243, 244 tag=GPA20SCIENCE
  ProxyAdvice "A <REQUIRED> GPA is required in science but your GPA is <APPLIED>."
```

```
# This financial aid award requires a 2.0 in the major
MinGPA 2.0 in (MAJOR) tag=GPAMAJOR
  ProxyAdvice "A 2.0 major GPA is required but your GPA is <APPLIED>."
```

```
MinGPA 2.0
  ProxyAdvice "A <REQUIRED> GPA is required but your GPA is <APPLIE
D>."
  Label GPA "GPA requirement"

# Student System GPA; Banner sites may prefer to use BannerGPA
If (SSGPA > 2.0) Then
  RuleComplete
  Label GPAGOOD "2.0 GPA requirement met"
Else #insufficient GPA
  RuleIncomplete
  ProxyAdvice "Your GPA is below the 2.0 required"
  Label GPABAD "2.0 GPA requirement not met";
```

## Notes

1. MinGPA indicates the minimum GPA for all courses used in the block. A GPA is calculated for the block by summing the grade points for all courses applied to the block, summing the graded credits earned for all courses applied to the block, and dividing the total grade points by the total graded credits earned.
2. When MinGPA is in the degree block, classes in fall-through are counted only if **Fall-through courses count in the overall GPA** is set to Yes in CFG020 DAP14. All classes in the insufficient section are also included in the degree GPA calculation.
3. Failed classes that could have applied to the major or minor or an OTHER block may be included in the GPA calculation depending on the UCX-CFG020 DAP14 **Failed classes count in GPA** flag settings.
4. In the degree block you may want to instead write an If statement that checks the SISGPA (or BannerGPA) and use RuleComplete or RuleIncomplete to give an appropriate message.
5. ProxyAdvice should always be used because the default advice is much less helpful.
6. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made. Or if the qualifier has a label then a label-tag will suffice.

## MinGPA (rule)

Updated: March 25, 2022

MinGPA indicates the minimum Grade Point Average for the classes applied to the rule.

## Template

```
MinGPA n.n tag=MINGPAXXX
```

## Examples

```
12 Credits in BUS 3@, ACCT 3@
MinGPA 2.5 tag=MINGPABUSACCT
Label ACCTELECT "Accounting electives";
```

```
BeginSub
  6 Credits in HIST 2@
  Label HIST2 "History 200-level";
  1 Class in HIST 1916
  Label HISTEIRE "History of Ireland";
EndSub
MinGPA 2.5 tag=HISTGPA
Label HIST "History requirement";
```

## Notes

1. MinGPA indicates the minimum GPA for all courses used in the rule. A GPA is calculated for the rule by summing the grade points for all courses applied to the rule, summing the graded credits earned for all courses applied to the rule, and dividing the total grade points by the total graded credits earned.
2. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## MinGrade (header)

Updated: March 24, 2023

MinGrade indicates the minimum numeric grade that must be achieved for each course applied to the block.

## Template

```
MinGrade n.n tag=MINGRADE
```

## Examples

```
# Classes with low grades go to the insufficient section
MinGrade 1.5 tag=GRADE15
```

```
# Classes with low grades go to the over-the-limit section
```

```
MaxClasses 0 in @ (With DWGrade<1.5) tag=GRD15
```

## Notes

1. MinGrade can be used to indicate the minimum passing grade that must be achieved for a requirement to be satisfied.
2. When MinGrade is in the starting block, a class with an insufficient grade that could have applied to any block is instead placed into the insufficient section and is counted in the Overall GPA. If the class could not have applied to any rules the class will end up in fall-through and will still count in the overall GPA.
3. When MinGrade is not in the starting block and a class that could have applied to the block is restricting from doing so because it has an insufficient grade the class will be counted in this block's GPA. The UCX-CFG020 DAP14 **MinGrade insufficient classes count in GPA** flag controls this behavior however.
4. You can instead use MaxCredits or MaxClasses and DWGrade to throw out classes with low grades so they do not count in the GPA.
5. In-progress classes always pass the MinGrade check. When DWGrade or DWGradeNumber are used, both in-progress and passed pass-fail classes are given a grade of 4.0.
6. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
7. Unlike other header qualifiers, the MinGrade (header) qualifier applies only to the block where it is scribed. It does not affect any child blocks that are called into the block. However, MinGrade in the starting/degree block does affect all blocks and fall-through.

## MinGrade (rule)

Updated: March 24, 2023

MinGrade indicates the minimum numeric grade that must be achieved for each course applied to the rule.

## Template

```
MinGrade n.n tag=MINGPAXXX
```

## Examples

```
12 Credits in BUS 3@, ACCT 3@
```



```
MinGrade 2.5 tag=MINGRADEABUSACCT
Label ACCTELECT "Accounting electives";

12 Credits in BUS 3@ (With Hide DWGrade>=2.5),
                ACCT 3@ (With Hide DWGrade>=2.0)
Label ACCTELECT2 "Accounting electives";
```

## Notes

1. MinGrade can be used to indicate the minimum passing grade that must be achieved for a requirement to be satisfied.
2. You may instead use DWGrade in With to control the minimum grade that is required.
3. In-progress classes always pass the MinGrade check. When DWGrade or DWGradeNumber are used, both in-progress and passed pass-fail classes are given a grade of 4.0.
4. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## Minor (header)

Updated: March 25, 2022

When Minor is used in ShareWith it refers to the Minor block in the audit. When used in an If-statement it refers to the minor on the student's curriculum.

## Examples

```
ShareWith (Minor)

If (Minor = XY) Then
    MinGPA 2.5 tag=MINGPA
    ProxyAdvice "You need a GPA 2.5 but currently have a <APPLIED> GP
A"
```

## Notes

1. When Minor is used within a block header, it is only used with an If-statement or ShareWith.

## Minor (rule)

Updated: March 25, 2022

When Minor is used in ShareWith, Blocktype or Block it refers to the Minor block in the audit.  
When used in an If-statement it refers to the minor on the student's curriculum.

### Examples

```
1 Blocktype (Minor)
  Label MNRBLOCK "Minor requirements";

# Only if the student has the SPAN minor on their curriculum will t
his block be used
1 Block (Minor=SPAN)
  Label MNRBLOCK "Spanish minor requirements";

If (Minor = RPD) Then
  3 Credits in MATH 400
  Label MA400 "Math 400";

6 Credits in BUS 1@
ShareWith (Minor)
Label BUS "Business";
```

### Notes

1. Minor is followed by a minor code except in a Blocktype rule and is optional in ShareWith.
2. The minor code must be valid in UCX-STU024.
3. Minor can be used with Blocktype, Block, If-statement, or ShareWith.
4. When Minor is used in a Block rule, the specific minor block specified will only be included in the audit if the student has that minor on their curriculum. For this reason, it is rare for use a Block rule with Minor.

## MinPerDisc (header)

Updated: March 25, 2022

MinPerDisc indicates the minimum number of credits/classes in each discipline listed that can be applied to a set of requirements.

---

## Template

```
MinPerDisc x Credits (XXX, YYY, ZZZ) tag=MINPD
ProxyAdvice "You have taken <APPLIED> XXX. YYY, ZZZ"
ProxyAdvice "credits but still need <NEEDED> more."
```

## Examples

```
MinPerDisc 3 Classes in (COMP) tag=MINPDCOMP
ProxyAdvice "You have taken <APPLIED> COMP classes but"
ProxyAdvice "still need <NEEDED> more."
```

```
# A minimum of 9 credits combined in chemistry, biology and physics
MinPerDisc 9 Credits (CHEM, BIOL, PHYS) tag=MINPDSCI
ProxyAdvice "You have taken <APPLIED> chemistry, biology and physics"
ProxyAdvice "credits but still need <NEEDED> more."
```

```
# A minimum of 9 credits in each area: chemistry, biology and physics
MinPerDisc 9 Credits (CHEM) tag=MINPDCHEM
ProxyAdvice "You have taken <APPLIED> chemistry"
ProxyAdvice "credits but still need <NEEDED> more."
MinPerDisc 9 Credits (BIOL) tag=MINPDBIOL
ProxyAdvice "You have taken <APPLIED> biology "
ProxyAdvice "credits but still need <NEEDED> more."
MinPerDisc 9 Credits (PHYS) tag=MINPDPHYS
ProxyAdvice "You have taken <APPLIED> physics"
ProxyAdvice "credits but still need <NEEDED> more."
```

## Notes

1. The MinPerDisc qualifier is applied by the auditor against all the classes used to fill the block. If MinPerDisc is in the starting block then it applies to the classes in all blocks and also those in the fall-through section.
2. ProxyAdvice should always be used because the default advice is much less helpful.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made. Or if the qualifier has a label then a label-tag will suffice.
4. SameDisc may be combined with MinPerDisc.

## MinPerDisc (rule)

Updated: March 25, 2022

MinPerDisc indicates the minimum number of credits/classes in each discipline listed that can be applied to a requirement.

### Template

```
MinPerDisc x Credits (XXX, YYY, ZZZ) tag=MINPD
```

### Examples

```
# Require 6 chemistry credits
16 Credits in BIO @, CHE @
  MinPerDisc 6 Credits in (CHE) tag=MINPDCHE
  Label SCI "Weird Science";

# Require 3 art and music classes
19 Credits in @ (With Attribute=SAKI)
  MinPerDisc 3 Classes (MUS, ART) tag=MPDMUSART
  Label SAKI "Sakinah Requirement";

# Require 3 art classes and 3 music classes
19 Credits in @ (With Attribute=SAKI)
  MinPerDisc 3 Classes (MUS) tag=MPDMUS
  MinPerDisc 3 Classes (ART) tag=MPDART
  Label SAKI "Sakinah Requirement";
```

### Notes

1. The MinPerDisc qualifier is applied by the auditor against all the classes used to fill the requirement.
2. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
3. SameDisc may be combined with MinPerDisc.

## MinRes (header)

Updated: March 25, 2022

MinRes indicates the minimum number of credits/classes that must be earned in residence.

### Template

```
MinRes x Credits tag=MINRES
ProxyAdvice "You have taken <APPLIED> credits in residence"
ProxyAdvice "but still need <NEEDED> more."
```

### Examples

```
MinRes 12 Classes tag=MINRES
ProxyAdvice "You have taken <APPLIED> classes in residence"
ProxyAdvice "but still need <NEEDED> more."
```

```
MinRes 36 Credits tag=MINRES
ProxyAdvice "You have taken <APPLIED> credits in residence"
ProxyAdvice "but still need <NEEDED> more."
```

```
MinRes 36 Credits
ProxyAdvice "You have taken <APPLIED> credits in residence"
ProxyAdvice "but still need <NEEDED> more."
Label MINRES "Residency Requirement"
```

### Notes

1. The MinRes qualifier is applied by the auditor against all the classes used to fill the block. If MinRes is in the starting block then it applies to the classes in all blocks and also those in the fall-through section.
2. ProxyAdvice should always be used because the default advice is much less helpful.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made. Or if the qualifier has a label then a label-tag will suffice.

## MinSpread (rule)

Updated: March 25, 2022

MinSpread indicates the minimum number of disciplines from the course list in which courses

must be earned to satisfy the requirement.

### Template

```
MinSpread x tag=MINSPDXXYYZZ
```

### Examples

```
12 Credits in PSY @, BIO @, CHE @  
MinSpread 2 tag=MINSPDSCI  
Label SCIENCE "Science";  
  
3 Classes in MATH @, HIST @, PHIL @  
MinSpread 2 tag=MINSPDFRESH  
Label FRESHSPECT "Freshman Spectrum";
```

### Notes

1. MinSpread indicates the minimum number of disciplines from the list that can be represented in the courses used to satisfy the requirement. If there are 3 disciplines listed and MinSpread is 2, then the courses that satisfy the requirement must be from at least 2 of the 3 disciplines.
2. MinSpread can be used only with a course rule or subset.
3. MinSpread can be combined with SameDisc.
4. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## MinTerm (header)

Updated: March 25, 2022

MinTerm indicates the minimum number of credits/classes that must be taken per term.

### Template

```
MinTerm 1 Class in XXX @ tag=MINTERMXXX  
ProxyAdvice "You are required to take at least"  
ProxyAdvice "one XXX class per term."
```

## Examples

```
MinTerm 1 Class in PE @ tag=MINTERMPE
ProxyAdvice "You are required to take at least"
ProxyAdvice "one PE class per term."
```

```
MinTerm 3 Credits in MUS 1@ tag=MINTERMMUSIC
Except MUS 106
ProxyAdvice "You are required to take at least"
ProxyAdvice "three music credits per term."
```

```
MinTerm 6 Credits
ProxyAdvice "You are required to take at least"
ProxyAdvice "six credits per term."
Label MINTERM6 "6 credits per term"
```

## Notes

1. When used in the block header, MinTerm is followed by the number of classes or credits, optionally followed by "in", followed by a course list in which comma/or is allowed but plus/and is not allowed.
2. ProxyAdvice should always be used because the default advice is much less helpful.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made. Or if the qualifier has a label then a label-tag will suffice.

## MinTerm (rule)

Updated: March 24, 2023

MinTerm indicates the minimum number of credits/classes that must be taken per term.

## Template

```
MinTerm x Credits tag=MINTERM
```

## Examples

```
16 Credits in BIO @, CHE @
MinTerm 2 Credits tag=MINTERM
Label SCI "Weird Science";
```

```
# In each term the student has taken classes at least one class must be taken with this attribute
12 Classes in @ (With Attribute=SAKI)
  MinTerm 1 Class tag=MINTERM1
  Label SAKI "Sakinah Requirement";
```

## Notes

1. MinTerm can only be used if all of a student's courses and transfer credits have been linked to a term.
2. MinTerm can be used only with a course rule.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## NoCount

Updated: March 25, 2022

NoCount allows courses to satisfy specific requirements without affecting the total credit count or GPA calculation.

## Template

```
NoCount
```

## Examples

```
2 Classes in MIL 100:299 NoCount, HIST 100:299
  Label HISTMIL "History and Military";
```

```
5 Classes in PE @ NoCount (With Attribute=ROTC), ROTC @ NoCount
  Label PEROTC "PE/ROTC";
```

## Notes

1. NoCount is only allowed on courses specified in a class rule. NoCount is not allowed with a course rule specified with credits or with classes and/or credits. The class rule must be expressed in terms of Classes.
2. NoCount can be listed only one time per course. If multiple courses are used, then place



NoCount after each course.

3. For ranges of course numbers, NoCount applies to all courses in the range. For example, "ART 100:102 NoCount" is the same as "ART 100 NoCount, ART 101 NoCount, ART 102 NoCount".
4. Only use NoCount after a course if you do not want the auditor to count the course in the number of credits and GPA. The NoCount keyword will mask courses applied to the requirement by using a credit value of 0. Rule qualifiers on a rule with the NoCount keyword will count classes applied to the requirement as a class but will treat these courses as 0 credit courses.
5. The NoCount courses are invisible to all block qualifiers. They are not counted in a block's overall class or credit counts.
6. NoCount courses do not count in a rule's GPA calculation, a block's GPA calculation, or in the overall GPA calculation.
7. A course that is applied to a count rule has higher priority over a NoCount location.
8. Using NoCount and ShareWith together could yield unpredictable results.

## NonCourse (rule)

Updated: March 25, 2022

NonCourse indicates a required non-course activity, such as a thesis, recital, or exam.

### Template

```
1 NonCourse (XXXX)
  ProxyAdvice "You must take the xxxx requirement"
  Label NONCXXXX "xxxxxxx Requirement";
```

### Examples

```
1 NonCourse (LIFE, COMM)
  ProxyAdvice "You need to choose one of these and complete it"
  Label LIFECOMM "Life Issues or Community Service";
```

```
1 NonCourse (CHAPEL > 50)
  ProxyAdvice "You need to attend chapel 50 times or more"
  Label CHAPEL "Chapel Attendance 50 times or more";
```

```
2 NonCourses (LIFE, COLL, COMM)
```

```
ProxyAdvice "You need 2 human services"  
Label HUMAN "Human Services - 2 required";  
  
# 2 recitals with a DONE status are required.  
2 NonCourses (RECITAL = DONE)  
ProxyAdvice "You need 2 completed recitals"  
Label RECITAL "Recital";
```

## Notes

1. The list of noncourses must be enclosed in parentheses. The number specifies how many of the noncourses listed are required. The number of noncourse codes listed in parentheses is not checked against the number specified. The number can be less than, equal to, or greater than the number of noncourse codes listed.
2. Multiple noncourses can be listed if separated by a comma. To specify that multiple, different noncourses are required, enter a NonCourse rule for each noncourse.
3. The code following NonCourse must appear in the institution's noncourse table (UCX-SCR003), where it is mapped to a field in the database.
4. NonCourses includes noncourse values. You can choose to ignore the value or you can use equals, not-equals, greater-than or less-than to check the value. Example, 1 NonCourse (Thesis > 67) – the student not only must have the THESIS noncourse but also must have a value greater than 67.
5. Allowable rule qualifiers: none.
6. ProxyAdvice should always be used because the default advice is much less helpful.

## NonExclusive (header)

Updated: March 25, 2022

NonExclusive can be used in place of ShareWith; it was the original name used. See [ShareWith \(header\)](#).

## NonExclusive (rule)

Updated: March 25, 2022

NonExclusive can be used in place of ShareWith; it was the original name used. See [ShareWith \(rule\)](#).

---

## NotGpa (rule)

Updated: March 25, 2022

NotGPA indicates courses that count in neither the block GPA nor the overall GPA.

### Template

```
NotGPA
```

### Examples

```
6 Credits in INS1 @
NotGPA
Label INTSTUD "International Studies";

1 Block (Other=NURSAAS)
NotGPA
Label NURSAAS "Entry Requirements";
```

### Notes

1. NotGPA indicates that the credits applied towards satisfying the rule should not be counted in the block GPA.
2. The Auditor Engine will calculate one GPA per block for a student. The NotGPA keyword indicates that the courses used to satisfy a particular rule should count toward neither the block GPA nor the overall GPA.
3. Degree Works will not calculate the cumulative GPA for a student. The student system calculates the cumulative GPA, and Degree Works then receives the cumulative GPA from the student system.
4. It is possible to construct special requirements blocks that serve only to calculate a GPA for a group of courses. For example, a science GPA is desired but there is no requirements block for science. Build a requirements block for science (OTHER=SCIENCE), put ShareWith in the block header, and create the rules that list the courses to be used in the science GPA. Then, in the blocks for specific science majors (e.g., MAJOR=BIO, MAJOR=CHEM, MAJOR=PHYS), add the following rule "1 Block (OTHER=SCIENCE)".
5. By placing NotGPA on a Block rule you are telling the auditor not to count the classes in the referenced block in the parent block's GPA calculation. However, this does not work if the parent block is the degree block because all classes are included in the overall or degree GPA calculation.

6. If a course is on one rule with NotGPA and on another rule without NotGPA, then the course will be counted in the block's GPA.

## NumConcs

Updated: March 25, 2022

NumberOfConcentrations is used in an if-statement to find out how many concentration blocks are in the current audit.

### Template

```
If (NumberOfConcentrations >= n) Then
```

### Examples

```
If (Concentration=COMM and NumberOfConcentrations = 1) Then  
  1 BlockType (Minor)  
  Label MINOR "Minor Requirements";
```

### Notes

1. NumConcs may be used in place of NumberOfConcentrations.
2. The auditor examines the count of concentration blocks that were pulled into the audit; it does not count the number of concentrations on the student's academic record – though normally the counts should match.

## NumMajors

Updated: March 25, 2022

NumberOfMajors is used in an if-statement to find out how many major blocks are in the current audit.

### Template

```
If (NumberOfMajors >= n) Then
```

## Examples

```
If (Major=COMM and NumberOfMajors = 1) Then  
  1 BlockType (Minor)  
    Label MINOR "Minor Requirements";
```

## Notes

1. NumMajors may be used in place of NumberOfMajors.
2. The auditor examines the count of major blocks that were pulled into the audit; it does not count the number of majors on the student's academic record – though normally the counts should match.

## NumMinors

Updated: March 24, 2023

NumberOfMinors is used in an if-statement to find out how many minor blocks are in the current audit.

## Template

```
If (NumberOfMinors >= n) Then
```

## Examples

```
If (Minor=COMM and NumberOfMinors = 1) Then  
  1 BlockType (Minor)  
    Label MINOR "Minor Requirements";
```

## Notes

1. NumMinors may be used in place of NumberOfMinors.
2. The auditor examines the count of minor blocks that were pulled into the audit; it does not count the number of minors on the student's academic record – though normally the counts should match.

## Optional (header)

Updated: March 25, 2022

Optional indicates a block that does not have to be completed for the overall degree to be completed.

### Template

```
Optional
```

### Examples

```
Optional
```

### Notes

1. Optional indicates that the block should be evaluated but that the degree can still be awarded if the requirements are not met.
2. Optional applies to the entire block, not just a specific rule or qualifier.
3. The credits in an Optional block are treated as non-required credits when dealing with the Elective Credits Allowed calculation.

## Or (header)

Updated: March 25, 2022

Or is a boolean operator and also a connector in a list of items.

### Template

```
Or
```

### Examples

```
# Block needs at least 36 credits or at least 12 classes  
36 Credits or 12 Classes
```

```
If (MAJOR=PE or MAJOR=HE) Then
  MaxClasses 20 in PE @
```

## Notes

1. Or can connect Credits and Classes.
2. Or can connect two expressions in an If-statement.
3. When there is a list of items (courses, codes, etc.) you can use a comma, or you can use "or".

## Or (rule)

Updated: March 25, 2022

Or is a boolean operator and also a connector in a list of items.

## Template

Or

## Examples

```
6 Credits or 3 Classes in MAT @
Label MATH "Math";
```

```
If (MAJOR=PE or MAJOR=HE) Then
  RuleComplete
  Label ED "Education";
```

```
1 Class in INS 101, 102
Label INTSTUD "International Studies";
```

```
1 Class in INS 101 or 102
Label INTSTUD "International Studies";
```

```
1 Group in
  (6 Credits in FRE @, SPA @, GER @
   Label "French/Spanish/German") or
  (9 Credits in GRK @, LAT @
   Label "Greek/Latin")
```

```
Label LANGUAGE "LANGUAGE";
```

### Notes

1. Or can connect Credits and Classes.
2. Or can connect rules within a group rule.
3. Or can connect two expressions in an If-statement.
4. When there is a list of items (courses, codes, etc) you may use a comma or you may use "or".

## Other (header)

Updated: March 25, 2022

When Other is used in ShareWith it refers to the Other block in the audit.

### Template

```
ShareWith (Other=xxxx)
```

### Examples

```
ShareWith (Other=GENED)
```

```
DontShare (Other=GENED)
```

### Notes

1. When Other is used within a block header, it is only used with an If-statement or ShareWith or DontShare.

## Other (rule)

Updated: March 25, 2022

When Other is used in ShareWith it refers to the specified Other block in the audit. When used in an If-statement it refers to the Other on the student's curriculum.



## Examples

```
If (Other = PLAN) Then
  1 Block (Other=PLAN)
    Label PLAN "Planned Requirements"
else
  1 Block (Other=GENED)
    Label PLAN "Gen Ed Requirements";

6 Credits in BUS 1@
  ShareWith (Other=CORE)
  Label BUS "Business";
```

## Notes

1. Other is followed by an Other code that already is saved as a block.
2. The Other code must exist as an OTHER block in the database.
3. Other can be used with If-statement or ShareWith.

## Previous

Updated: March 24, 2023

PREVIOUS translates to the term before the active term on the student's record. Typically this is used with the Financial Aid audit.

## Template

```
MinCredits nn in XXX @ (With DWTerm=PREVIOUS)
```

## Examples

```
MinCredits 12 in @ (With DWTerm=PREVIOUS)
```

## Notes

1. Maps to the term before the rad\_term on the rad\_student\_mst.
2. Although PREVIOUS is typically used on the Financial Aid audit it could be used on any type of audit in any WITH statement using DWTerm.

3. PREVIOUS must be all uppercase.

## Previous2TermsEarnedCredits

Updated: March 25, 2022

These credits are taken from the classes in the previous two terms.

### Template

```
if (Previous2TermsEarnedCredits > nn) then
```

### Examples

```
if (Previous2TermsEarnedCredits > 18) then
  RuleComplete
  Label PREV2 "18 credits required last two terms";
```

### Notes

1. These credits are automatically calculated for Athletic Eligibility audits.

## Previous3TermsEarnedCredits

Updated: March 25, 2022

These credits are taken from the classes in the previous three terms.

### Template

```
if (Previous3TermsEarnedCredits > nn) then
```

### Examples

```
if (Previous3TermsEarnedCredits > 18) then
  RuleComplete
  Label PREV2 "18 credits required last three terms";
```

**Notes**

1. These credits are automatically calculated for Athletic Eligibility audits.

**PreviousAcademicYearEarnedCredits**

Updated: March 25, 2022

These credits are taken from the classes in the previous year – excluding summer.

**Template**

```
if (PreviousAcademicYearEarnedCredits > nn) then
```

**Examples**

```
if (PreviousAcademicYearEarnedCredits > 18) then  
  RuleComplete  
  Label PREV2 "18 credits required in previous year";
```

**Notes**

1. These credits are automatically calculated for Athletic Eligibility audits.

**PreviousFullYearEarnedCredits**

Updated: March 25, 2022

These credits are taken from the classes in the previous year – including summer.

**Template**

```
if (PreviousFullYearEarnedCredits > nn) then
```

**Examples**

```
if (PreviousFullYearEarnedCredits > 18) then  
  RuleComplete
```

```
Label PREV2 "18 credits required in previous full year";
```

### Notes

1. These credits are automatically calculated for Athletic Eligibility audits.

## PreviousTermEarnedCredits

Updated: March 25, 2022

These credits are taken from the classes in the previous term – excluding summer.

### Template

```
if (PreviousTermEarnedCredits > nn) then
```

### Examples

```
if (PreviousTermEarnedCredits > 6) then  
  RuleComplete  
  Label PREV6 "6 credits required in previous term";
```

### Notes

1. These credits are automatically calculated for Athletic Eligibility audits.

## PreviousTermEarnedCredits-Fall

Updated: March 25, 2022

These credits are taken from the classes in the previous fall term.

### Template

```
if (PreviousTermEarnedCredits-Fall > nn) then
```

## Examples

```
if (PreviousTermEarnedCredits-Fall > 6) then
  RuleComplete
  Label PREV6 "6 credits required in previous fall term";
```

## Notes

1. These credits are automatically calculated for Athletic Eligibility audits.

## Program (header)

Updated: March 25, 2022

When Program is used in ShareWith it refers to the program block in the audit. When used in an If-statement it refers to the program on the student's curriculum.

## Template

```
If (Program = XX) Then
  XXX
```

## Examples

```
ShareWith (Program)
```

```
DontShare (Program)
```

```
If (Program = XY) Then
  MinGPA 2.5 tag=MINGPA
  ProxyAdvice "You need a GPA 2.5 but currently have a <APPLIED> GP
A"
```

## Notes

1. When Program is used within a block header, it is only used with an If-statement, DontShare or ShareWith.

---

## Program (rule)

Updated: March 25, 2022

When Program is used in ShareWith, Blocktype, or Block it refers to the program block in the audit. When used in an If-statement it refers to the program on the student's curriculum.

### Template

```
Blocktype (Program)
```

### Examples

```
1 Blocktype (Program)
  Label PROGBLOCK "Program requirements";

# Only if the student has the SPAN program on their curriculum will
this block be used
1 Block (Program=SPAN)
  Label PROGBLOCK "Spanish Program requirements";

If (Program = RPD) Then
  3 Credits in MATH 400
  Label MA400 "Math 400";

6 Credits in BUS 1@
  ShareWith (Program)
  Label BUS "Business";
```

### Notes

1. Program is followed by a Program code except in a Blocktype rule and is optional in ShareWith.
2. The Program code must be valid in UCX-STU316.
3. Program can be used with Blocktype, Block, If-statement, or ShareWith.
4. When Program is used in a Block rule, the specific program block specified will only be included in the audit if the student has that program on their curriculum. For this reason, it is rare for use a Block rule with Program.

## ProxyAdvice (header)

Updated: March 25, 2022

ProxyAdvice specifies the advice text to show in place of the normal advice that would display for the header qualifier.

The ProxyAdvice text will appear as long as the header qualifier is not complete; as soon as the header qualifier has been completed, the ProxyAdvice text will be suppressed. ProxyAdvice can be used on the following block header qualifiers: Classes, Credits, LastRes, MinGPA, MinPerDisc, MinRes, MinCredits, MinClasses, MinTerm, MinSpread, Under.

### Template

```
ProxyAdvice "<REQUIRED> credits are required. "  
ProxyAdvice "You have taken <APPLIED> credits but"  
ProxyAdvice "still need <NEEDED> more."
```

### Examples

```
MinCredits 8 in PE @ tag=MINPE  
  ProxyAdvice "You have taken <APPLIED> PE credits but"  
  ProxyAdvice "still need <NEEDED> more to reach a total of <REQUIR  
ED> credits."
```

```
MinGPA 2.0 tag=GPA20  
  Display "Your major GPA is <APPLIED>."  
  ProxyAdvice "A <REQUIRED> GPA is required but your GPA is <APPLIE  
D>."
```

### Notes

1. ProxyAdvice is followed by up to 200 bytes of text enclosed in quotes.
2. ProxyAdvice can be repeated as many times as needed; the text is appended together.
3. No space is needed at the end of each line of advice; a space will be inserted when each line of text is put together on the worksheet.
4. Advice text appears as long as header qualifier is not complete.
5. The <REQUIRED> tag is useful for when an exception is made to the qualifier. The new required classes/credits appear in the text instead of the original value.
6. The <APPLIED> tag shows how many classes/credits are applying to the qualifier.

7. The <NEEDED> tag show many classes/credits are still needed to satisfy the qualifier.
8. Proxy-Advice can be used instead of ProxyAdvice.

## ProxyAdvice (rule)

Updated: March 25, 2022

ProxyAdvice specifies the advice text to show in place of the normal advice that would display for the rule.

The ProxyAdvice text will appear as long as the rule is not complete; as soon as the rule has been completed the ProxyAdvice text will be suppressed. ProxyAdvice should be used on rules containing long lists of classes or complex group rules.

### Template

```
ProxyAdvice "<REQUIRED> credits are required. "  
ProxyAdvice "You have taken <APPLIED> credits but"  
ProxyAdvice "still need <NEEDED> more."
```

### Examples

```
3 Credits in MATH 104, 109, 115, 119, 135, 156, 178, 198, 148, 199  
ProxyAdvice "<REQUIRED> Credits in some math class not related to  
business"  
ProxyAdvice "or computers is required. You have taken <APPLIED> c  
redits"  
ProxyAdvice "and need <NEEDED> more."  
Label MATH "Math requirement";
```

```
1 Group in  
  (9 Credits in SPAN 1@ Label SPANISH "Spanish") or  
  (9 Credits in FREN 1@ Label FRENCH  "French") or  
  (9 Credits in IRSH 1@ Label IRISH   "Irish")  
ProxyAdvice "9 Language Credits are required"  
Label LANG "LANGUAGE REQUIREMENT";
```

```
2 NonCourses (RECITAL = DONE)  
ProxyAdvice "You need 2 completed recitals"  
Label RECITAL "Recital";
```

```
5 Credits in @ (WITH Attribute=WRIT)  
ProxyAdvice "Click here to see classes that meet this requiremen
```



```
t."  
Ruletag AdviceJump="http://myschool.edu/catatlog/"  
Ruletag AdviceJump="englishdepartment/writing.html"  
Label WRITING "Writing Requirement";
```

## Notes

1. ProxyAdvice is followed by up to 200 bytes of text enclosed in quotes.
2. ProxyAdvice can be repeated as many times as needed; the text is appended together.
3. No space is needed at the end of each line of advice; a space will be inserted when each line of text is put together on the worksheet.
4. Advice text appears as long as rule is not complete.
5. Proxy-Advice can be used instead of ProxyAdvice.
6. ProxyAdvice may be used in any rule that gives advice.
7. <REQUIRED>, <APPLIED>, and <NEEDED> can only be used on course rules; they cannot be used on group, subset, rules etc. The values will be credits if it is a credits rule and the values will be classes if it is a classes rule. The value will also be credits if the rule is specified as credits and/or classes.
8. The <REQUIRED> tag is useful for when an exception is made to the rule. The new required classes/credits appear in the text instead of the original value.

## Pseudo

Updated: March 25, 2022

Pseudo is used in a block that does not have a strict credit limit. When Pseudo is specified the credits qualifier will always be satisfied and thus no advice will appear. See the [Credits \(header\)](#) topic for more information.

## Regterm

Updated: March 24, 2023

REGTERM translates to the registration term. Typically this is used with the Prerequisite Checking audit.

## Template

```
MinCredits nn in XXX @ (With DWTerm=REGTERM)
```

## Examples

```
MinCredits 12 in @ (With DWTerm=REGTERM)
```

## Notes

1. REGTERM maps to the rad\_term on the rad\_student\_mst for all audits except the Prerequisite Checking audit. On that audit, it maps to the term for which registration is taking place.
2. Although REGTERM is typically used on the Prerequisite Checking audit, it could be used on any type of audit in any WITH statement using DWTerm. For non-prerequisite audits, the REGTERM matches the student's active term.
3. REGTERM must be all uppercase.

## Remark

Updated: March 25, 2022

Remarks are additional comments associated with the header or specific rules that are displayed on the worksheet to help the student understand the requirements.

## Template

```
Remark "xxxx xxxx xxxxx"  
Remark "xx xxxx xx xxxxxx"
```

## Examples

```
Remark "BSC103 is recommended for freshmen."
```

```
Remark "Proficiency in English demonstrated to the satisfaction of"  
Remark "the English Proficiency Council (a score of at least 44 on"  
Remark "the Test of Standard Written English, or a grade of C or"  
Remark "better in ENG 111)."
```

```
If (Attribute = HONR and NumberOfMajors >= 2) Then
  BeginIf
    5 Classes in HIST 100:199
    Label HIST100A "Honors: History 100 level classes";
    Remark "As an honors student you need to take 5 classes"
    Remark "at the 100 level."
  EndIf
Else
  BeginElse
    3 Classes in HIST 100:199
    Label HIST100B "History 100 level classes";
    Remark "You need to take 3 classes"
    Remark "at the 100 level."
  EndElse

2 Classes in EVA 101, RORY @, ELENA @
  RuleTag RemarkJump="http://some.place.edu/onthinternet/anywherei
sfine/"
  RuleTag RemarkJump="support/getmemoreinfo.html"
  RuleTag RemarkHint="More info on Gen Ed option-a"
  Label "Gen Ed option A";
  Remark "You can click this link to find out more information";
  Remark "about this requirement.";
```

## Notes

1. Remark is used after a rule statement or at the start of the body section of the block. If used for a rule, the Remark follows the label that ends the previous rule statement. Remarks placed after the first semicolon are considered header remarks and appear in the block header on the worksheet.
2. Remark is followed by a free-text comment enclosed in quotes. The text can be up to 200 characters long and cannot include quotation marks. Remarks are limited to 200 bytes per line but you can have as many Remarks as you like; this means you can have unlimited remark text.
3. A semicolon may optionally follow the closing quote.
4. The free-text comment is used as a description on Degree Works output. It should be a description phrased for students and advisors, similar to the college catalog.
5. Remark statements are outside the scope of rule; they stand alone.
6. Remarks are not allowed within a subset or within a Group's list of rules.
7. A RemarkJump and RemarkHint may be used along with a Remark to allow hyper links to additional information about the requirements.

---

## ResidenceCompletedTermCount

Updated: March 25, 2022

Counts the terms where at least one native class was taken.

### Template

```
if (ResidenceCompletedTermCount > nn) then
```

### Examples

```
if (ResidenceCompletedTermCount > 3) then
  BeginIf
    MinGPA 3.2
    ProxyAdvice "You need a GPA of 3.2 now that you have 3 terms co
mpleted at UW"
  EndIf
```

### Notes

1. This can only be used in any audit but is more appropriate to be used in the Athletic Eligibility Audit or the Financial Aid Audit.
2. For Athletic Eligibility Audits, summer terms are excluded, the active term counts and counting starts at the first full-time term.
3. For Financial Aid and academic audits the summer term is counted, the active term is not counted and counting starts at the first term.
4. ResidenceTermCount is an alias.

## ResidenceCreditsEarned

Updated: March 25, 2022

These credits are taken from the rad\_cum\_cr\_earn field on the rad\_term\_dtl and can only be used in the Athletic Eligibility Audit or the Financial Aid Audit.

### Template

```
if (ResidenceCreditsEarned > nn) then
```

## Examples

```
if (ResidenceCreditsEarned >= 75) then
  RuleComplete
  Label "Term credits earned satisfied"
else
  RuleIncomplete
  ProxyAdvice "You have not yet earned 75 credits"
  Label "Term credits earned- not met";
```

## Notes

1. This can only be used in the Athletic Eligibility Audit or the Financial Aid Audit.

## RuleComplete (rule)

Updated: March 25, 2022

RuleComplete is a dummy rule that is always 100 percent complete and has no requirements. Although it can be used anywhere a course rule can be used, its main purpose is to be used within an If statement.

## Template

```
RuleComplete
  Label RCXXX "Xxxx xxxxxx xx xxx";
```

## Examples

```
If (ALISTAT = GOOD) then
  RuleComplete
  Label "Thank you for completing the Ali Application - you are fantastic!"
Else
  RuleIncomplete
  ProxyAdvice "See your advisor"
  Label "You have not completed the Ali Application"
```

## Notes

1. Allowable rule qualifiers: none.
2. Rule-Complete with a hyphen can be used also.

---

## RuleIncomplete (rule)

Updated: March 25, 2022

RuleIncomplete is a dummy rule that is always 0 percent complete and has no requirements. Although it can be used anywhere a course rule can be used, its main purpose is to be used within an If statement.

### Template

```
RuleIncomplete
  ProxyAdvice "XXXXX xx XXXXX xxx XXXX"
  Label RICXXX "Xxxx XXXXX xx xxx XXXXXXXX";
```

### Examples

```
If (ALISTAT = GOOD) then
  RuleComplete
    Label "Thank you for completing the Ali Application - you are fantastic!"
  Else
    RuleIncomplete
      ProxyAdvice "See your advisor"
      Label "You have not completed the Ali Application"
```

### Notes

1. ProxyAdvice should always be used with this requirement.
2. Allowable rule qualifiers: none.
3. Rule-Incomplete with a hyphen can be used also.

## RuleTag

Updated: March 25, 2022

RuleTag is a special qualifier you can place on any rule to give it special meaning when the audit worksheet is being displayed.

Any RuleTag name-value pair you add to your rule will be available for use within the stylesheets, to show the rule in a special way - hide it, use a different color, etc. Both the name and the value following RuleTag are limited to 200 characters each. The value must be enclosed in double-quotes if it contains non-alphanumeric characters.

## Template

```
RuleTag XXXXXX="YYYYY"
```

## Examples

```
5 Credits in @ (WITH ATTRIBUTE=WRIT)
  RuleTag AdviceJump="writing.html"
  ProxyAdvice "Click here to see classes that meet this requiremen
t"
  Label WRITING "Writing Requirement";

2 Classes in ART 101, ARTH @, GRAPH @
  RuleTag RemarkJump="http://some.place.edu/ontheinternet/anywherei
sfine/"
  RuleTag RemarkJump="support/getmemoreinfo.html"
  RuleTag RemarkHint="More info on Gen Ed option-a"
  Label GENEDA "Gen Ed option A";
Remark "You can click this link to find out more information";
Remark "about this requirement.";

5 Credits in BIOL 2@, PHYS 250:270
  RuleTag Category="Special Science Classes"
  Label SCIENCE "Science classes";

3 Classes in in ACCT 2@
  RuleTag DWCredits=9
  Label ACCT "Accounting classes";

1 Group in
  (2 Classes in ART 100:199 Label ART "Art option") or
  (3 Classes in ARTH 100:199 Label ARTH "Art history option") or
  RuleTag DWCredits=7
  Label MONET "Art requirement";
```

## Notes

1. Allowable on any rule.
2. RuleTag names of AdviceJump, RemarkJump and RemarkHint have special meaning and are used by the standard Degree Works worksheets. See the [AdviceJump](#), [RemarkJump](#) and [RuleTag](#) topics for more information.
3. RuleTag name of DWCredits is used with HeaderTag DWCredits=DWRuleSum. See the

[HeaderTag](#) topic for more details.

## SameDisc (header)

Updated: March 25, 2022

SameDisc is used to equate two disciplines.

### Template

```
SameDisc (XXX=YYY, AAA=BBB) tag=SAMESAME
```

### Examples

```
MaxPerDisc 2 Classes (FRE, SPA)  
SameDisc (FRT=FRE, FRN=FRE, SPT=SPA)
```

```
MaxPerDisc 8 Credits (PE)  
SameDisc (HE=PE)
```

```
MaxPerDisc 8 Credits (PE, HE)
```

### Notes

1. SameDisc sets an equivalence between two disciplines. The equivalence is used when evaluating MaxPerDisc, MaxSpread, MinPerDisc, and MinSpread, not when matching courses taken by the student to required courses. Typically, SameDisc is used to equate obsolete disciplines with current disciplines.
2. The second discipline must appear in the discipline list associated with MaxPerDisc/MinPerDisc.

## SameDisc (rule)

Updated: March 25, 2022

SameDisc is used to equate two disciplines.



## Template

```
SameDisc (XXX=YYY, AAA=BBB) tag=SAMESAME
```

## Examples

```
# Counts FRT and SPT as the ENG discipline. A student who took FRT
100
# and SPT100 has not satisfied the MinSpread requirement for 2 disc
iplines
# because FRT and SPT both count as the ENG discipline.
11 Credits in ENG @, FRT @, GER @, PHL @,
                REL @, SPA @, SPT @
MinSpread 2
SameDisc (FRT=ENG,SPT=ENG) tag=SAME
Label GENSTUDIES "General Studies";
```

## Notes

1. SameDisc sets an equivalence between two disciplines. The equivalence is used when evaluating MaxPerDisc, MaxSpread, MinPerDisc, and MinSpread, not when matching courses taken by the student to required courses. Typically, SameDisc is used to equate obsolete disciplines with current disciplines.
2. The second discipline must appear in the discipline list associated with MaxPerDisc/MinPerDisc.

## School (header)

Updated: March 25, 2022

When School is used in ShareWith it refers to the School block in the audit. When used in an If-statement it refers to the school on the student's curriculum.

## Template

```
If (School = XY) Then
  XXX
```

## Examples

```
ShareWith (School)
```

```
If (School = XY) Then
  MinGPA 2.5 tag=MINGPA
  ProxyAdvice "You need a GPA 2.5 but currently have a <APPLIED> GP
A"
```

## Notes

1. When School is used within a block header, it is only used with an If-statement or ShareWith.

## School (rule)

Updated: March 25, 2022

When School is used in ShareWith, Blocktype or Block it refers to the School block in the audit.  
When used in an If-statement it refers to the school on the student's curriculum.

## Template

```
1 Blocktype (School)
```

## Examples

```
1 Blocktype (School)
  Label SCHBLOCK "School requirements";
```

```
# Only if the student has the UG school on their curriculum will th
is block be used
```

```
1 Block (School=UG)
  Label UGBLOCK "Undergrad requirements";
```

```
If (School = RPD) Then
  3 Credits in MATH 400
  Label MA400 "Math 400";
```

```
6 Credits in BUS 1@
ShareWith (School)
```

```
Label BUS "Business";
```

## Notes

1. School is followed by a school code except in a Blocktype rule and is optional in ShareWith.
2. The school code must be valid in UCX-STU350.
3. School can be used with Blocktype, Block, If-statement, or ShareWith.
4. When School is used in a Block rule, the specific school block specified will only be included in the audit if the student has that school on their curriculum. For this reason, it is rare to use a Block rule with School.

## ShareWith (header)

Updated: March 25, 2022

ShareWith indicates that credits/classes can fulfill multiple requirements.

## Template

```
ShareWith (XXXXX) tag=SHAREXXXXX
```

## Examples

```
# 10 Credits from this block can also be applied to the
# minor block or the major block
Share 10 CREDITS (MINOR, MAJOR) tag=KL23K3 # one of these blocks -
but not both
```

```
# all credits from this block can also be applied to any other bloc
k;
# however, a class that is applied here can only be applied to
# one other block - not to all of the blocks
ShareWith (AllBlocks) tag=KLQ2345
```

```
# all credits from this block can be applied to the general educati
on block and
# to multiple rules within this block
ShareWith(OTHER=GENED, THISBLOCK) tag=KLA9K23
```

```
# all credits from this block can also be applied to any
# English or Literature Major block
ShareWith (MAJOR=ENGL, MAJOR=LIT) tag=KALK43 # one of these blocks
- but not both

# all credits from can also be applied to both English and Literatu
re Major block
ShareWith (MAJOR=ENGL) tag=2345K # share with this block
ShareWith (MAJOR=LIT) tag=LK340 # and also share with this block

# all credits from this block can also be applied to any Undergrad
College block
ShareWith (COLLEGE=U@) tag=2K3L43

## all credits from this block can be shared with any major except
CHEM
ShareWith (MAJOR, MAJOR<>CHEM) tag=K2L390Q # share with any major e
xcept for CHEM

## all credits from this block can be shared with any block except
HIST major block
ShareWith (AllBlocks, MAJOR<>HIST) tag=AKSDF

# Share with the student's first major
ShareWith (MAJOR=1st) tag=AKLSDUFA

# Share with major that is associated with this conc block
ShareWith (MAJOR=associated) tag=SHAREMAJASSOC
# Share with concentration that is associated with this major block
ShareWith (CONC=associated) tag=SHARECONASSOC
# Share with any concentration except the one that is associated wi
th this major block
ShareWith (CONC, CONC<>associated) tag=SHARECONCNOTASSOC
```

## Notes

1. Share can be used in place of ShareWith.
2. NonExclusive can be used in place of ShareWith; it was the original name used.
3. ShareWith indicates that the credits or classes applied by the Auditor Engine towards satisfying the requirements in this block can also be used to satisfy requirements in other blocks. Any single class will share with only one of the blocks listed – not all of the blocks to which they may apply. If you have "Share 5 Classes (Major, Minor)" it is possible to have 3 classes shared with the Major and 2 with the Minor or the auditor might share 1 class with the

Major and 4 with the Minor – but any single class will only be shared with one of those blocks – not both.

4. If you do want to share with both the Major and Minor you can simply use two separate qualifiers. For example, Share 5 Classes (Major) and also Share 5 Classes (Minor). However, there is no guarantee that the same 5 classes will be shared between all three blocks. It is possible 5 classes will be shared with the Major and a different 5 classes will be shared with the Minor.
5. ShareWith is needed as part of the rule only if the block has not been declared as being shared.
6. You can specify a number of credits or classes to indicate how many credits or classes the auditor can apply to this block as well as to other blocks. A scope or scope list is required to indicate the blocks in which the courses can be applied non-exclusively.
7. When specified, the number of credits or classes must be less than or equal to the number of credits or classes specified before ShareWith. Subtract the number following Share or ShareWith from the number before Share or ShareWith to get the number of credits or classes that will not be shared; i.e., will be exclusive. If the number of credits or classes is not specified, then all credits and classes in the block or rule can be applied here and in subsequent rules and blocks.
8. ShareWith is followed by a block type (COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, OTHER, PROGRAM, SCHOOL, SPEC) or scope (ALLBLOCKS, THISBLOCK) to indicate which blocks can share the credits or classes. Any of the keywords used as block type can be used here, except ID. If OTHER is used, then a code must be specified, e.g., "Share 6 Credits (OTHER=GENED)". A value can also be used with block types other than OTHER, e.g., "ShareWith (MAJOR=ENGL, MAJOR=LIT)". Values also support wildcards (@).
9. To treat some classes/credits as exclusive and the rest as sharing, use ShareWith as a block qualifier and use DontShare with the number of classes or credits as a rule qualifier.
10. Do not use ShareWith as a block qualifier and use ShareWith with the number of classes or credits as a rule qualifier as you will get unpredictable results.
11. Using NoCount and ShareWith together could yield unpredictable results. Using SpMaxCredits and Share or ShareWith together could also yield unpredictable results.
12. Also see StandAloneBlock. It allows sharing in a more global way and may meet your requirements.
13. The not-equals (< >) block specification has precedence in that it overrides any other sharing specified on the qualifier.
14. 1st, 2nd, 3rd, 4th, etc. You can use an ordinal value to specify that sharing should only occur with the 1st, 2nd, 3rd, etc block of a certain type. For example, ShareWith (Major=1st) will only allow sharing with the first major. Conversely, ShareWith(Major, Major<>1st) will allow sharing with all majors except for the first one. The auditor determines which is the 1st, 2nd, etc block of the specific type based on the order of the blocks that were passed to the auditor – and this is related to the sequence number on the rad\_goalData\_dtl. Caveat: if the student has multiple majors and multiple concentrations within each major specifying ShareWith(Conc=1st) may not result in the concentration you intended. The ordinal specification is not relative to the

block doing the sharing. That is, the auditor does not find the 1st concentration for that given major; the auditor simply finds the first concentration in its list of concentrations. If this is an issue for you then you might consider using ASSOCIATED instead.

15. ASSOCIATED: Instead of specifying a specific block you can use ASSOCIATED to let the auditor figure out which specific block is associated with the parent block. ShareWith(Major=associated) can be used to find the major that is associated with this concentration block (or any other type of block but usually it is majors and concentrations that are associated). The reverse can also be used: ShareWith(Conc=associated) would be placed in the major block. The auditor will attempt to find the associated block in two ways: First, the auditor examines the secondary tags of both the major and the conc blocks to see if one is tied to the other. Second, if the secondary tags don't lead to an association then the auditor examines the ATTACH values on the rad\_goalData\_dtl records. If the auditor finds a CONC record that has an ATTACH code for the parent major block with the ShareWith qualifier then the auditor has found an association. Note, if two concentrations are associated with the same major and the major has ShareWith (conc=associated) then the major will only share with the first concentration. However, if the concentration blocks instead have ShareWith(major=associated) then both concentrations will share with the major. As mentioned previously, associations can go from any block type to any other block type but most often they are setup between the concentration and major. You can always use this as a way to prevent sharing with the associated block: ShareWith (Conc, Conc<>associated). This will allow sharing with any conc block except for the associated one. And yes, when you think ASSOCIATED you can also think ATTACHED because of the field names on the rad\_goalData\_dtl. DontShare may not work like you want it to. For example, in your major if you have DontShare (Conc=XYZ) and also ShareWith(Minor) and the minor and the concentration are sharing (because they have ShareWith) then a class may be applied to all three blocks. This is because the auditor is seeing that the minor and concentration are sharing and the major and minor are sharing; it does consider the major and concentration to be sharing.
16. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made. Or if the qualifier has a label then a label-tag will suffice.

## ShareWith (rule)

Updated: March 25, 2022

ShareWith indicates that credits/classes can fulfill multiple requirements.

### Template

```
ShareWith (XXXXX)tag=SHAREXXXXX
```

### Examples

```
##6 credits from this rule can also be applied to the MAJOR block a
```

```
nd
##3 of those credits can also be applied to the MINOR block
6 Credits in INS1 @
  Share 6 Credits (MAJOR) tag=KLAS
  Share 3 Credits (MINOR) tag=BJLN
  Label G3ET3T "International Studies";

##courses from this rule can also be applied to the MINOR and/or CO
NC block
3 Classes in HIST @
  Share 2 Classes (MINOR, CONC) tag=BJLN34
  Label ASDF34 "History";

##all courses from this rule can also be applied to other rules in
this block
12 Credits in ENG 100:199, PHI 100:199
  ShareWith (THISBLOCK) tag=BJLN345634
  Label 346RER "English and Philosophy";

##courses from this rule can also be applied to English and/or Lite
rature MAJOR blocks
3 Classes in HIST @
  Share 2 Classes (MAJOR=ENGL, MAJOR=LIT) tag=BJLN234
  Label RE35RR "History";

##courses from this rule can also be applied to rules in any Underg
raduate COLLEGE block
12 Credits in ENG 100:199, PHI 100:199
  ShareWith (COLLEGE=U@) tag=AILIME
  Label 345RDG "English and Philosophy";

## courses from this rule can also be applied any major except for
LIT
3 Classes in HIST @
  Share 2 Classes (MAJOR, MAJOR<>LIT) tag=BJLN345
  Label QWE343 "History";

# Share with the student's first major
3 Classes in HIST @
  ShareWith (MAJOR=1st) tag=BJLNADF
  Label FDSSE "History";

# Share with major that is associated with this conc block
3 Classes in HIST @
  ShareWith (MAJOR=associated) tag=BJLNQ345345
```

```
Label SSIEAS "History";

# Share with concentration that is associated with this major block
3 Classes in HIST @
  ShareWith (CONC=associated) tag=BJLN23434
  Label 34R3FE "History";

# Share with any concentration except the one that is associated with this major block
3 Classes in HIST @
  ShareWith (CONC, CONC<>associated) tag=BJLN2345
  Label FDS45FD "History";

# Sharing can be placed on group or subset also:
BeginSub
  12 Credits in ENG 100:199, PHI 100:199
  Label 345RDG "English and Philosophy";
  3 Classes in HIST @
  Label QWE343 "History";
EndSub
  ShareWith (Other=GENED) tag=AILIME
  Label 9234RJ "English and Philosophy and History";
```

## Notes

1. ShareWith can be used on a course rule, group or subset.
2. Share can be used in place of ShareWith.
3. NonExclusive can be used in place of ShareWith; it was the original name used.
4. ShareWith indicates that the credits or classes applied towards satisfying this rule can be used to satisfy requirements in other blocks or in subsequent rules in this block.
5. ShareWith is needed as part of the rule statement only if the block has not been declared as being shared.
6. ShareWith must follow a course list.
7. ShareWith is optionally followed by either credits or classes to indicate how many credits or classes the auditor can apply to this rule as well as to subsequent rules.
8. When specified, the number of credits or classes must be less than or equal to the number of credits or classes specified before ShareWith. Subtract the number following ShareWith from the number before ShareWith to get the number of credits or classes that will be exclusive. If the number of credits or classes is not specified, then all credits and classes in the block or rule can be applied here and in subsequent rules and blocks.



9. ShareWith is followed by a block type (COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, OTHER, PROGRAM, SCHOOL, SPEC) or scope (ALLBLOCKS, THISBLOCK) to indicate which blocks can share the credits or classes. Any of the keywords used as block type can be used here, except ID. If OTHER is used, then a OTHER code must be specified, e.g., "Share 6 Credits (OTHER=GENED)". A value can also be used with block types other than OTHER, e.g., "ShareWith (MAJOR=ENGL, MAJOR=LIT)". Values also support wildcards (@).
10. DontShare cannot be used in the same rule as ShareWith.
11. Disallow duplicates of block types if ShareWith is repeated in a rule. For example: "Share 6 Credits (MAJOR)" and "Share 12 Credits (MAJOR)" on the same rule is not allowed.
12. To treat some classes/credits as not sharing and the rest as sharing, use ShareWith as a block qualifier and use DontShare with the number of classes or credits as a rule qualifier.
13. Do not use ShareWith as a block qualifier and use ShareWith with the number of classes or credits as a rule qualifier.
14. Using NoCount and ShareWith together could yield unpredictable results.
15. Do not use ShareWith with SpMaxCredits or SpMaxTerm.
16. 1st, 2nd, 3rd, 4th, etc. See the notes on this option in the [ShareWith \(header\)](#) topic.
17. ASSOCIATED. See the notes on this option in the [ShareWith \(header\)](#) topic.
18. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## Spec (header)

Updated: March 25, 2022

When Spec is used in ShareWith it refers to the specialization block in the audit. When used in an If-statement it refers to the specialization on the student's curriculum.

### Template

```
If (Spec = XY) Then  
  XXX
```

### Examples

```
ShareWith (Spec)
```

```
If (Spec = XY) Then
```

```
MinGPA 2.5 tag=MINGPA
ProxyAdvice "You need a GPA 2.5 but currently have a <APPLIED> GPA"
```

## Notes

1. When Spec is used within a block header, it is only used with an If-statement or ShareWith.

## Spec (rule)

Updated: March 25, 2022

When Spec is used in ShareWith, Blocktype or Block it refers to the specialization block in the audit. When used in an If-statement it refers to the specialization on the student's curriculum.

## Template

```
1 Blocktype (Spec)
```

## Examples

```
1 Blocktype (Spec)
  Label SCHBLOCK "Spec requirements";

# Only if the student has the UG spec on their curriculum will this
block be used
1 Block (Spec=UG)
  Label UGBLOCK "Undergrad requirements";

If (Spec = RPD) Then
  3 Credits in MATH 400
  Label MA400 "Math 400";

6 Credits in BUS 1@
  ShareWith (Spec)
  Label BUS "Business";
```

## Notes

1. Spec is followed by a specialization code except in a Blocktype rule and is optional in

ShareWith.

2. The specialization code must be valid in UCX-STU323.
3. Spec can be used with Blocktype, Block, If-statement, or ShareWith.
4. When Spec is used in a Block rule, the specific specialization block specified will only be included in the audit if the student has that specialization on their curriculum. For this reason, it is rare to use a Block rule with Spec.

## SpMaxCredits (header)

Updated: March 25, 2022

SpMaxCredits indicates a maximum number of credits. The maximum is a strict cap. The excess credits are split across blocks.

### Template

```
SpMaxCredits n in XXXX @ tag=SPMAXXXX
```

### Examples

```
SpMaxCredits 9 in MUS 1@ (With DWResident=Y) tag=SPMAXMUSIC
```

```
SpMaxCredits 3 in GEOL 109, 145 tag=SPMAXGEOL
```

```
SpMaxCredits 12 in PSY 100:199  
  Except PSY 108, 117 tag=SPMAXPSY
```

```
# Max of 15 in MUS and ART combined  
SpMaxCredits 15 in MUS @, ART @ tag=SPMAXMUSART
```

```
# Max of 15 in MUS and max of 15 in ART  
SpMaxCredits 15 in MUS @ tag=SPMAXMUS  
SpMaxCredits 15 in ART @ tag=SPMAXART
```

```
# Disallow classes older than 10 years - but allow any ANTH to be older  
SpMaxCredits 0 in @ (With DWAge>10)  
  Except ANTH @ tag=SPMAX10
```

## Notes

1. The connector in the course list following SpMaxCredits can be a comma or "or". The plus sign or "and" is not allowed. If multiple courses are listed, then the sum of all credits for the classes that satisfy the list is compared against the maximum specified.
2. Credits in excess of the number specified may cause courses to be split—some credits applied to this block and some applied to other blocks. If the block is the starting block, then the excess credits go to Over-The-Limit.
3. The course list associated with SpMaxCredits allows Except but not Including.
4. Using SpMaxCredits and ShareWith may lead to unpredictable results.
5. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

## SpMaxTerm (header)

Updated: March 25, 2022

MaxTerm indicates the maximum number of credits/classes that can be applied to a set of requirements each term. The maximum is a strict cap. The excess credits are split across blocks.

## Template

```
SpMaxTerm n Credits in XXXX @ tag=SPMAXXXX
```

## Examples

```
SpMaxTerm 1 Class in MUS 1@ (With DWResident=Y) tag=SPMAXMUSIC
```

```
SpMaxTerm 12 Credits in PSY 100:199  
Except PSY 108, 117 tag=SPMAXPSY
```

```
# Max of 15 in MUS and ART combined  
SpMaxTerm 15 Credits in MUS @, ART @ tag=SPMAXMUSART
```

```
# Max of 15 in MUS and max of 15 in ART  
SpMaxTerm 15 Credits in MUS @ tag=SPMAXMUS  
SpMaxTerm 15 Credits in ART @ tag=SPMAXART
```

**Notes**

1. The number indicates the maximum number of classes or credits that can be taken per term. This number is a strict cap. Credits in excess of the number specified may cause courses to be split—some credits applied to this block and some applied to other blocks. If the block is the starting block, then the excess credits go to Over-The-Limit.
2. The course list associated with SpMaxTerm allows Except but not Including.
3. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.
4. Using SpMaxTerm and ShareWith may lead to unpredictable results.
5. A qualifier tag should always be used to ensure exceptions apply to the correct qualifier even after block changes are made.

**StandAloneBlock (header)**

Updated: March 25, 2022

StandAloneBlock denotes that classes in this block are applied without regard to classes applied in the other blocks of the audit.

**Template**

```
StandAloneBlock
```

**Examples**

```
StandAloneBlock
```

**Notes**

1. The Auditor will place classes in this block and will ignore this block when determining how or if any sharing is done among the other blocks.
2. Example: Major1 can share with Core and Major2 can share with Core. Major1 and Major2 cannot share in some colleges but can share in other colleges. Using StandAloneBlock in the Core block lets the major blocks deal with sharing without having to worry about the Core block.
3. A good general rule is this: if the block can share with all blocks then use StandAloneBlock.

## Tag

Updated: March 25, 2022

Tag is used on a header or rule qualifier to ensure that any exceptions applied to the qualifier will apply even after the block contents have changed.

## Template

```
Tag=XXXXXX
```

## Examples

```
MinCredits 8 in PE @ tag=MINPE
  ProxyAdvice "You have taken <APPLIED> PE credits but"
  ProxyAdvice "still need <NEEDED> more."
```

```
MaxPassfail 12.5 Credits tag=MAXPF
```

```
12 Classes in BUS 3@, ACCT 3@
  MinClasses 1 in BUS 321, ACCT 306 tag=MINBUS321306
  MinClasses 1 in BUS 327, ACCT 312 tag=MINBUS327312
  Label ACCTELECT "Accounting electives";
```

## Notes

1. Tag is also referred to as a Qualifier Tag.
2. All qualifiers should have a tag – which is a code with a maximum length of 20 alphanumeric characters that is placed at the end of the qualifier (but before the ProxyAdvice on header qualifiers).
3. When a Label is on a header qualifier the qualifier does not need a tag since the Label tag is used to identify the qualifier.
4. A tag is required on a qualifier if the type of qualifier repeats in the header or on the rule. However, the best practice is to always add a tag in case another qualifier is added in the future.
5. The tag value has to be unique in the block. However, it is strongly recommended that all tag have alphanumeric values. Numeric tags like "4" or "8.3" are not recommended because it is too easy for someone to renumber the tags and thus causing exceptions to apply to the wrong requirements. Tags like "MINHIST" or "MAXCHEM200" are preferred as they are much less likely to be changed by a novice user. You might even consider using a random (such as

LP23ADF9X) so that it has no meaning whatsoever and therefore is much less likely to be changed accidentally.

6. See the [Label Tags](#) topic for more information about qualifier tags.

## Then

Updated: March 25, 2022

Used as part of a conditional statement. May be used in the block header and in the body to control the qualifiers and rules used in the block based on the student's data. See [If-Then](#).

## ThisBlock (header)

Updated: March 25, 2022

This indicates the scope for the ShareWith qualifier. This scope signifies that all blocks are to be considered for sharing of classes with the current block.

## Template

```
ShareWith (ThisBlock)
```

## Examples

```
Share 10 Credits (ThisBlock)
```

```
ShareWith (ThisBlock)
```

## Notes

1. Each class can be shared among all of the rules in the block.

## ThisBlock (rule)

Updated: September 30, 2022

Indicates the scope for the ShareWith qualifier. This scope signifies that this requirement is considered for sharing of classes with another requirement in this block.

## Template

```
ShareWith (ThisBlock)
```

## Examples

```
6 Credits in BUS 1@
Share 3 Credits (ThisBlock) tag=RPDI94843
Label BUS "Business";
```

```
9 Credits in CHEM @, BIOL @, PHYS @
ShareWith (ThisBlock) tag=ELD90243
Label SCI "Science";
```

## Notes

1. Each class from a rule can share with one other rule within the same block.
2. Class A can share with requirement X and class B can share with requirement Y in this block.

## TotalCreditsAttempted

Updated: March 25, 2022

These credits are taken from the rad\_cum\_gr\_att field on the rad\_term\_dtl and can only be used in the Athletic Eligibility Audit or the Financial Aid Audit.

## Template

```
if (TotalCreditsAttempted > nn) then
```

## Examples

```
if (TotalCreditsAttempted >= 75) then
  RuleComplete
  Label "Total credits attempted satisfied"
else
  RuleIncomplete
  ProxyAdvice "You have not yet attempted 75 credits"
  Label "Total credits attempted- not met";
```



## Notes

1. This can only be used in the Athletic Eligibility Audit or the Financial Aid Audit.

## TotalCreditsEarned

Updated: March 25, 2022

These credits are taken from the rad\_cum\_tot\_earn field on the rad\_term\_dtl and can only be used in the Athletic Eligibility Audit or the Financial Aid Audit.

## Template

```
if (TotalCreditsEarned > nn) then
```

## Examples

```
if (TotalCreditsEarned >= 75) then
  RuleComplete
  Label "Total credits earned satisfied"
else
  RuleIncomplete
  ProxyAdvice "You have not yet earned 75 credits"
  Label "Total credits earned- not met";
```

## Notes

1. This can only be used in the Athletic Eligibility Audit or the Financial Aid Audit.

## Under (header)

Updated: March 25, 2022

Under specifies a limit on the number of credits or classes than can be used for the particular set of coursework.

Unlike MaxClasses/MaxCredits the Under qualifier does not remove classes from the block or audit—it simply checks to see that the limit was not exceeded. The qualifier will be satisfied as long as the number specified is not exceeded.

## Template

```
Under x Credits in XXXX @ tag=UNDERLKLASD
ProxyAdvice "You have exceeded the x credit limit."
Label ADFAL "xxx xxx xxxxx"
```

## Examples

```
Under 10 Classes in ART @, MUS @
ProxyAdvice "You have exceeded the 10 class limit."
Label ARTMUS "Only 10 art and music classes are allowed."
```

```
Under 30 Credits in @ (With Attribute=DEV)
Except PE @, MUS @
ProxyAdvice "You have exceeded the 30 credit limit."
Label DEV "Only 30 developmental credits are allowed."
```

## Notes

1. Under can be used in any block but is most often used in an AWARD block in a Financial Aid audit.

## With

Updated: March 24, 2023

Additional custom class specifiers in a course list. The With keyword is particularly useful when course content is determined by the course key and additional data, such as grade or term.

## Template

```
(With xxxx=yyyy)
```

## Examples

```
MaxClasses 3 in MUS100 (With DWGradeNumber < 2.0)
```

```
# MYCODE points to some field on the rad_class-dtl via UCX-SCR044
MaxCredits 3 in HUM @ (With MYCODE="AB@" or MYCODE=XYZ), ENGL 323
```

```
# You can also string the values together; these two examples are e
quivalent:
MaxCredits 6 in ENGL @ (With Attribute=ABCD, WXYZ)
MaxCredits 6 in ENGL @ (With Attribute=ABCD or Attribute=WXYZ)
```

```
##In this example only those HIST 229 classes taken before term 199
9A
##can be used in the MAXCREDITS calculation
MaxCredits 15 in HIST 213, 214, 224, {HIDE HIST 229 (With DWTERM <
1999A),} HIST 235
```

```
MinCredits 12 in ACCT 31000, 3200, 3300, 4350, 4600, 4700 (With DWR
esident=Y), 4800
```

```
# Look up ADMITTERM on the student's rad_custom_dtl; the STUDENT- p
refix indicates
# that this is a variable name - not an actual value
MaxClasses 0 in @ (With Hide DWTerm < STUDENT-AdmitTerm)
```

```
30 Credits in MUS @ (With Attribute=DEV), PE @ (With Attribute=DEV)
ProxyAdvice "You need 30 developmental credits."
ProxyAdvice "Currently you only have <APPLIED>."
Label DEV "30 developmental credits";
```

```
If (MATH 101 (With DWInprogress=N) was PASSED) Then
```

## Notes

1. With offers additional specification of a particular course by qualifying the preceding course With custom data from the student system's class. The custom data is defined in UCX-SCR044 and can reference any piece of data in the class record.
2. The token following With must either be defined in UCX-SCR044, with Custom Class Data, or be a DW named field. After the UCX-SCR044 custom data, an operator must exist, followed by a valid value for that custom data item. The value can contain a wildcard (@) to signify zero or more occurrences of any character.
3. The entire With expression must be enclosed in parentheses, including With.
4. Quotation marks are required around the code to the right of the equals sign if a non-alphanumeric character is used.
5. If a list of values following With are needed, then separate the values with a comma, for example, ENG100 (With DWSection=01, 02). Doing this indicates that the class must contain one of the values in the list. This is the same as separating the values using OR: (With DWSection=01 or DWSection=02).

6. Only the last course before With is qualified by the With custom data. For example: ENG 101, 110 (With MYCODE="XX") qualifies only ENG 110 as requiring MYCODE=XX.
7. For ranges of course numbers, With applies to all courses in the range. For example, ART 100:102 (With DWSection="G@") is the same as ART 100 (With DWSection="G@"), ART 101 (With DWSection="G@"), ART 102 (With DWSection="G@").
8. With is only allowed after a course.
9. The following DW named fields can be used instead of using UCX-SCR044: DWAge, DWCredits, DWCreditType, DWCourseNumber, DWDiscipline, DWGradeNumber, DWGradeLetter, DWGradeType, DWLocation, DWPassFail, DWResident, DWSchool, DWSection, DWTerm, DWTitle, DWTransfer, DWTransferCourse, DWTransferSchool, DWTransferSchoolID, DWInprogress, DWPreregistered, DWTermType, DWPassed, DWEtsType. See core.audit.useEtsType to enable use of DWEtsType.
10. DWResident will have a value of Y if the class is taken in residence and a value of N if the class is a transfer class. DWTransfer will have a value of Y if the class is a transfer class and a value of N if the class was taken in residence.
11. A wildcard may be needed if the length of the value specified is not the same as the length of the field being referenced: (With DWGradeLetter="B@"). Wildcards are not needed at end of title fields.
12. The With operator supports the relational operators: > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), <> (not equal to), and = (equal to).
13. You can string multiple With codes together. For example: 1 class in ENGL 1@ (With DWTerm=1822 and DWSection=XY, AB and DWResident=Y) Label English class.
14. You cannot mix AND with OR, however. Doing this will cause a parser error. For example, 1 class in ENGL 1@ (With DWTerm=1822 or DWSection=XY, AB and DWResident=Y) Label English class is not valid.
15. DWTerm can be used with CURRENT and PREVIOUS (With DWTerm=CURRENT). This option is most often used in an AWARD block when performing a Financial Aid audit. CURRENT is defined as the student's active term, while PREVIOUS is defined as the term previous to the active term for which the student took classes. DWTerm can also be used with REGTERM for prereq checking (With DWTerm = REGTERM). Scribing DWTerm = REGTERM will ensure that the course is a coreq; scribing DWTerm < REGTERM will ensure that the course is a true prereq and not a coreq.
16. DWInprogress is only Y when the class's term value matches the active-term on the rad\_student\_mst and when the Inprogress flag on the rad\_class\_dtl is Y.
17. DWPreregistered is only Y when the class's term value is greater than the active-term on the rad\_student\_mst and when the Inprogress flag on the rad\_class\_dtl is Y.
18. DWTransferCourse uses whatever is bridged to the rad\_transfer\_dtl.rad\_tr\_crse\_key field. If MATH101 is in this field then you need to use With DWTransferCourse=MATH101. If MATH 101 (With a space) is in this field then you need to use With DWTransferCourse=MATH 101. Be sure to use whatever spaces are in the actual value bridged to this field. The Banner extract pulls over the transfer course without any space between the discipline and number so

Banner schools should follow the first example here.

19. When using DWTransferSchool you can either use the school's full name (for example, University of Learning) or you may use the school's ETS/FICE ID code. Because school names change (for example, College changes to University) and because of inconsistency on how your users have entered the school names into your ERP (for example, Univ. vs U. vs University) you might consider using the school's ID code instead of the school name in your With qualifiers. Both of these are supported: With DWTransferSchool=University of Learning and With DWTransferSchool=123456. However, if the school ID you are using begins with an alpha character (for example, W382191), you must use DWTransferSchoolId instead as it will always treat the value as an ID and never as a name. You can also use DWTransferSchoolId even when the ID is a number like 123456, but using it when the ID starts with an alpha character is what is required. To keep things simple you can use DWTransferSchoolId when using an ID and use DWTransferSchool when dealing with a name.
20. You can place a STUDENT- prefix in front of the With value on the right of the operator to tell the auditor to lookup the specified name on the rad-custom-dtl and use the value found. Example, With DWTerm > STUDENT-AdmitTerm - the auditor would lookup ADMITTER on the rad-custom-dtl and use the value associated with that record. AUDITTER must be defined in UCX-SCR002 also. Currently STUDENT- only works with DWTerm.
21. DWAge looks at the age of the class in years by looking at the class's term value and subtracting it from the current/active term for the student. A term value of 200510 is converted to a decimal value of 2005.10 and then subtracted from the active term, which might be 201220, which is converted to a decimal value of 2012.20. Subtracting the two gives 2012.20 – 2005.10 = 7.1 years old. If the rule specifies DWAge < 10 then this age of 7.1 years will be within the age limit and would apply to the rule. Three different formats of terms are supported with lengths 6, 5 and 4: 6 characters: 200510. The first four characters are the year with the last two characters being the term within the year; 5 characters: 20051 – the first four characters are the year with the last character being the term within the year; 4 characters: 1051 - the first three characters are the year with the last character being the term within the year (here the leading 1 means 21st century with terms before the year 2000 having a leading 0 instead – for example, 0991). If your terms have some other format you may not be able to use DWAge. Planned classes have a term value of PLANNED normally. For the DWAge calculation these planned classes are given an age of zero months/years.
22. DWGrade, DWGradeNumber, and DWGradeNum are all synonyms. However, when you use DWGradeNum or DWGradeNumber, you must specify a numeric grade. For example, DWGradeNumber > 2.5. When you use DWGrade, you can specify a numeric grade or a letter grade. However, if a letter grade is specified, the grade will be looked up on UCX-STU385 and will be translated to a numeric grade. For example, DWGrade > C will result in meaning DWGrade > 2.0. You may have the need to use DWGradeLetter (or DWLetterGrade) for special cases where multiple letter grades have the same value. This usually happens with grades of Incomplete, Withdrawn, and so on where they all have a grade of 0.0. In a header qualifier, you may want to do something like this: MaxClasses 0 in @ (With DWGradeLetter = I). When the letter grade is specified like this, no lookup on UCX-STU385 is performed to get the numeric equivalent of the letter. For this reason, you should not use less-than or greater-than with DWGradeLetter. When DWGrade or DWGradeNumber are used, both in-progress and passed pass-fail classes are given a grade of 4.0.
23. DWDiscipline and DWCourseNumber can be used for special situations such as when you want to use a wildcard but want to exclude a special discipline. For example: MaxCredits 5 in @ (With DWPassfail=Y and DWDiscipline<>ANTH). Here you are putting a max on the

number of passfail credits but are allowing an unlimited number of ANTH passfail credits; that is, the ANTH classes are being excluded. You will not be using DWDiscipline and DWCourseNumber often, but they can come in handy in special situations. Note, when an equivalence is in place, it is the new discipline and number that is being used and not the original values. For example, if SOC 123 was renamed to ANTH 145, it will be ANTH and 145 that will be used in DWDiscipline and DWCourseNumber and not SOC or 123.

24. In planner, look ahead and what-if audits courses may be passed to be included in the audit. These are not real classes and thus do not have many of the DW values associated with them. The school is not part of the course record, but these course records often have a DW-SCHOOL attribute. When DWSchool is scribed and a course is encountered that does not have a school value, the value from the DW-SCHOOL attribute is used as the school value for comparison against the scribed DWSchool value.
25. Placing Hide after the With will suppress the With information from appearing in the advice on the worksheet.

# Transfer Equivalency Admin

Updated: March 25, 2022

Transfer Equivalency Admin is the data entry interface for managing Transfer Equivalency coursework articulation.

## Mappings

Updated: March 25, 2022

Use the Mappings module to create school or test mappings. Use school mappings to align a transfer school's courses to your school's equivalent courses. Use test mappings to align test scores to your school's (local) equivalent courses.

## School Mappings

Updated: March 25, 2022

Use school mappings to align a transfer school's courses to your school's equivalent courses.

### Create a new school mapping

Updated: March 25, 2022

You can create a new school mapping to align the courses from a transfer school to your school.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Mappings**.
2. Click **School Mappings**, if necessary.
3. In the **School ID** column, click the transfer school ID number.

**Note:** The school list shows favorite schools by default. Clear the **Show favorite schools only** check box to view all schools. The list of favorite schools comes from UCX-TRQ060, which contains a user-defined list of transfer schools. These transfer schools are the schools for which mappings are maintained frequently and from which your school receives many transcripts.

4. Click **Create new school mapping**.
5. In the **Transfer course** section, enter the applicable information.

Transfer course field	Description
Transfer discipline	Enter the course category. For example, Chemistry or Chem.
Transfer course num	Enter your school's or the 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 years the grade is valid after the class is complete.
When taken between catalog years	The catalog year range between the start and the end of the course.

6. In the **Mapping to local course** section, enter the applicable information.

Mapping to local course field	Description
Local discipline	Select the course category. For example, Chemistry or Chem.
Local course num	Select your school's course catalog number.
Title	Enter the full name of the course.
Credits	Enter the number of credits for the course.
Valid for catalog years between	Select the catalog year range within which a student completes a course for this mapping to be valid.

7. In the **Authorized by** field, enter the name of the person who can approve the mapping.
8. In the **Comments** field, enter any additional information.
9. **Optional:** In the **Transfer condition** section, enter the applicable information.

Transfer condition field	Description
Condition	Select the condition from drop-down list. For example, select <b>MAJOR</b> .
Operator	Select the comparison operator. For example, select <b>Is equal to</b> .
Value(s)	Enter the value to compare. For example, type <b>Chemistry</b> .

10. Click **Save**.



## Edit school mappings



Updated: March 25, 2022

You can add, edit, or delete school mappings as required.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Mappings**.
2. Click **School Mappings**, if necessary.
3. In the **School ID** column, click the transfer school ID number.

**Note:** Clear the **Show favorite schools only** check box to view all schools. The list of favorite schools comes from UCX-TRQ060, which contains a user-defined list of transfer schools. These transfer schools are the schools for which mappings are maintained frequently and from which your schools receive many transcripts.

4. In the **Mapping ID** column, click the ID number.
5. Add, edit, or delete transfer course or local course:
  - To add a transfer course or local course, click  and **Save** when finished.
  - To edit a transfer course or local course, click  and **Save** when finished.
  - To delete a transfer course or local course, select one or more rows and click **Remove** and **Save** when finished.

## Test Mappings

Updated: March 25, 2022

Use test mappings to align the transfer school's test scores to your school's equivalent scores.

### Create a new test mapping

Updated: March 25, 2022

You can create a new test mapping, to align test scores from a transfer school to your school.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Mappings**.
2. Click **Test Mappings**.
3. Click **Create new test mapping**.

4. In the **Test score** section, enter the applicable information.

Test Score field	Description
Test	Enter the test key.
Title	Enter the full name of the test.
Min score	Enter the minimum point total for the test.
Max score	Enter the maximum point total for the test.
Years ago	Enter the maximum number of years the score is valid after the test is taken.
When taken between catalog years	The catalog year range between the start and the end of the test.

5. In the **Mapping to local course** section, enter the applicable information.

Mapping to local course field	Description
Local discipline	Select the course category. For example, Chem.
Local course num	Select your school's course catalog number.
Title	Enter the full name of the course.
Credits	Enter the number of credits for the course.
Valid for catalog years between	The catalog year range within which a student completes a course for this mapping to be valid.

6. In the **Authorized by** field, enter the name of the person who can approve the mapping.
7. In the **Comments** field, enter any additional information.
8. **Optional:** In the **Transfer condition** section, enter the applicable information.

Transfer condition field	Description
Condition	Select the condition from drop-down list. For example, select <b>MAJOR</b> .
Operator	Select the comparison operator. For example, select <b>Is equal to</b> .
Value(s)	Enter the value to compare. For example, type <b>Chemistry</b> .



9. Click **Save**.

## Edit test mappings

Updated: March 25, 2022

You can add, edit, or delete test mappings as required.

## Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Mappings**.
2. Click **Test Mappings**.
3. In the **Mapping ID** column, click the ID number.
4. Add, edit, or delete the test score or local course.
  - To add new test score or local course, click  and **Save** when finished.
  - To edit a test score or local course, click  and **Save** when finished.
  - To delete a test score or local course, select one or more mappings. Click **Remove** and **Save** when finished.

## Transcript

Updated: March 25, 2022


You can use the Transcript module to add transcripts from multiple schools and maintain test scores. You can also update student records to add the detail of previously attended schools.

## Add or modify student information

Updated: March 25, 2022

The Student Information tab displays basic student details. You can also add student goals and specify articulation conditions.

## Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Transcript**.
2. In the **Student ID** column, click the ID number.
3. Click **Student Information**, if necessary.
4. In the **Student details** section, click . View or edit the applicable information.

Student details fields	Description
Student ID	The student ID number.
Status	The status of the student.
Degree	Select the course of study.
Level	Select the degree level of the student.

Student details fields	Description
Catalog year	Select the catalog year that is applicable to the student.

5. In the **Student goals** section, click  to add a goal. Click  to edit an existing goal.

Student goals	Description
Goal type	Select the type of degree. For example, Major.
Goal value	Select the course of study. For example, Chemistry.

6. In the **Student conditions** section, click  to add student conditions that mappings require.

To be able to view the condition fields, set the `treq.treqadmin.applicant.show.articulationConditions` setting to **True** in Controller, if enabled. You can add a maximum of ten conditions.



7. Click **Save**.

## Add or modify test scores

Updated: March 25, 2022

The Tests tab displays the list of tests a student took previously. You can add these test scores to articulate the test score to a course credit at your school.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Transcript**.
2. In the **Student ID** column, click the ID number.
3. Click **Test Scores**.
4. In the **Placement test scores** section, click  to add a test score. Click  to edit an existing test score.

Fields	Description
Test name	Select the name of the test.
Test title	Enter the title of the test.
Score	Enter the test score.
Test date	Select the date on which the student took the test.
Condition	Add user-defined conditions associated with this test score. You can add a maximum of four conditions.

5. Click **Save**.



---

## Add or modify transcript classes

Updated: March 25, 2022

You can add, edit, or delete transcript classes from each school transcript for a student. You can add as many transcript classes as necessary.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Transcript**.
2. In the **Student ID** column, click the ID number.
3. Click **Transcripts**.
4. From the **Transcript classes** field, select the applicable school.
5. In the **Transcript classes** section, click  to add a transcript class. Click  to edit an existing transcript class.

Fields	Description
Discipline	Enter the discipline.
Course num	Enter the course number.
Title	Enter the course title.
Grade	Select the grade the student achieved for this class.
Credits	Enter the number of credits the student earned for this class
Credit type	Select the type of credit.
Grade type	Select the type of grade.
Term	Select the term during which the student completed the class
Calendar	Select the calendar, the type of instruction followed for this class.
Condition	Add user-defined conditions associated with this class. You can add a maximum of four conditions.

6. Click **Save**.

## Add school transcripts

Updated: March 25, 2022

You can add transcripts from each of the schools a student has previously attended.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Transcript**.

2. In the **Student ID** column, click the ID number.
3. Click **Add School Transcripts**.
4. Search and select a transcript school.

**Note:** The school list shows favorite schools by default. Clear the **Show favorite schools only** check box to view all schools. The list of favorite schools comes from UCX-TRQ060, which contains a user-defined list of transfer schools. These transfer schools are the schools for which mappings are maintained frequently and from which your school receives many transcripts.

5. Click **Add transcript**.
6. Add the transcript classes. For more information, see *Add transcript classes*.

## Articulation

Updated: March 25, 2022

Use the Articulate module to view or edit articulated transfer classes for students. You can process the student transcripts against defined articulation rules and resolve classes with multiple, duplicate, or missing mappings.

On the Articulation page you can view a student's last articulation results, and you can perform articulation for a student if not performed previously. You can also view the last articulated date for that student.

The Articulation summary shows the percentages of the articulated, undecided, leftover, and duplicate transfer classes for the respective student. When you resolve all of the undecided, leftover, and duplicate mappings, the articulated summary shows 100 percent.

There are two sections:

- **Articulated classes:** You can view and edit the articulated classes from the Articulated classes section. When you resolve all classes and articulation is 100 percent, you can click **New Audit** to process the transfer audit.
- **Unresolved classes:** The Unresolved classes section includes undecided, leftover, and duplicate classes. You can resolve these classes to make them articulated classes.
  - Undecided classes are the transfer classes for which you can see multiple mappings.
  - Leftover classes are the classes for which there are no mappings.
  - Duplicate classes are the classes where more than one transfer course maps to the same class at your school.

## Perform a re-articulation

Updated: March 25, 2022

You can perform a re-articulation when you update mappings for a particular school or update any transcript classes for a student. When you re-articulate a student, Transfer Equivalency Admin overwrites the previous articulation with the updated articulation.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Articulate**.
2. Search for the applicable student.
3. On the student's row, in the **Action** column, click **View** or **Articulate**, whichever applies.

**Note:** **Articulate** appears only when you have not yet articulated the student at all. Otherwise, **View** appears.


4. In the **Articulation Summary** section, click **Re-articulate**.

## Edit articulated classes

Updated: March 25, 2022

You can edit already articulated classes to change the local course to which the transfer course is mapped.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Articulate**.
2. Search for the applicable student.
3. On the student's row, in the **Action** column, click **View**.
4. In the **Articulated classes** section, click  to edit the articulated classes.
5. In the **Local Course** section, in the **Course** field, select the course from your school to which you want to map the transfer course.

**Result:** The title of the course auto fills based on your selection.

6. In the **Credits** field, specify the equivalent credits for the course at your school.
7. Specify the equivalent **Grade** and the **Grade Points**.
8. Click **Save**.

## Resolve unresolved classes

Updated: March 25, 2022

You can resolve undecided, leftover, and duplicate types of unresolved classes.

### Resolve undecided classes

Updated: March 25, 2022

You can resolve the undecided classes from the Unresolved classes section. Undecided classes are the transfer classes for which multiple mappings were found.

#### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Articulate**.
2. Search for the applicable student.
3. On the student's row, in the **Action** column, click **Articulate** or **View**, whichever applies.
4. In the **Unresolved classes** section, on the applicable class with a status of Undecided, click **Resolve**.

**Result:** You see local classes paired with different transfer classes.

5. Select the most suitable option.
6. Click **Save**.

### Resolve leftover classes

Updated: March 25, 2022

Leftover classes are the transfer classes for which no mappings are found. To resolve these classes, use the **Articulate as zero credits** or the **Special mapping** option.

#### *Resolve leftover classes through zero credits*

*Updated: March 25, 2022*

You can resolve the leftover class by articulating them as zero credit. Here, the class is marked with zero credit and transfers to your school without any credit value.

#### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Articulate**.
2. Search for the applicable student.



3. On the student's row, in the **Action** column, click **Articulate** or **View**, whichever applies.
4. In the **Unresolved classes** section, on the applicable class with a status of Leftover, click **Resolve**.
5. Click **Articulate as zero credits** to move the class to the zero credits grid.
6. Click **Save**.

### *Resolve leftover classes through special mappings*


*Updated: March 25, 2022*

You can resolve leftover classes and transfer them to your school through special mappings.

#### **Procedure**

1. From the toolbar, go to **Transfer Equivalency Admin > Articulate**.
2. Search for the applicable student.
3. On the student's row, in the **Action** column, click **Articulate** or **View**, whichever applies.
4. In the **Unresolved classes** section, on the applicable class with a status of Leftover, click **Resolve**.
5. Click **Special Mapping** to move the class to the **Edit special mapping** window.
6. Select the mapping type:
  - **Global mapping - for all future articulation:** This is the default option selected. It makes this mapping applicable for all future articulations and specifies the applicable period.
  - **Student mapping – for this student only:** You can use this option to create a unique mapping for the selected student only.

**Note:** The following fields are disabled if you select **Student mapping – for this student only**:

  - **Beginning of time** and **End of time** when taken between catalog years.
  - **Comment**
7. To articulate the transfer course to a single course at your school, in the **Mapping to local course** section, select the course at your school and specify the credits, grade, and grade points.
8. To add more than one local course at your school, in the **Mapping to local course** section, click .
9. Click **Save**.

## Resolve duplicate classes

Updated: March 25, 2022

You can resolve the duplicate classes from the Unresolved classes section. Duplicate classes are classes where more than one transfer course maps to the same class at your school.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Articulate**.
2. Search for the applicable student.
3. On the student's row, in the **Action** column, click **Articulate** or **View**, whichever applies.
4. In the **Unresolved classes** section, on the applicable class with a status of Duplicate, click **Resolve**.
5. Select one or more appropriate courses for which the student should receive the credits.

**Note:** A class is articulated to zero credits if you do not select any transfer course from the list.

6. Click **Save**.

## 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 school, 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.




## Perform a degree audit

Updated: March 25, 2022

You can perform a degree audit and view the degree audit report. These goals will not be saved.

### Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Audit**.

2. In the **Student ID** column, click the applicable student ID number.
3. In the **Student Details** section, click  to update the student information.
4. In the **Student Goals** section, click  to add new goals, or click  to update the existing goals as necessary.
5. Click **Process new audit**.

## Roll

Updated: March 25, 2022

Use the Roll function to update Banner with articulated student data from Transfer Equivalency Admin.

If the student's data is articulated successfully but not yet rolled to the student system, Transfer Equivalency Admin allows the roll of 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.

During the roll process, a student's dap\_transfer\_dtl records are loaded in Banner's **SHRTRTK** table, which is used in Banner's transfer articulation process. Transfer Equivalency Admin will not write to the Banner transfer class tables **SHRTRCE** and **SHRTRCR**, which exist in Banner's academic history.

Before Transfer Equivalency Admin writes records to **SHRTRTK**, certain configurations are required on the Banner side. The **SHRTRTK** 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.

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.

## Roll articulation results

Updated: March 25, 2022

You can send the results of a student's articulation to the student system using the roll function.

## Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Roll**. **Roll** appears in the **Action** column for 100 percent articulated students.
2. On the student's row, in the **Action** column, click **Roll**.

## Processes and tables

Updated: March 25, 2022

You can use below processes and tables in the roll as necessary.


- **ban60**: A subroutine that writes `dap_transfer_dtl` data to **SHRTRTK**.
- Banner tables:
  - **SHRTRAM**: Student's prior attendance at a transfer college. An existing **SHRTRIT** table is required.
  - **SHRTRIT**: Student's link to a transfer college.
  - **SHRTRCE**, **SHRTRCR**: Academic history transfer classes extracted from Banner to Degree Works.
  - **SHRTRTK**: Banner's transfer articulation table. An existing **SHRTRAM** is required.
- Banner screens:
  - **SHATRNS**: Used to create the student relationship with the transfer college. Linked to the **SHRTRIT** and **SHRTRAM** tables.
  - **SHATAEQ**: Used for transfer articulation. Linked to the **SHRTRTK** table.

## View history

Updated: March 25, 2022

You can see who created or modified student records and when.

## Procedure

1. From the toolbar, go to **Transfer Equivalency Admin > Transcript**.
2. In the **Student ID** column, click the ID number.
3. Click **History** .

# Print report

Updated: March 25, 2022

You can send a mapping or audit report to PDF or to a printer.

## About this task

### Procedure

1. Access the applicable mapping or audit record.
2. Click **Print Report**.

---

# Transit

Updated: September 25, 2023

Transit is a web application that allows users of Degree Works to launch batch reports and processes.

## Transit Run Jobs

Updated: September 29, 2023

The Run Jobs tab in Transit is where authorized users define the criteria to select the pool of IDs on which a job should be run, provide answers for job questions, and launch, schedule, or save the parameters of that job.

## Run ADMIN - Administrative tasks

Updated: September 29, 2023

Instead of contacting your IT staff, you can use Transit to run certain administrative tasks yourself.

### Procedure

1. In the **Report or processor** field, select **ADMIN - Administrative tasks**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the **Select command to run** field, select the report or process you want to run.
4. In the space provided, enter a student or user ID if you are running dap22dbg, rad30dbg, or weblogon.
5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.

- Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
6. Choose when to run the job.
- Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run AUD01 - List unhooked and unenforced exceptions

Updated: September 29, 2023

The AUD01 report shows all unhooked and all unenforced exceptions.

### About this task

Ellucian recommends running this report regularly to identify whether exceptions have become unhooked. If you use label tags on all of your labels, however, unhooked exceptions should rarely occur. For more information, see the [Label Tags](#) topic.

This report runs against all records, as there are no selection options from Transit.

### Procedure

1. In the **Report or processor** field, select **AUD01 - List unhooked and unenforced exceptions**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.

- 
- Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
4. Choose when to run the job.
- Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run AUD02 - Delete audits by freeze type and date

Updated: September 29, 2023

With the AUD02 processor you can delete frozen audits older than a certain date.

### About this task

The primary use of the AUD02 is to clean out certain frozen audits periodically. For example, you may have created frozen audits using DAP27 to run batch what-if audits.

**Note:** Requiring a value in the **Select freeze type** field serves to prevent you from deleting all audits by accident.

### Procedure

1. In the **Report or processor** field, select **AUD02 - Delete audits by freeze type and date**.
  2. In the **Saved job parameters** field, select a saved job parameter option.
    - None - do not use existing nor create new saved job parameters.
    - New - create new saved job parameters. You are prompted for a job parameter description (required).
    - Previously saved job parameters, if they exist.
- Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.
3. In the **Select freeze type** field, select the type of audit freeze.
  4. Under **Delete audits older than**, select an applicable date.
  5. Select **Enable debugging** if you would like to debug.



**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

## 6. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run BAN62 - Banner SAP Processor

Updated: September 29, 2023

With BAN62, you can extract course information from a student's degree audit and store it in the Banner database for SAP determination.

### About this task

BAN62 populates the SHRSAPP and SHRSARJ tables in Banner. You have the option to refresh the student data and run a new degree audit or to run BAN62 on existing degree audits. For more information about SAP processing, see the [Satisfactory academic progress](#) topic.

### Procedure

1. In the **Report or processor** field, select **BAN62 - Banner SAP Processor**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>Student ID</b>	<ol style="list-style-type: none"> <li>Select <b>Student ID</b>.</li> <li>In the <b>Student ID</b> field, enter the applicable student ID.</li> <li>Click <b>Add Student ID</b>.</li> </ol>
<b>ID File</b>	<ol style="list-style-type: none"> <li>Select <b>ID File</b>.  <b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</li> <li>Click <b>Choose File</b>.</li> <li>In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</li> </ol>
<b>Enter SQL</b>	<ol style="list-style-type: none"> <li>Select <b>Enter SQL</b>.</li> <li>Enter the SQL Query in the space provided.  <b>Note:</b> <ul style="list-style-type: none"> <li>The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li> <li>You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul> </li> </ol>
<b>Student data</b>	<ol style="list-style-type: none"> <li>Select <b>Student data</b>.</li> <li>Make the applicable selections for each drop-down section.  <b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA or the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</li> </ol>

- If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

- If you select **Create new Audit**, make your additional selections.
- In the **Select term** field, select the applicable term.
- Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

## 8. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP16 - Parse Requirements Processor

Updated: September 29, 2023

DAP16 processes all Scribe requirement blocks it finds in the database.

### About this task

DAP16 reparses each block and updates the dap-req-crs-dtl, dap-req-link-dtl, and dap-result-dtl tables. In addition, it replaces the syntax and remarks trees in the daptrees directory. If there are errors during the parsing phase, DAP16 updates the parse status with No. If there are no errors, it sets the parse status to OK.

### Procedure

1. In the **Report or processor** field, select **Run DAP16 - Parse Requirements Processor**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
4. Choose when to run the job.
- Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP21 - Extract Articulated Transfers

Updated: September 29, 2023

The DAP21 batch program creates an XML file of articulated transfer data from DAPDB. You should use DAP21 if you use Degree Works Transfer Equivalency and want to FTP the articulated transfer data back to the student system.

### Procedure

1. In the **Report or processor** field, select **DAP21 - Extract Articulated Transfers**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>Student ID</b>	<ol style="list-style-type: none"><li>a. Select <b>Student ID</b>.</li><li>b. In the <b>Student ID</b> field, enter the applicable student ID.</li><li>c. Click <b>Add Student ID</b>.</li></ol>

Option	Description
<b>ID File</b>	<p>a. Select <b>ID File</b>.</p> <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> <p>b. Click <b>Choose File</b>.</p> <p>c. In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</p>
<b>Enter SQL</b>	<p>a. Select <b>Enter SQL</b>.</p> <p>b. Enter the SQL Query in the space provided.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li> <li>You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul>
<b>Student data</b>	<p>a. Select <b>Student data</b>.</p> <p>b. Make the applicable selections for each drop-down section.</p> <p><b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA <i>or</i> the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</p>

4. If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

5. If desired, select **Re-roll students who were already rolled**.
6. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

---

7. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP22 - Generate Audits

Updated: September 29, 2023

The DAP22 processor runs audits in batch, which means it produces an audit for every student, for every active student, or for a group of students selected by some criteria.

### About this task

It is rare you would need to do this from Transit, however, because batch audits should run after RAD11 completes, meaning you should have updated audits for your students as their data changes.

### Procedure

1. In the **Report or processor** field, select **DAP22 - Generate Audits**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>Student ID</b>	<ol style="list-style-type: none"><li>a. Select <b>Student ID</b>.</li><li>b. In the <b>Student ID</b> field, enter the applicable student ID.</li><li>c. Click <b>Add Student ID</b>.</li></ol>

Option	Description
<b>ID File</b>	<p>a. Select <b>ID File</b>.</p> <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> <p>b. Click <b>Choose File</b>.</p> <p>c. In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</p>
<b>Enter SQL</b>	<p>a. Select <b>Enter SQL</b>.</p> <p>b. Enter the SQL Query in the space provided.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li> <li>You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul>
<b>Student data</b>	<p>a. Select <b>Student data</b>.</p> <p>b. Make the applicable selections for each drop-down section.</p> <p><b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA <i>or</i> the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</p>

4. If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

5. In the **Select audit type** field, select the applicable audit type.
6. If you select **Create new audit**, make your additional selections.
7. In the **Select output** field, make your selection.

Option	Description
<b>No output</b>	Proceed to the next step.
<b>Create PDF file</b>	a. In the <b>Select PDF page dimensions</b> field, select the size and

Option	Description
	<p>orientation for the PDF output.</p> <p>b. If desired, select the applicable Locale.</p> <p>c. In the <b>Select audit report</b> field, select the applicable report.</p> <p>d. In the <b>Custom data filter</b> field, select the user class to represent the users who will be viewing the PDF. For example, if you select ADV, custom data that should be shown only to advisors will appear in the PDF.</p> <p>e. If desired, select <b>Create individual output files for each audit</b>.</p>
<b>Create raw XML</b>	Proceed to the next step.

8. If desired, select **Build audit results to allow queries against the database (CPA)**.
9. If desired, in the **Select sort** field, select how to sort the data.
10. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

11. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP27 - Generate What-if Audits

Updated: September 29, 2023

With DAP27, you can determine if some students have completed a certificate or another program.



## Procedure

1. In the **Report or processor** field, select **DAP27 - Generate What-if Audits**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>Student ID</b>	<ol style="list-style-type: none"> <li>a. Select <b>Student ID</b>.</li> <li>b. In the <b>Student ID</b> field, enter the applicable student ID.</li> <li>c. Click <b>Add Student ID</b>.</li> </ol>
<b>ID File</b>	<ol style="list-style-type: none"> <li>a. Select <b>ID File</b>.               <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> </li> <li>b. Click <b>Choose File</b>.</li> <li>c. In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</li> </ol>
<b>Enter SQL</b>	<ol style="list-style-type: none"> <li>a. Select <b>Enter SQL</b>.</li> <li>b. Enter the SQL Query in the space provided.               <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li> <li>• You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul> </li> </ol>
<b>Student data</b>	<ol style="list-style-type: none"> <li>a. Select <b>Student data</b>.</li> <li>b. Make the applicable selections for each drop-down section.</li> </ol>

Option	Description
	<p><b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA or the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</p>

- If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

- Select the applicable information in all required fields and additional fields as needed.
- In the **Select output** field, make your selection.

Option	Description
<b>No output</b>	Proceed to the next step.
<b>Create PDF file</b>	<ol style="list-style-type: none"> <li>In the <b>Select PDF page dimensions</b> field, select the size and orientation for the PDF output.</li> <li>If desired, select the applicable Locale.</li> <li>In the <b>Select audit report</b> field, select the applicable report.</li> <li>In the <b>Custom data filter</b> field, select the user class to represent the users who will be viewing the PDF. For example, if you select ADV, custom data that should be shown only to advisors will appear in the PDF.</li> <li>If desired, select <b>Create individual output files for each audit</b>.</li> </ol>
<b>Create raw XML</b>	Proceed to the next step.

- If desired, select **Include In-progress classes** and **Include Preregistered** classes.
- Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

- Choose when to run the job.
  - Click **Launch** to run the job now.

- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP28 - Generate Alternate What-if audits

Updated: September 29, 2023

You can run DAP28 on a set of selected students based on an alternate curriculum you have bridged for each student.

### Procedure

1. In the **Report or processor** field, select **DAP28 - Generate Alternate What-if audits**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
Student ID	<ol style="list-style-type: none"><li>a. Select <b>Student ID</b>.</li><li>b. In the <b>Student ID</b> field, enter the applicable student ID.</li><li>c. Click <b>Add Student ID</b>.</li></ol>
ID File	<ol style="list-style-type: none"><li>a. Select <b>ID File</b>.</li></ol> <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> <ol style="list-style-type: none"><li>b. Click <b>Choose File</b>.</li><li>c. In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</li></ol>
Enter SQL	<ol style="list-style-type: none"><li>a. Select <b>Enter SQL</b>.</li></ol>

Option	Description
	<p>b. Enter the SQL Query in the space provided.</p> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li> <li>You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul>
<b>Student data</b>	<p>a. Select <b>Student data</b>.</p> <p>b. Make the applicable selections for each drop-down section.</p> <p><b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA or the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</p>

4. If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

5. In the **Select freeze type** field, select the type of audit freeze.
6. In the **Select output** field, make your selection.

Option	Description
<b>No output</b>	Proceed to the next step.
<b>Create PDF file</b>	<p>a. In the <b>Select PDF page dimensions</b> field, select the size and orientation for the PDF output.</p> <p>b. If desired, select the applicable Locale.</p> <p>c. In the <b>Select audit report</b> field, select the applicable report.</p> <p>d. In the <b>Custom data filter</b> field, select the user class to represent the users who will be viewing the PDF. For example, if you select ADV, custom data that should be shown only to advisors will appear in the PDF.</p> <p>e. If desired, select <b>Create individual output files for each audit</b>.</p>
<b>Create raw XML</b>	Proceed to the next step.

- 
7. If desired, select **Include In-progress classes** and **Include Preregistered** classes.
  8. If desired, in the **Select sort** field, select how to sort the data.
  9. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

10. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP40 - Unload Scribe Blocks

Updated: September 29, 2023

The DAP40 batch program unloads your Scribe blocks to a zip file that you can download from Transit.

### About this task

You can then use the DAP41 job to upload this file into a different environment. This allows you to move blocks from your test to production environment, for example.

### Procedure

1. In the **Report or processor** field, select **DAP40 - Unload Scribe blocks**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

---

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the **Select blocks to unload** field, make your selection.
  - All blocks – unloads all blocks.
  - All but Planner blocks (RA blocks) – Planner blocks (with block IDs starting with RB) are excluded.
  - Only Planner blocks (RB blocks) – Only planner blocks are included.
4. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

5. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP41 - Load Scribe Blocks

Updated: September 29, 2023

The DAP41 batch program loads your Scribe blocks from a specified zip file that you upload to Transit.

### About this task

The zip file must have come from you running DAP40 to unload blocks from the same or another environment. This allows you to move blocks from your test to production environment, for example.

### Procedure

1. In the **Report or processor** field, select **DAP41 - Load Scribe blocks**.
2. Under **Select the blocks file to load**, click **Choose File**.

- 
3. In the **File Upload** window, navigate to and select the .zip file containing the blocks to load, and click **Open**.
  4. In the **Select blocks to load** field, make your selection.
    - All blocks – unloads all blocks.
    - All but Planner blocks (RA blocks) – Planner blocks (with block IDs starting with RB) are excluded.
    - Only Planner blocks (RB blocks) – Only planner blocks are included.
  5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
6. Choose when to run the job.
    - Click **Launch** to run the job now.
    - Click **Schedule** to select a date and time to run the job.

**Note:** You do not see the **Schedule** button unless you have sufficient permissions.

## Run DAP42 - Unload Mappings

Updated: September 29, 2023

The DAP42 batch program unloads your mappings to a zip file that you can download from Transit.

### About this task

You can then use the DAP43 job to upload this file into a different environment.

### Procedure

1. In the **Report or processor** field, select **DAP42 - Unload Mappings**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).

- Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP43 - Load Mappings

Updated: September 29, 2023

The DAP43 batch program loads your mappings from a specified zip file that you upload to Transit.

### Procedure

1. In the **Report or processor** field, select **DAP43 - Load Mappings**.
2. Under **Select the mappings file to load**, click **Choose File**.
3. In the **File Upload** window, navigate to and select the .zip file containing the mappings to load, and click **Open**.
4. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.



---

5. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job.

**Note:** You do not see the **Schedule** button unless you have sufficient permissions.

## Run DAP44 - Unload UCX Tables

Updated: September 29, 2023

The DAP44 batch program unloads your UCX tables to a zip file that you can download from Transit.

### About this task

You can then use the DAP45 job to upload this file into a different environment.

### Procedure

1. In the **Report or processor** field, select **DAP44 - Unload UCX Tables**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.

- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP45 - Load UCX Tables

Updated: September 29, 2023

The DAP45 batch program loads your UCX tables from a specified zip file that you upload to Transit.

### Procedure

1. In the **Report or processor** field, select **DAP45 - Load UCX Tables**.
2. Under **Select the UCX file to load**, click **Choose File**.
3. In the **File Upload** window, navigate to and select the .zip file containing the UCX tables to load, and click **Open**.
4. Select **Enable debugging** if you would like to debug.

#### **Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
5. Choose when to run the job.
    - Click **Launch** to run the job now.
    - Click **Schedule** to select a date and time to run the job.

**Note:** You do not see the **Schedule** button unless you have sufficient permissions.

## Run DAP54 - Create Plan from Template Processor

Updated: September 29, 2023

You can assign plans in batch to students using this processor. You can specify the template for all students in your pool, or you can let the processor determine the template for students based on catalog year, school, degree, and major.

## Procedure

1. In the **Report or processor** field, select **DAP54 - Create Plan from Template Processor**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>Student ID</b>	<ol style="list-style-type: none"> <li>a. Select <b>Student ID</b>.</li> <li>b. In the <b>Student ID</b> field, enter the applicable student ID.</li> <li>c. Click <b>Add Student ID</b>.</li> </ol>
<b>ID File</b>	<ol style="list-style-type: none"> <li>a. Select <b>ID File</b>.</li> </ol> <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> <ol style="list-style-type: none"> <li>b. Click <b>Choose File</b>.</li> <li>c. In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</li> </ol>
<b>Enter SQL</b>	<ol style="list-style-type: none"> <li>a. Select <b>Enter SQL</b>.</li> <li>b. Enter the SQL Query in the space provided.</li> </ol> <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li> <li>• You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul>
<b>Student data</b>	<ol style="list-style-type: none"> <li>a. Select <b>Student data</b>.</li> <li>b. Make the applicable selections for each drop-down section.</li> </ol>

Option	Description
	<p><b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA or the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</p>

- If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

- In the space provided, enter the applicable template ID.
- In the **Select starting term** field, select the applicable term with which to start.
- If applicable, select **Are these plans temporary**.
- Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
- Choose when to run the job.
    - Click **Launch** to run the job now.
    - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
    - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP58 - Batch Tracking Processor

Updated: September 29, 2023

Using DAP58, you can track whether your students are following their educational plans. The processor updates the tracking status for each requirement, term, and plan, allowing you to run reports against the data.

## Procedure

1. In the **Report or processor** field, select **DAP58 - Batch Tracking Processor**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>Student ID</b>	<ol style="list-style-type: none"> <li>a. Select <b>Student ID</b>.</li> <li>b. In the <b>Student ID</b> field, enter the applicable student ID.</li> <li>c. Click <b>Add Student ID</b>.</li> </ol>
<b>ID File</b>	<ol style="list-style-type: none"> <li>a. Select <b>ID File</b>.               <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> </li> <li>b. Click <b>Choose File</b>.</li> <li>c. In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</li> </ol>
<b>Enter SQL</b>	<ol style="list-style-type: none"> <li>a. Select <b>Enter SQL</b>.</li> <li>b. Enter the SQL Query in the space provided.               <p><b>Note:</b></p> <ul style="list-style-type: none"> <li>• The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li> <li>• You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul> </li> </ol>
<b>Student data</b>	<ol style="list-style-type: none"> <li>a. Select <b>Student data</b>.</li> <li>b. Make the applicable selections for each drop-down section.</li> </ol>

Option	Description
	<p><b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA or the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</p>

4. If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

5. If applicable, select **Is this an official run of the processor**.
6. In the **Select cutoff term** field, select the applicable term.
7. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

8. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run DAP59 - Batch Timetabling Processor

Updated: September 29, 2023

Using DAP59, you can run audits against student educational plans.

### Before you begin

Before running DAP59, ensure that the UCX-CFG020 RESULTS Build Stinfo flag is set to Y so that all the data required for timetabling is created.

## About this task

The timetabling processor runs an audit for each of the student's active and locked/approved plans for the given school/level specified. If the processor cannot find an active and locked/approved plan for a student, it looks for a temporary plan DAP54 created from a template. The processor includes in the audit all classes after the student's active term, up to and including the cutoff term specified. If the student has preregistered for the same classes that appear on the plan, the processor does not include the duplicate classes. The processor saves the audit to the CPA tables so you can see how it applied each of the planned classes in the audit.

## Procedure

1. In the **Report or processor** field, select **DAP59 - Batch Timetabling Processor**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>Student ID</b>	<ol style="list-style-type: none"> <li>a. Select <b>Student ID</b>.</li> <li>b. In the <b>Student ID</b> field, enter the applicable student ID.</li> <li>c. Click <b>Add Student ID</b>.</li> </ol>
<b>ID File</b>	<ol style="list-style-type: none"> <li>a. Select <b>ID File</b>.               <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> </li> <li>b. Click <b>Choose File</b>.</li> <li>c. In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</li> </ol>
<b>Enter SQL</b>	<ol style="list-style-type: none"> <li>a. Select <b>Enter SQL</b>.</li> <li>b. Enter the SQL Query in the space provided.</li> </ol> <p><b>Note:</b></p>

Option	Description
	<ul style="list-style-type: none"> <li>The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li> <li>You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul>
<b>Student data</b>	<p>a. Select <b>Student data</b>.</p> <p>b. Make the applicable selections for each drop-down section.</p> <p><b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA or the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</p>

4. If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

5. In the **Select level** field, select the applicable college level.
6. In the **Select cutoff term** field, select the applicable term.
7. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

8. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.



---

## Run RAD11 - Radbridge Batch Processor

Updated: September 29, 2023

RAD11 loads extracted data from the student information systems (SIS) into the Degree Works database.

### About this task

For more information, see the *Bridge Interface Format* topic.

### Procedure

1. In the **Report or processor** field, select **RAD11 - Radbridge Batch Processor**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the space provided, enter the name of the applicable BIF file.

**Note:** The BIF file must already exist in your admin/data directory on the classic server.

4. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

5. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run RAD30 - Banner Student Extract and Bridge

Updated: September 29, 2023

You can use RAD30 to run the Banner STUDENT extract, which bridges active students' academic and basic biographic data from Banner into Degree Works.

### About this task

You can select the students to bridge either by specifying a list of student IDs or configuring the SQL in the `integration.banner.extract.student.sql.daily` Shepherd setting.

RAD30 generates new audits for students with data changes.

Be sure your UCX-CFG020 BANNER settings are correct before running the extract.

### Procedure

1. In the **Report or processor** field, select **RAD30 - Banner Extract and Bridge**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>SIS ID</b>	<ol style="list-style-type: none"><li>a. Select <b>SIS ID</b>.</li><li>b. In the <b>SIS ID</b> field, enter the applicable SIS ID.</li><li>c. Click <b>Add SIS ID</b>.</li></ol>
<b>ID file</b>	<ol style="list-style-type: none"><li>a. Select <b>ID File</b>.  <b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</li><li>b. Click <b>Choose File</b>.</li><li>c. In the <b>File Upload</b> window, navigate to and select the file containing SIS IDs, and click <b>Open</b>.</li></ol>

---

Option	Description
Default SQL File	This option uses the SQL defined in the integration.banner.extract.student.sql.daily Shepherd setting.

4. If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

5. If desired, select **Force new audits to be run for students even when there are no data changes**.
6. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

7. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run RAD32 - Banner Applicant Extract and Bridge

Updated: September 29, 2023

You can use RAD32 to run the Banner APPLICANT extract, which bridges admissions applicants academic and basic biographic data from Banner into Degree Works.

### About this task

You can select the applicants to bridge either by specifying a list of applicant IDs or configuring the SQL in the integration.banner.extract.applicant.sql.daily Shepherd setting.

The job generates new audits for applicants with data changes.

Be sure your UCX-CFG020 BANNER settings are correct before running the extract.

---

## Procedure

1. In the **Report or processor** field, select **Run RAD32 - Banner Applicant Extract and Bridge**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>SIS ID</b>	<ol style="list-style-type: none"><li>a. Select <b>SIS ID</b>.</li><li>b. In the <b>SIS ID</b> field, enter the applicable SIS ID.</li><li>c. Click <b>Add SIS ID</b>.</li></ol>
<b>ID file</b>	<ol style="list-style-type: none"><li>a. Select <b>ID File</b>.</li></ol> <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> <ol style="list-style-type: none"><li>b. Click <b>Choose File</b>.</li><li>c. In the <b>File Upload</b> window, navigate to and select the file containing SIS IDs, and click <b>Open</b>.</li></ol>
<b>Default SQL File</b>	This option uses the SQL defined in the integration.banner.extract.applicant.sql.daily Shepherd setting.

4. If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

6. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run RAD33 - Banner Non-Student Extract and Bridge

Updated: September 29, 2023

You use RAD33 to run the Banner NON-STUDENT extract, which creates access records for staff who require access to Degree Works. You can use RAD33 to extract advisors, deans, and so on.

### About this task

You can specify a list of staff IDs or indicate that the default SQL should be used to select the pool of users for the user class specified.

Be sure your UCX-CFG020 BANNER settings are correct before running the extract.

### Procedure

1. In the **Report or processor** field, select **Run RAD33 - Banner Non-Student Extract and Bridge**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>SIS ID</b>	<ol style="list-style-type: none"><li>a. Select <b>SIS ID</b>.</li><li>b. In the <b>SIS ID</b> field, enter the applicable SIS ID.</li><li>c. Click <b>Add SIS ID</b>.</li></ol>

Option	Description
<b>ID file</b>	<p>a. Select <b>ID File</b>.</p> <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> <p>b. Click <b>Choose File</b>.</p> <p>c. In the <b>File Upload</b> window, navigate to and select the file containing SIS IDs, and click <b>Open</b>.</p>
<b>Default SQL file</b>	This option uses the SQL defined in the integration.banner.extract.<user class>.sql.daily setting. If you choose a user class of ADVX, the setting that will be used to select these users will be integration.banner.extract.advx.sql.daily, for example. You can use Controller to create additional sql.daily settings for whichever user classes from AUD012 you use.

- If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

- Select the user class of the users being extracted.

**Note:** You can use the **Show in RAD33** flag in UCX-AUD012 User Class Codes to configure which user classes display in this drop-down.

- Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

- Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run RAD34 - Banner Course Extract

Updated: September 29, 2023

You can use RAD34 in to run the Banner COURSE extract, which extracts all current courses from your course catalog in Banner to Degree Works.

### About this task

This job only adds or updates rad\_course\_mst records, but deletes and re-adds rad\_crs\_attr\_dtl records.

Be sure your UCX-CFG020 BANNER settings are correct before running the extract.

### Procedure

1. In the **Report or processor** field, select **Run RAD34 - Banner Course Extract**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run RAD35 - Banner Curriculum Rules Extract

Updated: September 29, 2023

You can use RAD35 to run the Banner CURRRULE extract, which bridges curriculum rule data from Banner to Degree Works.

### About this task

This job first deletes the old curriculum rules and then re-adds the new curriculum rules.

### Procedure

1. In the **Report or processor** field, select **Run RAD35 - Banner Curriculum Rules Extract**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.



---

## Run RAD36 - Banner Validation table Extract (UCX)

Updated: September 29, 2023

You can use RAD36 to run the Banner UCX extract, which creates Degree Works validation tables using data from the Banner validation tables.

### About this task

Be sure to check the UCX-CFG020 RADBRIDGE setting for Add UCX Entries Only before running the extract. If UCX-CFG020 RADBRIDGE Add UCX Entries Only = N, the job deletes all records and re-adds them. If UCX-CFG020 RADBRIDGE Add UCX Entries Only = Y, it only adds new records (no updates).

### Procedure

1. In the **Report or processor** field, select **Run RAD36 - Banner Validation table Extract**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run RAD37 - Banner Transfer School Extract (ETS)

Updated: September 29, 2023

You can use RAD37 to run the Banner ETS extract, which bridges transfer institution ETS codes, names, and identification data from Banner to Degree Works.

### About this task

RAD37 only adds or updates rad\_ets\_mst records; it does not delete.

### Procedure

1. In the **Report or processor** field, select **Run RAD37 - Banner Transfer School Extract (ETS)**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run RAD38 - Banner Equivalencies Extract

Updated: September 29, 2023

You can use RAD38 to run the Banner EQUIV extract, which bridges historic courses and their current equivalent courses from Banner to Degree Works.

### About this task

The job first deletes both the dap\_eqv\_crs\_mst and UCX-CFG070 and re-adds all equivalencies.

### Procedure

1. In the **Report or processor** field, select **Run RAD38 - Banner Equivalencies Extract**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run RAD39 - Banner Transfer Equivalency Extract (Mappings)

Updated: September 29, 2023

You can use RAD39 to run the Banner MAPPINGS extract, which bridges transfer equivalent rules from Banner to Degree Works.

### About this task

For each school ID included in the extract, RAD 39 first deletes the dap\_mapping\_dtl where dap\_create\_who=BRIDGE before adding new records. Because of this, each time you run the extract, it assigns a new set of dap\_map\_id values.

Be sure your properties settings are correct before running the extract.

### Procedure

1. In the **Report or processor** field, select **Run RAD39 - Banner Transfer Equivalency Extract (Mappings)**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run RAD40 - Student Data Delete Processor

Updated: September 29, 2023

RAD40 can be used to permanently delete student records from the Degree Works database. This deletes Degree Works data only. The student's data in the SIS is not deleted.

### Procedure

1. In the **Report or processor** field, select **RAD40 - Student Data Delete Processor**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Make your selection.

Option	Description
<b>Student ID</b>	<ol style="list-style-type: none"><li>a. Select <b>Student ID</b>.</li><li>b. In the <b>Student ID</b> field, enter the applicable student ID.</li><li>c. Click <b>Add Student ID</b>.</li></ol>
<b>ID File</b>	<ol style="list-style-type: none"><li>a. Select <b>ID File</b>.</li></ol> <p><b>Note:</b> The ID file must be a plain ASCII text file with one ID per line. Do not import Excel, Word, or any other type of file.</p> <ol style="list-style-type: none"><li>b. Click <b>Choose File</b>.</li><li>c. In the <b>File Upload</b> window, navigate to and select the file containing student IDs, and click <b>Open</b>.</li></ol>
<b>Enter SQL</b>	<ol style="list-style-type: none"><li>a. Select <b>Enter SQL</b>.</li><li>b. Enter the SQL Query in the space provided.</li></ol> <p><b>Note:</b></p> <ul style="list-style-type: none"><li>• The SQL query can be any valid query from the Degree Works database. It should select only one column that must be an ID code.</li></ul>

Option	Description
	<ul style="list-style-type: none"> <li>You do not see the <b>Enter SQL</b> option unless you have sufficient permissions.</li> </ul>
<b>Student data</b>	<p>a. Select <b>Student data</b>.</p> <p>b. Make the applicable selections for each drop-down section.</p> <p><b>Note:</b> Multiple selections in the same data item will be an <i>or</i>, while selections from different data items are an <i>and</i>. For example, if you select Degree=BA and Degree=BS and Major=ANTH, you are saying to select any student with either the BA or the BS degree <i>and</i> the ANTH major, so BA ANTH or BS ANTH students.</p>

4. If desired, click **Find** to see how many records the job will contain based on the criteria you entered.

**Note:** The find results feature is available only for the SIS ID selection.

5. If desired, select **Delete non-bridged data**.

This will completely delete all of the student's data from Degree Works, academic history and goal data, in addition to audits, exceptions, and plans. After deletion, there is no way to recover this data without restoring it from a database backup, so be sure that your selection pool contains only the students that you intend to delete.

6. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

7. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run SCR02 - Find blocks where this course is referenced

Updated: September 29, 2023

Using SCR02, you can find blocks that reference certain courses and copy and paste the report into a spreadsheet.

### Procedure

1. In the **Report or processor** field, select **SCR02 - Find blocks where this COURSE is referenced**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the space provided, enter the course discipline.
4. In the space provided, enter the course number.
5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

6. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run SCR05 - List blocks changed by date range

Updated: September 29, 2023

Using SCR05, you can get a list of all the blocks you have in Degree Works, or you can limit the number of blocks in the list by date range.

### About this task

This report sorts by block type and value. If you do specify a date range, the report also lists the modify date for each block selected.

### Procedure

1. In the **Report or processor** field, select **SCR05 - List blocks changed by date range**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. If desired, in the **Start date** field, enter or select the oldest date in the range.
4. If desired, in the **End date** field, enter the most recent date in the range.
5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
6. Choose when to run the job.
    - Click **Launch** to run the job now.
    - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
    - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.



---

## Run SCR06 - List block primary and secondary tags

Updated: September 29, 2023

Using SCR06, you can get a list of all blocks showing their primary and secondary tags.

### About this task

For example, you can easily see which blocks use a Program or Student ID secondary tag.

### Procedure

1. In the **Report or processor** field, select **SCR06 - List block primary and secondary tags**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run SCR07 - List block text from Scribe

Updated: September 29, 2023

Using SCR07, you can get a report listing all of your blocks with all tags and text lines, sorted by the block type and value.

### About this task

You may want this information to save as a backup or view in bulk.

### Procedure

1. In the **Report or processor** field, select **SCR07 - List block text from Scribe**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the **Select catalog year** field, select the applicable year.
4. In the **Select block type** field, select the applicable block type.
5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
6. Choose when to run the job.
    - Click **Launch** to run the job now.
    - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
    - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run SCR08 - List log entries from Scribe text

Updated: September 29, 2023

Using SCR08, you can get a list of all blocks with LOG text lines, sorted by the block type and value and text sequence number. This job looks only for lines with LOG in columns 1-3 of the text line.

### Procedure

1. In the **Report or processor** field, select **SCR08 - List LOG entries from Scribe text**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run SCR09 - List todo entries from Scribe text

Updated: September 29, 2023

Using SCR09, you can get a list of all blocks with TODO text lines, sorted by the block type and value and text sequence number. This job looks only for lines with TODO in columns 1-4 of the

---

text line.

### Procedure

1. In the **Report or processor** field, select **SCR09 - List TODO entries from Scribe text**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

4. Choose when to run the job.
  - Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run SCR10 - Find blocks where this text is referenced

Updated: September 29, 2023

Using SCR10, you can find the blocks that make use of remarks or proxy-advice or use certain values. The report lists the text lines containing the specified string in addition to the block information.

### Procedure

1. In the **Report or processor** field, select **SCR10 - Find blocks where this text is referenced**.

---

2. In the **Saved job parameters** field, select a saved job parameter option.

- None - do not use existing nor create new saved job parameters.
- New - create new saved job parameters. You are prompted for a job parameter description (required).
- Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the space provided, enter the text string you want to find.

**Note:** The search is case sensitive. For example, if you type "Remark" it will not find "remark" or "REMARK."

4. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

5. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run SCR11 - Find-and-Replace block text

Updated: September 29, 2023

Using SCR11, you can find any string of text up to 30 characters and replace it with a new string. The new text can be shorter or longer than the old text, but the new text also has a limit of 30 characters.

### About this task

Before running SCR11, you should first run SCR10 to find the text that you want to replace to understand what will happen when you run SCR11. Like SCR10, the search in SCR11 is case sensitive so if you search for "remark" it will only find the lowercase occurrences of that word. You may have to run SCR11 again for "Remark" and then again for "REMARK."

---

You should not use this report to change names of courses. For example, if you have a rule of "1 Class in HIST 102, 106" you cannot use this report to change "HIST 106" to something else because it would not find "HIST 106." This find-and-replace does not understand the Scribe language.

Because this is such a powerful tool, you could end up replacing too much text or the wrong text, even if you study the results of SCR10 first. For this reason, you should make a backup of your blocks using SCR07 or DAP40 in addition to your normal database backup, so you have a way to undo your work in case of a mistake. There is no "undo" button in this report. If you make a mistake, you must restore from a saved backup or fix the mistakes manually.

You should run DAP16 after making changes using SCR11 to ensure your blocks parse successfully and to allow new audits to pull in your changes.

## Procedure

1. In the **Report or processor** field, select **SCR11 - Find-and-Replace block text**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the first space provided, enter the text you want to find.
4. In the second space provided, enter the replacement text.
5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
6. Choose when to run the job.
    - Click **Launch** to run the job now.
    - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
    - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

---

## Run SCR91 - Test Banner Prerequisite Checker Service

Updated: September 29, 2023

Using SCR91, you can test the data Degree Works will return given this mock Banner registration data.

### Procedure

1. In the **Report or processor** field, select **SCR91 - Text Banner Prerequisite Checker Service**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the space provided, enter the applicable SPRIDEN ID.
4. If desired, in the **Select level** field, select the applicable school level.
5. If desired, in the **Select degree** field, select the applicable degree.
6. Enter the information for the first course.
  - a. In the **Select registration term** field, select the applicable term.
  - b. In the space provided, enter the Banner CRN.
  - c. In the space provided, enter the course discipline.
  - d. In the space provided, enter the applicable course number.
  - e. In the space provided, enter the section number.
7. **Optional:** Enter information for the second course registration.
8. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
9. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run SCR92 - Test Banner Prerequisite Description Service

Updated: September 29, 2023

Using SCR92, you can test the data Degree Works will return given this mock Banner catalog request.

### Procedure

1. In the **Report or processor** field, select **SCR92 - Test Banner Prerequisite Description Service**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the space provided, enter the applicable SPRIDEN ID.
4. If desired, in the **Select level** field, select the applicable school level.
5. If desired, in the **Select degree** field, select the applicable degree.
6. In the space provided, enter the Banner CRN.
7. In the **Select registration term** field, select the applicable term.
8. In the space provided, enter the course discipline.
9. In the space provided, enter the applicable course number.
10. In the space provided, enter the section number.
11. Select **Enable debugging** if you would like to debug.

**Note:**



- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

12. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run SCR93 - Report by Catalog

Updated: September 29, 2023

SCR93 can show the REQUISITE block data for all SCBCRSE records with the Prereq-in-Degree Works flag enabled.

### About this task

If the job does not find a match, it displays an error. You can restrict the report to only Banner records that are missing Degree Works blocks, or to display all of the REQUISITE block text.

### Procedure

1. In the **Report or processor** field, select **SCR93 - Report by CATALOG**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. If desired, select **Only show Banner records that are missing Degree Works blocks**.
4. If desired, select **Show block text for each requisite block found**.
5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

## 6. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run SCR94 - Report by Schedule

Updated: September 29, 2023

SCR94 can show the REQUISITE block data for all SCBCRSE and SSBSECT records with the Prereq-in-Degree Works flag enabled. If the job does not find a match, it displays an error. You can restrict the report to only Banner records that are missing Degree Works blocks, or display all of the REQUISITE block text.

### About this task

If the job does not find a match, it displays an error. You can restrict the report to only Banner records that do not have the flag enabled.

### Procedure

1. In the **Report or processor** field, select **SCR94 - Report by SCHEDULE**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. If desired, select **Only show Banner records that are missing Degree Works blocks**.

4. If desired, select **Show block text for each requisite block found**.
5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
- Debugging cannot be enabled when scheduling jobs.
- The **Enable debugging** selection is not included when creating saved job parameters.

6. Choose when to run the job.

- Click **Launch** to run the job now.
- Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
- Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run SCR95 - Report by Requisite block

Updated: September 29, 2023

SCR95 can show the REQUISITE block data for all SCBCRSE and SSBSECT records with the Prereq-in-Degree Works flag enabled. If the job does not find a match, it displays an error. You can restrict the report to only Banner records that do not have the flag enabled.

### Procedure

1. In the **Report or processor** field, select **SCR95 - Report by REQUISITE block**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. If desired, select **Only show REQUISITE records that are disabled in Banner**.
4. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.
  - Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
5. Choose when to run the job.
- Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.

**Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Run UCX01 - UCX Records Modified

Updated: September 29, 2023

UCX01 lists all UCX records modified during a specific date range. The report lists the UCX table number, UCX key, and modified date.

### Procedure

1. In the **Report or processor** field, select **UCX01 - UCX Records Modified**.
2. In the **Saved job parameters** field, select a saved job parameter option.
  - None - do not use existing nor create new saved job parameters.
  - New - create new saved job parameters. You are prompted for a job parameter description (required).
  - Previously saved job parameters, if they exist.

**Note:** You do not see the **Saved job parameters** field unless you have sufficient permissions.

3. In the **Start date** field, enter the first date in the range for which you want to see modified UCX records.
4. If desired, in the **End date** field, enter the last date in the range for which you want to see modified UCX records.
5. Select **Enable debugging** if you would like to debug.

**Note:**

- You do not see the **Enable debugging** check box unless you have sufficient permissions.

- Debugging cannot be enabled when scheduling jobs.
  - The **Enable debugging** selection is not included when creating saved job parameters.
6. Choose when to run the job.
- Click **Launch** to run the job now.
  - Click **Schedule** to select a date and time to run the job. If you would like to make it a recurring job, click **Schedule recurring job** and make the date, time, and recurrence changes as needed.
  - Click **Save** to save the job parameters without launching or scheduling the job.
- Note:** You do not see the **Schedule** or **Save** buttons unless you have sufficient permissions.

## Transit Completed Jobs

Updated: September 29, 2023

You can view the output of, download job artifacts for, and delete completed jobs.

### View completed jobs

Updated: September 29, 2023

You can view and download job artifacts as needed.

#### Procedure

1. Click **Completed Jobs**.
2. Filter or sort the jobs.

Option	Description
<b>Filter</b>	In the space provided, enter the applicable criteria.
<b>Sort</b>	Click any of the available column headings to sort in ascending or descending order.

**Note:** To the right of a job, you can click **Information** ⓘ to see details about that job.

3. On the far right, click **Expand row** ▾.
4. Click **View** to view the artifact output, or **Download** to save the artifact to your computer.

---

## Delete completed jobs

Updated: September 29, 2023

You can delete completed or failed jobs as needed.

### Procedure

1. Click **Completed Jobs**.
2. Filter or sort the jobs.

Option	Description
<b>Filter</b>	In the space provided, enter the applicable criteria.
<b>Sort</b>	Click any of the available column headings to sort in ascending or descending order.

**Note:** To the right of a job, you can click **Information** ⓘ to see details about that job.

3. Select the applicable job or jobs. You can select one or multiple individually, or you can select all of the jobs visible on the page by clicking the check box at the top of the column. You cannot delete jobs that are in a pending or running state.
4. Delete the job or jobs.

Option	Description
<b>Delete one job</b>	To the right of the job, click <b>Delete</b> .
<b>Delete more than one or all jobs visible on the page</b>	At the top right, click <b>Delete</b> .

## Transit Scheduled Jobs

Updated: September 29, 2023

You can view the next scheduled time of and edit or delete scheduled and recurring jobs.

## View or edit scheduled jobs

Updated: September 29, 2023

You can view and edit scheduled and recurring jobs as needed.

---

## Procedure

1. Click **Scheduled Jobs**.
2. Filter or sort the jobs.

Option	Description
<b>Filter</b>	In the space provided, enter the applicable criteria.
<b>Sort</b>	Click any of the available column headings to sort in ascending or descending order.

**Note:** To the right of a job, you can click **Information**  to see details about that job.

3. Click **Type** to view or edit the scheduled job.

## Delete scheduled jobs

Updated: September 29, 2023

You can delete a scheduled job as needed.

## Procedure

1. Click **Scheduled Jobs**.
2. Filter or sort the jobs.

Option	Description
<b>Filter</b>	In the space provided, enter the applicable criteria.
<b>Sort</b>	Click any of the available column headings to sort in ascending or descending order.

**Note:** To the right of a job, you can click **Information**  to see details about that job.

3. Delete the applicable scheduled job, and select whether to delete only the next scheduled instance or the entire scheduled series.

**Note:** You cannot delete a scheduled job if you do not have sufficient permissions to that job.

---

## Transit Saved Jobs

Updated: September 29, 2023

You can view when job parameters are in use by a scheduled job, and edit or delete saved job parameters.

### View or edit saved job parameters

Updated: September 29, 2023

You can view and edit saved job parameters as needed.

#### Procedure

1. Click **Saved Jobs**.
2. Filter or sort the jobs.

Option	Description
<b>Filter</b>	In the space provided, enter the applicable criteria.
<b>Sort</b>	Click any of the available column headings to sort in ascending or descending order.

**Note:** If you have the applicable authorization key, you can see the next time saved job parameters will be used in a scheduled or recurring job by clicking the number in the **In Use** column.

3. In the **Description** column, click the name of the saved job to view or edit the parameters.

### Delete saved job parameters

Updated: September 29, 2023

You can delete saved job parameters as needed.

#### Procedure

1. Click **Saved Jobs**.
2. Filter or sort the jobs.

Option	Description
<b>Filter</b>	In the space provided, enter the applicable



---

Option	Description
	criteria.
<b>Sort</b>	Click any of the available column headings to sort in ascending or descending order.

3. Delete the applicable saved job parameters.

**Note:** You cannot delete saved job parameters that are in use by a scheduled or recurring job or if you do not have sufficient permissions to that job.