



Ellucian Degree Works

Reference, 5.1.2

February 21, 2024

Notices and Privacy

© 2024 Ellucian.

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

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

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

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

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

Contents

Advanced Reporting	20
Data generation with the Curriculum Planning Assistant	20
CPA database tables	20
DAP_RESULT_DTL	20
DAP_RESNONCR_DTL	21
DAP_RESCLASS_DTL	22
Database views	23
Database space needed	24
UCX-CFG020 RESULTS configuration	25
Result types	26
AIDCUMCREDITS	27
AIDGRADE	28
AIDTERM	29
AIDTERMCREDITS	29
AUDITERROR	30
AUDITID	30
BLOCKCRCLAPPLIED	31
BLOCKCRCLNEEDED	32
BLOCKGPA	32
CATALOGYEAR	33
CLASSAPPLIED	34
CLASSNEEDED	34
CLASSREQUIRED	35
CRCLNEEDED	36
CRCLREQUIRED	36
EXCEPTION	37
FALLCLASSAPPLIED	38
FALLCRCL	39
FALLNONCRAPPLIED	39
GOALDATA	40
HEADERADVICE	41
HEADERQUAL	41

HEADERQUAL.....	49
INSUFFCLASSAPPLIED.....	56
INSUFFCRCL.....	57
LABEL.....	57
LABEL.....	58
NONCRAPPLIED.....	58
NONCRNEEDED.....	59
NONCRNUMNEED.....	60
OTLCLASSAPPLIED.....	60
OTLCRCL.....	61
PLAN.....	62
QUALADVICE.....	62
REMARK.....	63
RULEADVICE.....	63
Node-Types.....	64
Numeric fields.....	66
Rule ID.....	67
MAYTAKE vs MUSTTAKE vs CANTTAKE.....	67
Academic audits in CPA.....	68
DAP16 labels.....	68
NEW audits.....	69
Create academic audit CPA data.....	69
Batch generation of audit results using DAP22.....	70
Dynamic generation of audit results using DAP25.....	71
Planner audits in CPA.....	73
Results reporting.....	73
Reporting with other Degree Works data.....	74
Student Educational Planner.....	74
Plan data structures.....	75
SEP_PLAN.....	77
SEP_PLAN_NOTE.....	80
SEP_PLAN_TERM.....	81
SEP_PLAN_TERM_NOTE.....	83

SEP_PLAN_GROUP.....	84
SEP_PLAN_GROUP_NOTE.....	87
SEP_PLAN_GPA.....	88
SEP_PLAN_GPA_NOTE.....	90
SEP_PLAN_CLASS.....	91
SEP_PLAN_CLASS_NOTE.....	94
SEP_PLAN_TEST.....	95
SEP_PLAN_TEST_NOTE.....	97
SEP_PLAN_NONCOURSE.....	98
SEP_PLAN_NONCOURSE_NOTE..	99
SEP_PLAN_PLACEHOLDER.....	100
SEP_PLAN_PLACEHOLDER_NOTE	102
Plan entity relation diagrams.....	103
Template data structures.....	103
SEP_TMPL_MST.....	103
SEP_TMPL_NOTE.....	104
SEP_TMPL_TAG.....	105
SEP_TMPL_TERM.....	106
SEP_TMPL_TERM_NOTE.....	107
SEP_TMPL_GROUP.....	108
SEP_TMPL_GROUP_NOTE.....	110
SEP_TMPL_GPA.....	111
SEP_TMPL_GPA_NOTE.....	113
SEP_TMPL_CLASS.....	114
SEP_TMPL_CLASS_NOTE.....	116
SEP_TMPL_TEST.....	117
SEP_TMPL_TEST_NOTE.....	119
SEP_TMPL_NONCOURSE.....	120
SEP_TMPL_NONCOURSE_NOTE..	121
SEP_TMPL_PLACEHOLDER....	122
SEP_TMPL_PLACEHOLDER_NOTE	124
Template entity relation diagram.....	125
Requirements.....	125
DAP_REQ_BLOCK.....	125
DAP_REQ CRS_DTL.....	128

DAP_REQ_LINK_DTL.....	128
Exceptions.....	129
DAP_EXCEPT_DTL.....	130
Notes.....	133
DAP_NOTE_DTL.....	133
DAP_NOTE_TXT_DTL.....	135
Banner Integration.....	136
Banner attributes.....	136
Banner class attributes.....	136
Transfer classes applied to program.....	137
Current/history classes applied to program.....	137
Banner course attributes.....	139
Banner student attributes.....	139
Banner degree coding structure.....	140
Banner degree coding structure examples.....	140
Required access to Banner.....	142
Database table access.....	143
Function access.....	149
Satisfactory academic progress.....	150
SHRSAPP table layout.....	153
SHRSARJ table layout.....	157
Applicants in Degree Works.....	157
Banner applicant processing.....	158
Extract applicants.....	162
Applicant extract process configuration flags.....	163
Extract process.....	163
integration.banner.extract.config setting.....	165
Applicant user class.....	165
Financial aid data in Degree Works.....	166
Create Scribe custom data using Banner data.....	167
Scribing against test scores.....	168
Populate SHP_USER_ATTRIB from Banner data.....	169
Preferred name.....	170
Banner Data Mapping for BIF.....	170

Banner Bridge.....	171
R011PRIM - Primary Record.....	171
R027GOAL - Goal Record.....	173
R033GDTA - Goal Data Record.....	189
R062TERM - Term Record.....	212
R071CLAS - Class Record - CURRENT.....	214
R071CLAS - Class Record - HISTORIC.....	220
R083TRAN - Transfer Record.....	226
R085ATTR - Attribute Record.....	232
R091TEST - Test Record.....	233
R100PDEG - Previous Institution Record.....	234
R111NCRS - Non Course Record.....	235
R121CUST - Athletic Data - Custom Record....	236
R121CUST - Dynamic Retrieval - Custom Record	237
R121CUST - Student Attributes - Custom Record	238
R126REPT - Dynamic Retrieval - Report Record.	239
R171SHPU - SHP User Record.....	240
R190DEQV - DAP Equivalent Course Record...	242
R602CRSE - COURSE Record.....	250
R605CRSA - Course Attribute Record.....	253
R652MAPD - DAP Mapping Record.....	255
R655MAPA - DAP Map Attributes Record.....	261
R658MAPC - DAP Map Conditions Record....	261
R702ETSM - ETS Record.....	262
R800UCXT - UCX Record.....	264
R900CURR - Curriculum Rules Record.....	272
Class repeats/multiple occurrences.....	279
Repeated classes versus repeatable classes...	279
In-progress repeats identification.....	281
Repeats for renumbered courses identification..	284
Bridge Interface Format.....	285
Degree Works data storage.....	285
Degree Works components.....	285
Degree Works bridge specifications.....	288

Minimum data	289
Setup considerations	291
Transfer Equivalency setup considerations	292
Bridge program	293
Transfer transcripts	295
Repeat policy	296
Record layout	297
R011PRIM - Primary Record	298
R026APPL Applicant Record	300
R027GOAL - Goal Record	303
R033GDTA - Goal Data Record	305
R062TERM - Term Record	309
R071CLAS - Class Record	311
Classes falling into the Insufficient Section	317
R083TRAN - Transfer Class Record	318
R085ATTR - Class Attribute Record	325
R091TEST - Test Record	327
R100PDEG - Prev Inst Record	330
R111NCRS - Non Crse Record	332
R121CUST - Custom Record	334
R126REPT - Report Record	336
R128AID - Financial Aid Record	338
R130NOTE - DAP Note Record	339
R140NTXT - DAP Note Text Record	341
R171SHPU - SHP User Record	341
R180SWAP - Swap ID Record	344
R190DEQV - DAP Equivalent Course Record	345
R201TREQ - DAP Transfer Record	346
R500DELI Delete ID Record	350
R602CRSE - Course Record	351
R605CRSA - Course Attribute Record	354
R652MAPD - DAP Mapping Record	356
R658MAPC - DAP Map Conditions Record	360
R656MAPA - DAP Map Attributes Record	362
R702ETSM - ETS Record	363

R800UCXT UCX Tables.....	364
R900CURR - Curriculum Rules.....	367
Prerequisite Checking.....	369
Scribing for prerequisite checking.....	369
Elements of a REQUISITE block.....	370
Block sharing considerations.....	377
Block naming considerations.....	379
Scribing for the Requisite Check service.....	380
Scribing for the Requisite Description service.....	381
Student Educational Planner prerequisite checking.....	381
Banner prerequisite checking.....	382
Technical Configuration Reference.....	387
Shepherd settings.....	387
Spec value.....	387
Value types.....	388
Module settings.....	388
articulation.....	388
classicConnector.....	389
classic.daemons.....	390
core.....	391
core.amqp.....	393
core.articulation.....	395
core.audit.....	396
core.security.....	399
core.site.....	410
core.whatif.....	411
core.workflow.....	415
dash.....	416
email.....	425
integration.banner.....	426
integration.bridge.audits.....	431
localization.....	432
scribe.....	433

studentPlanner.....	434
studentPlanner.planner.....	435
studentPlanner.template.....	442
theme.....	444
transferFinder.....	444
transferFinder.central.....	446
transferFinder.service.....	447
transit.....	448
treq.applicant.....	450
treq.goal.....	450
treq.selfservice.....	451
treq.treqadmin.....	453
UCX tables.....	454
UCX-AUD012 User Class Codes.....	454
UCX-AUD013 User Class Permissions.....	455
UCX-AUD014 Exception Type Codes.....	456
UCX-AUD015 Exception Status Codes.....	457
UCX-AUD027 Major What-If Picklist.....	458
UCX-AUD029 Minor What-If Picklist.....	460
UCX-AUD031 Athletic Eligibility Types.....	461
UCX-AUD032 Audit Freeze Types.....	461
UCX-AUD033 Financial Aid Awards.....	462
UCX-AUD034 Audit Type.....	462
UCX-AUD047 Repeat Policies.....	463
UCX-BAN001 Schedule Type Codes.....	464
UCX-BAN002 Section Codes.....	465
UCX-BAN003 Gmod Codes.....	466
UCX-BAN080 Banner Custom Data Definitions.....	466
UCX-CFG020 BANNER.....	474
Load a current grade special edits.....	496
UCX-CFG020 BLOCKPRI.....	498
UCX-CFG020 COURSELINK.....	501
UCX-CFG020 DAP13.....	502
Process Cross-Listings.....	504
Process Equivalences.....	505

UCX-CFG020 DAP14.....	506
Check CFG020 TIEBREAK First.....	517
UCX-CFG020 DAP15ADVICE.....	518
UCX-CFG020 DAP15LABEL.....	518
UCX-CFG020 DAP15LIST.....	519
UCX-CFG020 EXCEPTIONS.....	519
UCX-CFG020 RADBRIDGE.....	520
UCX-CFG020 REFRESH.....	523
UCX-CFG020 RESULTS.....	524
UCX-CFG020 TIEBREAK.....	526
UCX-CFG020 TRANSFER.....	529
UCX-CFG020 TREQ.....	530
UCX-CFG020 WEB.....	534
UCX-CFG020 WEBPARAMS.....	537
UCX-CFG068 Course Sequencing.....	539
UCX-CFG070 Course Equivalence Records.....	541
UCX-CFG071 Note Text.....	542
UCX-CFG072 Planned Courses.....	543
UCX-CFG073 Cross Listed Courses.....	544
UCX-CFG074 Reused Courses.....	546
UCX-CFG075 Repeatables.....	546
UCX-CFG078 Split Courses.....	547
UCX-CST001 – CST010 Custom Tables.....	549
UCX-RPT036 Audit Report Formats.....	550
UCX-RPT046 Custom Report Data.....	555
UCX-RPT050 Course Link Title Settings.....	557
UCX-RPT052 Course Link Attribute Settings.....	558
UCX-RPT054 Course Link Sections Settings.....	559
UCX-RPT056 Course Link Transfer Settings.....	560
UCX-SCR001 Reserved Words.....	561
UCX-SCR002 Custom Data.....	563
UCX-SCR003 Noncourse Codes.....	565
UCX-SCR004 Requirement Block Type Codes.....	568
UCX-SCR007 Block Parsed Status.....	569
UCX-SCR008 Note Type Codes.....	569

UCX-SCR044 WITH Custom Class Data.....	570
UCX-SCR045 DECIDE Options.....	572
UCX-SEP001 Template Tags.....	574
UCX-SEP002 Template Term Schemes.....	575
UCX-SEP003 Requirement Delivery Codes.....	577
UCX-SEP004 Requirement Types.....	577
UCX-SEP005 Placeholder Requirement Types.....	578
UCX-SEP006 Test Codes.....	579
UCX-SEP007 Test Score Sort Order.....	579
UCX-SEP008 GPA Requirement Types.....	580
UCX-SEP009 Choice Scribe Pointers.....	581
UCX-SEP011 Plan Scope Values.....	582
UCX-SEP060 - UCX-SEP069 User-Defined Template Tags..	582
UCX-SHP078 SHP Keys.....	583
UCX-SHP080 Locale Codes.....	584
UCX-STU016 DegreeWorks Term Descriptions.....	584
UCX-STU023 Student Major Codes.....	586
UCX-STU024 Student Minor Codes.....	588
UCX-STU035 Degree Works Catalog Year.....	589
UCX-STU050 Course Attributes.....	590
UCX-STU305 Student Level Codes.....	590
UCX-STU306 Student Status Codes.....	591
UCX-STU307 Degree Codes.....	592
UCX-STU314 Test Codes.....	594
UCX-STU316 Program Codes.....	595
UCX-STU323 Specialization Codes.....	596
UCX-STU324 Liberal Learning Codes.....	597
UCX-STU346 Transfer Calendar.....	598
UCX-STU350 School Code.....	599
UCX-STU352 Discipline Codes.....	600
UCX-STU355 Credit Type Codes.....	601
UCX-STU356 Grade Type Codes.....	602
UCX-STU385 Grade Table.....	603
UCX-STU398 Transfer Grade Mapping.....	607
UCX-STU560 College Codes.....	608

UCX-STU563 Concentration Codes.	609
UCX-STU576 Campus or Location Codes.	611
UCX-SYS001 Table of Contents.	612
UCX-SYS098 Error or Status Messages.	613
UCX-SYS100 PDF Audit Page Dimensions.	614
UCX-SYS933 Data Names.	615
UCX-SYS935 Web Server Function Buffers.	617
UCX-SYS998 System Standard Error Messages.	618
UCX-SYS999 Data Dictionary.	619
UCX-TRF100 Term Global Mapping Codes.	621
UCX-TRQ060 TreQ Favorite Schools.	621
UCX-TRQ061 Transfer Mapping Conditions.	622
UCX-TRQ062 Applicant TreQ Status.	624
UCX-TRQ063 Transfer Articulation Status.	625
UCX-TRQ065 Transfer Articulation Codes.	625
 Technical Considerations.	627
Parser Engine (DAP13).	627
Auditor Engine (DAP14).	629
GPA calculations.	631
Redemption algorithm.	632
Fall-through redemption.	632
Nonexclusive redemption.	633
Too many classes on a rule.	633
Match level.	634
Fit rank.	635
Group processing.	636
Remove classes when too many fit on a rule.	637
Class evaluation on a rule.	638
Percent complete calculation.	641
Rule completeness.	641
Block completeness.	642
Overall audit completeness.	643
Output options.	643
Output Engine (DAP15).	643

Custom data items.	644
NonCourse data items.	645
Advisee filtering.	645
Degree Works bridge.	648
Degree Works static bridge.	648
Degree Works dynamic refresh.	649
Equivalent course tracking.	652
Set up equivalent course tracking.	654
CFG070 Equivalence course records.	656
Process equivalences into scribed courses.	656
Process cross-listings into scribed courses.	659
Freeze audits.	660
Multi-entity processing in Degree Works.	662
Multiple-campus case study.	663
Multi-entity processing database tables.	666
Repeated classes.	668
Split credits.	670
Syntax for split credits.	670
Auditor Engine processing of split credits.	671
Class count.	671
Exclusivity.	672
Multiple splits.	673
Output.	673
Split power.	673
Other qualifiers.	674
Best fit.	675
Fall through.	675
Over-the-limit.	676
Fall-through split.	676
Multiple split qualifiers.	676
Course Link configuration.	677
Degree Works accessibility compliance.	678
Western character support.	678
Database tables.	680
DAP tables.	681

RAD tables.....	682
SHP tables.....	684
SEP tables.....	684
Transit tables.....	685
Degree Works special scripts.....	686
changepassword script.....	693
dap22dbg.....	693
dapauditstopdffiles script.....	693
dapauditstoxmlfiles script.....	694
dapauditstoxml script.....	695
dapblockinsert script.....	696
dapfindbadaudits script.....	697
dapfindorphanedaudits script.....	698
dbbuild script.....	699
debugon and debugoff.....	700
dgwversion.....	700
dwsettings script.....	701
getxmlaudit script.....	702
packdebug script.....	703
preqstats.....	703
profiledbg script.....	705
sharegen script.....	706
shareinfo script.....	708
webanalyze.....	708
webstats.....	709
webtime.....	711
Degree Works security.....	711
HTTPS/SSL security.....	712
Authentication security.....	712
Multiple paths to authentication.....	713
Degree Works native login.....	714
LDAP user database.....	715
CAS single sign-on.....	716
SAML single sign-on and sign-off....	718
External access manager.....	720

Multiple authentication entry points.....	721
Persistent authentication for a session.....	722
API Authentication.....	722
Stateless token creation.....	723
Token utilization.....	724
User creation in User API for Transfer Equivalency Self-Service.....	725
Authentication debugging in API Services.....	726
Access control (authorization).....	726
Key assignment with SHPCFG.....	727
Keys and keyrings.....	727
SHPCFG maintenance.....	727
Services.....	731
Groups.....	739
Users.....	741
Database privileges.....	742
Encrypted data.....	742
Degree Works applications.....	743
Deployment considerations.....	743
Degree Works intercommunication.....	744
Degree Works external communication requirements.....	746
Request-response.....	746
Degree Works and your student data.....	747
Degree Works maintenance.....	748
Applications restart.....	748
Cron setup for Degree Works.....	749
After system failure.....	750
Classic server operating system change.....	751
Daily maintenance tasks.....	751
Monthly maintenance tasks.....	752
Semi-yearly or yearly maintenance tasks.....	752
Code patches between releases.....	752
Update internal Tomcat libraries in standalone Java applications	753
As-needed tasks.....	754
Degree Works updates processing.....	754

Degree Works user addition.	754
Blocks transfer between two different environments	754
Mappings transfer between two different environments	757
UCX tables transfer between two different environments	759
Old student data removal from your Degree Works database	
.	760
Database cloning from test to production.	761
Email notification configuration maintenance. . .	761
Database credentials changes.	762
Degree Works standing daemons.	763
webshow.	764
webstart.	765
webrestart.	765
webstop.	765
dapshow.	765
dapstart.	766
daprestart.	766
dapstop.	766
radshow.	766
radstart.	767
radrestart.	767
radstop.	767
resstart.	768
resrestart.	768
resstop.	768
tbestart.	768
tberestart.	768
tbestop.	769
preqstart.	769
preqrestart.	769
preqstop.	769
Load balancing and proxies.	769
Classic load balancing.	770
Java application load balancing.	770
System performance.	771

System management and performance.....	771
Database server configuration.....	772
Java database pooling configuration.....	772
Degree Works web server daemon configuration.....	773
Batch processing performance.....	775
Custom indexes.....	775
 Transit Jobs.....	776
ADMIN Administrative tasks.....	776
AUD01 - List unhooked and unenforced exceptions.....	778
AUD02 - Delete audits by freeze type and date.....	779
BAN62 - Satisfactory Academic Progress Processor.....	779
DAP16 - Parse Requirements Processor.....	780
DAP21 - Extract Articulated Transfers.....	781
Email message capability.....	782
DAP21 errors, warnings, and success.....	783
Warnings.....	783
Fatal errors.....	783
Success.....	783
Sample output XML data file.....	786
DAP22 - Generate Audits.....	806
Report outputs.....	807
dwfop.....	808
DAP27 - What-if Audits.....	809
DAP28 - Alternate What-if Audits.....	811
DAP40 Unload Scribe Blocks.....	813
DAP40 errors, warnings, and success.....	814
DAP41 Load Scribe Blocks.....	814
DAP41 errors, warnings, and success.....	815
DAP42 Unload Mappings.....	815
DAP43 Load Mappings.....	816
DAP44 Unload UCX tables.....	816
DAP45 Load UCX tables.....	817
DAP54 - Template processor plan creation.....	817
Degree template search.....	819

DAP58 - Batch tracking processor.	822
DAP59 - Batch timetabling processor.	823
RAD11 - The Traditional Bridge Batch Processor.	823
Data files.	824
FTP process.	824
Run RAD11 processor.	825
RAD3x - Banner Extract and Bridge.	829
RAD30 - Banner Student Extract and Bridge.	829
RAD32 - Banner Applicant Extract and Bridge.	830
RAD33 - Banner Non-Student Extract and Bridge.	830
RAD34 - Banner Course Extract.	831
RAD35 - Banner Curriculum Rules Extract.	831
RAD36 - Banner Validation Table Extract (UCX).	831
RAD37 - Banner Transfer School Extract (ETS).	832
RAD38 - Banner Equivalencies Extract.	832
RAD39 - Banner Transfer Equivalency Extract (Mappings).	832
RAD40 - Student Data Delete Processor.	832
SCR02 - Find blocks where this COURSE is referenced.	833
SCR05 - List blocks changed by date range.	833
SCR06 - List block primary and secondary tags.	834
SCR07 - List block text from Scribe.	834
SCR08 - List LOG entries from Scribe text.	835
SCR09 - List TODO entries from Scribe text.	835
SCR10 - Find blocks where this text is referenced.	835
SCR11 - Find and Replace block text.	836
SCR91 - Test Banner Prerequisite Checker Service.	836
SCR92 - Test Banner Prerequisite Description Service.	837
SCR93 - Report by CATALOG.	838
SCR94 - Report by SCHEDULE.	838
SCR95 - Report by REQUISITE Block.	838
UCX01 - UCX Records Modified.	839
Schema.	840

Advanced Reporting

Updated: September 30, 2022

Degree Works can provide detailed guidance to the Registrar and to an institution's research staff through reports the institution writes that access Degree Works database tables.

There are multiple data storehouses in Degree Works. The Curriculum Planning Assistant (CPA) contains registration history, student advice, and Student Educational Planner (SEP) data from academic and planner audits. Degree Works stores the actual audit in a proprietary format that is not easily accessed by standard tools, but the CPA unloads the audit data into a format accessible with SQL reporting tools.

Scribe requirement blocks, exceptions, notes, plan, and template data are also available for reporting.

Data generation with the Curriculum Planning Assistant

Updated: September 30, 2022

Academic and planner audit data for reporting can be loaded into the Curriculum Planning Assistant (CPA) tables by DAP22 and DAP59, respectively.

CPA database tables

Updated: September 30, 2023

Curriculum Planning Assistant results data are stored in the dap_result_dtl, dap_resnoncr_dtl and dap_resclass_dtl database tables.

DAP_RESULT_DTL

Updated: March 25, 2022

This table contains the results of an audit.

Each set of audit records is grouped by the first five fields as shown below.

Name	Null?	Type
DAP_STU_ID	NOT NULL	CHAR(10)
DAP_AUDIT_TYPE		CHAR(2)
DAP SCHOOL		CHAR(12)
DAP_DEGREE		CHAR(12)
DAP_ACTIVE_TERM		CHAR(8)
DAP_BLOCK_SEQ_NUM		NUMBER(2)
DAP_RESULT_SEQ_NUM		NUMBER(6)
DAP_REQ_ID		CHAR(8)
DAP_RULE_ID		CHAR(20)
DAP_NODE_TYPE		NUMBER(4)
DAP_RESULT_TYPE		CHAR(20)
DAP_VALUE1		CHAR(12)
DAP_VALUE2		CHAR(12)
DAP_VALUE3		CHAR(12)
DAP_VALUE4		CHAR(12)
DAP_NUMBER1		NUMBER(8,3)
DAP_NUMBER2		NUMBER(8,3)
DAP_NUMBER3		NUMBER(8,3)
DAP_NUMBER4		NUMBER(8,3)
DAP_FREETEXT		CHAR(80)
DAP_CREATE_DATE		DATE
UNIQUE_ID	NOT NULL	NUMBER(10)
UNIQUE_KEY		NOT NULL CHAR(36)

DAP_RESNONCR_DTL

Updated: March 25, 2022

The records here represent the noncourses a student has completed that were used in the degree audit.

As with the dap_result_dtl, each set of records is grouped by the first five fields as shown below.

Name	Null?	Type
DAP_STU_ID	NOT NULL	CHAR(10)
DAP_AUDIT_TYPE		CHAR(2)
DAP SCHOOL		CHAR(12)
DAP_DEGREE		CHAR(12)
DAP_ACTIVE_TERM		CHAR(8)
DAP_CLASS_ID		CHAR(6)
DAP_NONCRS_TYPE		CHAR(12)
DAP_NONCRS_VALUE		CHAR(12)
DAP_COURSE_TITLE		CHAR(50)
DAP_TERM		CHAR(8)
DAP_CREATE_DATE		DATE
UNIQUE_ID	NOT NULL	NUMBER(10)
UNIQUE_KEY		NOT NULL CHAR(36)

DAP_RESCLASS_DTL

Updated: September 30, 2022

The records here represent the classes a student has taken (or will take) that were used in the degree audit.

As with the dap_result_dtl, each set of records is grouped by the first five fields as shown below.

Note: The dap_fallthrough field below does not signify that the class ended up in the fall-through section. This flag records that the class was bridged with the force-fallthrough flag set on the rad_class_dtl record when it was bridged. In the same manner, the dap_passfail, dap_incomplete, and dap_insufficient fields also record how the class was bridged to Degree Works.

Name	Null?	Type
DAP_STU_ID	NOT NULL	CHAR(10)
DAP_AUDIT_TYPE		CHAR(2)
DAP SCHOOL		CHAR(12)
DAP_DEGREE		CHAR(12)
DAP_ACTIVE_TERM		CHAR(8)
DAP_CLASS_ID		CHAR(6)
DAP_DISCIPLINE		CHAR(12)
DAP_COURSE_NUM		CHAR(12)
DAP_COURSE_TITLE		CHAR(50)
DAP_TERM		CHAR(8)
DAP_TRANSFER		CHAR(1)
DAP_REPEAT_PLCY1		CHAR(1)
DAP_CLASS_STATUS		CHAR(2)
DAP_REPEAT_DISC		CHAR(12)
DAP_REPEAT_NUM		CHAR(12)
DAP_LOCATION		CHAR(12)
DAP_GRADE_TYPE		CHAR(6)
DAP_TRANSFER_TYPE		CHAR(4)
DAP_GRADE		CHAR(6)
DAP_GRADE_NUM		NUMBER(7,3)
DAP_GRADE_POINTS		NUMBER(7,3)
DAP_SS_CREDITS		NUMBER(7,3)
DAP_AUD_CREDITS		NUMBER(7,3)
DAP_GPA_CREDITS		NUMBER(7,3)
DAP_GPA_GRD PTS		NUMBER(7,3)
DAP_PASSFAIL		CHAR(1)
DAP_INCOMPLETE		CHAR(1)
DAP_PASS		CHAR(1)
DAP_FALLTHROUGH		CHAR(1)
DAP_INPROGRESS		CHAR(1)
DAP_GRADEERROR		CHAR(1)
DAP_INSUFFICIENT		CHAR(1)
DAP_INSUFFMAJOR		CHAR(1)
DAP_REASON_INSUF		CHAR(2)
DAP_ERROR_NUMBER		CHAR(4)
DAP_TR_COURSE		CHAR(19)
DAP_TR_SCHL_NAME		CHAR(30)
DAP_WITH_DATA		CHAR(30)
DAP_TR_TITLE		CHAR(50)
DAP_CRN		CHAR(10)
DAP_CREATE_DATE		DATE
UNIQUE_ID	NOT NULL	NUMBER(10)
UNIQUE_KEY	NOT NULL	CHAR(36)

Database views

Updated: March 25, 2022

Creating database views can help in reporting off the data contained in tables.

Sample views defined by Ellucian can be used for reporting with your SQL reporting tool and to help you create your own views. You must define these views in the Oracle database.

To define the necessary views in your Oracle database, you must be an Oracle user that has permission to create views. It is recommended that your DBA create the views. The sample views can be found in \$DGHOME/sql/cpa_views.sql. You can run that file through sqlplus to create the views if they do not already exist.

Note: The views are set up to look at only one type of audit. The dap_audit_type field is used to find only the Academic Audit records or the Financial Aid Audit records.

Database space needed

Updated: September 30, 2022

You must consider the implications of storing CPA data for reporting purposes.

The information below is intended as a guide to help you determine the database size needed based on your student population.

Scribe data

If you have 200 blocks using an average of 8 labels per block, you will get 1,600 additional result-dtl records. This is an extra 416K bytes, but is a fixed number that will not change as your student population changes.

Dap-result-dtl records are also built for each of the labels in your blocks.

Dap-result-dtl - 262 bytes

Approximately 500 records per student – depends on student level, requirements, and UCX-CFG020 RESULTS flags.

For one student: 500 records x 262 bytes = 131,000 bytes

Dap-resclass-dtl – 184 bytes

Freshman may have fewer than 20 each, while seniors may have 30-50 classes each.

For one student: 25 records x 184 bytes = 4,600

Dap-resnoncr-dtl – 110 bytes

The number of UCX-SCR003 data items defined determines how many of these will be built per student.

For one student: 5 records x 110 bytes = 550 bytes

Approximate Total: 131K + 4.6K + .5K = 136K per student

UCX-CFG020 RESULTS configuration

Updated: September 29, 2023

The CFG020 RESULTS record should be used to control the kind of records that are created for each degree audit report.

By setting a flag to N in this CFG020 record, you are telling Degree Works to not build results records for that kind of information. Because of this, you will be limited in the types of questions you can ask of the data. Turning off the remarks, for example, is probably acceptable, because you probably do not ever want to query the remarks. Turning off course rule advice, on the other hand, would probably be problematic, because you will not know what classes are needed by your student body.

Note: There is not a CFG020 RESULTS flag corresponding to every record type that appears in DAP_RESULT_DTL. If a record type does not appear here (AUDITID, for example), that means it is always created and there is no flag to turn it on or off.

CFG020 RESULTS flag	Which result-types it controls
Build CATALOGYEAR and GOALDATA records	CATALOGYEAR, GOALDATA
Build the Block Header Credits/Classes needed	BLOCKCRCLNEEDED
Build the Block Header Credits/Classes Applied	BLOCKCRCLAPPLIED
Build the Block Header Advice	HEADERADVICE
Build the Block Header Qualifier Requirements	HEADERQUAL
Build the list of classes applied to rules	CLASSAPPLIED, NONCRAPPLIED, FALLCLASSAPPLIED, FALLNONCRAPPLIED, OTLCLASSAPPLIED, INSUFFCLASSAPPLIED
Build Remarks for Headers and for Rules	REMARK
Build Fallthrough (electives) Section	FALLCRCL, FALLCLASSAPPLIED, FALLNONCRAPPLIED
Build Over-the-limit (not used) Section	OTLCRCL, OTLCLASSAPPLIED
Build Insufficient (failed) Section	INSUFFCRCL, INSUFFCLASSAPPLIED
Build the Audit Errors Section	AUDITERROR
Build Rule ProxyAdvice	RULEADVICE
Build the Block label and percent	LABEL
Build the Block advice	RULEADVICE, QUALADVICE
Build the Blocktype label and percent	LABEL
Build the Blocktype advice	RULEADVICE, QUALADVICE
	LABEL
Build the Group label and percent	LABEL
Build the Group advice	RULEADVICE, QUALADVICE

CFG020 RESULTS flag	Which result-types it controls
Build the Subset label and percent	LABEL
Build the Subset advice	RULEADVICE, QUALADVICE
Build the Noncourse label and percent	LABEL
Build the Noncourse advice	NONCRNUMNEED, NONCRNEEDED
Build the Course rule details	LABEL
Build the Course rule advice	RULEADVICE, QUALADVICE, CRCLNEEDED, CLASSNEEDED
Build the Course rule Classes/Credits required	CRCLREQUIRED

Result types

Updated: September 30, 2022

The result-type field on each dap-result-dtl specifies the type of data each record contains.

Result type	Description
AIDCUMCREDITS	Financial Aid: Cum. Total credits earned (res and transfer), cum. credits earned (res)
AIDGRADE	Financial Aid: Cum. graded credits attempted, cum. Grade points, and cum. GPA
AIDTERM	Financial Aid: Last completed term type, active term and previous term
AIDTERMCREDITS	Financial Aid: Credits attempted this term and completed term count
AUDITERROR	Error from audit
AUDITID	Audit ID used to create the results records.
BLOCKCRCLAPPLIED	Number of classes and credits applied to this block
BLOCKCRCLNEEDED	Number of classes and credits still needed on this block
BLOCKGPA	GPA, percent-complete and GPA-grade-points applied to this block
CATALOGYEAR	Student's catalog year
CLASSAPPLIED	Class applied to this rule
CLASSNEEDED	Class needed on rule
CLASSREQUIRED	Class listed on the original requirement; only available for a timetabling audit
CRCLNEEDED	Number of classes and credits still needed on rule

Result type	Description
CRCLREQUIRED	Number of classes and credits originally required on rule
EXCEPTION	Exception applied to audit
FALLCLASSAPPLIED	Fall-through/Electives class applied
FALLCRCL	Fall-through/Electives number of classes and credits applied
FALLNONCRAPPLIED	Fall-through/Electives noncourse applied
GOALDATA	Student's list of majors/minors, concentrations, advisors, etc.
HEADERADVICE	Block header qualifier advice
HEADERQUAL	Block header qualifier text
INSUFFCLASSAPPLIED	Insufficient section class applied
INSUFFCRCL	Insufficient section number of classes and credits applied
LABEL	Label for this rule
NONCRAPPLIED	NonCourses applied
NONCRNEEDED	NonCourses needed
NONCRNUMNEED	Number of NonCourses needed
OTLCLASSAPPLIED	Over-The-Limit class applied
OTLCRCL	Over-The-Limit number of classes and credits applied
PLAN	Created for planner/timetabling audits only.
QUALADVICE	Rule qualifier advice
REMARK	Block header and rule remarks
RULEADVICE	Rule advice – list classes still needed

For each student, the following fields in the dap_result_dtl contain the same data regardless of the result type.

- STU-ID – must be valid in the dap_student_mst
- AUDIT-TYPE – is AA (Academic), FA (Financial Aid), AE (Athletic), or PL (Plan)
- SCHOOL – school against which the audit was run
- DEGREE – degree against which the audit was run
- CREATE-DATE – date the result record was created
- ACTIVE-TERM – the student's active-term when the data was created

AIDCUMCREDITS

Updated: March 25, 2022

This result type shows financial aid cumulative credits.

Field	Description	Example
Block-Seq-Num	Always 95	95
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	AID	AID
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Cumulative total credits earned (residence and transfer)	37
Value2	Cumulative credits earned (residence only)	28
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	Cumulative total credits earned (residence and transfer)	37
Number2	Cumulative credits earned (residence only)	28
Number3	(not used)	0
Number4	(not used)	0
Freetext	(not used)	(blank)

AIDGRADE

Updated: March 25, 2022

This result type shows financial aid GPA information.

Field	Description	Example
Block-Seq-Num	Always 95	95
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	AID	AID
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Cumulative graded credit attempted	37
Value2	Cumulative grade points	123
Value3	Cumulative GPA	3.07
Value4	(not used)	(blank)

Field	Description	Example
Number1	Cumulative graded credit attempted	37
Number2	Cumulative grade points	123
Number3	Cumulative GPA	3.07
Number4	(not used)	0
Freetext	(not used)	(blank)

AIDTERM

Updated: March 25, 2022

This result type shows financial aid term information.

Field	Description	Example
Block-Seq-Num	Always 95	95
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	AID	AID
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Last completed term type	SPRING
Value2	Active Term	200910
Value3	Previous Term	200830
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	(not used)	(blank)

AIDTERMCREDITS

Updated: March 25, 2022

This result type shows financial aid term credits.

Field	Description	Example
Block-Seq-Num	Always 95	95
Result-Seq-Num	Used to sort result records for this student	167

Field	Description	Example
Req-ID	AID	AID
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Credits attempted this term	12
Value2	Completed term count	3
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	Credits attempted this term	12
Number2	Completed term count	3
Number3	(not used)	0
Number4	(not used)	0
Freetext	(not used)	(blank)

AUDITERROR

Updated: March 25, 2022

This result type shows errors from audits.

Field	Description	Example
Block-Seq-Num	Always 93	93
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	AUDERROR	AUDERROR
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Error from UCX-SYS998	3712
Value2	(not used)	(blank)
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Error nnnn occurred in audit	Error 3712 occurred in audit

AUDITID

Updated: March 25, 2022

This result type shows audit ID, date, and time of audit.

Field	Description	Example
Block-Seq-Num	(not used)	0
Result-Seq-Num	Used to sort result records for this student	1
Req-ID	Always set to STUINFO	STUINFO
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Audit-ID from dap-audit-dtl	AA000123
Value2	Date when the audit was run	20050828
Value3	Time when the audit was run	1456
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	(not used)	(blank)

BLOCKCRCLAPPLIED

Updated: March 25, 2022

This result type shows number of classes and credits applied to a block.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Credits applied	87
Value2	Classes applied	20
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	Credits applied	87
Number2	Classes applied	20

Field	Description	Example
Number3	(not used)	0
Number4	(not used)	0
Freetext	(not used)	(blank)

BLOCKCRCLNEEDED

Updated: March 25, 2022

This result type shows number of classes and credits still needed on a block.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Credits needed	43
Value2	Classes needed	15
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	Credits needed	43
Number2	Classes needed	15
Number3	(not used)	0
Number4	(not used)	0
Freetext	(not used)	(blank)

BLOCKGPA

Updated: March 25, 2022

This result type shows percent complete, GPA and GPA-grade-points applied to a block.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67

Field	Description	Example
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Block percent complete	76.392
Value2	Block GPA	3.459
Value3	Block grade points used in GPA calculation	278.5
Value4	Block type	MAJOR
Number1	Block percent complete	76.392
Number2	Block GPA	3.459
Number3	Block grade points used in GPA calculation	278.5
Number4	(not used)	0
Freetext	(not used)	(blank)

CATALOGYEAR

Updated: March 25, 2022

This result type shows a student's overall catalog year.

Field	Description	Example
Block-Seq-Num	(not used)	0
Result-Seq-Num	Used to sort result records for this student	2
Req-ID	Always set to STUINFO	STUINFO
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Student's overall catalog year	20052006
Value2	(not used)	(blank)
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	(not used)	(blank)

CLASSAPPLIED

Updated: March 25, 2022

This result type shows the class applied to a rule.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Always 4200 for course rules	4200
Value1	Discipline	ACCT
Value2	Number	112
Value3	dap-class-id on the dap-resclass-dtl	0031
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Discipline and number	ACCT 112

CLASSNEEDED

Updated: March 25, 2022

This result type shows the class needed on a rule.

Note: When the WITH advice in Value3 contains an attribute, the word ATTRIBUTE is shortened to ATTR so the value can fit into the database field. For example, if ATTRIBUTE = RORY is found in the WITH advice, it is shortened to ATTR = RORY. In addition, when the WITH advice is longer than 12 characters, only the first 12 characters appear in this field.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67

Field	Description	Example
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Always 4200 for course rules	4200
Value1	Discipline	ACCT
Value2	Number	112
Value3	WITH advice	ATTR = RORY
Value4	MUSTTAKE or MAYTAKE or CANTTAKE (see special topic on these tags)	MUSTTAKE
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Discipline and number and WITH advice	ACCT 112 Special Grade

CLASSREQUIRED

Updated: March 25, 2022

This result type shows a class from the original requirement. It is available only for a timetabling audit.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Always 4200 for course rules	4200
Value1	Discipline	ACCT
Value2	Number	112
Value3	WITH advice	Special grade
Value4	(not used)	(blank)

Field	Description	Example
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Discipline and number and WITH advice	ACCT 112 Special Grade

CRCLNEEDED

Updated: March 25, 2022

This result type shows the number of classes and credits still needed on a rule.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Always 4200 for course rules	4200
Value1	Credits still needed	10
Value2	Classes still needed	3
Value3	Operator – AND or OR	AND
Value4	Course connector - comma or plus	,
Number1	Credits still needed	10
Number2	Classes still needed	3
Number3	(not used)	0
Number4	(not used)	0
Freetext	Credits and/or Classes	10 Credits and 3 Classes

CRCLREQUIRED

Updated: March 25, 2022

This result type shows the number of classes and credits originally required on a rule.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Always 4200 for course rules	4200
Value1	Credits-begin:Credits-end	10:15
Value2	Classes-begin:Classes-end	3:4
Value3	Operator – AND or OR	AND
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Credits:Credits and/or Classes/Classes	5 Credits and 2 Classes

EXCEPTION

Updated: March 25, 2022

This result type shows an exception applied to the audit.

You can link to the dap_except_dtl to get more details about the exception by linking by the dap_stu_id field on both tables and from the dap_number1 field on this table to the dap_exc_num field on the dap_except_dtl. The dap_exc_buffer field has the course information at specific locations in the buffer.

See the *Exceptions* topic for details on the contents of the dap_except_dtl.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Depends on type of rule	4200

Field	Description	Example
Value1	Exception ID number (dap_except_dtl.dap_exc_num)	8
Value2	Exception type – see UCX-AUD014	AH
Value3	Audit tree node id where exception was applied (dap_except_dtl.dap_node_id)	178
Value4	ENFORCED or NOTENFORCED	ENFORCED
Number1	Exception ID number (dap_except_dtl.dap_exc_num)	8
Number2	(not used)	0
Number3	Audit tree node id where exception was applied (dap_except_dtl.dap_node_id)	178
Number4	(not used)	0
Freetext	Reason why exception was not applied or text saying it was applied	Exception was applied successfully

FALLCLASSAPPLIED

Updated: March 25, 2022

This result type shows the fall-through/electives class applied.

Field	Description	Example
Block-Seq-Num	Always 90	90
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	FALLTHRU	FALLTHRU
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Discipline	BIOL
Value2	Number	107
Value3	dap-class-id on the dap-resclass-dtl	24
Value4	ECA-OK or ECA-OVERFLOW (if CheckElectiveCreditsAllowed is used)	ECA-OVERFLOW
Number1	(not used)	0

Field	Description	Example
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Discipline Number	BIOL 107

FALLCRCL

Updated: September 30, 2022

This result type shows the fall-through/electives number of classes and credits applied.

Field	Description	Example
Block-Seq-Num	Always 90	90
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	FALLTHRU	FALLTHRU
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Credits applied	23
Value2	Classes applied	6
Value3	Noncourses applied	2
Value4	Overflow credits (if CheckElectiveCreditsAllowed is used)	9
Number1	Credits applied	23
Number2	Classes applied	6
Number3	Noncourses applied	2
Number4	Overflow credits (if CheckElectiveCreditsAllowed is used)	9
Freetext	Classes and credits applied	23 Credits and 6 Classes

FALLNONCRAAPPLIED

Updated: September 30, 2022

This result type shows fall-through/electives noncourse applied.

Field	Description	Example
Block-Seq-Num	Always 90	90

Field	Description	Example
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	FALLTHRU	FALLTHRU
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Code	THESIS
Value2	Value	NOTDONE
Value3	dap-class-id on the dap-resnoncr-dtl	24
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Code Value	THESIS NOTDONE

GOALDATA

Updated: March 25, 2022

This result type shows student's majors, minors, concentrations, advisors, and other codes.

Field	Description	Example
Block-Seq-Num	(not used)	0
Result-Seq-Num	Used to sort result records for this student	4
Req-ID	Always set to STUINFO	STUINFO
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Goal code: MAJOR, MINOR, CONC, ADVISOR, PROGRAM, COLLEGE, STUSTATUS, LIBL, SPEC	MATH
Value2	Value of goal	ENGL
Value3	Catalog year associated with this goal	20122013
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0

Field	Description	Example
Number4	(not used)	0
Freetext	(not used)	(blank)

HEADERADVICE

Updated: March 25, 2022

This result type shows block header qualifier advice.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	(not used)	(blank)
Value2	(not used)	(blank)
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Proxy-Advice or real qualifier advice	Lastres not satisfied

HEADERQUAL

Updated: March 25, 2022

This result type shows block header qualifier text, tied to student audits.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123

Field	Description	Example
Rule-ID	0 - plus the qualifier's node-id	0-23
Node-Type	The qualifier type number	4101
Value1	Qualifier name	MAXCLASSES
Value2	Classes or credits required	15
Value3	CREDITS or CLASSES or blank	CLASSES
Value4	May have a value or may be blank – use Number4 below	(blank)
Number1	(not used – will always be 0)	0
Number2	(see table below)	15
Number3	(not used – will always be 0)	0
Number4	(see table below)	3
Freetext	List the classes associated with the qualifier	PE @

For ResultType records of HEADERQUAL, the Value1 field will contain the type of qualifier and will contain these values in the Value and Number fields.

Value1	CLASSES
Value2	Minimum number of classes that must be taken
Value3	CLASSES
Value4	Number of classes taken
Number1	0
Number2	Minimum number of classes that must be taken
Number3	0
Number4	Number of classes taken
Value1	CREDITS
Value2	Minimum number of credits that must be taken
Value3	CREDITS
Value4	Number of credits taken
Number1	0
Number2	Minimum number of credits that must be taken
Number3	0
Number4	Number of credits taken
Value1	HIGHPRIORITY
Value2	(blank)

Value3	(blank)
Value4	(blank)
Number1	0
Number2	0
Number3	0
Number4	0
Value1	LASTRES
Value2	Number of classes or credits required in residence
Value3	CREDITS or CLASSES
Value4	Number of in residence classes/credits applied – as of the last transfer class
Number1	0
Number2	Number of classes or credits required in residence
Number3	0
Number4	Number of in residence classes/credits applied – as of the last transfer class
Value1	LOWPRIORITY
Value2	(blank)
Value3	(blank)
Value4	(blank)
Number1	0
Number2	0
Number3	0
Number4	0
Value1	MAXCLASSES
Value2	Maximum number of classes that can be taken from a specific list of courses
Value3	CLASSES
Value4	Number of classes applied
Number1	0
Number2	Maximum number of classes that can be taken from a specific list of courses
Number3	0
Number4	Number of classes applied

Value1	MAXCREDITS
Value2	Maximum number of credits that can be taken from a specific list of courses
Value3	CREDITS
Value4	Number of credits applied
Number1	0
Number2	Maximum number of credits that can be taken from a specific list of courses
Number3	0
Number4	Number of credits applied
Value1	MAXPASSFAIL
Value2	Maximum number of classes/credits that can be taken passfail
Value3	CREDITS or CLASSES
Value4	Number of passfail classes/credits applied
Number1	0
Number2	Maximum number of classes/credits that can be taken passfail
Number3	0
Number4	Number of passfail classes/credits applied
VALUE1	MAXPERDISC
Value2	Maximum number of classes/credits that can be taken from this list of disciplines
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied
Number1	0
Number2	Maximum number of classes/credits that can be taken from this list of disciplines
Number3	0
Number4	Number of classes/credits applied
Value1	MAXTERM
Value2	Maximum number of classes/credits that can be taken per term in the specified list of courses
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied to the last term processed - mostly meaningless data

Number1	0
Number2	Maximum number of classes/credits that can be taken per term in the specified list of courses
Number3	0
Number4	Number of classes/credits applied to the last term processed - mostly meaningless data
Value1	MAXTRANSFER
Value2	Maximum number of classes/credits that can be taken from a transfer institution
Value3	CREDITS or CLASSES
Value4	Number of transfer classes/credits applied
Number1	0
Number2	Maximum number of classes/credits that can be taken from a transfer institution
Number3	0
Number4	Number of transfer classes/credits applied
Value1	MINCLASSES
Value2	Minimum number of classes that must be taken from a specific list of courses
Value3	CLASSES
Value4	Number of classes applied
Number1	0
Number2	Minimum number of classes that must be taken from a specific list of courses
Number3	0
Number4	Number of classes applied
Value1	MINCREDITS
Value2	Minimum number of credits that must be taken from a specific list of courses
Value3	CREDITS
Value4	Number of credits applied
Number1	0
Number2	Minimum number of credits that must be taken from a specific list of courses
Number3	0
Number4	Number of credits applied

Value1	MINGPA
Value2	Minimum GPA that must be achieved
Value3	(blank)
Value4	GPA applied
Number1	0
Number2	Minimum GPA that must be achieved
Number3	0
Number4	GPA applied
Value1	MINGRADE
Value2	Minimum grade classes must have to be used on the rules in this block
Value3	(blank)
Value4	(blank)
Number1	0
Number2	Minimum grade classes must have to be used on the rules in this block
Number3	0
Number4	0
Value1	MINPERDISC
Value2	Minimum number of classes/credits that must be taken from this list of disciplines
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied
Number1	0
Number2	Minimum number of classes/credits that must be taken from this list of disciplines
Number3	0
Number4	Number of classes/credits applied
Value1	MINRES
Value2	Minimum number of classes/credits that must be taken in residence
Value3	CREDITS or CLASSES
Value4	Number of classes/credits taken in residence
Number1	0
Number2	Minimum number of classes/credits that must be taken in residence

Number3	0
Number4	Number of classes/credits taken in residence
Value1	MINTERM
Value2	Minimum number of classes/credits that must be taken per term in the specified list of courses
Value3	CREDITS or CLASSES
Value4	(blank)
Number1	0
Number2	Minimum number of classes/credits that must be taken per term in the specified list of courses
Number3	0
Number4	0
Value1	NONEXCLUSIVE
Value2	Maximum number of classes/credits that can be shared with the blocks listed
Value3	CREDITS or CLASSES
Value4	(blank)
Number1	0
Number2	Maximum number of classes/credits that can be shared with the blocks listed
Number3	0
Number4	0
Value1	OPTIONAL
Value2	(blank)
Value3	(blank)
Value4	(blank)
Number1	0
Number2	0
Number3	0
Number4	0
Value1	SAMEDISC
Value2	(blank)
Value3	(blank)
Value4	(blank)

Number1	0
Number2	0
Number3	0
Number4	0
Value1	SPMAXCREDITS
Value2	Maximum number of split credits that can be taken from a specific list of courses
Value3	CREDITS
Value4	Number of credits applied
Number1	0
Number2	Maximum number of split credits that can be taken from a specific list of courses
Number3	0
Number4	Number of credits applied
Value1	SPMAXTERM
Value2	Maximum number of split classes/credits that can be taken per term in the specified list of courses
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied to the last term processed - mostly meaningless data
Number1	0
Number2	Maximum number of split classes/credits that can be taken per term in the specified list of courses
Number3	0
Number4	Number of classes/credits applied to the last term processed - mostly meaningless data
Value1	UNDER
Value2	Maximum number of classes/credits that should be taken out of the specified list of courses
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied from this list of courses
Number1	0

Number2	Maximum number of classes/credits that should be taken out of the specified list of courses
Number3	0
Number4	Number of classes/credits applied from this list of courses

HEADERQUAL

Updated: March 25, 2022

This result type shows block header qualifier text, tied to parsed blocks (dap_stu_id=req-id).

Field	Description	Example
Block-Seq-Num	Always 1	1
Result-Seq-Num	Used to sort the qualifiers in this block	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	0 - plus the qualifier's node-id	0-23
Node-Type	The qualifier type number	4101
Value1	Qualifier's node-id	MAXCLASSES
Value2	Qualifier tag (if found)	15
Value3	CREDITS or CLASSES or blank	CLASSES
Value4	May have a value or may be blank – use Number4 below	(blank)
Number1	(not used – will always be 0)	0
Number2	(see table below)	15
Number3	(not used– will always be 0)	0
Number4	(see table below)	3
Freetext	The qualifier's label text or some text in square brackets describing the qualifier	PE @

For ResultType records of HEADERQUAL, the Value1 field will contain the type of qualifier and will contain these values in the Value and Number fields.

Value1	CLASSES
Value2	Minimum number of classes that must be taken
Value3	CLASSES
Value4	Number of classes taken

Number1	0
Number2	Minimum number of classes that must be taken
Number3	0
Number4	Number of classes taken
Value1	CREDITS
Value2	Minimum number of credits that must be taken
Value3	CREDITS
Value4	Number of credits taken
Number1	0
Number2	Minimum number of credits that must be taken
Number3	0
Number4	Number of credits taken
Value1	HIGHPRIORITY
Value2	(blank)
Value3	(blank)
Value4	(blank)
Number1	0
Number2	0
Number3	0
Number4	0
Value1	LASTRES
Value2	Number of classes or credits required in residence
Value3	CREDITS or CLASSES
Value4	Number of in residence classes/credits applied – as of the last transfer class
Number1	0
Number2	Number of classes or credits required in residence
Number3	0
Number4	Number of in residence classes/credits applied – as of the last transfer class
Value1	LOWPRIORITY
Value2	(blank)
Value3	(blank)

Value4	(blank)
Number1	0
Number2	0
Number3	0
Number4	0
Value1	MAXCLASSES
Value2	Maximum number of classes that can be taken from a specific list of courses
Value3	CLASSES
Value4	Number of classes applied
Number1	0
Number2	Maximum number of classes that can be taken from a specific list of courses
Number3	0
Number4	Number of classes applied
Value1	MAXCREDITS
Value2	Maximum number of credits that can be taken from a specific list of courses
Value3	CREDITS
Value4	Number of credits applied
Number1	0
Number2	Maximum number of credits that can be taken from a specific list of courses
Number3	0
Number4	Number of credits applied
Value1	MAXPASSFAIL
Value2	Maximum number of classes/credits that can be taken passfail
Value3	CREDITS or CLASSES
Value4	Number of passfail classes/credits applied
Number1	0
Number2	Maximum number of classes/credits that can be taken passfail
Number3	0
Number4	Number of passfail classes/credits applied
VALUE1	MAXPERDISC

Value2	Maximum number of classes/credits that can be taken from this list of disciplines
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied
Number1	0
Number2	Maximum number of classes/credits that can be taken from this list of disciplines
Number3	0
Number4	Number of classes/credits applied
Value1	MAXTERM
Value2	Maximum number of classes/credits that can be taken per term in the specified list of courses
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied to the last term processed - mostly meaningless data
Number1	0
Number2	Maximum number of classes/credits that can be taken per term in the specified list of courses
Number3	0
Number4	Number of classes/credits applied to the last term processed - mostly meaningless data
Value1	MAXTRANSFER
Value2	Maximum number of classes/credits that can be taken from a transfer institution
Value3	CREDITS or CLASSES
Value4	Number of transfer classes/credits applied
Number1	0
Number2	Maximum number of classes/credits that can be taken from a transfer institution
Number3	0
Number4	Number of transfer classes/credits applied
Value1	MINCLASSES
Value2	Minimum number of classes that must be taken from a specific list of courses
Value3	CLASSES
Value4	Number of classes applied

Number1	0
Number2	Minimum number of classes that must be taken from a specific list of courses
Number3	0
Number4	Number of classes applied
Value1	MINCREDITS
Value2	Minimum number of credits that must be taken from a specific list of courses
Value3	CREDITS
Value4	Number of credits applied
Number1	0
Number2	Minimum number of credits that must be taken from a specific list of courses
Number3	0
Number4	Number of credits applied
Value1	MINGPA
Value2	Minimum GPA that must be achieved
Value3	(blank)
Value4	GPA applied
Number1	0
Number2	Minimum GPA that must be achieved
Number3	0
Number4	GPA applied
Value1	MINGRADE
Value2	Minimum grade classes must have to be used on the rules in this block
Value3	(blank)
Value4	(blank)
Number1	0
Number2	Minimum grade classes must have to be used on the rules in this block
Number3	0
Number4	0
Value1	MINPERDISC
Value2	Minimum number of classes/credits that must be taken from this list of disciplines

Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied
Number1	0
Number2	Minimum number of classes/credits that must be taken from this list of disciplines
Number3	0
Number4	Number of classes/credits applied
Value1	MINRES
Value2	Minimum number of classes/credits that must be taken in residence
Value3	CREDITS or CLASSES
Value4	Number of classes/credits taken in residence
Number1	0
Number2	Minimum number of classes/credits that must be taken in residence
Number3	0
Number4	Number of classes/credits taken in residence
Value1	MINTERM
Value2	Minimum number of classes/credits that must be taken per term in the specified list of courses
Value3	CREDITS or CLASSES
Value4	(blank)
Number1	0
Number2	Minimum number of classes/credits that must be taken per term in the specified list of courses
Number3	0
Number4	0
Value1	NONEXCLUSIVE
Value2	Maximum number of classes/credits that can be shared with the blocks listed
Value3	CREDITS or CLASSES
Value4	(blank)
Number1	0
Number2	Maximum number of classes/credits that can be shared with the blocks listed

Number3	0
Number4	0
Value1	OPTIONAL
Value2	(blank)
Value3	(blank)
Value4	(blank)
Number1	0
Number2	0
Number3	0
Number4	0
Value1	SAMEDISC
Value2	(blank)
Value3	(blank)
Value4	(blank)
Number1	0
Number2	0
Number3	0
Number4	0
Value1	SPMAXCREDITS
Value2	Maximum number of split credits that can be taken from a specific list of courses
Value3	CREDITS
Value4	Number of credits applied
Number1	0
Number2	Maximum number of split credits that can be taken from a specific list of courses
Number3	0
Number4	Number of credits applied
Value1	SPMAXTERM
Value2	Maximum number of split classes/credits that can be taken per term in the specified list of courses
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied to the last term processed - mostly meaningless data

Number1	0
Number2	Maximum number of split classes/credits that can be taken per term in the specified list of courses
Number3	0
Number4	Number of classes/credits applied to the last term processed - mostly meaningless data
Value1	UNDER
Value2	Maximum number of classes/credits that should be taken out of the specified list of courses
Value3	CREDITS or CLASSES
Value4	Number of classes/credits applied from this list of courses
Number1	0
Number2	Maximum number of classes/credits that should be taken out of the specified list of courses
Number3	0
Number4	Number of classes/credits applied from this list of courses

INSUFFCLASSAPPLIED

Updated: September 30, 2022

This result type shows insufficient class applied.

Field	Description	Example
Block-Seq-Num	Always 92	92
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	INSUFF	INSUFF
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Discipline	BIOL
Value2	Number	107
Value3	dap-class-id on the dap-resclass-dtl	24
Value4	(not used)	(blank)
Number1	(not used)	0

Field	Description	Example
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Discipline Number	BIOL 107

INSUFFCRCL

Updated: March 25, 2022

This result type shows insufficient number of classes and credits applied.

Field	Description	Example
Block-Seq-Num	Always 92	92
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	INSUFF	INSUFF
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Credits applied	23
Value2	Classes applied	6
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	Credits applied	23
Number2	Classes applied	6
Number3	(not used)	0
Number4	(not used)	0
Freetext	Classes and credits applied	23 Credits and 6 Classes

LABEL

Updated: March 25, 2022

This result type shows the label for this rule, tied to student audits.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123

Field	Description	Example
Rule-ID	Rule sequence number	5-2
Node-Type	Depends on type of rule	4200
Value1	Rule percent-complete	65
Value2	Credits applied	12
Value3	Classes applied	3
Value4	(not used)	(blank)
Number1	Rule percent-complete	65
Number2	Credits applied	12
Number3	Classes applied	3
Number4	(not used)	0
Freetext	Label text	History Requirement

LABEL

Updated: March 25, 2022

This result type shows the label for this rule, tied to parsed blocks (dap_stu_id=req-id).

Field	Description	Example
Block-Seq-Num	Always 1	1
Result-Seq-Num	Used to sort the labels in this block	12
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Depends on type of rule	4200
Value1	Node ID	178
Value2	Label tag	HISTREQ
Value3	(not used)	3
Value4	(not used)	(blank)
Number1	Node ID	178
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Label text	HISTREQ

NONCRAPPLIED

Updated: March 25, 2022

This result type shows noncourses applied.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Always 4800 for noncourse rules	4800
Value1	Noncourse code (from UCX-SCR003)	THESIS
Value2	Noncourse value or blank	DONE
Value3	dap-class-id on the dap-resnoncr-dtl	42
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Code Value	THESIS DONE

NONCRNEEDED

Updated: March 25, 2022

This result type shows noncourses needed.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Always 4800 for noncourse rules	4800
Value1	Noncourse code (from UCX-SCR003)	THESIS

Field	Description	Example
Value2	Noncourse relational operator	=
Value3	Noncourse value or blank	DONE
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Code relational-operator Value	THESIS = DONE

NONCRNUMNEED

Updated: March 25, 2022

This result type shows the number of noncourses needed.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Always 4800 for noncourse rules	4800
Value1	Number of noncourses	1
Value2	(not used)	(blank)
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	Number of noncourses	1
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Num-noncourses "Noncourses"	1 Noncourses

OTLCLASSAPPLIED

Updated: March 25, 2022

This result type shows over-the-limit class applied.

Field	Description	Example
Block-Seq-Num	Always 91	91
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	OTL	OTL
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Discipline	BIOL
Value2	Number	107
Value3	dap-class-id on the dap-resclass-dtl	24
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Discipline Number	BIOL 107

OTLCRCL

Updated: March 25, 2022

This result type shows over-the-limit number of classes and credits applied.

Field	Description	Example
Block-Seq-Num	Always 91	91
Result-Seq-Num	Used to sort result records for this student	167
Req-ID	OTL	OTL
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	Credits applied	23
Value2	Classes applied	6
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	Credits applied	23
Number2	Classes applied	6
Number3	(not used)	0
Number4	(not used)	0

Field	Description	Example
Freetext	Classes and credits applied	23 Credits and 6 Classes

PLAN

Updated: September 30, 2022

This result type shows audit ID, date, and time of an audit.

Field	Description	Example
Block-Seq-Num	(not used)	0
Result-Seq-Num	Used to sort result records for this student	1
Req-ID	Always set to STUINFO	STUINFO
Rule-ID	(not used)	0
Node-Type	(not used)	0
Value1	P if data created from a plan; T P if data came from a template	
Value2	Cutoff term as specified in Transit – DAP59	201510
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	sep_plan.plan_id or sep_tmpl_mst.tmpl_mst_id	ASDFKA897ASDFKA

QUALADVICE

Updated: March 25, 2022

This result type shows rule qualifier advice.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2

Field	Description	Example
Node-Type	Depends on type of rule	4200
Value1	(not used)	(blank)
Value2	(not used)	(blank)
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Rule qualifier advice text	Minspread not satisfied

REMARK

Updated: March 25, 2022

This result type shows block header and rule remarks.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	0 if part of header; non-zero if part of a rule	5
Node-Type	(not used)	0
Value1	(not used)	(blank)
Value2	(not used)	(blank)
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Remark text	2.5 GPA required

RULEADVICE

Updated: September 30, 2022

This result type shows textual rule advice.

Field	Description	Example
Block-Seq-Num	Sequence of the block in the audit	1
Result-Seq-Num	Used to sort result records for this student	67
Req-ID	Requirement ID of this block – valid in dap-req-block	RA000123
Rule-ID	Rule sequence number	5-2
Node-Type	Depends on type of rule. For more information, see the Node-Types topic.	4200
Value1	If the rule is not yet complete, this will contain the number of remaining groups or non-courses	2
Value2	(not used)	(blank)
Value3	(not used)	(blank)
Value4	(not used)	(blank)
Number1	(not used)	0
Number2	(not used)	0
Number3	(not used)	0
Number4	(not used)	0
Freetext	Proxy-Advice or rule specific advice	See major block

Node-Types

Updated: March 24, 2023

Some result types have a value in the node-type. This number indicates the type of rule or qualifier the result is for.

Rule node-types

Type of rule	Node type
COURSE_RULE	4200
IF_STMT_RULE	4300
BLOCK_RULE	4400
BLOCKTYPE_RULE	4500
GROUP_RULE	4600

Type of rule	Node type
SUBSET_RULE	4700
NONCOURSE_RULE	4800
COMPLETE_RULE	4900
INCLUDEBLOCKS_RULE	4950
INCOMPLETE_RULE	4971

Qualifier node-types

Type of qualifier	Node type
MAXCLASS	4101
MINCLASS	4102
MAXCREDIT	4103
MINCREDIT	4104
MAXTERM_B	4105
MINTERM_B	4106
MAXPERDISC	4107
MINPERDISC	4108
SAMEDISC	4109
MAXTRANSFER	4110
NONEXCLUSIVE	4111
LASTRES_CLASSES	4112
LASTRES_CREDITS	4113
MAXPF_CLASSES	4114
MAXPF_CREDITS	4115
MINGPA	4116
MINGRADE	4117
MINRES_CLASSES	4118
MINRES_CREDITS	4119
OPTIONAL	4120
CLASSES_CREDITS	4121
EXCLUSIVE	4122
MINAREAS	4123
MAXTERM_CLASSES	4124
MAXTERM_CREDITS	4125
MINTERM_CLASSES	4126
MINTERM_CREDITS	4127
MAXSPREAD	4128

Type of qualifier	Node type
MINSPREAD	4129
NOTGPA	4130
SPMAXCREDIT	4131
SPMAXTERM_B	4132
KEEPNEWEST	4133
KEEPOLDEST	4134
HIGH_PRIORITY	4135
LOW_PRIORITY	4136
STANDALONEBLOCK	4137
RULETAG	4138
UNDER	4139
CHECKELECTIVES	4140
LOWEST_PRIORITY	4141
HEADERTAG	4142

Numeric fields

Updated: March 25, 2022

Record types containing numbers in the Value1-4 fields will also be the corresponding values stored in the Number1-4 fields, but as actual database numeric values.

The number fields get loaded with real values for these result types:

- BLOCKGPA
- BLOCKCRCLAPPLIED
- BLOCKCRCLNEEDED
- LABEL
- CRCLNEEDED
- NONCRNUMNEED
- FALLCRCL
- OTLCRCL
- INSUFFCRCL

For the other result types, the number fields contain 0.0.

Rule ID

Updated: March 25, 2022

Each rule in a block has a unique rule-ID used to help identify each rule in the dap-result-dtl.

The first rule has an ID of 1, the second rule 2, and so on. When rules appear within groups or subsets or if-statements, additional ID information is added allowing you to keep track of the hierarchy. As the rules get deeper, a hyphen and number are added at each level to represent the relationship to the parent rule.

However, if the rule-id is greater than 20 characters, the rule's node-id will be used instead.

Rule ID	Rule
1	5 Credits in math 105 Label "Intro to Algebra"
2	5 Credits in math 112 Label "Algebra II"
3-0	Beginsub 5 Credits in HIST 124 Label "European History";
3-1	5 Credits in HIST 125 Label "American History";
3-2	5 Credits in HIST 126 Label "Current History";
3	EndSub Label "History requirement";
4-0	1 group in (6 credits in CHEM 104, 105 Label "Chemistry for non science majors") or
4-1	(6 credits in CHEM 114, 115 Label "chemistry for science majors")
4	Label "Chemistry requirement";

MAYTAKE vs MUSTTAKE vs CANTTAKE

Updated: March 25, 2022

Classes listed as options are reported in the database as MAYTAKE while classes listed in "plus" lists or as part of an "including" list are marked as MUSTTAKE. Classes that cannot be used to satisfy a rule are coded with CANTTAKE.

The classes listed in these rules will be marked as MUSTTAKE.

Rule	Reason
1 Class in Math 123	Only class listed

Rule	Reason
5 Credits in Math 123	Only class listed
2 Classes in Math 123 + 124	Part of a + list
2 Classes in Math 123, 124	Number of classes specified
10 Credits in Math 123 + 124	Part of a + list
15 Credits in Math @ Including Math 123	Part of the Including list

The classes listed in these rules will be marked as **MAYTAKE**.

Rule	Reason
1 Class in Math 123, 124	Only 1 class is needed but multiple classes are listed.
10 Credits in Math 123, 124	Credits are specified; possibly could take one class to satisfy the requirement.
1 group in (1 Class in math 123) or (1 Class in math 124)	Part of a group rule

The classes listed in the Except list of these rules will be marked as **CANTTAKE**.

Rule	Reason
10 Credits in Math 12@ Except Math 128	Math 128 cannot be used to satisfy this rule.

Academic audits in CPA

Updated: September 30, 2022

The Curriculum Planning Assistant (CPA) information can use academic audit data to make advising a central component of enrollment management, which you can use for institutional planning and resource allocation.

The CPA academic audit data is comprehensive, term-specific, and gathered from every college, school, division, and academic department within the institution.

DAP16 labels

Updated: March 25, 2022

To help some CPA reporting, we store each of the labels in each of the blocks in the dap_result_dtl.

When a block is saved to the database the parser saves each label found in the block along with its rule_ID to the dap_result_dtl table. The dap_stu_id and dap_req_id fields are loaded with the

block's RA number. A dummy dap_student_mst is created for the block with the stu_id field as its RA number – this is done for data integrity.

Before trying to use the CPA reports for the first time after installation you may need to run DAP16 to build the dap_result_dtl records for each of the blocks.

NEW audits

Updated: March 24, 2023

New audits created by either bridging student data through the extract, running DAP22 in Transit, or by a user in Responsive Dashboard will have the dap_what field on dap_audit_dtl set to NEW. Audit results can then be generated with either DAP22 or DAP25.

Audit results can be dynamically created from NEW audits using DAP25. This process searches for newly created audits and creates the audit results records. It is a low-level background process and thus uses the CPU only when it is not in use and should not affect system performance significantly. DAP25 can be started and stopped at any time using the resrestart and resstop scripts, but normally would run all day long watching for any new audits. It might be a good idea, however, to shut down this job while your nightly extract is run and then start it again when it is finished. You might consider setting up your cron job to run resstop before running your extract, and issue the resrestart after your extract finishes. For more information, see the [Dynamic generation of audit results using DAP25](#) topic.

Create academic audit CPA data

Updated: September 30, 2022

You can opt to create Curriculum Planning Assistant (CPA) data when you are generating audits and then use that data in your institution's reporting.

About this task

CPA data can be built by running batch audits (DAP22 through Transit) and specifying you want to also build the CPA results records. Before a new set of records is saved for a student, the current set of CPA results is first deleted. However, because an institution may want to keep results for multiple degrees for a student, either simultaneous or consecutive, only the CPA results for the school and degree against which the audit was run are deleted. As a result, a student could have two sets of CPA results in the database, for example, one for a BA in Art History degree and one for an MBA degree.

When CPA results are built, they are added to the database student by student. Old CPA results data for that particular student for the same school and degree are deleted before the new records are added based on the core.audit.cpa.latestTermOnly Shepherd setting. For more information, see the [core.audit](#) topic.

Typically, the pool of students processed by the CPA are currently enrolled students. Those students that have graduated, transferred, or taken time off will most likely not be in this pool, but the CPA results data for these students from the previous term will most likely still be in the Degree Works database. You may create a report that helps look for trends on what classes are generally used to satisfy certain requirements or what classes have a high failing rate. Keeping historic CPA results data can help provide valuable information to your institution's administrators.

Batch generation of audit results using DAP22

Updated: September 30, 2022

You can use DAP22 in Transit to read audits and generate results.

You can generate audits results in batch using DAP22 in Transit. Under **Questions** you would select **Build audit results to allow queries against the database (CPA)?**. New audits may be created to feed the audit results by selecting **Create new audit?**. If new audits are not created, the most recent audit for each student will be used to build the audit results. Be sure to choose the correct audit type when building results. You can choose Academic Audit, Athletic Eligibility Audit, or Financial Aid Audit. The set of students is based on those students chosen in your **Selection criteria** in Transit. When students are selected, results will be built for all of their most recent degree audits. If they have two degrees on their curriculum, they will get two sets of results generated.

DAP22 is designed not to build audit results for a particular audit if the audit results were already built and still exist in dap_result_dtl. You can override this behavior by running the following command, and then issuing a tberestart.

```
$ export DAP32_FORCE_RESULTS=1
```

With this flag set, the existing records for the audit are removed and new records for the audit are built. This may be useful if you change any of the UCX-CFG020 RESULTS flags or if new software is delivered that fixes a problem with the results records. In most situations, however, you should not need to set this flag.

These settings control whether results are built for audits generated after the bannerextract or RAD30JOB is run and whether in-progress and preregistered classes are included in the audit.

```
integration.bridge.audits.buildCpaResults  
integration.bridge.audits.includeInProgress  
integration.bridge.audits.includePreregistered
```

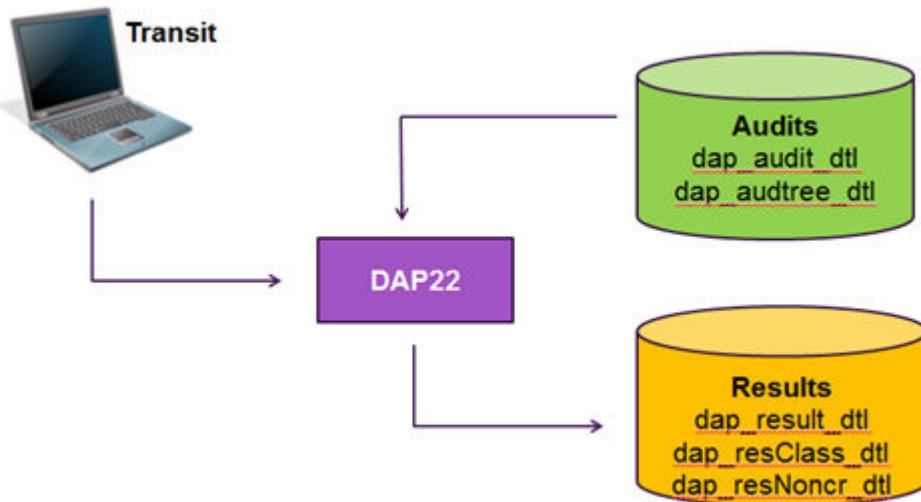
Set integration.bridge.audits.buildCpaResults to true so that results will be generated for every audit launched by the extract/bridge.

You should not use both DAP22 and DAP25 because they both serve the same purpose. You may, however, want to generate new audits for your students and build results. If you do this, be sure to do a resstop to terminate the daemon process so it does not interfere with your running of DAP22.

However, if you set the integration.bridge.audits.buildCpaResults flag, you will find that only those audits created from the extract get turned into CPA records. The audits that are run during the day (perhaps because of the UCX-CFG020 REFRESH flags) will not have CPA data generated. For this reason, it is imperative that you do a resrestart to have the dynamic processor running to catch newly created audits. You do not, however, want this running when you are running the extract/bridge. In cron you will want to do something like this regardless of whether you have the integration.bridge.audits.buildCpaResults flag set. It is best to wait until the extract has run and all audits are finished to have the dynamic processor create the CPA data.

```
resstop  
launchjob --wait $ADMIN_HOME/myjobs/rad30.standardSql.json  
resrestart
```

The following diagram shows how DAP22 reads the audits and generates results.



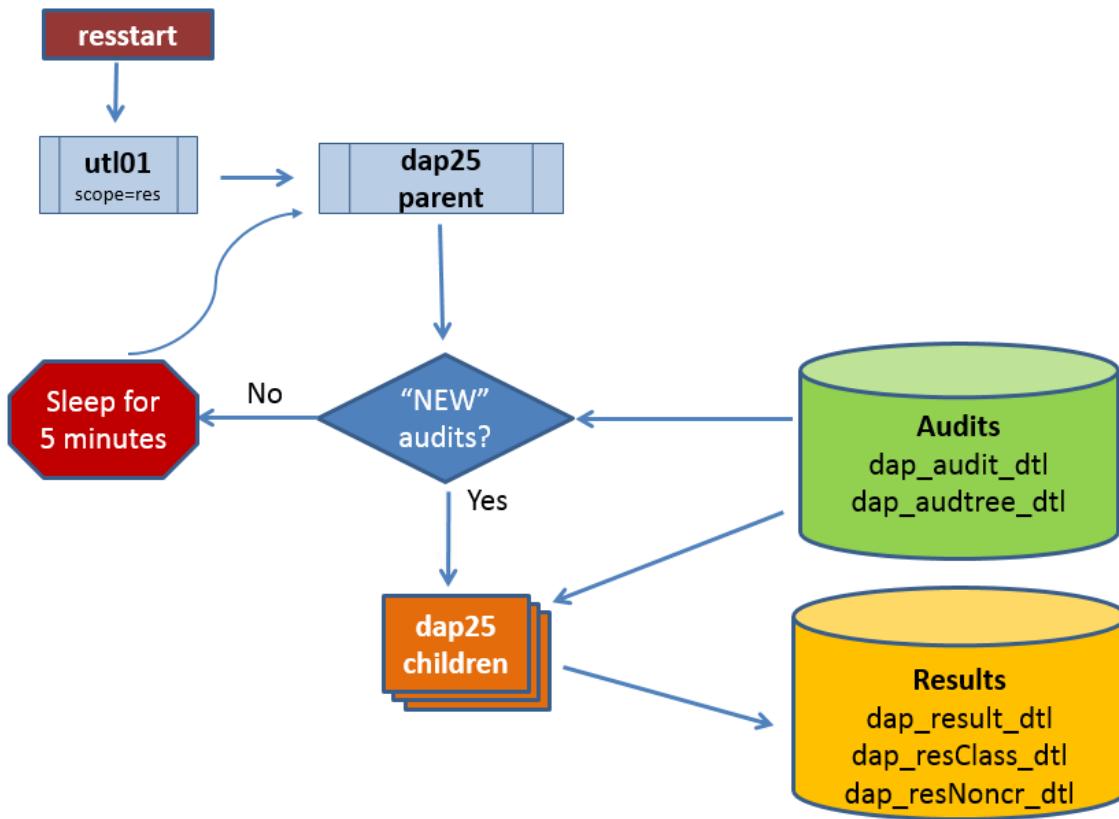
Dynamic generation of audit results using DAP25

Updated: September 30, 2022

As it can take time to build the results from an audit and because schools may want to always have the most up-to-date results information, dynamic generation of audit results is an option.

Results built this way, however, are done based only on Academic Audits.

1. When a new audit is run it is saved with dap_what set to NEW on the dap_audit_dtl.
2. The resrestart script is run to start the daemon/listener to dynamically generate audit results. A utl01 is started; it spawns off exactly one dap25 parent process.
3. The parent dap25 process will wake up and check for more new audits according to the classic.daemons.dap25.sleepSeconds Shepherd setting. It will query for dap_audit_dtl records with a dap_what = NEW, stopping when it reaches the maximum defined in the classic.daemons.dap25.auditMaximum Shepherd setting.
4. The parent dap25 process will spawn a sufficient number of child processes to process the new audits. The maximum number of audits per child is controlled by the classic.daemons.dap25.auditsPerChild Shepherd setting. If classic.daemons.dap25.auditMaximum is set to 100 and classic.daemons.dap25.auditsPerChild is set to 20 then the dap25 parent will create 5 child process to handle the 100 audits – giving each child 20 audits each to process.
5. The dap_what on the dap_audit_dtl is set to RESULTS so it is no longer considered “NEW”.
6. If you want the dap25 parent process to continue checking forever for new audits then ensure the classic.daemons.dap25.loopMaximum Shepherd setting is set to 0. You may set it to 10, for example, if you only want to it run 10 times and then quit. This is generally used for testing; normally this setting should be set to 0.



Scripts/commands

- resrestart – stops and then starts a new dap25 daemon process
- resshow – shows the dap25 parent and children that are running
- resstop – stops the dap25 daemon process
- restart – starts a new dap25 daemon process
- dapreslook – tool to show the results records built for a given student

You should do a resrestart at least one time a week to limit the size of the dap25.log file created in the logdebug directory.

Even if you did not run resrestart, you can still run resshow to display the number of NEW academic audits in the dap_audit_dtl along with the number of academic audits currently in the dap_result_dtl.

When running resshow, supply the counts parameter.

```
$ resshow counts
```

After running resrestart and run resshow without (or with) the counts parameter, you will most likely see just the utl01 process along with the parent dap25. The utl01 process is there to monitor dap25 in case it dies. A new dap25 will be restarted if it dies.

This parent dap25 sleeps most of the time, but when it wakes up it queries the db to check for NEW audits. Only when there are NEW audits to run will the parent dap25 spawn off a list of child dap25 processes. You will see those child processes appear in the resshow display. When the results have been built, each dap25 child will die.

Planner audits in CPA

Updated: September 30, 2022

With DAP59, planner audit data can be loaded into CPA and then used to help determine the student demand for classes or timetabling.

This data can be used to plan class schedules and to determine the resources that will be required for classes, for example, classroom size. The information can also be used by third-party class scheduling tools. Information from the Student Educational Planner (SEP) is used to generate this data in the CPA.

Using a student's active and approved plan, or a temporary template plan (in cases where a student does not have an active, approved plan), an institute can forecast the courses and sections that need to be available for registration in a future term.

Timetabling data is generated by running the Batch Timetabling Processor (DAP59), which is launched through Transit. For more information, see the [DAP59 - Batch timetabling processor](#) topic. The pool of students that have to be processed is determined either with selection criteria, SQL or a file of ID numbers. The cutoff term for which timetabling data is to be generated must also be specified in Transit. DAP59 will generate a planner audit for the student's active and approved plan for the selected school. If an active and approved plan for that student and school is not found, DAP59 may use a temporary plan for the student instead. A temporary plan is used if the Create Plan from Template Processor (DAP54) was run with the temporary option before running DAP59. For more information, see the [DAP54 - Template processor plan creation](#) topic.

Before running DAP59, ensure that the UCX-CFG020 RESULTS Build CATALOGYEAR and GOALDATA records flag is set to Y so that all the data required for timetabling is created.

Results reporting

Updated: March 25, 2022

DAP59 loads the timetabling results in dap_result_dtl. The additional class data is also loaded in dap_resclass_dtl and non-course data is loaded in dap_resnoncr_dtl. The dap_audit_type is PL on these records.

The dap_result_dtl record with the dap_result_type = PLAN can be used to determine if the results came from a plan (dap_value1=P) or a temporary plan/template (dap_value1=T). Additionally, dap_freetext contains the plan_id from the sep_plan or the tmpl_mst_id from the sep_tmpl_mst.

The dap_req_id on the dap_result_dtl indicates the block of the audit to which the pre-registered or planned class would apply. If the class would apply to the Fallthrough, Insufficient or Over the Limit sections of the audit, the dap_req_id would be FALLTHRU, INSUFF or OTL, respectively.

The dap_value3 column on the dap_result_dtl contains an identifier that can be used to link the class to the dap_class_id on the dap_resclass_dtl. You must also link the records based on the dap_stu_id, dap_audit_type, dap_school, and dap_degree fields.

The dap_term field on the dap_resclass_dtl is the term code from the plan. Planned classes will have a dap_grade of 'PLAN' and a blank dap_class_status.

A dap_result_dtl record is created for each course listed on the original requirement. This result-type is CLASSREQUIRED and is used only for timetabling.

For more information, see the [Result types](#) topic.

Reporting with other Degree Works data

Updated: March 25, 2022

Degree Works stores valuable data from Scribe requirement blocks, exceptions, notes, plan and template data that can be used for reporting.

Student Educational Planner

Updated: March 24, 2023

The Student Educational Planner gives students and their advisors the ability to plan out their educational goals with confidence that they will complete their program on time.

A plan consists of a series of academic terms each containing one or more requirements. A requirement is something that the student must fulfill to keep on track with their program completion goal.

There are several different types of requirements that can be chosen by the student.

- The most common type of requirement is a course requirement. These are planned courses a student expects to take.
- Another type of requirement is a GPA requirement. This means that the student plans to achieve an overall, major, or class list GPA of at least a certain level.
- A third type of requirement is a test requirement. This means the student is intending to pass a certain test, for example, English Composition, with a minimum grade.
- The fourth type of requirement is the non-course requirement. This requirement allows for the inclusion of specific requirements that do not fall in the realm of the previous three types, such as honors papers, or recitals.

- The last requirement type is the placeholder criteria, which is meant as a catch all requirement that accepts any free-form requirement the student or advisor may want to place on the plan. Unlike the previous requirements, placeholder requirements cannot be represented in Scribe, so cannot be checked by an audit, and cannot be tracked.

Requirements can also be grouped together to form a complex requirement.

Requirement type	Description
Course	Courses the student plans to take.
GPA	Minimum GPA the student expects to achieve.
Test	A minimum score on a test.
Non-course	A specific requirement chosen from a set of requirements defined by the school.
Placeholder	A free-form catchall requirement that cannot be automatically tracked.
Group	A group of one or more requirements.

Each requirement may express different data items that are specific to that type, and there are data items that all have in common. For example, the course requirement has the data items: course discipline, course number, and minimum grade, while the test requirement has test code and minimum score. Common data items include tracking status, critical flag, and notes. Free format notes are provided at every level of the plan, including the plan itself, terms, groups, and requirements. There are internal notes and public notes, and your role will determine which notes you may see.

These requirements may be grouped to form more complex requirement, such as (MATH 101 or ACCT 200) AND (MATH 102 or ACCT 202). There is no limit to the depth of parenthesis that can be expressed in the planner structures. However, the user interface currently limits not only the depth but also the types of requirements that can be grouped. At the current time, only two levels of sub-grouping are allowed and only course requirements are allowed to be included. In fact, the user interface treats these groupings like another type of requirement, called choice. The choice requirement, however, is really just a grouping of course requirements.

Plans can be created from templates. The structure of templates closely mimics that of plans, and much of the following discussion of plan structure can be applied to templates. Templates have an additional structure, SEP_TMPL_TAG, which allows the templates to be tagged to facilitate their organization and search.

Plan data structures

Updated: March 24, 2023

The overall structure is tree-like with nodes hanging under the root plan node.

Groups can be of type union (UN) or choice (CH). Union groups contain requirements that must all be met, for example: ENGL 101 and MATH 200. Choice groups contain requirements where only one must be met, for example: ENGL 101 or MATH 200. A group will contain 1 or more requirements and groups. There is no limit to the number of groups or requirements in a group.

Only one group is required, and that is the root group under a term, and it is always a union type group.

For example, a student's plan might contain the following term requirements.

Table 1: Term: Fall 2022

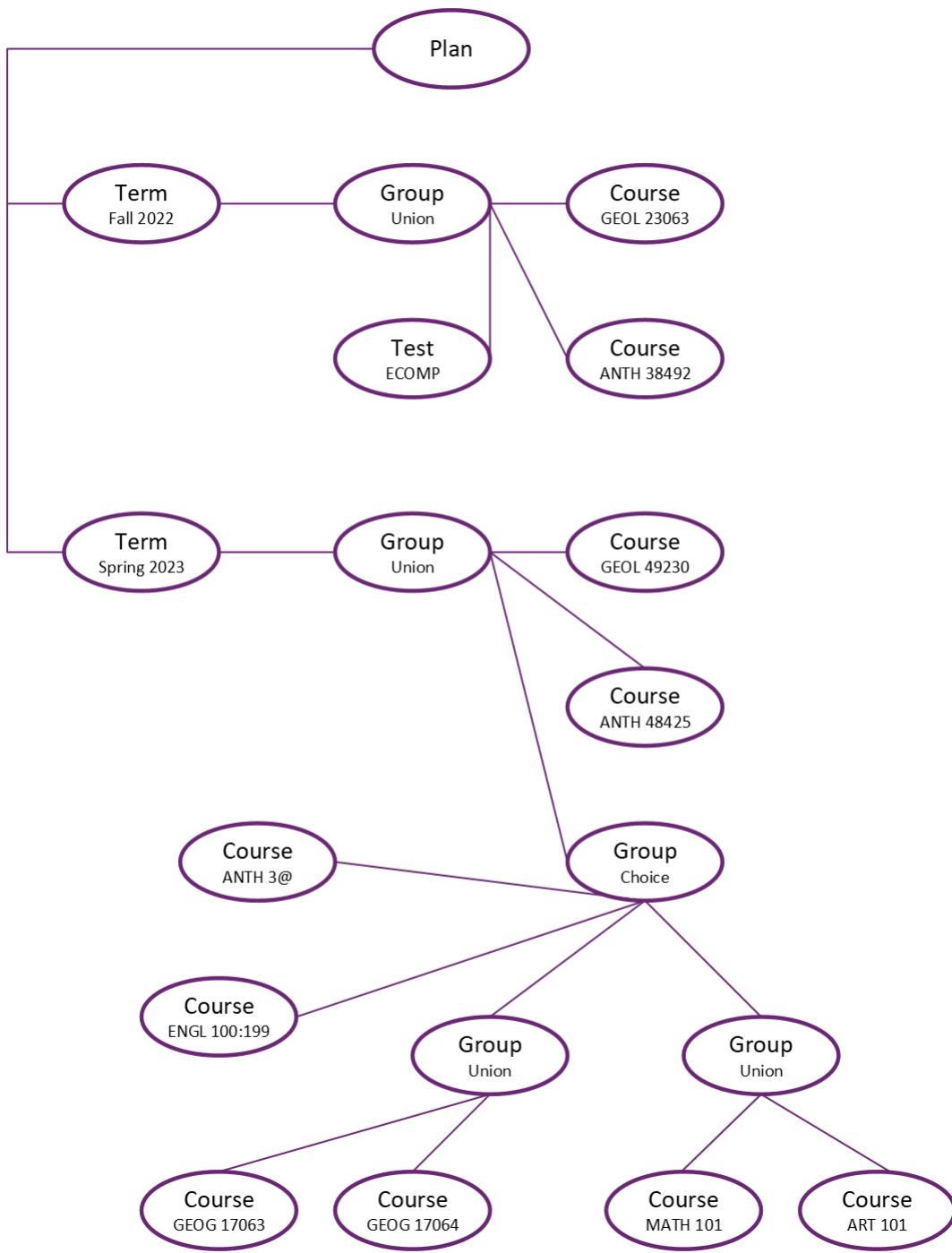
Course Requirement	GEOL 23063
Course Requirement	ANTH 38492
Test Requirement	ECOMP (English Composition)

Table 2: Term: Spring 2023

Course Requirement	GEOL 49230
Course Requirement	ANTH 48425
Choice Requirement	ANTH 3@ or ENGL 100:199 or (MATH 101 and ART 101) or (GEOG 17063 and GEOG 17064)

Note: The second level group (choice) in the Spring 2023 term is shown in the user interface as a choice requirement, although it is really just a grouping of courses. The user interface will currently not allow for more depth than shown here.

This can be represented by the following diagram.



SEP_PLAN

Updated: September 30, 2022

This table is the overall parent table for a plan. In this table, there is only one entry for every plan.

Item name	Data type	Maximum size	Description
PLAN_ID	CHAR	36	The plan object ID, a unique identifier generated for the plan.
STUDENT_ID	CHAR	10	The SIS ID. References the DAP_STU_ID column in the DAP_STUDENT_MST table.
TMPL_MST_ID	CHAR	36	A reference to the TMPL_MST_ID column of the SEP_TMPL_MST. If not empty, it represents the template from which this plan was created.
DESCRIPTION	VARCHAR	80	A free-format description of the plan.
IS_ACTIVE	CHAR	1	"Y" or "N", indicating whether or not this is an active plan.
IS_LOCKED	CHAR	1	"Y" or "N", indicating whether or not the plan is locked.
APPROVAL_STATUS	CHAR	2	A coded value, one of the following: AA – Auto approved AP – Approved NO – Needs Approval PE – Pending Approval RJ – Rejected
SCOPE	CHAR	6	A coded value from UCX_SEP011, representing the type of plan. For example, COMP – Comprehensive Plan.
SCOPE_DATE	TIMESTAMP	7	The date the scope was modified.

Item name	Data type	Maximum size	Description
SCHOOL	VARCHAR	12	A coded value from UCX_STU350, representing the school/level to which this plan applies. For example, UG – Undergraduate.
DEGREE	VARCHAR	12	A coded value from UCX_STU307, representing the degree to which this plan applies. For example, BA – Bachelor of Arts.
OFFICIAL_TRACKING_STATUS	VARCHAR	12	<p>The official tracking status for the plan. One of the following:</p> <p>ONTRACK – Plan is on track to completion.</p> <p>OFFTRACK – Plan is not on track to completion.</p> <p>NOTEVALUED – Tracking status of the plan has not yet been evaluated.</p>
UNOFFICIAL_TRACKING_STATUS	VARCHAR	12	<p>The unofficial tracking status for the plan. One of the following:</p> <p>ONTRACK – Plan is on track to completion.</p> <p>OFFTRACK – Plan is not on track to completion.</p> <p>NOTEVALUED – Tracking status of the plan has not yet been evaluated.</p>
IS_TRACKING_STATUS_CURRENT	CHAR	1	“Y” or “N”. Indicates whether or not any changes have been

Item name	Data type	Maximum size	Description
			made to the plan from when the tracking status was last calculated.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the plan.
MODIFY_DATE	TIMESTAMP	7	The date and time the plan was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the plan.
CREATE_WHO	VARCHAR	14	The ID of the user who created the plan.
CREATE_DATE	TIMESTAMP	7	The date the plan was created.
CREATE_WHAT	VARCHAR	30	The process that created the plan.
OVERRIDE_WHO	VARCHAR	14	The ID of the user who saved a with prerequisite errors.
OVERRIDE_DATE	TIMESTAMP	7	The date that a plan was saved with prerequisite errors.

SEP_PLAN_NOTE

Updated: September 30, 2022

This table contains notes related to the overall plan. There may be an unlimited number of notes tied to a plan. It is linked to the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
PLAN_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
INTERNAL	CHAR	1	"Y" or "N" indicating whether or not the

Item name	Data type	Maximum size	Description
			note has restricted visibility.
COPIED_FROM_TEMPLATE	CHAR	1	"Y" or "N" indicating whether or not the note was originally copied from a template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_PLAN_TERM

Updated: September 30, 2022

This table records information about the term in which the requirements reside. In this table, there is one entry for every term that contains requirements. It is linked to the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
TERM_ID	CHAR	36	The term object ID, a unique identifier generated for the term.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this term belongs.
GROUP_ID	CHAR	36	The root group for requirements in this term. All requirements must be part of a group, and there is one group that is at the top of the hierarchy.

Item name	Data type	Maximum size	Description
TERM	VARCHAR	8	A coded value from UCX_STU016 representing the term associated with all the requirements contained within.
OFFICIAL_TRACKING_STATUS	VARCHAR	12	<p>The official tracking status for the term. One of the following:</p> <p>ONTRACK – Term is on track to completion.</p> <p>OFFTRACK – Term is not on track to completion.</p> <p>NOTEVALUED – Tracking status of the term has not yet been evaluated.</p>
UNOFFICIAL_TRACKING_STATUS	VARCHAR	12	<p>The unofficial tracking status for the term. One of the following:</p> <p>ONTRACK – Term is on track to completion.</p> <p>OFFTRACK – Term is not on track to completion.</p> <p>NOTEVALUED – Tracking status of the term has not yet been evaluated.</p>
IS_TRACKING_STATUS_CURRENT	CHAR	1	“Y” or “N”. Indicates whether or not any changes have been made to the term from when the tracking status was last calculated.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the term.

Item name	Data type	Maximum size	Description
MODIFY_DATE	TIMESTAMP	7	The date and time the term was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the term.
CREATE_WHO	VARCHAR	14	The ID of the user who created the term.
CREATE_DATE	TIMESTAMP	7	The date the term was created.
CREATE_WHAT	VARCHAR	30	The process that created the term.

SEP_PLAN_TERM_NOTE

Updated: September 30, 2022

This table contains notes related to a plan term. There can be an unlimited number of notes. It is linked to the SEP_PLAN_TERM table by the TERM_ID column and the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
TERM_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
TERM_ID	CHAR	36	Reference to the TERM_ID column of the SEP_PLAN_TERM table to which this note belongs.
PLAN_ID	CHAR	36	The text of the note.
NOTE_TEXT	CLOB	unlimited	The user ID of the note's author.
AUTHOR	VARCHAR	14	A number that records the order in which the user placed the note.
INTERNAL	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility.
COPIED_FROM_TEMPLATE	CHAR	1	"Y" or "N" indicating whether or not the note was originally copied from a template.

Item name	Data type	Maximum size	Description
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_PLAN_GROUP

Updated: September 30, 2022

This table organizes and contains requirements. All requirements link to this table. It is linked to the SEP_PLAN table by the PLAN_ID column and the SEP_PLAN_TERM table by the TERM_ID column.

Item name	Data type	Maximum size	Description
GROUP_ID	CHAR	36	The group object ID, a unique identifier generated for the group.
TERM_ID_4	CHAR	36	Reference to the TERM_ID column of the SEP_PLAN_TERM table to which this group belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this group belongs.
GROUP_TYPE	VARCHAR	2	A coded value representing the type of group. One of the following: UN – Union. All contained requirements must be fulfilled.

Item name	Data type	Maximum size	Description
			CH – Choice. One of the contained requirements must be fulfilled.
GROUP_ID_PARENT_4CHAR		36	The parent group of this group. For the root group, this element will be the same as the GROUP_ID.
SEQUENCE	DECIMAL	6	A number that records the order in which the user placed the group within the parent group together with other requirements.
IS_CRITICAL	CHAR	1	“Y” or “N”, indicating whether or not this group is critical for tracking purposes.
IS_HONORS	CHAR	1	“Y” or “N”, indicating whether or not the courses in this group must be taken for honors.
IS_GROUP_SELECTION	CHAR	1	“Y” if the student has selected this group out of a choice of requirements. “N” if the student has not selected this group out of a choice of requirements. “null” if the student has made no choice.
CAMPUS	CHAR	12	A coded value from UCX_STU576 representing the campus where the student plans to take the courses listed in this group. For example, MA - Main.
DELIVERY	VARCHAR	10	A coded value from UCX_SEP003 representing the method of coursework delivery for the

Item name	Data type	Maximum size	Description
			courses listed in this group. For example, SELF – Self-paced course.
CREDITS	DECIMAL	6,3	The minimum total number of credits for all the courses in this group. Contains 3 whole digits and 3 decimal digits.
MINIMUM_GRADE	VARCHAR	6	The minimum grade that every course in this group must meet to fulfill the requirement.
TRACKING_STATUS	VARCHAR	12	<p>The tracking status for the group. One of the following:</p> <p>ONTRACK – Group is on track to completion.</p> <p>OFFTRACK – Group is not on track to completion.</p> <p>NOTEVALUATED – Tracking status of the group has not yet been evaluated.</p>
SCRIBE_POINTER	VARCHAR	12	A coded value from UCX_SEP009 representing the audit requirement as written in Scribe that this group will fulfill.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the group.
MODIFY_DATE	TIMESTAMP	7	The date and time the group was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the group.
CREATE_WHO	VARCHAR	14	The ID of the user who created the group.

Item name	Data type	Maximum size	Description
CREATE_DATE	TIMESTAMP	7	The date the group was created.
CREATE_WHAT	VARCHAR	30	The process that created the group.

SEP_PLAN_GROUP_NOTE

Updated: September 30, 2022

This table contains notes related to a plan group. There can be an unlimited number of notes. It is linked to the SEP_PLAN_GROUP table by the GROUP_ID column and the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
GROUP_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_PLAN_GROUP table to which this note belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
INTERNAL	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility.
COPIED_FROM_TEMPLATE	CHAR	1	"Y" or "N" indicating whether or not the note was originally copied from a template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.

Item name	Data type	Maximum size	Description
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_PLAN_GPA

Updated: September 30, 2022

This table records GPA requirements, which means that the student plans to achieve an overall, major, or class list GPA of at least a certain level. There can be an unlimited number of requirements. It is linked to the SEP_PLAN table by the PLAN_ID column and the SEP_PLAN_GROUP by the GROUP_ID column.

Item name	Data type	Maximum size	Description
GPA_ID	CHAR	36	The GPA requirement object ID, a unique identifier generated for the GPA requirement.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this requirement is critical for tracking purposes.
GPA_TYPE	VARCHAR	10	A coded value from UCX_SEP008 representing the type of GPA being required. For example, MAJOR - Major GPA
MINIMUM_GPA	DECIMAL	23,9	The minimum GPA expected. Contains a maximum of 14 whole digits and 9 decimal digits.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_PLAN_GROUP table to which this requirement belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this requirement belongs.

Item name	Data type	Maximum size	Description
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
TRACKING_STATUS	VARCHAR	12	<p>The tracking status for the requirement. One of the following:</p> <p>ONTRACK – Requirement is on track to completion.</p> <p>OFFTRACK – Requirement is not on track to completion.</p> <p>NOTEVALUATED – Tracking status of the requirement has not yet been evaluated.</p>
GPA_CODE	VARCHAR	12	A coded value from UCX_STU023 representing the major associated with this GPA requirement, only if the GpaType is "MAJOR". For example, ARTH - Art History.
CLASS_LIST	VARCHAR	4000	A list of classes used to evaluate the GPA when the GpaType is "CLASS". For example, MATH 101 + MATH 102.
CALCULATED_GPA	DECIMAL	23,9	The actual GPA as calculated by the tracking software. Contains a maximum of 14 whole digits and 9 decimal digits.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.

Item name	Data type	Maximum size	Description
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_PLAN_GPA_NOTE

Updated: September 30, 2022

This table contains notes related to a plan GPA requirement. There can be an unlimited number of notes. It is linked to the SEP_PLAN_GPA table by the GPA_ID column and the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
GPA_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
GPA_ID	CHAR	36	Reference to the GPA_ID column of the SEP_PLAN_GPA table to which this note belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
INTERNAL	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility.

Item name	Data type	Maximum size	Description
COPIED_FROM_TEMPLATE	CHAR	1	"Y" or "N" indicating whether or not the note was originally copied from a template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_PLAN_CLASS

Updated: September 30, 2022

This table records course requirements. These are planned courses a student expects to take. There may be an unlimited number of requirements. It is linked to the SEP_PLAN table by the PLAN_ID column and the SEP_PLAN_GROUP table by the GROUP_ID column.

Item name	Data type	Maximum size	Description
CLASS_ID	CHAR	36	The course requirement object ID, a unique identifier generated for the requirement.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this requirement is critical for tracking purposes.
IS_HONORS	CHAR	1	"Y" or "N", indicating whether or not the courses in this group must be taken for honors.
IS_GROUP_SELECTION	CHAR	1	"Y" if the student has selected this requirement out of a choice of requirements. "N" if the student has not

Item name	Data type	Maximum size	Description
			selected this requirement out of a choice of requirements. "null" if the student has made no choice.
COURSE_DISCIPLINE	VARCHAR	12	The course discipline. For example, "MATH". Forms the first part of the course identifier (For example, "MATH 101").
COURSE_NUMBER	VARCHAR	12	The course number. For example, "101". Forms the second part of the course identifier (For example, "MATH 101").
COURSE_ATTRIBUTE	VARCHAR	12	A coded value from UCX_STU050 representing an attribute that might be attached to the course. For example, MAJORONLY - Major Only.
CAMPUS	CHAR	12	A coded value from UCX_STU576 representing the campus where the student plans to take the courses listed in this group. For example, MA - Main.
DELIVERY	VARCHAR	10	A coded value from UCX_SEP003 representing the method of coursework delivery for the courses listed in this group. For example, SELF – Self-paced course.
CREDITS	DECIMAL	7,3	The minimum total number of credits for all the courses in this group. Contains 3

Item name	Data type	Maximum size	Description
			whole digits and 3 decimal digits.
MINIMUM_GRADE	VARCHAR	6	The minimum grade that every course in this group must meet to fulfill the requirement.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_PLAN_GROUP table to which this requirement belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this requirement belongs.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
TRACKING_STATUS	VARCHAR	12	<p>The tracking status for the group. One of the following:</p> <p>ONTRACK – Group is on track to completion.</p> <p>OFFTRACK – Group is not on track to completion.</p> <p>NOTEVALUATED – Tracking status of the group has not yet been evaluated.</p>
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.

Item name	Data type	Maximum size	Description
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_PLAN_CLASS_NOTE

Updated: September 30, 2022

This table contains notes related to a plan class requirement. There can be an unlimited number of notes. It is linked to the SEP_PLAN_CLASS table by the CLASS_ID column and the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
CLASS_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
CLASS_ID	CHAR	36	Reference to the CLASS_ID column of the SEP_PLAN_CLASS table to which this note belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
INTERNAL	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility.
COPIED_FROM_TEMPLATE	CHAR	1	"Y" or "N" indicating whether or not the note was originally

Item name	Data type	Maximum size	Description
			copied from a template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_PLAN_TEST

Updated: September 30, 2022

This table records test requirements. This means the student is intending to pass a certain test with a minimum grade. There can be an unlimited number of requirements. It is linked to the SEP_PLAN table by the PLAN_ID column and the SEP_PLAN_GROUP table by the GROUP_ID column.

Item name	Data type	Maximum size	Description
TEST_ID	CHAR	36	The test requirement object ID, a unique identifier generated for this requirement.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this requirement is critical for tracking purposes.
TEST_CODE	VARCHAR	10	A coded value from UCX_SEP006 representing the type of test being required. For example, ALEV_CHEM - A-Level Chemistry.
MINIMUM_SCORE	VARCHAR	6	The minimum test score expected.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_PLAN_GROUP

Item name	Data type	Maximum size	Description
			table to which this requirement belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this requirement belongs.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
TRACKING_STATUS	VARCHAR	12	The tracking status for the group. One of the following: ONTRACK – Group is on track to completion. OFFTRACK – Group is not on track to completion. NOTEVALUATED – Tracking status of the group has not yet been evaluated.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_PLAN_TEST_NOTE

Updated: September 30, 2022

This table contains notes related to a plan test requirement. There can be an unlimited number of notes. It is linked to the SEP_PLAN_TEST table by the TEST_ID column and the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
TEST_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
TEST_ID	CHAR	36	Reference to the TEST_ID column of the SEP_PLAN_TEST table to which this note belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
INTERNAL	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility.
COPIED_FROM_TEMPLATE	CHAR	1	"Y" or "N" indicating whether or not the note was originally copied from a template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_PLAN_NONCOURSE

Updated: September 30, 2022

This table records non-course requirements. It allows for the inclusion of non-course requirements like an honors paper or recital. There can be an unlimited number of requirements. It is linked to the SEP_PLAN table by the PLAN_ID column and the SEP_PLAN_GROUP table by the GROUP_ID column.

Item name	Data type	Maximum size	Description
NONCOURSE_ID	CHAR	36	The non-course requirement object ID, a unique identifier generated for this requirement.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this requirement is critical for tracking purposes.
NONCOURSE_CODE	VARCHAR	12	A coded value from UCX_SCR003 representing the type of non-course requirement. E.g. PIANO - Piano recital.
NONCOURSE_SCORE	VARCHAR	12	An optional score or status code, if applicable for the type of non-course requirement.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_PLAN_GROUP table to which this requirement belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this requirement belongs.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
TRACKING_STATUS	VARCHAR	12	The tracking status for the group. One of the following:

Item name	Data type	Maximum size	Description
			ONTRACK – Group is on track to completion.
			OFFTRACK – Group is not on track to completion.
			NOTEVALUATED – Tracking status of the group has not yet been evaluated.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_PLAN_NONCOURSE_NOTE

Updated: September 30, 2022

This contains notes related to a plan non-course requirement. There can be an unlimited number of notes. It is linked to the SEP_PLAN_NONCOURSE table by the NONCOURSE_ID column and the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
NONCOURSE_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
NONCOURSE_ID	CHAR	36	Reference to the NONCOURSE_ID column of the SEP_PLAN_NONCOURSE

Item name	Data type	Maximum size	Description
			table to which this note belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
INTERNAL	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility.
COPIED_FROM_TEMPLATE	CHAR	1	"Y" or "N" indicating whether or not the note was originally copied from a template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_PLAN_PLACEHOLDER

Updated: September 30, 2022

This table records placeholder requirements. These are catchall requirements that accept any free-form requirement the student or advisor may want to place on the plan. There can be an unlimited number of requirements. It is linked to the SEP_PLAN table by the PLAN_ID column and the SEP_PLAN_GROUP table by the GROUP_ID column.

Item named	Data types	Maximum size	Description
PLACEHOLDER_ID	CHAR	36	The placeholder requirement object ID, a unique identifier

Item named	Data types	Maximum size	Description
			generated for this requirement.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_PLAN_GROUP table to which this requirement belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this requirement belongs.
PLACEHOLDER_TYPE	VARCHAR	12	A coded value from UCX_SEP005 representing the type of placeholder requirement. For example, FORLANG – Foreign Language.
PLACEHOLDER_VALUE	VARCHAR	50	Free-form text describing the requirement.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_PLAN_PLACEHOLDER_NOTE

Updated: September 30, 2022

This table contains notes related to a plan placeholder requirement. There can be an unlimited number of notes. It is linked to the SEP_PLAN_PLACEHOLDER table by the PLACEHOLDER_ID column and the SEP_PLAN table by the PLAN_ID column.

Item name	Data type	Maximum size	Description
PLACEHOLDER_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
PLACEHOLDER_ID	CHAR	36	Reference to the PLACEHOLDER_ID column of the SEP_PLAN_PLACEHOLDER table to which this note belongs.
PLAN_ID	CHAR	36	Reference to the PLAN_ID column of the SEP_PLAN table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
INTERNAL	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility.
COPIED_FROM_TEMPLATE	CHAR	1	"Y" or "N" indicating whether or not the note was originally copied from a template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

Plan entity relation diagrams

Updated: March 24, 2023

Ellucian provides entity relationship diagrams to help you understand how key entities relate to one another.

Click Attachment  on this page to download the SEP templates diagram.

Template data structures

Updated: March 25, 2022

The template data structures are very similar to those for the plans. They lack several of the items in the plans and add a few others.

SEP_TMPL_MST

Updated: September 30, 2022

This table is the overall parent table for a template. There is only one entry in this table for every template.

Item name	Data type	Maximum size	Description
TMPL_MST_ID	CHAR	36	The template object ID, a unique identifier for the plan. It is generated by the system and generally not seen by users.
TEMPLATE_ID	CHAR	8	A convenient ID that is created by the user.
DESCRIPTION	VARCHAR	80	A free-format description of the template.
IS_ACTIVE	CHAR	1	"Y" or "N", indicating whether or not this is an active template.
TERM_SCHEME	VARCHAR	27	A coded value defined in UCX_SEP002. It defines a typical term sequence that should be used as the basis for a plan created by the template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the template.

Item name	Data type	Maximum size	Description
MODIFY_DATE	TIMESTAMP	7	The date and time the template was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the template.
CREATE_WHO	VARCHAR	14	The ID of the user who created the template.
CREATE_DATE	TIMESTAMP	7	The date the template was created.
CREATE_WHAT	VARCHAR	30	The process that created the template.

SEP_TMPL_NOTE

Updated: September 30, 2022

This table contains notes related to the overall template. There can be an unlimited number of notes tied to a template. It is linked to the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Date type	Maximum size	Description
TMPL_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
COPY_TO_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note should be copied to plans created from this template.
INTERNAL_ON_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility after it is copied to a plan created from this template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the plan.

Item name	Date type	Maximum size	Description
MODIFY_DATE	TIMESTAMP	7	The date and time the plan was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the plan.
CREATE_WHO	VARCHAR	14	The ID of the user who created the plan.
CREATE_DATE	TIMESTAMP	7	The date the plan was created.
CREATE_WHAT	VARCHAR	30	The process that created the plan.

SEP_TMPL_TAG

Updated: September 30, 2022

This table records information used to organize templates. It is used in the tree view and when creating plans for student in batch mode (DAP54). It is linked to the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data type	Maximum size	Description
TAG_ID	CHAR	36	The tag object ID, a unique identifier for the tag.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this tag belongs.
TAG_CODE	VARCHAR	12	A coded value from UCX_SEP001 representing the type of tag associated with the template. For example, "DEGREE" for degree codes.
TAG_VALUE	VARCHAR	12	A coded value from one of several UCX tables depending on the code recorded in TAG_CODE. It represents the tag value associated with the TAG_CODE. For example, "BA" for Bachelor of Arts

Item name	Data type	Maximum size	Description
			associated with the "DEGREE" tag code.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the tag.
MODIFY_DATE	TIMESTAMP	7	The date and time the tag was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the tag.
CREATE_WHO	VARCHAR	14	The ID of the user who created the tag.
CREATE_DATE	TIMESTAMP	7	The date the tag was created.
CREATE_WHAT	VARCHAR	30	The process that created the tag.

SEP_TMPL_TERM

Updated: September 30, 2022

This table records information about the term in which the requirements reside. There is one entry in this table for every term that contains requirements. It is linked to the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data type	Maximum size	Description
TERM_ID	CHAR	36	The term object ID, a unique identifier for the term.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this term belongs.
GROUP_ID	CHAR	36	The root group for requirements in this term. All requirements must be part of a group, and there is one group that is at the top of the hierarchy.
TERM_SEQ	DECIMAL	6	A number that order of the term within the plan.
DESCRIPTION	VARCHAR	80	A description for the term. This is initially

Item name	Data type	Maximum size	Description
			set from UCX_SEP002.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the term.
MODIFY_DATE	TIMESTAMP	7	The date and time the term was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the term.
CREATE_WHO	VARCHAR	14	The ID of the user who created the term.
CREATE_DATE	TIMESTAMP	7	The date the term was created.
CREATE_WHAT	VARCHAR	30	The process that created the term.

SEP_TMPL_TERM_NOTE

Updated: September 30, 2022

This table contains notes related to a template term. There can be an unlimited number of notes. It is linked to the SEP_TMPL_TERM table by the TERM_ID column and the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data type	Maximum size	Description
TERM_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
TERM_ID	CHAR	36	Reference to the TERM_ID column of the SEP_TMPL_TERM table to which this note belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
COPY_TO_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note should be copied

Item name	Data type	Maximum size	Description
			to plans created from this template.
INTERNAL_ON_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility after it is copied to a plan created from this template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_TMPL_GROUP

Updated: September 30, 2022

This table organizes and contains requirements. All requirements link to this table. It is linked to the SEP_TMPL_MST table by the TMPL_MST_ID column and the SEP_TMPL_TERM table by the TERM_ID column.

Item name	Data type	Maximum size	Description
GROUP_ID	CHAR	36	The group object ID, a unique identifier generated for the group.
TERM_ID	CHAR	36	Reference to the TERM_ID column of the SEP_TMPL_TERM table to which this group belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this group belongs.

Item name	Data type	Maximum size	Description
GROUP_ID_PARENT	CHAR	36	The parent group of this group. For the root group, this element will be the same as the GROUP_ID.
GROUP_TYPE	VARCHAR	2	A coded value representing the type of group. One of the following: UN – Union. All contained requirements must be fulfilled. CH – Choice. One of the contained requirements must be fulfilled.
SEQUENCE	DECIMAL	6	A number that records the order in which the user placed the group within the parent group together with other requirements.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this group is critical for tracking purposes.
CAMPUS	CHAR	12	A coded value from UCX_STU576 representing the campus where the student plans to take the courses listed in this group. For example, MA - Main.
DELIVERY	VARCHAR	10	A coded value from UCX_SEP003 representing the method of coursework delivery for the courses listed in this group. For example, SELF – Self-paced course.

Item name	Data type	Maximum size	Description
CREDITS	DECIMAL	6,3	The minimum total number of credits for all the courses in this group. Contains 3 whole digits and 3 decimal digits.
MINIMUM_GRADE	VARCHAR	6	The minimum grade that every course in this group must meet to fulfill the requirement.
SCRIBE_POINTER	VARCHAR	12	A coded value from UCX_SEP009 representing the audit requirement as written in Scribe that this group will fulfill.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the group.
MODIFY_DATE	TIMESTAMP	7	The date and time the group was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the group.
CREATE_WHO	VARCHAR	14	The ID of the user who created the group.
CREATE_DATE	TIMESTAMP	7	The date the group was created.
CREATE_WHAT	VARCHAR	30	The process that created the group.

SEP_TMPL_GROUP_NOTE

Updated: September 30, 2022

This table contains notes related to a group. There can be an unlimited number of notes. It is linked to the SEP_TMPL_GROUP table by the GROUP_ID column and the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data type	Maximum size	Description
GROUP_NOTE_ID	CHAR	36	The note object ID, a unique generated identifier for the note.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the

Item name	Data type	Maximum size	Description
			SEP_TMPL_GROUP table to which this note belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
COPY_TO_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note should be copied to plans created from this template.
INTERNAL_ON_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility after it is copied to a plan created from this template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_TMPL_GPA

Updated: September 30, 2022

This table records GPA requirements, which means that the student plans to achieve an overall, major, or class list GPA of at least a certain level. There can be an unlimited number of requirements. It is linked to the SEP_TMPL_MST table by the TMPL_MST_ID column and the SEP_TMPL_GROUP table by the GROUP_ID column.

Item name	Data type	Maximum size	Description
GPA_ID	CHAR	36	The GPA requirement object ID, a unique identifier generated for the GPA requirement.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_TMPL_GROUP table to which this requirement belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this requirement belongs.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this requirement is critical for tracking purposes.
GPA_TYPE	VARCHAR	10	A coded value from UCX_SEP008 representing the type of GPA being required. For example, MAJOR - Major GPA
MINIMUM_GPA	DECIMAL	23,9	The minimum GPA expected. Contains a maximum of 14 whole digits and 9 decimal digits.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
GPA_CODE	VARCHAR	12	A coded value from UCX_STU023 representing the major associated with this GPA requirement, only if the GpaType is "MAJOR". For example, ARTH - Art History.
CLASS_LIST	VARCHAR	4000	A list of classes used to evaluate the GPA

Item name	Data type	Maximum size	Description
			when the GpaType is "CLASS". For example, MATH 101 + MATH 102.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_TMPL_GPA_NOTE

Updated: September 30, 2022

This table contains notes related to a plan GPA requirement. There can be an unlimited number of notes. It is linked to the SEP_TMPL_GPA table by the GPA_ID column and the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data Type	Maximum size	Description
GPA_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
GPA_ID	CHAR	36	Reference to the GPA_ID column of the SEP_TMPL_GPA table to which this note belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.

Item name	Data Type	Maximum size	Description
AUTHOR	VARCHAR	14	The user ID of the note's author.
COPY_TO_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note should be copied to plans created from this template.
INTERNAL_ON_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility after it is copied to a plan created from this template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_TMPL_CLASS

Updated: September 30, 2022

This table records course requirements. These are planned courses a student expects to take. There can be an unlimited number of requirements. It is linked to the SEP_TMPL_MST table by the TMPL_MST_ID column and the SEP_TMPL_GROUP table by the GROUP_ID column.

Description	Data type	Maximum size	Description
CLASS_ID	CHAR	36	The course requirement object ID, a unique identifier generated for the requirement.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_TMPL_GROUP table to which this requirement belongs.

Description	Data type	Maximum size	Description
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this requirement belongs.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this requirement is critical for tracking purposes.
COURSE_DISCIPLINE	VARCHAR	12	The course discipline. For example, "MATH". Forms the first part of the course identifier (For example, "MATH 101").
COURSE_NUMBER	VARCHAR	12	The course number. For example, "101". Forms the second part of the course identifier (For example, "MATH 101").
COURSE_ATTRIBUTE	VARCHAR	12	A coded value from UCX_STU050 representing an attribute that might be attached to the course. For example, MAJORONLY - Major Only.
CAMPUS	CHAR	12	A coded value from UCX_STU576 representing the campus where the student plans to take the courses listed in this group. For example, MA - Main.
DELIVERY	VARCHAR	10	A coded value from UCX_SEP003 representing the method of coursework delivery for the courses listed in this group. For example, SELF – Self-paced course.

Description	Data type	Maximum size	Description
CREDITS	DECIMAL	7,3	The minimum total number of credits for all the courses in this group. Contains 3 whole digits and 3 decimal digits.
MINIMUM_GRADE	VARCHAR	6	The minimum grade that every course in this group must meet to fulfill the requirement.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_TMPL_CLASS_NOTE

Updated: September 30, 2022

This table contains notes related to a plan class requirement. There can be an unlimited number of notes. It is linked to the SEP_TMPL_CLASS table by the CLASS_ID column and the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data type	Maximum size	Description
CLASS_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.

Item name	Data type	Maximum size	Description
CLASS_ID	CHAR	36	Reference to the CLASS_ID column of the SEP_TMPL_CLASS table to which this note belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
COPY_TO_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note should be copied to plans created from this template.
INTERNAL_ON_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility after it is copied to a plan created from this template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_TMPL_TEST

Updated: September 30, 2022

This table records test requirements. This means the student is intending to pass a certain test with a minimum grade. There can be an unlimited number of requirements. It is linked to the

SEP_TMPL_MST table by the TMPL_MST_ID column and the SEP_TMPL_GROUP table by the GROUP_ID column.

Item name	Data type	Maximum size	Description
TEST_ID	CHAR	36	The test requirement object ID, a unique identifier generated for this requirement.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_TMPL_GROUP table to which this requirement belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this requirement belongs.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this requirement is critical for tracking purposes.
TEST_CODE	VARCHAR	10	A coded value from UCX_SEP006 representing the type of test being required. For example, ALEV_CHEM - A-Level Chemistry.
MINIMUM_SCORE	VARCHAR	6	The minimum test score expected.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.

Item name	Data type	Maximum size	Description
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_TMPL_TEST_NOTE

Updated: September 30, 2022

This table contains notes related to a plan test requirement. There can be an unlimited number of notes. It is linked to the SEP_TMPL_TEST table by the TEST_ID column and the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data type	Maximum size	Description
TEST_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
TEST_ID	CHAR	36	Reference to the TEST_ID column of the SEP_TMPL_TEST table to which this note belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
COPY_TO_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note should be copied to plans created from this template.
INTERNAL_ON_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility after it is copied to a plan

Item name	Data type	Maximum size	Description
			created from this template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_TMPL_NONCOURSE

Updated: September 30, 2022

This table records non-course requirements. It allows for the inclusion of non-course requirements like an honors paper or recital. There can be an unlimited number of requirements. It is linked to the SEP_TMPL_MST table by the TMPL_MST_ID column and the SEP_TMPL_GROUP table by the GROUP_ID column.

Item name	Data type	Maximum size	Description
NONCOURSE_ID	CHAR	36	The non-course requirement object ID, a unique identifier generated for this requirement.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_TMPL_GROUP table to which this requirement belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this requirement belongs.
IS_CRITICAL	CHAR	1	"Y" or "N", indicating whether or not this requirement is critical for tracking purposes.

Item name	Data type	Maximum size	Description
NONCOURSE_CODE	VARCHAR	12	A coded value from UCX_SCR003 representing the type of non-course requirement. For example, PIANO - Piano recital.
NONCOURSE_SCORE	VARCHAR	12	An optional score or status code, if applicable for the type of non-course requirement.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_TMPL_NONCOURSE_NOTE

Updated: September 30, 2022

This table contains notes related to a plan non-course requirement. There can be an unlimited number of notes. It is linked to the SEP_TMPL_NONCOURSE table by the NONCOURSE_ID column and the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data type	Maximum size	Description
NONCOURSE_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
NONCOURSE_ID	CHAR	36	Reference to the NONCOURSE_ID column of the SEP_TMPL_NONCOURSE table to which this note belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
COPY_TO_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note should be copied to plans created from this template.
INTERNAL_ON_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility after it is copied to a plan created from this template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

SEP_TMPL_PLACEHOLDER

Updated: September 30, 2022

This table records placeholder requirements. These are catchall requirements that accept any free-form requirement the student or advisor may want to place on the plan. There can be an unlimited number of requirements. It is linked to the SEP_TMPL_MST table by the TMPL_MST_ID column and the SEP_TMPL_GROUP table by the GROUP_ID column.

Item name	Data type	Maximum size	Description
PLACEHOLDER_ID	CHAR	36	The placeholder requirement object ID, a unique identifier generated for this requirement.
GROUP_ID	CHAR	36	Reference to the GROUP_ID column of the SEP_TMPL_GROUP table to which this requirement belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this requirement belongs.
PLACEHOLDER_TYPE	VARCHAR	12	A coded value from UCX_SEP005 representing the type of placeholder requirement. For example, FORLANG – Foreign Language.
PLACEHOLDER_VALUE	VARCHAR	50	Free-form text describing the requirement.
SEQUENCE	DECIMAL	6	A number that records the order in which the student placed the requirement within its parent group.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the requirement.
MODIFY_DATE	TIMESTAMP	7	The date and time the requirement was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the requirement.

Item name	Data type	Maximum size	Description
CREATE_WHO	VARCHAR	14	The ID of the user who created the requirement.
CREATE_DATE	TIMESTAMP	7	The date the requirement was created.
CREATE_WHAT	VARCHAR	30	The process that created the requirement.

SEP_TMPL_PLACEHOLDER_NOTE

Updated: September 30, 2022

This table contains notes related to a plan placeholder requirement. There can be an unlimited number of notes. It is linked to the SEP_TMPL_PLACEHOLDER table by the PLACEHOLDER_ID column and the SEP_TMPL_MST table by the TMPL_MST_ID column.

Item name	Data type	Maximum size	Description
PLACEHOLDER_NOTE_ID	CHAR	36	The note object ID, a unique identifier generated for the note.
PLACEHOLDER_ID	CHAR	36	Reference to the PLACEHOLDER_ID column of the SEP_TMPL_PLACEHOLDER table to which this note belongs.
TMPL_MST_ID	CHAR	36	Reference to the TMPL_MST_ID column of the SEP_TMPL_MST table to which this note belongs.
NOTE_TEXT	CLOB	unlimited	The text of the note.
AUTHOR	VARCHAR	14	The user ID of the note's author.
COPY_TO_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note should be copied to plans created from this template.
INTERNAL_ON_PLAN	CHAR	1	"Y" or "N" indicating whether or not the note has restricted visibility after it is copied to a plan

Item name	Data type	Maximum size	Description
			created from this template.
MODIFY_WHO	VARCHAR	14	The ID of the user who last modified the note.
MODIFY_DATE	TIMESTAMP	7	The date and time the note was last modified.
MODIFY_WHAT	VARCHAR	30	The process that last modified the note.
CREATE_WHO	VARCHAR	14	The ID of the user who created the note.
CREATE_DATE	TIMESTAMP	7	The date the note was created.
CREATE_WHAT	VARCHAR	30	The process that created the note.

Template entity relation diagram

Updated: March 24, 2023

Ellucian provides entity relationship diagrams to help you understand how key entities relate to one another.

Click Attachment  on this page to download the SEP templates diagram.

Requirements

Updated: March 25, 2022

Requirement blocks created by Scribe are saved in the dap_req_block table. All of the tags and text are saved in the one table. The text is saved in a CLOB field so special care is needed if you are to read the contents of that field.

DAP_REQ_BLOCK

Updated: September 30, 2022

Requirement blocks created by Scribe are saved in the dap_req_block table.

Field	Data type	Length	Description
REQUIREMENT_ID	VARCHAR2	8	Requirement block ID. RAnnnnnn for academic blocks and RBnnnnnn for blocks created by the planner.

Field	Data type	Length	Description
BLOCK_TYPE	VARCHAR2	12	Type of block. Valid in UCX-SCR004.
BLOCK_VALUE	VARCHAR2	12	Value of the block. Valid in the respective UCX table based on the type. Values for block types of OTHER do not exist in any UCX table and value for block types of ID must be valid on rad_student_mst.rad_id.
TITLE	VARCHAR2	50	The title of the block appearing on worksheets.
PERIOD_START	VARCHAR2	12	The starting catalog year. For REQUISITE blocks this is a starting term.
PERIOD_STOP	VARCHAR2	12	The ending catalog year. For REQUISITE blocks this is an ending term.
SCHOOL	VARCHAR2	12	Optionally the block can be tied to this school. Valid in UCX-STU350.
DEGREE	VARCHAR2	12	Optionally the block can be tied to this degree. Valid in UCX-STU307.
COLLEGE	VARCHAR2	12	Optionally the block can be tied to this college. Valid in UCX-STU560.
MAJOR1	VARCHAR2	12	Optionally the block can be tied to this major. Valid in UCX-STU023.
MAJOR2	VARCHAR2	12	Optionally the block can be tied to this major. Valid in UCX-STU023.
CONCENTRATION	VARCHAR2	12	Optionally the block can be tied to this

Field	Data type	Length	Description
			concentration. Valid in UCX-STU563.
MINOR	VARCHAR2	12	Optionally the block can be tied to this minor. Valid in UCX-STU024.
LIBERAL_LEARNING	VARCHAR2	12	Optionally the block can be tied to this liberal learning. Valid in UCX-STU324.
SPECIALIZATION	VARCHAR2	12	Optionally the block can be tied to this specialization. Valid in UCX-STU323.
PROGRAM	VARCHAR2	12	Optionally the block can be tied to this program. Valid in UCX-STU350.
STUDENT_ID	VARCHAR2	10	Optionally the block can be tied to this student ID. Valid on rad_student_mst.rad_id.
PARSE_STATUS	VARCHAR2	2	Valid in UCX-SCR007. OK means the last block parse was successful. NO means the last block parse resulted in errors.
PARSE_DATE	TIMESTAMP	7	Date the parse status changed or when changes were made to the block.
PARSE_WHO	VARCHAR2	14	ID of person who changed the block.
PARSE_WHAT	VARCHAR2	30	CONVERSION (from old Scribe) or blank
LOCK_VERSION	NUMBER	18	Used to ensure to users are not modifying the same block at the same time.
REQUIREMENT_TEXT	CLOB	unlimited	The text of the block saved as a CLOB field. Newline characters separate each line of text.

Field	Data type	Length	Description
CREATE_DATE	TIMESTAMP	7	The date the block was created.
CREATE_WHO	VARCHAR2	14	The ID of the person who created the block.
CREATE_WHAT	VARCHAR2	30	CONVERSION (from old Scribe) or SCRIBE
MODIFY_DATE	TIMESTAMP	7	The date the block was last modified.
MODIFY_WHO	VARCHAR2	14	The ID of the person who last modified the block.
MODIFY_WHAT	VARCHAR2	30	DAP36 or SCRIBE

DAP_REQ_CRS_DTL

Updated: March 25, 2022

When a block is parsed, the courses found in the block are saved to the dap_req_crs_dtl.

Field	Data type	Length	Description
DAP_REQ_ID	CHAR	8	Requirement block ID. Corresponds with dap_req_block.requirement_id.
DAP_DISCIPLINE	CHAR	12	Course discipline as it is scribed. This may contain the wildcard character - @.
DAP_NUMBER_BEGIN	CHAR	12	Course number as it is scribed. This may contain the wildcard character - @. If the course that was scribed is a range this will be the starting course number.
DAP_NUMBER_END	CHAR	12	Ending course number of a range. If the course is not a range this will be blank.

DAP_REQ_LINK_DTL

Updated: March 24, 2023

When a block is parsed, the OTHER blocks that are referenced are saved to the dap_req_link_dtl.

When an audit is run, the degree, major, minor, and so on blocks are included based on what is on the student's curriculum. For each of these blocks, the links in this table for the OTHER blocks they reference are then processed to know which OTHER blocks need to be included.

Field	Data type	Length	Description
DAP_REQ_ID	CHAR	8	Requirement block ID of the calling block. Corresponds with dap_req_block.requirement_id.
DAP_TYPE_FROM	CHAR	12	Block type of the calling block. Example, DEGREE. Corresponds with dap_req_block.block_type.
DAP_VALUE_FROM	CHAR	12	Block value of the calling block. Example, BS. Corresponds with dap_req_block.block_value.
DAP_TYPE_TO	CHAR	12	Block type of the referenced block. This is always OTHER. (additional values may be in this field but only the OTHER type is actually used) Corresponds with dap_req_block.block_type.
DAP_VALUE_TO	CHAR	12	Block value of the referenced block. Example, GENED. Corresponds with dap_req_block.block_value.

Exceptions

Updated: September 30, 2022

Exceptions are applied to audits and saved in the dap_except_dtl table. You can use the Exceptions Report in Exception Management to review these exceptions, or you can use your own tools to report against this data.

When an exception is added, the node-id of the rule or qualifier on which the exception was placed is saved to the dap_node_id and the label tag or qualifier tag is saved to the dap_label_tag. When subsequent changes are made to the block where the exception was placed are made, the label tag is used to locate the new node-id where the exception should apply. If no label tag exists then the node-id is looked up on the old block and then that location it compared to the same location on the new block to find the new node-id with the help of the label text on the requirement. The bottom line is that without the label or qualifier tag in place the system may not be able to locate

the correct requirement when block changes are made in Scribe. For more information, see the [Label Tags](#) topic.

DAP_EXCEPT_DTL

Updated: September 30, 2022

Exceptions applied to audits are saved in the dap_except_dtl.

Field	Data type	Length	Description
DAP_STU_ID	CHAR	10	Student ID
DAP_SCHOOL	CHAR	12	Valid in UCX-STU350. Example: UG. Exceptions may be tied to the school by the UCX-CFG020 EXCEPTIONS Exceptions Tied to School flag.
DAP_DEGREE	CHAR	12	Valid in UCX-STU307. Example: BS. Exceptions may be tied to the degree by the UCX-CFG020 EXCEPTIONS Exceptions Tied to Degree flag.
DAP_EXC_TYPE	CHAR	2	Valid in UCX-AUD014. Example, AH
DAP_EXC_STATUS	CHAR	2	Valid in UCX-AUD015. Example, RG.
DAP_REQ_ID	CHAR	8	Requirement/Block ID where exception is applied. This will be RA000000 if it is global exception, applying to all blocks in the student's audit.
DAP_NODE_ID	NUMBER	6	Points to a particular requirement. Example, 167.
DAP_NODE_TYPE	NUMBER	4	Obsolete
DAP_LABEL_TAG	CHAR	20	Scribed label tag or qualifier tag where exception was placed. Example, CHEMELC
DAP_EXC_REQ_OLD	CHAR	40	Obsolete

Field	Data type	Length	Description
DAP_EXC_CLUSTER	CHAR	4	Obsolete
DAP_EXC_BUFFER	CHAR	80	The course and credit information for the exception. See DAP_EXC_BUFFER .
DAP_EXC_LABEL	CHAR	50	Short description of exception.
DAP_EXC_REMARK	CHAR	72	WITH data for the course. See DAP_EXC_REMARK .
DAP_EXC_DETAILS	CHAR	220	Long description of exception often explaining the reason behind the description.
DAP_NOTE_NUM	NUMBER	4	Obsolete
DAP_APPLY_STATUS	CHAR	2	OK if applied or some other code signifying why the exceptions was not applied. See DAP_APPLY_STATUS .
DAP_CREATE_DATE	TIMESTAMP	7	Date the exception was created
DAP_CREATE_WHO	CHAR	14	ID of user who created the exception
DAP_CREATE_ID	CHAR	14	ID of user who created the exception
DAP_USER_LAST	CHAR	4	user class of the user who last modified the exception
DAP_MOD_DATE	DATE	7	Date the exception was last modified
DAP_WHO	CHAR	14	ID of user who last modified the exception
DAP_MOD_ID	CHAR	14	ID of user who last modified the exception
DAP_WHAT	CHAR	8	Name of routine that modified the exception. Usually DAP44
DAP_TIMESTAMP	TIMESTAMP	7	Date the exception was last modified.
DAP_EXC_NUM	NUMBER	4	Sequence number for the exceptions added for this student.

DAP_EXC_BUFFER

The dap_exc_buffer field has data at different offsets based on the exception type. The first 23 bytes are not used.

Apply here

Field	Offset	Length
Discipline	24	12
Course Number	36	12

Also allow

Field	Offset	Length
Discipline	24	12
Course Number	36	12

Substitution (aka replace requirement)

Field	Offset	Length
Old Discipline	24	12
Old Number	36	12
New Discipline	48	12
New Number	60	12

Not needed/change the limit

Discipline	24	12
Course Number	36	12
Classes/Credits	48	7
Type (CR or CL)	55	2

DAP_EXC_REMARK

For the Apply Here, Also Allow, and Substitute exceptions, there might be WITH data in the dap_exc_remark field also.

Field	Offset	Length
"WITH:"	1	5
With Code	6	12
With Operator	18	2
With Value	20	30

DAP_APPLY_STATUS

Status	Description
OK	Applying to requirement
UN	Unhooked – caused by a scribe change; location of requirement could not be determined
BK	Block was not found in the audit - change of major maybe?
CR	Requirement is not a course rule – exception and requirement no longer match up
CO	The course to be substituted was not found on the requirement
IF	IF-statement was resolved and the rule where the exception was to apply is not being used
ND	Node-id not found in block - changes to rules maybe? Should not occur if you have label tags in place
PA	Rule is a plus-list but Also Allow exceptions are not allowed on these types of rules
PN	Rule is a plus-list and the course is not on the rule so the Also Allow or Apply Here exception cannot be used
R1	Not enforced but we don't know why; this should never happen; this signifies a coding problem perhaps
R2	Not enforced and an unknown error code was found; this should never happen; this signifies a coding problem perhaps

Notes

Updated: March 24, 2023

Academic advising notes are added to student records using Notes or Petitions in the tools menus. Both types of notes are saved to the dap_note_dtl and dap_note_txt_dtl.

DAP_NOTE_DTL

Updated: September 29, 2023

The DAP_NOTE_DTL is the primary record for an academic audit note.

Field	Data type	Length	Description
DAP_STU_ID	CHAR	10	Student ID
DAP SCHOOL	CHAR	12	Valid in UCX-STU350. Example: UG. Notes may be tied to the school by the UCX-CFG020 WEB Notes tied to School flag.
DAP_DEGREE	CHAR	12	Valid in UCX-STU307. Example: BS. Notes may be tied to the degree by the UCX-CFG020 WEB Notes tied to Degree flag.
DAP_NOTE_TYPE	CHAR	2	Valid in UCX-SCR008. Petitions have a type of PE and normal notes have some other value. For normal notes, a flag on the SCR008 record indicates that this is an internal note.
DAP_NOTE_STATUS	CHAR	2	Notes have a status of OK, while petitions can have several different values because of the approval process: <ul style="list-style-type: none"> • PA - Petition Approved • PP - Petition Applied • PR - Petition Rejected • PW - Petition Awaiting
DAP_EXC_NUM	NUMBER	4	Obsolete
DAP_CREATE_DATE	TIMESTAMP	7	Date the note was created
DAP_CREATE_WHO	CHAR	14	ID of user who created the note
DAP_CREATE_ID	CHAR	14	ID of user who created the note

Field	Data type	Length	Description
DAP_USER_LAST	CHAR	4	user class of the user who last modified the note
DAP_MOD_DATE	TIMESTAMP	7	Date the note was last modified
DAP_WHO	CHAR	14	ID of user who last modified the note
DAP_MOD_ID	CHAR	14	ID of user who last modified the note
DAP_WHAT	CHAR	8	Name of routine that modified the note. Usually DAP44
DAP_TIMESTAMP	TIMESTAMP	7	Date the note was last modified.
DAP_NOTE_NUM	NUMBER	4	Sequence number for the notes added for this student. Together with the dap_stu_id this field is used to find the child dap_note_txt_dtl records.

DAP_NOTE_TXT_DTL

Updated: March 24, 2023

The DAP_NOTE_TXT_DTL is linked to the DAP_NOTE_DTL and contains the note text.

Field	Data type	Length	Description
DAP_STU_ID	CHAR	10	Student ID
DAP_NOTE_TEXT	CHAR	72	The note text. The complete note text is broken up into 72 byte chunks and saved on separate records. No carriage-return or newline characters are saved here.
DAP_NOTE_NUM	NUMBER	4	The note number corresponding to the dap_note_dtl.dap_note_num.
DAP_NOTE_SEQ	NUMBER	4	The sequence of this text record for the overall note.

Banner Integration

Updated: March 25, 2022

Review the processes and procedures users must follow to integrate Banner and Degree Works.

Banner attributes

Banner class attributes

Updated: March 25, 2022

Banner class attributes are codes that are not part of the standard database tables passed for class records (current, historic, and transfer) from the student system to Degree Works.

These class attributes are stored in the following Banner database tables and are retrieved using different pieces of a student's class data:

- SSRATTR - current classes by Crn and Term
- SHRATTR - historic classes by PIDM, Sequence Numbers and Effective Term
- SHRATTC - historic classes by Crn and Effective Term
- SHRTATT - transfer classes by PIDM and Sequence Numbers

Historic class attributes are stored in two tables: SHRATTR and SHRATTC. A UCX-CFG020 BANNER flag Always Process SHRATTC controls whether attributes from SHRATTC are extracted if attributes from both tables exist. SHRATTR class attributes are processed first. If any SHRATTR attributes are found for a given class they are written to the rad_attr_dtl. Then the UCX-CFG020 BANNER Always Process SHRATTC flag is checked:

- N - the SHRATTC attributes will be skipped for a given class if attributes from SHRATTR already exist for that class. This is the default value if this flag is left BLANK.
- Y - the SHRATTC class attributes will always be processed and written to the rad_attr_dtl if found for a given class, even if SHRATTR attributes exist for that class. In this case class attributes from both the SHRATTR and SHRATTC Banner tables would be written to the rad_attr_dtl.

These values are available for use with the WITH keyword in Scribe and a single entry of ATTRIBUTE must be defined in UCX-SCR044. Class Attributes may be used where a requirement varies based on class data. For example, classes in the Honors program might have an HONR Class Attribute assigned to each Honors class. If a student is in the Honors program and must complete 5 credits in Honors English, then the requirement could be written as follows:

5 Credits in ENGL @ (With Attribute = HONR)

The above requirement will only work properly if the ATTRIBUTE is added to UCX-SCR044 with the ATTR Element assigned. The ATTR tells Degree Works that this item is defined in the rad_attr_dtl. Be sure to set the Offset and Length fields to 00.

Transfer classes applied to program

Updated: September 30, 2022

In addition to class attributes generated from the standard Banner attribute tables, it is also possible to generate a rad_attr_dtl record for Transfer Classes that have been applied to a program in Banner.

Such classes are identified by linking the Banner Transfer Equivalence record, SHTRCE, to the Transfer Course Degree Applied table SHTRCD where SHTRCD_APPLIED_IND = Y. The rad_attr_value will be populated with the student's Program Code, which is extracted from the associated Degree table SHRDGMR. To enable this feature, set the UCX-CFG020BANNER **Transfer Program Attr** flag to Y.

You could then scribe the Program value in the related requirement blocks so that the Transfer Classes are applied appropriately. For instance, "do not accept any transfer classes for this requirement that are not applied to the AA program (coded attribute of PRAA)" could be scribed as:

MaxClasses 0 in @ (With DWTransfer = Y and Attribute <> PRAA)

Current/history classes applied to program

Updated: September 30, 2022

In addition to class attributes generated from the standard Banner attribute tables, it is also possible to generate rad_attr_dtl records for current classes (SFRSTCR) and historic classes (SHRTCKN) that have been applied to a program in Banner.

You can enable this feature by setting the UCX-CFG020BANNER **Curt/Hist Program Attr** flag to Y.

Current classes are identified by linking the Banner Current record, SFRSTCR, to the SGBSTDN record that has a TERM_CODE_EFF less than or equal to the SFRSTCR_TERM_CODE. The SGBSTDN_PROGRAM_1 code from that record is used to populate the RAD attribute record (as the rad_attr_value on the rad_attr_dtl).

Historic classes are identified by linking the Banner Historic record, SHRTCKN, to the Institutional Course Term Degree Applied Repeating Table, SHRTCKD, where SHRTCKD_APPLIED_IND is not null. The SHRTCKD_DGMR_SEQ_NO is used to lookup the correct record from the associated Degree table, SHRDGMR. The SHGRDGMR_PROGRAM code from this record is used to populate the RAD attribute record (as the rad_attr_value on the rad_attr_dtl).

Example 1

List of rad_attr_dtl records for a given student:

rad_attr_key	rad_attr_code	rad_attr_value
SWK430-016	ATTRIBUTE	15SH
LAW211-017	ATTRIBUTE	15SH
WEL116-018	ATTRIBUTE	15SH
HRM540-019	ATTRIBUTE	20AA
LAW516-020	ATTRIBUTE	20AA
POL01C-021	ATTRIBUTE	15SW
POL01C-022	ATTRIBUTE	15SW
PSY01C-023	ATTRIBUTE	15SW

Use Scribe to modify the requirement blocks that need to restrict classes that qualify by using the Program attribute values so that the current and historic classes are applied appropriately. For instance, "do not accept any classes for this requirement that are not applied to the 15SW program" could be scribed as:

```
MaxClasses 0 in ☐ (With Attribute <> 15SW )
```

The last three classes listed above (POL01C-021, POL01C-022, and PSY01C-023) are the only records that could be used to satisfy the requirements for a block with this rule.

Example 2

Sample block for a Theology Degree – only classes with an attribute of 11TH will be applied (other rules like DWGRADE must be satisfied as well):

```
##DEGREE=1160TH
##Diploma in Theology
##2008-9999

BEGIN

128 Credits
    Proxy-Advice "128 points are required. You have either completed
"
    Proxy-Advice "or enrolled in <APPLIED>; "
    Proxy-Advice "you still need <NEEDED> more points."

MaxCredits 64 in ☐ (WITH DWCREDITTYPE=TR)

MinRes 64 Credits
    Proxy-Advice "A minimum of 64 points must be studied in this course
at CSU."

MaxClasses 0 in ☐ (WITH DWGRADE=US)
MaxClasses 0 in ☐ (WITH DWGRADE=FL)
MaxClasses 0 in ☐ (WITH DWGRADE=FW)
MaxClasses 0 in ☐ (WITH DWGRADE=AW)

MaxClasses 0 in SSS ☐
```

```
MaxClasses 0 in XLV @  
MaxClasses 0 in @ (WITH ATTRIBUTE <> 11TH)
```

```
;
```

Banner course attributes

Updated: September 30, 2022

Banner course attributes are codes that are not part of the standard database tables passed for course records from the student system to Degree Works.

These course attributes are stored in the SCRATTR Banner database table and are retrieved when courses are being pulled into Degree Works. These attributes are stored in the rad-crs-attr-dtl linked from the rad-course-mst by the course key. When a planner audit is performed the attributes associated with each course in the plan is sent to the auditor in case they are needed to satisfy requirements using WITH Attribute=.

Banner student attributes

Updated: September 30, 2022

Banner student attributes are codes that are not part of the standard database tables passed for class records (current, historic, and transfer) from the student system to Degree Works.

These student attributes are stored in the SGRSATT Banner database table. These attributes are retrieved using the student's PIDM. These values are available for use in If-statements in Scribe and may be used to control what appears in the student header on the audit worksheets. Student Attributes may be used where a requirement varies based on the presence or absence of an attribute. For example, students in the Honors program might have to take an additional set of classes. The Student Attribute code of HONR, for example, will be pulled from Banner into Degree Works and will be used to control what requirements the student must meet. Such a requirement might be written as follows:

```
If (Attribute = HONR) then  
 15 Credits in ENGL 4@  
  Label "15 upper-division credits required for honor students";
```

All Banner Student Attributes are placed into the rad-custom-dtl in Degree Works with a custom-code of ATTRIBUTE and a custom-value of the attribute code, such as HONR.

UCX-SCR002 must contain an ATTRIBUTE entry telling Degree Works to retrieve all rad-custom-dtl ATTRIBUTE records and send them to the auditor. Create this entry if it does not already exist:

Field	Value
Description	Banner Student Attributes
Data Element	R323

Field	Value
Edit Element 1	R322
Edit Type 1	EV (Edit Value 1)
Edit Value 1	ATTRIBUTE

Banner degree coding structure

Updated: March 25, 2022

For each student, Degree Works creates a degree record for each active SORLCUR record it finds with a unique sorlcur_degc_code.

One or more SORLFOS records are linked back to the associated SORLCUR(s) by the lcur_seq_no. Each SORLFOS major/minor/concentration is placed on this degree record. Each degree record built in Degree Works will result in a discrete degree audit.

Note that Degree Works does not use the sorlcur_program field to identify and extract the student's degree-major combination. In Degree Works, the degree is specifically taken from sorlcur_degc_code and the major is specifically taken from the associated SORLFOS record(s). However, the sorlcur_program is extracted and stored separately and can be referenced with Scribe to create a Program Requirement block, or differentiate between degree-major combinations by using the Program as a secondary tag.

Banner degree coding structure examples

One degree - two majors

Definition of degree and location of data in Banner	Results in Degree Works
SORLCUR - sorlcur_degc_code =BS; seq-no=1	This will result in one degree record in Degree Works - a BS degree with two majors. The degree audit will be run against this degree with two majors sharing or not sharing classes based on the requirements.
SORLFOS - sorlfos_majr_code =CHEM; lcur_seq_no=1	
SORLFOS - sorlfos_majr_code =BIOL; lcur_seq_no=1	

Two degrees of different type (same level) - two majors

Definition of degree and location of data in Banner	Results in Degree Works
SORLCUR - sorlcur_degc_code=BS; seq-no=1	This will result in one degree record in Degree Works - a BS degree with two majors. The degree audit will be run against this degree with two majors sharing or not sharing classes based on the requirements.
SORLFOS - sorlfos_majr_code =MATH; lcur_seq_no=1	
SORLCUR - sorlcur_degc_code =BS; seq-no=2	
SORLFOS - sorlfos_majr_code =PHYS; lcur_seq_no=2	

Two different degrees (same level) - two majors

Definition of degree and location of data in Banner	Results in Degree Works
SORLCUR - sorlcur_degc_code =BS; seq-no=1	This will result in two degree records in Degree Works - a BS degree with a CHEM major and a BA degree with an ARTH major. Two discrete degree audits will be produced - all classes will be applied to both sets of degrees/majors regardless of sharing policies.
SORLFOS - sorlfos_majr_code =CHEM; lcur_seq_no=1	
SORLCUR - sorlcur_degc_code =BA; seq-no=2	
SORLFOS - sorlfos_majr_code =ARTH; lcur_seq_no=2	

Two degrees (different level) - two majors

Definition of degree and location of data in Banner	Results in Degree Works
SORLCUR - sorlcur_degc_code =BS; level=UG; seq-no=1	This will result in two degree records in Degree Works - a BS degree with a CHEM major and an MA degree with a PHIL major. Two discrete degree audits will be produced. The undergraduate classes will be applied to the BS degree/major and the graduate classes will be applied to the MA/PHIL degree/major.
SORLFOS - sorlfos_majr_code =CHEM; lcur_seq_no=1	
SORLCUR - sorlcur_degc_code =MA; level=GR; seq-no=2	
SORLFOS - sorlfos_majr_code =PHIL; lcur_seq_no=2	You can change the configuration flag to allow all classes to apply to both degrees thus ignoring the level filter that is in place by default.

Two different BS degrees (same level) - two majors - not recommended

If concurrent curriculum is not used for the student being processed, this data will be extracted from the fixed columns found on the SGBSTDN table.

Definition of degree and location of data in Banner	Results in Degree Works
SORLCUR - sorlcur_degc_code=BSMATH; seq-no=1	This will result in two degree records in Degree Works - a BSMATH degree with a MATH major and a BSPHYS degree with a PHYS major.
SORLFOS - sorlfos_majr_code =MATH; lcur_seq_no=1	Two discrete degree audits will be produced - all classes will be applied to both sets of degrees/majors regardless of sharing policies. This is not a recommended approach. It is best to stick with Example A and link both majors to a single BS degree record.
SORLCUR - sorlcur_degc_code =BSPHYS; seq-no=2	Students normally cannot double-count all classes between the two majors and the only way to prevent the double-counting is to link them to the same degree.
SORLFOS - sorlfos_majr_code =PHYS; lcur_seq_no=2	

One BS degree (same level) - with two different programs and program as Degree = Y

Definition of degree and location of data in Banner	Results in Degree Works
SORLCUR - sorlcur_degc_code=BS; program=ANTH	This will result in two degree records in Degree Works - a rad_goal_dtl with degree=ANTH another with degree=CHEM. You will also see two rad_goalData_dtl records with goal-code=PROGRAM and goal-value=BS but one will have degree=ANTH and the other will have degree=CHEM.
SORLCUR - sorlcur_degc_code=BS; program=CHEM	Two discrete degree audits will be produced - all classes will be applied to both sets of degrees/majors regardless of sharing policies. Prevent the double-counting is to link them to the same degree.

Required access to Banner

Updated: September 30, 2022

Required access consists of access to your Banner database and the f_class_calc_fnc function.

Database table access

Updated: September 30, 2022

Access to your Banner database is required for the extract process, therefore read access must be provided to specific tables.

Two scripts, bannergrants.sql and bannergrantsverify.sql, can be helpful to establish or verify access to these tables for the Degree Works user created in your Banner database. Both scripts reside in \$DGWHOME/sql. The bannergrants script must be run by a Banner user with DBA privileges, and bannergrantsverify must be run by the Degree Works user through the dbb command.

Note: If your site uses the custom SQL in the UCX-BAN080 table to extract non-standard pieces of data from Banner, those database tables will NOT be listed here. However, SELECT access must be provided for those tables as well.

The SAP (Student Academic Progress) Processor BAN62 requires SELECT and INSERT access to tables SHRSAPP and SHRSARJ. In addition, SELECT access is required for SHRSAPP_SEQUENCE and SHRSARJ_SEQUENCE.

Banner table	Extract	Description
GOBUMAP	Advisors/Staff	UDC_ID/SPRIDEN_PIDM Mapping
	Students	
	Applicants	
GORADID	Advisors/Staff	ADDITIONAL_ID – moved to shp_user_mst
	Students	
	Applicants	
GOREMAL	Advisors/Staff	Email Address Data
	Students	
	Applicants	
SARADAP	Applicants	Applicant Degree Data
SCBCRKY	Course Equivalents	Course Start/End Dates
SCBCRSE	Course Catalog	Course Master Data
	Course Equivalents	
SCBDESC	Course Catalog	Course Description
SCRATTR	Course Catalog	Course Attributes

Banner table	Extract	Description
SCRLEVL	Course Catalog	Course School Attributes (DW-SCHOOL)
SCREQIV	Course Equivalents	Course Equivalents
SCRRTST	Course Catalog	Course Prerequisites
SCRTEXT	Course Equivalents	Course Description
SFRSTCR	Students	Current Class Data
	Applicants	
SGBSTDN	Students	General Student Degree Data
	Applicants	
SGRADVR	Advisors/Staff	Advisor Data
	Students	
	Applicants	
SGRSATT	Students	Student Attributes by PIDM and Current Term
	Applicants	
SGRATHE	Students	Student Athlete – first date of attendance for athletic eligibility
SGRCATT	Students	Class Attribute Repeating Table
SGRCLSR	Students	Student Classification Table
SHBRPTS	Students	Title Indicator
SHBTATC	Transfer Equivalency Admin/ Self-Service	Transfer Institution Transfer Catalog Data
SHRATTC	Students	Student Attributes by CRN and Historic Term
	Applicants	
SHRATTR	Students	Student Attributes by PIDM and Historic Sequence Number
	Applicants	
SHRDGMR	Students	Student Degree table
SHRGRDE	Students	Grade Codes (UCX-STU385)
	Applicants	Grade Types (UCX-STU356)

Banner table	Extract	Description
	UCX	
SHRGRDO	Students	Grade Codes - valid combo of level/gmod/grade (UCX-STU385) Grade Types (UCX-STU356)
	Applicants	
	UCX	
SHRICMT	Transfer Equivalency Admin/ Self-Service	Transfer Articulation Institution Course Comment
SHRLGPA	Students	Summary GPA/Credits Data
	Applicants	
SHRNCRS	Students	Non Course Data
	Applicants	
SHRQPNM	Students	Non Course Data - Papers and Exams
	Applicants	
SHRTATC	Transfer Equivalency Admin/ Self-Service	Transfer Institution Catalog Equivalent Data
SHRTATT	Students	Student Attributes by PIDM and Transfer Sequence Number
	Applicants	
SHRTCKD	Students	Institutional Course Term Degree Applied Repeating Table
	Applicants	
SHRTCKG	Students	Historic Grade Values
	Applicants	
SHRTCKL	Students	Historic Level (School) Data
	Applicants	
SHRTCKN	Students	Historic Class Data
	Applicants	
SHRTGPA	Students	Detail GPA/Credits by Term

Banner table	Extract	Description
	Applicants	
SHTRCD	Students	Transfer Course Degree Applied table
SHTRCE	Students	Transfer Equivalent Data
	Applicants	
SHTRCR	Students	Transfer Class Data
	Applicants	
SHTRIT	Students	Transfer School Data
	Applicants	
SHTRAT	Transfer Equivalency Admin/ Self-Service	Transfer Attributes
SOBCACT	Students	SORLCUR active indicator validation
	Applicants	
SOBCURR	Curriculum Rules	Curriculum Rules Base Table
SOBSBGI	ETS (transfer)	ETS Address Data
SORBTAG	ETS (transfer)	ETS Calendar Data
SORCCON	Curriculum Rules	Curriculum Rules Concentration Data
SORCMJR	Curriculum Rules	Curriculum Rules Major Data
SORCMNR	Curriculum Rules	Curriculum Rules Minor Data
SORDEGR	Students	Previous Degree Data
	Applicants	
SORLCUR	Students	Concurrent Degree Data
	Applicants	
SORLFOS	Students	Field of Study Data
	Applicants	
SORMCRL	Curriculum Rules	Curriculum Rules Control Table
SORTEST	Students	Test Score Data

Banner table	Extract	Description
	Applicants	
SMRPRLE	Students	Program codes
	Applicants	
SPRIDEN	Advisors/Staff	Primary Name Data
	Students	
	Applicants	
SSBSECT	Students	Schedule Master Data
	Applicants	
SSBXLST	Course Catalog	Cross Listing Table – seat count
SSRATTR	Students	Student Attributes by CRN and Current Term
	Applicants	
SSRMEET	Course Catalog	Section meeting times
SSRXLST	Course Catalog	Cross Listing table - grouping
STVACCL	UCX	Calendar Codes (UCX-STU346) used by Transfer Equivalency Self-Service
STVACYR		Catalog Year Codes (UCX-STU035)
STVATTR	Course Catalog	Attribute Codes (UCX-STU050) and used by Course Link
	UCX	
STVCLAS	UCX	Student Level Codes (UCX-STU305)
STVCOLL	UCX	College Codes (UCX-STU560)
STVCSTA	Course Catalog	Course Status Codes
	Course Equivalents	
	Curriculum Rules	
STVDEGC	UCX	Degree Codes (UCX-STU307)
STVGMOD	UCX	Grade Types (UCX-STU356)
STVLEVL	UCX	School Codes (UCX-STU350)

Banner table	Extract	Description
STVMAJR	UCX	Major Codes (UCX-STU023), Minor Codes (UCX-STU024), Concentration Codes (UCX-STU563)
STVNCRQ	Students Applicants	Non Course Codes
STVCNST	Students Applicants	Non Course Status Codes
STVQPTP	Students Applicants	Non Course Exam/Paper Codes
STVRSTS	Students Applicants	Course Registration Status
STVSBGI	ETS (transfer) Students Applicants UCX	ETS School Names
STVSSTS	Course Catalog	Course section status (active indicator)
STVSTST	Students Applicants	Student Status (DW student type)
STVSTYP	Students Applicants UCX	Student Type (DW Student Status Codes UCX-STU306)
STVSUBJ	UCX	Discipline Codes (UCX-STU352)
STVTAST	Transfer Equivalency Admin/ Self-Service	Transfer Articulation Course Status
STVTERM	Course Catalog	Term Codes (UCX-STU016)

Banner table	Extract	Description
	Course Equivalents	
	Curriculum Rules	
	Students	
	Applicants	
	UCX	
SURVERS	Banner General	Version of Oracle being used
TWGBWSES	Banner General	Banner Self-Service

Transfer Equivalency Customers: If you have licensed Transfer Equivalency, write access is required for the following Banner tables:

Banner Table	Description
SHRTRIT	Transfer Institution
SHRTRAM	Attendance Period by Transfer Institution
SHRTRTK	Transfer Institution Transfer Course Taken

Function access

Updated: September 30, 2022

The f_class_calc_fnc function must be made available so that student class levels can be calculated.

If you are using Banner Self-Service single sign-on for Degree Works, grant execute on the following packages/procedures to the DB_LOGIN_BANNER user. The DB_LOGIN_BANNER user must also be granted create public synonym and drop public synonym privileges to install dwssbfaculty.sql and dwssbstudent.sql.

- TWBKWBIS
- TWBKFRMT
- BWLKOIDS
- BWLKOSTM
- BWCKFRMT
- BWLKILIB
- BWCKLIBS

Satisfactory academic progress

Updated: September 30, 2022

Banner clients can monitor the academic progress of each student who applies for federal financial assistance and certify that students are making satisfactory academic progress (SAP) towards earning their degrees.

You can use results from Degree Works degree audits to determine whether a student is successfully completing coursework for a degree and remains eligible for federal financial aid.

The Banner Satisfactory Academic Progress Review (SHISAPP) page, the Satisfactory Academic Progress Course Data Table (SHRSAPP), and the Satisfactory Academic Progress Course Rejection Reason Table (SHRSARJ) are used with Degree Works degree audit processing to capture the data used for checking satisfactory academic progress. A Degree Works processor, BAN62, will extract course information from a student's degree audit and store it in the Banner database.

BAN62 is run from Transit and populates the Banner tables SHRSAPP and SHRSARJ. The user can use selection criteria to define a pool of students to process. The user has the option to refresh the student data and run a new degree audit, or to run BAN62 on existing degree audits. In addition, the user will have the option of freezing the new audits for future reference. For more information, see the [Run BAN62 - Banner SAP Processor](#) topic.

Before running BAN62, check the following items:

- The **UCX-CFG020DAP14 Calculate Elective Credits Allowed** flag must be set to Y for SAP processing.
- SELECT and INSERT access for tables SHRSAPP and SHRSARJ must be granted to your Degree Works user in your Banner database.
- SELECT access for SHRSAPP_SEQUENCE and SHRSARJ_SEQUENCE must be granted to your Degree Works user in your Banner database.
- A new freeze status must be added to UCX-AUD032 if you want to freeze audits that are generated for input into the BAN62 processor.

When the BAN62 processor completes its operations, every course from the most recent degree audit will have an entry written to the Banner table SHRSAPP. If a course did not meet a degree requirement, in addition to SHRSAPP, an entry with the rejection reason will be written to SHRSARJ. Maintenance of the SHRSAPP and SHRSARJ tables is performed through Banner's SAP Purge Processor SMPCSAP.

Dual degree students

- If the **UCX-CFG020 DAP14 Filter Classes by School** flag is set to N, then any class the student takes will be part of both of the student's audits if the student is a dual degree student.
- If the **UCX_CFG020 BANNER SAP All Degrees** flag is set to Y, this means that two sets of SAP records will be created in Banner for each audit, one for each Degree. If this flag is N or

blank, then SAP records will only be created for the Primary Degree with the lowest SORLCUR_PRIORITY_NO.

SAP audit freeze

It is recommended to freeze audits that are generated for use with BAN62. A new UCX-AUD032 freeze status of SAPFRZ has been added with the 4.1.1 update, and should be available in the Transit freeze-type picklist. You may want to create other freeze types, such as SAPFAL for Fall terms, SAPSPG for spring terms, or SAPWNT for winter terms. For more information about adding and configuring entries in UCX-AUD032, see the [UCX-AUD032 Audit Freeze Types](#) topic.

Multiple-entity processing

If your Banner database is configured for Multiple-Entity Processing (MEP), be sure to review configuring your Degree Works environment's VPDI code. For more information, see the [Enable multiple-entity processing for Banner](#) topic. If a VPDI code is set in Degree Works, it will be written to the SHRSAPP and SHRSARJ tables in Banner.

Repeated classes

No special processing is required for repeated classes. Each occurrence of a repeated class should receive an appropriate reject reason if it ends up in the insufficient section of the audit. The GPA credits and GPA grade points are also returned for each repeated class. See the [SHRSAPP table layout](#) topic, and the associated columns SHRSAPP_HOURS_GPA and SHRSAPP_QUALITY_POINTS to determine how the SHRSAPP_GPA value is calculated for a repeated course.

Split credits

When a class is split between two or more requirement blocks, only one instance of the class is processed by BAN62. However if a class is split between a requirement and Over-the-Limit or Fall-through, then two instances of the class will be processed by BAN62, generating two distinct entries in SHRSAPP. The course from Over-the-Limit or Fall-through will have an entry written to SHRSARJ.

In-progress and pre-registered courses

If an in-progress or pre-registered course does not meet a requirement and is not in the Over-the-Limit, Fall-through, or Insufficient sections, then a SHRSARJ record will be written with the In-Progress: audit was run with the **ApplyInProgress=N** rejection reason. This indicates that the audit should be rerun for the student with one of these options:

- set the **UCX-CFG020 DAP14Apply In Progress** flag to Y
- when running BAN62 in Transit, deselect the boxes for **Include In-progress and Pre-registered classes**

Courses that did not meet a degree requirement will generate a SHRSARJ entry, which includes the reason that the class was rejected. Possible reasons for a course not meeting a requirement are:

Rejection type	Rejection reason
Insufficient	Below the minimum grade required
Insufficient	Repeat, bridged with force-insufficient=Y
Insufficient	Because of repeat policy
Insufficient	Because it was failed
Insufficient	Because it was audited
Insufficient	Because it was withdrawn
Insufficient	Because it was incomplete
Over-the-limit	Too many credits
Over-the-limit	Too many classes
Over-the-limit	Maximum number of credits exceeded
Over-the-limit	Maximum number of classes exceeded
Fall-through	Elective credits allowed exceeded (this is only possible if the UCX-CFG020 DAP14 Calculate Elective Credits Allowed flag = Y).

Note that if a class does not apply to a requirement and does not go to the Over-the-Limit, Fall-through, Insufficient, or In-Progress sections, the class will be written to SHRSARJ with the **Unknown where class applies in audit** rejection reason. This situation means that the auditor recognized the class yet it was not found in the resulting audit. A Service Request should be opened with the Degree Works Action Line if this error occurs.

Similarly, if a class does not apply to a requirement and a rejection reason was not determined, a default **Rejected reason undefined** rejection reason will be written to SHRSARJ_REJECTION_REASON. A Service Request should be opened with the Degree Works Action Line if this situation occurs.

BAN62 output

After BAN62 completes, two files are created: ban62.act and ban62.log. The .act file is a summary of the process, and will list the number of audits that were generated (if the **Create New Audit** box was checked in Transit). The .log file will contain one line for each student that was processed, and totals for students processed and records written to SHRSAPP and SHRSARJ.

The BAN62 .log file should be reviewed after each run. It is possible a course may have been skipped by BAN62 for one of the following reasons:

- An associated SSBSECT entry was not found for a course (CRN and Term).
- An SFRSTCR record was not found for a course (CRN and Term).
- An SHRTRCE record was not found for a transfer course.
- An associated SHRGRDE entry was not found for a course.
- An associated SHRTCKG or SHRTCKL record was not found for a course's SHRTCKN record.

If your BAN62 log file lists any classes as being skipped, run BAN62 with debug enabled for that student to determine which classes were skipped and why. A case should be opened with the Degree Works Action Line if this situation occurs.

SHRSAPP table layout

Updated: March 25, 2022

Familiarize yourself with the information each Banner SHRSAPP table column contains.

Column	Populated with
SHRSAPP_PIDM	Student's PIDM
SHRSAPP_REQUEST_NO	Transit Job Number for the SAP processor
SHRSAPP_SEQ_NO	Unique one up number that identifies the record in this table.
SHRSAPP_TERM_CODE	The term code associated with the course.
SHRSAPP_CRN	In-Progress and Historic Class: CRN (Course Reference Number) associated with the course. Null for Transfer courses.
SHRSAPP_SUBJ_CODE	Subject Code of the course.
SHRSAPP_CRSE_NUMB	Course Number of the course.
SHRSAPP_PROGRAM_REQ_HOURS	The total credit hours required for the Degree.
SHRSAPP_LEVL_CODE_COMP	The Level associated with the Degree Audit.
SHRSAPP_CAMP_CODE_COMP	Null
SHRSAPP_MAJR_CODE_COMP	The Major associated with the Degree Audit.
SHRSAPP_DEGC_CODE_COMP	The Degree associated with the Degree Audit.
SHRSAPP_TERM_CODE_EVAL	The term in which BAN62 was run to represent the term for which academic progress is being evaluated.
SHRSAPP_PROGRAM_COMP	The Program associated with the Degree Audit.
SHRSAPP_TYPE_IND	If Y, the course was applied in the Degree Audit. If N, the course was not applied.
SHRSAPP_REJECTION_IND	Indicator used to identify that a rejection reason exists for the unused course. Set to N if <ul style="list-style-type: none"> • the course was used in the audit • the course was not used in the audit and there is no rejection reason. Set to Y only if the course was not used in the audit and a rejection reason exists. This

Column	Populated with
	indicates that an associate SHRSARJ record exists for this course.
SHRSAPP_CRSE_TITLE	In-Progress: SSBSECT_CRSE_TITLE for the CRN and Term from SFRSTCR. Historic: SHRTCKN_CRSE_TITLE. Transfer: SHRTRCE_CRSE_TITLE.
SHRSAPP_CRSE_SOURCE	In-Progress: set to R. Historic: set to H. Transfer: set to T.
SHRSAPP_LEVL_CODE	The level of the course.
SHRSAPP_CAMP_CODE	The campus of the course. In-Progress SFRSTCR_CAMP_CODE, Historic SHRTCKN_CAMP_CODE. Null for Transfer courses.
SHRSAPP_GRDE_CODE	The grade assigned to the course. In-Progress: Null. Historic: SHRTCKG_GRDE_CODE_FINAL. Transfer: SHRTRCE_GRDE_CODE.
SHRSAPP_GMOD_CODE	The GMOD code associated with the course. In-Progress: SFRSTCR_GMOD_CODE. Historic: SHRTCKG_GMOD_CODE. Transfer: SHRTRCE_GMOD_CODE.
SHRSAPP_CREDIT_HOURS	The credit hours assigned to the course. In-Progress: SFRSTCR_CREDIT_HR. Historic: SHRTCKG_CREDIT_HOURS. Transfer: SHRTRCE_CREDIT_HOURS
SHRSAPP_REG_CREDIT_HOURS	The credit hours assigned in Registration for an In-Progress course. In-Progress: SFRSTCR_CREDIT_HR. Null for Historic and Transfer courses.
SHRSAPP_REG_HOURS_ATTEMPTED	The credit hours attempted for an In-Progress course, SFRSTCR_CREDIT_HR. Null for Historic and Transfer courses.
SHRSAPP_HOURS_ATTEMPTED	The credit hours attempted for the course. Historic: SHRTCKG_HOURS_ATTEMPTED if associated SHGRDE_ATTEMPTED_IND = Y. Transfer: SHRTRCE_CREDIT_HOURS if associated SHGRDE_ATTEMPTED_IND = Y. Null for In-Progress courses.
SHRSAPP_HOURS_PASSED	The credit hours passed for Historic and Transfer courses.

Column	Populated with
	<p>Historic: SHRTCKG_CREDIT_HOURS if SHRTCKN_REPEAT.Course_IND = "I" or is blank, and the associated SHRGRDE_PASSED_IND = Y.</p> <p>Transfer: SHRTRCE_CREDIT_HOURS if SHRTRCE_REPEAT.Course = "I" or is blank, and the associated SHRGRDE_PASSED_IND = Y.</p> <p>Null for In-Progress courses.</p>
SHRSAPP_HOURS_EARNED	<p>The credit hours earned for Historic and Transfer courses.</p> <p>Historic: SHRTCKG_CREDIT_HOURS if SHRTCKN_REPEAT.Course_IND = "I" or is blank, and the associated SHRGRDE_COMPLETED_IND = Y.</p> <p>Transfer: SHRTRCE_CREDIT_HOURS if SHRTRCE_REPEAT.Course = "I" or is blank, and the associated SHRGRDE_COMPLETED_IND = Y.</p> <p>Null for In-Progress courses.</p>
SHRSAPP_HOURS_GPA	<p>The credit hours used for GPA calculation for Historic and Transfer courses.</p> <p>Historic: SHRTCKG_CREDIT_HOURS if SHRTCKN_REPEAT.Course_IND = "I", or is blank and the associated SHRGRDE_GPA_IND = Y.</p> <p>Transfer: SHRTRCE_CREDIT_HOURS if SHRTRCE_REPEAT.Course = "I" or is blank, and the associated SHRGRDE_COMPLETED_IND = Y.</p> <p>Null for In-Progress courses.</p>
SHRSAPP_QUALITY_POINTS	<p>The quality points for Historic and Transfer courses. This value is taken from SHRGRDE_QUALITY_POINTS based on the grade, level and term for the course, multiplied by the Credit Hours Earned for the course.</p> <p>Historic: SHRGRDE_QUALITY_POINTS * SHRTCKG_CREDIT_HOURS if the associated</p>

Column	Populated with
	<p>SHRGRDE_GPA_IND = Y and SHRTCKN_REPEAT_COURSE_IND = "I" or is blank.</p> <p>Transfer: SHRGRDE_QUALITY_POINTS * SHRTRCE_CREDIT_HOURS if the associated SHRGRDE_GPA_IND = Y and SHRTRCE_REPEAT_COURSE = "I" or is blank.</p> <p>Null for In-Progress courses.</p>
SHRSAPP_GPA	The GPA is calculated for Historic and Transfer courses: SHRSAPP_QUALITY_POINTS / SHRSAPP_HOURS_GPA. Null for In-Progress courses.
SHRSAPP_GPA_TYPE_IND	Indicator used to identify the type of course for which the GPA is calculated. Values are I – Institutional Course, T – Transfer Course.
SHRSAPP_REPEAT_COURSE_IND	<p>Indicator to identify how the GPA for the course was used as a result of repeat processing. Values are E – Exclude, A – All, or I – Include.</p> <p>Historic courses: SHRTCKN_REPEAT_COURSE_IND. Transfer courses: SHRTRCE_REPEAT_COURSE. Null for In-Progress courses.</p>
SHRSAPP_TCKN_SEQ_NO	Historic: SHRTCKN_SEQ_NO. Null for In-Progress and Transfer courses.
SHRSAPP_TRIT_SEQ_NO	Transfer: SHRTRCE_TRIT_SEQ_NO. Null for In-Progress and Historic courses.
SHRSAPP_TRAM_SEQ_NO	Transfer: SHRTRCE_TRAM_SEQ_NO. Null for In-Progress and Historic courses.
SHRSAPP_TRCR_SEQ_NO	Transfer: SHRTRCE_TRCR_SEQ_NO. Null for In-Progress and Historic courses.
SHRSAPP_TRCE_SEQ_NO	Transfer: SHRTRCE_SEQ_NO. Null for In-Progress and Historic courses.
SHRSAPP_SURROGATE_ID	Reserved for future use.
SHRSAPP_VERSION	Reserved for future use.
SHRSAPP_USER_ID	Populated with "Degree Works"
SHRSAPP_ACTIVITY_DATE	Date the SAP Processor was run
SHRSAPP_DATA_ORIGIN	Populated with "Degree Works"
SHRSAPP_VPDI_CODE	If VPDI is in use, the associated VPDI code. Otherwise null.

SHRSARJ table layout

Updated: March 25, 2022

Familiarize yourself with the information each Banner SHRSAPP table column contains.

Column name	Populated with
SHRSARJ_PIDM	Student's PIDM
SHRSARJ_REQUEST_NO	Transit Job Number for the SAP processor
SHRSARJ_SAPP_SEQ_NO	The SHRSAPP_SEQ_NO for the course associated with this record.
SHRSARJ_SEQ_NO	Unique one up number that identifies the record in this table.
SHRSARJ_REJECTION_REASON	The reason that the course was not applied to any requirement on the degree audit.
SHRSARJ_SURROGATE_ID	Reserved for future use.
SHRSARJ_VERSION	Reserved for future use.
SHRSARJ_USER_ID	Populated with "Degree Works"
SHRSARJ_ACTIVITY_DATE	Date the SAP Processor was run
SHRSARJ_DATA_ORIGIN	Populated with "Degree Works"
SHRSARJ_VPDI_CODE	If VPDI is in use, the associated VPDI code. Otherwise null.

Applicants in Degree Works

Updated: September 30, 2022

There is an Applicant extract that can be executed to allow students who have an applicant record, but may not yet be considered a current student to be bridged into Degree Works.

For these students if they have transfer courses, test scores, or other appropriate data in Banner their data will be bridged over into Degree Works.

Advisors and REG users can see applicants within Degree Works just as they see other regular students. Applicants must log into Degree Works from within SSB or Luminis, so they must have at least that access on the Banner side. Applicants must have at least applied to the school and have a SGBSTDN, SARADAP, or SORLCUR/SORLFOS record to be able to be extracted and thus be able to log in to Degree Works.

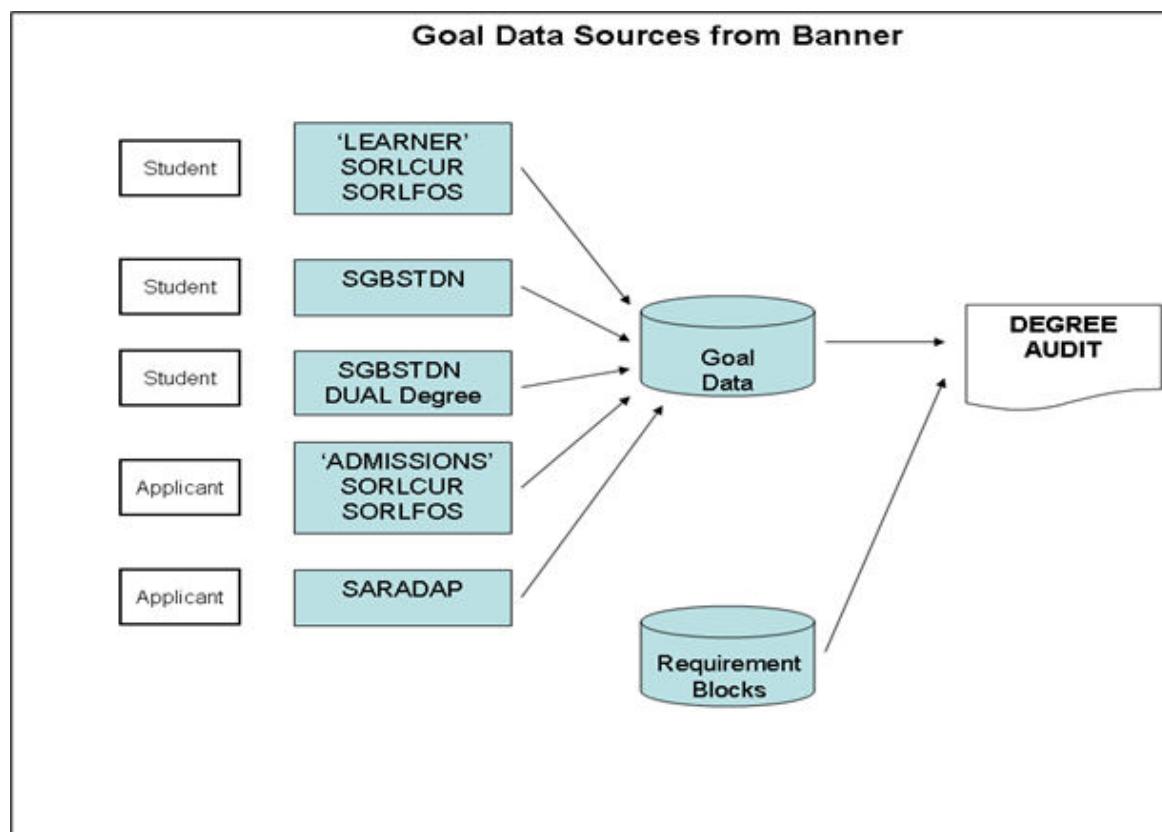
There is not a universal GUEST applicant extract or login that recruiters or admissions officers can use to log potential students into Degree Works. Unless the school has created a GUEST type user, which they can then use to access Degree Works.

Banner applicant processing

Updated: September 30, 2022

Degree Works allows the import of admissions data from Banner, allowing applicants to view degree audit worksheets. This can be a powerful recruiting tool for applicants who have transfer, test scores, custom records, or other appropriate data in Degree Works.

The Banner extract Transit job RAD32 may be run to extract applicant data to create the appropriate Goal and Goal Data records for each unique Level (school) and Degree combination (in addition to extracting all other appropriate Banner data for an applicant that may be used by Degree Works).



There are six paths that can be taken to load student/applicant Goal data.

- Student Goal Data may be loaded from LEARNER SORLCUR/SORLFOS curriculum records.
- Student Goal Data may be loaded from SGBSTDN curriculum data.
- Student Goal Data may be loaded from SGBSTDN DUAL degree data.
- Applicant Goal Data may be loaded from ADMISSIONS SORLCUR/SORLFOS curriculum records.

- Applicant Goal Data may be loaded from SARADAP curriculum data for unique Level (School)/Degree combinations.
- Applicant Goal Data may be loaded from SARADAP curriculum data for unique Level (School) codes.

The rules for determining how the extract decides what Goal Data to extract from Banner are defined below.

Three new configuration flags in the UCX-CFG020 BANNER record control how the applicant data is to be extracted and which rules will be followed.

Process Applicants - A Y/N flag. Set to Y if applicant processing by the extract is to be performed. Set to N if NO applicant data is to be imported into Degree Works. If this flag is set to Y and the REFRESH button on the web is clicked, then Banner applicant data will be looked up in addition to student data.

Process Both Goals - A Y/N flag. Set to Y if goal (degree) data from Banner Student (LEARNER) and Applicant (ADMISSIONS) data is to be imported into Degree Works. Set to N if the ADMISSIONS SORLCUR/SORLFOS records are NOT to be imported into Degree Works if 'LEARNER' SORLCUR/SORLFOS data is found. For example, if a student is working toward an undergraduate degree, but has applied to graduate school at the same institution applicant data could also exist at the same time. Thus, this flag would need to be set to Y so that both the undergraduate student data and the graduate applicant data is imported into Degree Works.

Load SARADAP Goals - A Y/N flag. Set to Y if SARADAP is to be processed if NO SORLCUR ADMISSIONS data is found. Set to N if NO SARADAP data is to be loaded into Degree Works.

Compare Applicant Degrees – A Y/N flag. Set to N if only the Level (School) is to be used when determining which SGBSTDN and SARADAP records to be extracted (if no SORLCUR/SORLFOS records are found for the given ID code). Otherwise unique Level (School)/Degree combinations will be used when determining which SGBSTDN and SARADAP records are to be extracted.

The Banner data extract is now a STUDENT and APPLICANT extract. When processing begins, the software will determine whether any of the following conditions exist and set the configuration flags appropriately.

- The LEARNER SORLCUR/SORLFOS records are retrieved using the SORLCUR query in integration.banner.extract.config (where the SORLCUR query specifies SORLCUR_LMOD_CODE = LEARNER along with all other required SQL). If found, set LEARNER_FOUND = Y.
- If the UCX-CFG020 BANNER Process Applicants = Y (for Refresh over the web) or the Banner extract is run with mode APPLICANT, then two additional checks will be made.
 - No LEARNER SORLCUR/SORLFOS records were found or the UCX-CFG020 BANNER Process Both Goals is set to Y.
 - If either condition is met then ADMISSIONS SORLCUR/SORLFOS records are retrieved using the SORLCUR2 query in integration.banner.extract.config (where the SORLCUR query specifies SORLCUR_LMOD_CODE = ADMISSIONS along with all other required SQL). If found then set ADMISSIONS_FOUND = Y.

- The SGBSTDN General Student record is looked up regardless of whether any SORLCUR/SORLFOS records are found. If an SGBSTDN record is found, SGBSTDN_FOUND will be set to Y.
- If ADMISSIONS_FOUND = Y then the associated SARADAP record will be identified using the SORLCUR_PIDM, SORLCUR_TERM_CODE and SORLCUR_KEY_SEQNO (linked to SARADAP_APPL_NO). If ADMISSIONS_FOUND = N, and the UCX-CFG020 BANNER Process Applicants flag = Y (or the APPLICANT mode is found in batch) and the UCX-CFG020 BANNER Load SARADAP Goals = Y the SARADAP record will be looked up using the SARADAP query in integration.banner.extract.config. If a valid record is found, SARADAP_FOUND = Y.

After the appropriate records have been read, the four flags (LEARNER_FOUND, ADMISSIONS_FOUND, SGBSTDN_FOUND, and SARADAP_FOUND) will be checked. If data was NOT retrieved for ANY of these four conditions then an error message will be written to the extract log and processing will quit for this ID Code.

Next, the software processes the curriculum data into using the following rules or paths.

- If LEARNER_FOUND = Y then the appropriate rad_goal_dtl and rad_goaldtrecords will be created from LEARNER SORLCUR/SORLFOS.
- If LEARNER_FOUND = N and ADMISSIONS_FOUND = N and SGBSTDN_FOUND = Y then the appropriate rad_goal_dtl and rad_goaldtrecords will be created from SGBSTDN.
- If the UCX-CFG020 BANNER Check Dual Degree = Y and SGBSTDN_FOUND = Y then the Dual Degree fields will be checked. If the Dual Degree contains data in the Dual Level (school) and Dual Degree fields, and the combination is unique (does not match any LEARNER combinations previously extracted) then the appropriate rad_goal_dtl and rad_goaldtrecords will be created from the Dual Degree.
- If ADMISSIONS_FOUND = Y and the Level (school)/Degree combination is unique (does not match any LEARNER combinations previously extracted), then the appropriate rad_goal_dtl and rad_goaldtrecords will be created from the ADMISSIONS SORLCUR/SORLFOS data.
- If ADMISSIONS_FOUND = N and UCX-CFG020 BANNER Process Applicants = "Y" and UCX-CFG020 BANNER Load SARADAP Goals = Y and SARADAP_FOUND = Y and UCX_CFG020 BANNER Compare Applicant Degrees = Y or BLANK then an additional edit is made.
If the SARADAP_TERM_CODE_ENTRY is GREATER THAN the SGBSTDN_TERM_CODE_EFF then any matching SGBSTDN goal records for that Level (school) and Degree combination are replaced with the SARADAP goal records for that Level (school) and Degree.

If, however, the SARADAP_TERM_CODE_ENTRY is LESS THAN or EQUAL TO the SGBSTDN_TERM_CODE_EFF then the SARADAP records for that Level (school) and Degree combination will be SKIPPED and NOT written to the rad_goal_dtl/rad_goaldt. Instead the previously loaded SGBSTDN goal data will be written to the rad_goal_dtl/rad_goaldt.

The same edit will be performed for the SGBSTDN_TERM_CODE_CTLG2 and SARADAP_TERM_CODE_CTLG2 entries. If the Level (school)/Degree combination is unique (does not match any LEARNER or SGBSTDN combinations previously extracted) then the appropriate rad_goal_dtl and rad_goaldtrecords will be created from SARADAP.

- If ADMISSIONS_FOUND = N and UCX-CFG020 BANNER Process Applicants = "Y" and UCX-CFG020 BANNER Load SARADAP Goals = Y and SARADAP_FOUND = Y and UCX_CFG020 BANNER Compare Applicant Degrees = N then an additional edit is made.

If the SARADAP_TERM_CODE_ENTRY is GREATER THAN the highest (most recent) SGBSTDN_TERM_CODE_EFF then any matching SGBSTDN goal records only the Level (school) are replaced with the SARADAP goal records for that Level (school).

If, however, the SARADAP_TERM_CODE_ENTRY is LESS THAN or EQUAL TO the highest (most recent) SGBSTDN_TERM_CODE_EFF then the SARADAP records for that Level (school) will be SKIPPED and NOT written to the rad_goal_dtl/rad_goaldata_dtl. Instead the previously loaded SGBSTDN goal data will be written to the rad_goal_dtl/rad_goaldata_dtl.

The same edit will be performed for the SGBSTDN_TERM_CODE_CTLG2 and SARADAP_TERM_CODE_CTLG2 entries. If the Level (school) is unique (does not match any LEARNER or SGBSTDN Level (school) previously extracted) then the appropriate rad_goal_dtl and rad_goaldata_dtl records will be created from SARADAP.

Processing will then continue so that other Banner data for the individual, such as transfer classes, test scores (for example, AP tests), and so on are extracted and imported into Degree Works.

Example of APPLICANT SQL in integration.banner.extract.config:

```
#####
SORLCUR2 - ADMISSIONS CURRICULUM (ban40) #####
# SORLCUR must be a; AND is required at the end of the WHERE
SORLCUR2-from:   FROM SORLCUR a, STVTERM t, SARADAP c
SORLCUR2-where:  WHERE (SELECT COUNT(*) FROM SHRTRCR
SORLCUR2-where:  WHERE SHRTRCR_PIDM = c.SARADAP_PIDM) > 0
SORLCUR2-where:  AND t.STVTERM_START_DATE > SYSDATE
SORLCUR2-where:  AND ((SELECT COUNT(*) FROM SGBSTDN
SORLCUR2-where:  WHERE SGBSTDN_PIDM = c.SARADAP_PIDM) < 1
SORLCUR2-where:  OR (SELECT COUNT(*) FROM STVSTST, SGBSTDN p
SORLCUR2-where:  WHERE STVSTST_CODE = p.SGBSTDN_STST_CODE
SORLCUR2-where:  AND STVSTST_REG_IND = 'Y'
SORLCUR2-where:  AND p.SGBSTDN_TERM_CODE_EFF = (SELECT MAX
SORLCUR2-where:    (o.SGBSTDN_TERM_CODE_EFF) FROM SGBSTDN o
SORLCUR2-where:  WHERE o.SGBSTDN_PIDM = p.SGBSTDN_PIDM
SORLCUR2-where:  AND o.SGBSTDN_TERM_CODE_EFF <
SORLCUR2-where:    c.SARADAP_TERM_CODE_ENTRY)) < 1)
SORLCUR2-where:  AND a.SORLCUR_CACT_CODE = 'ACTIVE'
SORLCUR2-where:  AND a.SORLCUR_LMOD_CODE = 'ADMISSIONS'
SORLCUR2-where:  AND a.SORLCUR_SEQNO = (SELECT MAX(b.SORLCUR_SEQNO)

SORLCUR2-where:  FROM SORLCUR b
SORLCUR2-where:  WHERE b.SORLCUR_PIDM = a.SORLCUR_PIDM
SORLCUR2-where:  AND b.SORLCUR_PRIORITY_NO = a.SORLCUR_PRIORITY_NO
SORLCUR2-where:  AND b.SORLCUR_LMOD_CODE = 'ADMISSIONS')
SORLCUR2-where:  AND a.SORLCUR_PIDM = c.SARADAP_PIDM
SORLCUR2-where:  AND a.SORLCUR_TERM_CODE = c.SARADAP_TERM_CODE_ENTR
Y
```

```

SORLCUR2-where:    AND a.SORLCUR_KEY_SEQNO = c.SARADAP_APPL_NO
SORLCUR2-where:    AND t.STVTERM_CODE = c.SARADAP_TERM_CODE_ENTRY
SORLCUR2-where: AND
#                           a.SORLCUR_PIDM = <students-pidm>
#####
##### SORLFOS2 - APPLICANT FIELD OF
STUDY (ban40) #####
# SORLFOS must be a; AND is required at the end of the WHERE
SORLFOS2-where: FROM SORLFOS a, SORLCUR b, STVTERM t, SARADAP d
SORLFOS2-where: WHERE (SELECT COUNT(*) FROM SHRTRCR
SORLFOS2-where: WHERE SHRTRCR_PIDM = d.SARADAP_PIDM) > 0
SORLFOS2-where: AND t.STVTERM_START_DATE > SYSDATE
SORLFOS2-where: AND ((SELECT COUNT(*) FROM SGBSTDN
SORLFOS2-where: WHERE SGBSTDN_PIDM = d.SARADAP_PIDM) < 1
SORLFOS2-where: OR (SELECT COUNT(*) FROM STVSTST, SGBSTDN p
SORLFOS2-where: WHERE STVSTST_CODE = p.SGBSTDN_STST_CODE
SORLFOS2-where: AND STVSTST_REG_IND = 'Y'
SORLFOS2-where: AND p.SGBSTDN_TERM_CODE_EFF = (SELECT MAX
SORLFOS2-where:          (o.SGBSTDN_TERM_CODE_EFF) FROM SGBSTDN o
SORLFOS2-where: WHERE o.SGBSTDN_PIDM = p.SGBSTDN_PIDM
SORLFOS2-where: AND o.SGBSTDN_TERM_CODE_EFF <
SORLFOS2-where:          d.SARADAP_TERM_CODE_ENTRY)) < 1)
SORLFOS2-where: AND b.SORLCUR_CACT_CODE = 'ACTIVE'
SORLFOS2-where: AND b.SORLCUR_LMOD_CODE = 'ADMISSIONS'
SORLFOS2-where: AND b.SORLCUR_SEQNO = (SELECT MAX(f.SORLCUR_SEQNO)
SORLFOS2-where: FROM SORLCUR f
SORLFOS2-where: WHERE f.SORLCUR_PIDM = b.SORLCUR_PIDM
SORLFOS2-where: AND f.SORLCUR_PRIORITY_NO = b.SORLCUR_PRIORITY_NO
SORLFOS2-where: AND f.SORLCUR_LMOD_CODE = 'ADMISSIONS')
SORLFOS2-where: AND b.SORLCUR_PIDM = d.SARADAP_PIDM
SORLFOS2-where: AND b.SORLCUR_TERM_CODE = d.SARADAP_TERM_CODE_ENTR
Y
SORLFOS2-where: AND b.SORLCUR_KEY_SEQNO = d.SARADAP_APPL_NO
SORLFOS2-where: AND t.STVTERM_CODE = d.SARADAP_TERM_CODE_ENTRY
SORLFOS2-where: AND a.SORLFOS_CSTS_CODE = 'INPROGRESS'
SORLFOS2-where: AND a.SORLFOS_CACT_CODE = 'ACTIVE'
SORLFOS2-where: AND a.SORLFOS_PIDM = b.SORLCUR_PIDM
SORLFOS2-where: AND a.SORLFOS_LCUR_SEQNO = b.SORLCUR_SEQNO
SORLFOS2-where: AND a.SORLFOS_SEQNO =
SORLFOS2-where:          (SELECT MAX(1.SORLFOS_SEQNO) FROM SORLFOS 1
SORLFOS2-where: WHERE 1.SORLFOS_PIDM = b.SORLCUR_PIDM
SORLFOS2-where: AND 1.SORLFOS_PRIORITY_NO = a.SORLFOS_PRIORITY_NO
SORLFOS2-where: AND 1.sorlfos_csts_code = 'INPROGRESS'
SORLFOS2-where: AND 1.SORLFOS_LCUR_SEQNO = b.SORLCUR_SEQNO)
SORLFOS2-where: AND
#
#                           a.SORLFOS_PIDM = <students-pidm>

```

Extract applicants

Updated: September 30, 2022

You can extract applicants.

About this task

You can:

- Use the **REFRESH** button (will work only if UCX-CFG020/BANNER->Process_Applicants = Y).
- Use the launchjob script in cron.
- In Transit, select **RAD32**, as the processor must use the SQL file or supply a list of IDs.

Applicant extract process configuration flags

Updated: March 25, 2022

Some configuration flags for the applicant extract process need to be set using Controller in CFG020/BANNER

- Process Applicants – Y/N – Setting this flag to “Y” will allow the applicant extract process to happen when the REFRESH button is pressed. The applicant data will be looked up in addition to the student data.
- Process Both Goals – Y/N – Setting this flag to “Y” loads both the “LEARNER” and the “ADMISSIONS” data from SORLCUR/SORLFOS into DGW. If it is set to “N”, then “ADMISSIONS” data is not loaded into DGW if “LEARNER” data is found.

Extract process

Updated: September 30, 2022

After refresh

When the **REFRESH** button is clicked, the following extract process is followed:

1. The SORLCUR/SORLFOS records are checked for a LEARNER record using the SORLCUR query within integration.banner.extract.config. If a LEARNER record is found, then the variable LEARNER_FOUND is set to Y.
2. If UCX-CFG020/BANNER->Process_Applicants = Y then if LEARNER_FOUND = N or if LEARNER_FOUND = Y and UCX-CFG020/BANNER->Process_both_goals = Y then the SORLCUR/SORLFOS records are searched for an ADMISSIONS record using the SORLCUR2 query in integration.banner.extract.config. If an ADMISSIONS record is found, then the variable ADMISSIONS_FOUND is set to Y.
3. Regardless of what happens in steps 1 & 2, the SGBSTDN record is looked up. If found, the variable SGBSTDN_FOUND is set to Y.
4. One of the following two paths will be followed:

- a. If the variable ADMISSIONS_FOUND = Y, then the associated SARADAP record will be looked up based on the associated values found in SORLCUR.
- b. If ADMISSIONS_FOUND = N and UCX-CFG020/BANNER->Process_Applicants = Y and UCX-CFG020/BANNER->Load_SARADAP_GOALS = Y the SARADAP record is looked up based on the SARADAP query in integration.banner.extract.config.

After execution

When the applicant extract is executed, the following process is followed:

1. Same as above. The LEARNER_FOUND flag is set to Y if found.
2. If LEARNER_FOUND = N or if LEARNER_FOUND = Y and UCX-CFG020/BANNER->Process_both_goals = Y then use the SORLCUR2 in integration.banner.extract.config to search for ADMISSIONS records. If found set ADMISSIONS_FOUND = Y.
3. Same as above.
4. One of the following two paths will be followed:
 - a. If the variable ADMISSIONS_FOUND = Y, then the associated SARADAP record will be looked up based on the associated values found in SORLCUR.
 - b. If ADMISSIONS_FOUND = N and UCX-CFG020/BANNER->Load_SARADAP_GOALS = Y the SARADAP record is looked up based on the SARADAP query in integration.banner.extract.config.

After extraction

After extraction, at least one of the variables (ADMISSIONS_FOUND, LEARNER_FOUND, SGBSTDN_FOUND, or SARADAP_FOUND) will have to be Y or an error will result. Then one and possibly up to three of the following paths will be taken (one path from the two student type paths, one path from the two applicant type paths, and one from the dual degree path):

1. (Student Path)
If LEARNER_FOUND = Y
The goal (degree) records are created from the SORLCUR/SORLFOS records.
2. (Student Path)
If LEARNER_FOUND = N and ADMISSIONS_FOUND = N and SGBSTDN_FOUND = Y
Load goal records from SGBSTDN records.
3. (Applicant Path)
If ADMISSIONS_FOUND = Y and the Level(school)/Degree combination does not match any of the LEARNER Level(School)/Degree combinations
Load goal records from the ADMISSIONS version of the SORLCUR/SORLFOS records.
4. (Applicant Path)
If ADMISSIONS_FOUND = N and SARADAP_FOUND = Y and UCX-CFG020/BANNER->Process_Applicants = Y and UCX-CFG020/BANNER->Load_SARADAP_goals = Y and the Level(school)/Degree combination does not match any of the LEARNER Level(School)/Degree combinations

Load goal records from SARADAP

5. (Dual Degree Path)

If UCX-CFG020/BANNER->Check_dual_degree = Y and SGBSTDN_FOUND = Y and the Dual Level(school)/Dual Degree combination does not match any of the LEARNER Level(School)/ Degree combinations

Load goal records from the SGBSTDN dual degree records

integration.banner.extract.config setting

Updated: September 30, 2022

There are three areas in the integration.banner.extract.config setting that need to be examined to determine if they are extracting the appropriate applicant data.

- SORLCUR2
- SORLFOS2
- SARADAP

Applicant user class

Updated: September 30, 2022

An applicant user class (APP) must be created in SHPCFG. This will be the lowest class. If the class does not already exist in AUD012, it will need to be added there.

```
#-----
-----  
#-- Degree Works keys for applicants  
#-----  
-----  
if (DGWUSERCLASS = "APP") then  
    addgroup = SRNAPP
```

By default, the SRNAPP group has the following accesses:

SDAUDREV

SDSTUME

SDWEB31

SDWHATIF

SDWORKS

SDXML31

SDAUDPDF

Financial aid data in Degree Works

Updated: March 25, 2022

You should set up UCX-BAN080 records to enable Financial Aid audits to help you determine if your aid students are meeting their financial aid obligations.

The following are examples of the types of entries you should add to UCX-BAN080:

AIDAWARD:AID	AWARD
AIDAWARD:COLUMN	RPRAWRD_FUND_CODE
AIDAWARD:ORDERBY	RPRAWRD_FUND_CODE
AIDAWARD:TABLE	RPRAWRD
AIDAWARD:WHERE_1	RPRAWRD_AIDY_CODE = '0506'
AIDAWARD:WHERE_2	AND RPRAWRD_AWST_CODE = 'ACPT'
AIDYEAR:AID	AIDYEAR
AIDYEAR:COLUMN	RPRAWRD_AIDY_CODE
AIDYEAR:ORDERBY	RPRAWRD_AIDY_CODE
AIDYEAR:TABLE	RPRAWRD
AIDYEAR:WHERE_1	RPRAWRD_AWST_CODE = 'ACPT'
AIDYEAR:WHERE_2	AND RPRAWRD_AIDY_CODE in
AIDYEAR:WHERE_3	(SELECT b.ROINST_AIDY_CODE FROM ROB
INST b	WHERE b.ROINST_STATUS_IND = 'A')
AIDYEAR:WHERE_4	ENROLLSTATUS
AIDENRSTATUS:AID	
AIDENRSTATUS:COLUMN	SGBSTDN_FULL_PART_IND
AIDENRSTATUS:ORDERBY	SGBSTDN_FULL_PART_IND
AIDENRSTATUS:TABLE	SGBSTDN a
AIDENRSTATUS:WHERE_1	a.SGBSTDN_TERM_CODE_EFF =
AIDENRSTATUS:WHERE_2	(SELECT MAX(b.SGBSTDN_TERM_CODE_EFF
)	
AIDENRSTATUS:WHERE_3	FROM SGBSTDN b

AIDENRSTATUS:WHERE_4	WHERE b.SGBSTDN_PIDM = a.SGBSTDN_PIDM
)	
AIDSTATUS:AID	AIDSTATUS
AIDSTATUS:COLUMN	RORSAPR_SAPR_CODE
AIDSTATUS:ORDERBY	RORSAPR_SAPR_CODE
AIDSTATUS:TABLE	RORSAPR
AIDSTATUS:WHERE_1	RORSAPR_SAPR_CODE IS NOT NULL

Create Scribe custom data using Banner data

Updated: March 25, 2022

There may be a time when a rule needs to be scribed against a variable from Banner that is not by default bridged into Degree Works.

Some examples of this type of variable include graduation status, academic standing, and campus code. Follow the procedure below to set up and use these types of variables.

1. Create the variable in the BAN080. In this table you will indicate the column, table, and where statements to retrieve the variable from Banner. The following shows an example of the code set up for the academic standing code. You pick a name for this variable. We will call it ACSTCODE.

Note: If you choose a table that is not a typical table DGW uses, you must make sure your DBA gives the DGW user read access to this table.

- a. Create a record in BAN080 with the key of ACSTCODE:TABLE. The Value1 should be the table name you are retrieving from in this case, SGBSTDN.
- b. Next create a record for the column with the key code: ACSTCODE:COLUMN. For this example, we will be retrieving the SGBSTDN_STST_CODE, so that should be entered in the Value1 field.
- c. You can create records that are specific to the school, degree, or both. When the audit is run the school and degree on the custom record will be matched against the school and degree of the audit. In doing this, you can select values that are school and degree specific. See the [UCX-BAN080 Banner Custom Data Definitions](#) topic for details about how to set this up.
- d. Any SQL statements that will be used to find the desired instance of the variable should now be entered into records with keys beginning with WHERE. For multiple where statements, add multiple where records. These records should be named WHERE_1, WHERE_2, etc. In this example, we will find the academic status value for the latest term. The SQL to accomplish this will be:

```
Where a.sgbstdn_term_code_eff = (Select Max(b.sgbstdn_term_code_eff) from
Sgbstdn b where b.sgbstdn_pidm = a.sgbstdn_pidm)
```

In this example, four records will need to be created. They can be named:
 ACSTCODE:WHERE_1, ACSTCODE:WHERE_2, ACSTCODE:WHERE_3, and
 ACSTCODE:WHERE_4.

A summary of the records in BAN080 for this variable should look like the following:

ACSTCODE:COLUMN	Sgbstdn_stst_code
ACSTCODE:TABLE	Sgbstdn a
ACSTCODE:WHERE_1	a.sgbstdn_term_code_eff =
ACSTCODE:WHERE_2	(Select Max(b.sgbstdn_term_code_eff)
ACSTCODE:WHERE_3	From SGBSTDN b
ACSTCODE:WHERE_4	Where b.sgbstdn_pidm = a.sgbstdn_pidm)

2. Next, a record needs to be added into the SCR002 table in which to scribe against. This record should have the same name as the variable created in step 1 above. The Data Element value should be the record from UCX-SYS999 that is to be used in the Scribe IF statement.

Because you are pulling data from BAN080, this data will go into the rad_custom_dtl record. The rad_custom_code with a value of 'R322' in SYS999 should then be entered in as the Data Element. The UCX table can be left blank because this is coming from BAN080 data.

The Edit Element1 should be the value of the data item. In this case, 'R323' in SYS999 points to the rad_custom_value field, this should be entered into the Edit Element1 field. We will be retrieving all the values for this data item, so the Type value should be set to EV. Finally the value of the SCR002 record should be the name of the variable from the BAN080 table. In this case ACADSTST.

3. After the tables have been set up in Controller for the BAN080 records and the SCR002 record, a webrestart should be issued. For students to get this data put into their rad_custom_data table, they will need to be re-bridged from Banner into Degree Works. If the data item does not exist in Banner for the student, the record will not be loaded in the rad_cust_dtl table.
4. Now you can put in rules into your blocks to use this data variable created above. For our example using the Academic Status code, we can scribe against this value to determine if a particular rule has been met.

Scribing against test scores

Updated: September 30, 2022

Test scores are brought into the DGW rad_custom_dtl table from the SORLCUR table by default based on your integration.banner.extract.config setting.

By default, all test scores are retrieved. You can modify integration.banner.extract.config to delimit what test scores are brought in by adding WHERE statements into the integration.banner.extract.config setting. Because these scores are already being brought in, it is not needed to create a BAN080 set of records to retrieve them. You will need to create SCR002 records for the test scores to be able to scribe against the tests. As an example, if a student has the following records in SORTEST:

SORTTEST_TESC_CODE DATE	SORTTEST_TEST_SCORE	SORTTEST_TEST_
A01 15-JAN-22	25	
A02 15-JAN-22	24	
A03 15-JAN-22	25	
A04 15-JAN-22	25	
A05 15-JAN-22	26	

To be able to scribe against one of these, the test code will need to be added to the SCR002 table. For example to scribe against test types of A01, the following SCR002 record needs to be created:

Key	A01
Description	Test Scores - A01
Data Element	R323
Edit Element 1	R322
Edit Type 1	EV (Edit Value 1)
Edit Value 1	A01

Populate SHP_USER_ATTRIB from Banner data

Updated: March 25, 2022

If your institution uses Banner data to assign Keys and Services to students, you will need to add an entry to UCX_BAN080 to identify the data to be added to the SHP_USER_ATTRIB table.

For example, if you want to allow access to Degree Works services based on academic standing, you must create an entry in BAN080 to generate a RAD_CUSTOM_DTL, then add a SHPCFG entry to create an associated SHP_USER_ATTRIB record.

1. Create the custom data to be pulled over from Banner in the BAN080 table. (For information on how to create BAN080 variables, please refer to the documentation on retrieving BAN080 variables.) In this example, we will create the custom value ACADSTANDING from the SHRTTRM_ASTD_CODE_END_OF_TERM column of SHRTTRM.

ACADSTANDING:COLUMN	SHRTTRM_ASTD_CODE_END_OF_TERM
ACADSTANDING:ORDERBY	SHRTTRM_ASTD_CODE_END_OF_TERM
ACADSTANDING:TABLE	SHRTTRM a
ACADSTANDING:WHERE_1	a.SHRTTRM_TERM_CODE =
ACADSTANDING:WHERE_2	(SELECT MAX(b.SHRTTRM_TERM_CODE)
ACADSTANDING:WHERE_3	FROM SHRTTRM b

- ACADSTANDING:WHERE_4
IDM)
WHERE b.SHRTTRM_PIDM = a.SHRTTRM_P
2. Add a new entry in BAN080 where the key is the custom code followed by a colon (:) and SHPCFG.
 3. Re-extract the students so they get the ACADSTANDING code and value loaded into their rad_custom_dtl. In addition, due to the ACADSTANDING:SHPCFG entry, they will also get a record written to SHP_USER_ATTRIB.

Preferred name

Updated: September 30, 2022

The name field for the student, applicant, advisor, or staff user is pulled from SPRIDEN's FIRST_NAME, LAST_NAME, and MI fields. The preferred name is pulled from SPBPERS_PREF_FIRST_NAME. However, a configuration change to integration.banner.extract.config can tell the extract to pull the preferred name from a special SPRIDEN record.

For information about how to format the name field and to enable the preferred name functionality, see the banner.extract.staff.name and the banner.extract.student.name settings in the [integration.banner](#) topic. Only when the pref.enabled settings are set to true will the preferred name be pulled from SPBPERS or SPRIDEN.

If you store the preferred name in SPRIDEN instead of in SPBPERS, you should use the code below instead of the default SPBPERS configuration in integration.banner.extract.config in Controller. Here, the FROM clause is replaced with a select from SPRIDEN with a specific SPRIDEN_NTYP_CODE. You need to change XXXX to the NTYP_CODE your school uses to signify the preferred name SPRIDEN record.

```
#####
SPBPERS -Pref-first-name (ban40/ban45) #####
#
# Standard is to only select by the SPBPERS_PIDM
# -but you may add to the WHERE clause if needed -
# but don't forget the final AND
SPBPERS-from: FROM
SPBPERS-from: (SELECT spriden_first_name spbpers_pref_first_name
SPBPERS-from: ,spriden_pidm spbpers_pidm
SPBPERS-from: FROM SPRIDEN, SPBPERS
SPBPERS-from: WHERE spriden_pidm = spbpers_pidm
SPBPERS-from: AND spriden_ntyp_code = 'XXXX') a
SPBPERS-where: WHERE
# a.SPBPERS_PIDM = <individual's pidm>
```

Banner Data Mapping for BIF

Updated: September 30, 2022

Review how data is extracted from the Banner database and bridged or copied to the RAD and DAP database tables in Degree Works.

Banner Bridge

Updated: March 25, 2022

Banner data is retrieved from many database tables, formatted based on various Degree Works rules, loaded into a set of records in the Bridge Interface Format (BIF), and then passed to the radbridge subroutine, which loads the Banner data into the RAD and DAP tables.

Each BIF record contains an 8-byte IDENTIFIER (for example, R011PRIM for the rad_primary_mst) and is the link to the Degree Works Bridge documentation. That document contains the actual BIF record layouts for each of the different record types. This document discusses the Banner database tables that are used and the rules that are used to convert the data into the format required by Degree Works.

R011PRIM - Primary Record

Updated: September 29, 2023

The Banner ID and name from SPRIDEN are used for most of the data in this record.

In the cases where different database tables are used they will be identified. Only the pieces of data obtained from Banner tables are defined below.

SQL used to read the SPRIDEN table

```
SELECT SPRIDEN_PIDM,
       SPRIDEN_ID,
       SPRIDEN_LAST_NAME,
       SPRIDEN_FIRST_NAME,
       SPRIDEN_MI,
       SPRIDEN_CHANGE_IND
  FROM SPRIDEN
 WHERE SPRIDEN_ID = :zStudentId
   AND SPRIDEN_CHANGE_IND IS NULL;
```

These records are bridged into the RAD_PRIMARY_MST table in Degree Works.

Field	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data. However, when retrieving Banner data in the Banner database the SPRIDEN_PIDM is used as the key to student and staff data.
rad_name	See the integration.banner.extract.student.name.* and integration.banner.extract.staff.name.* settings. You may control how the name is extracted and bridged using these settings.

Field	Banner Data
	For more information, see the Preferred name topic and Technical Configuration Reference .
rad_nickname	Not extracted. Loaded with BLANKS.
rad_format_name	Not extracted. Loaded with BLANKS.
rad_sex	Not extracted. Loaded with BLANKS.
rad_birthdate	Not extracted. Loaded with BLANKS.
rad_soc_sec_nbr	Not extracted. Loaded with BLANKS.
rad_term	<p>The Active Term (rad_term) is determined by comparing the Highest Class Term to the Highest Degree Term (or Future Degree Term for incoming freshman and transfers).</p> <p>The Highest Class Term is first calculated from the current classes found in SFRSTCR. The class with the most recent SFRSTCR_TERM_CODE that has an STVTERM_START_DATE less than or equal to the current date is used. Future classes are ignored in this calculation. If no valid current classes exist, the highest historic term from SHRTCKN (SHRTCKN_TERM_CODE) is used as the Highest Class Term.</p> <p>The Highest Degree Term is found using the SORLCUR_TERM_CODE that has an STVTERM_START_DATE less than or equal to the current date. The system evaluates the SORLCUR record with a SORLCUR_LMOD_CODE of LEARNER for students and ADMISSIONS for applicants. If integration.banner.extract.student.useAdmitTerm is TRUE, the SORLCUR_TERM_CODE_ADMIT is used if it is greater than the SORLCUR_TERM_CODE and it can be in the future.</p> <p>Note: The SQL used above to retrieve the desired class and degree records is defined in the integration.banner.extract.config Shepherd setting.</p> <p>The following hierarchy is used to set the active term:</p> <ol style="list-style-type: none"> 1. If the Highest Degree Term \geq Highest Class Term, the Highest Degree Term will be used as the Active Term. 2. If the Highest Degree Term $<$ Highest Class Term, the Highest Class Term will be used as the Active Term. 3. If the Highest Degree Term cannot be determined and the Highest Class Term cannot be determined, the Future Degree Term will be used (incoming freshmen and incoming transfers).
rad_address1	Not extracted. Loaded with BLANKS.
rad_address2	Not extracted. Loaded with BLANKS.
rad_city	Not extracted. Loaded with BLANKS.

Field	Banner Data
rad_zip	Not extracted. Loaded with BLANKS.
rad_phone	Not extracted. Loaded with BLANKS.
rad_email	GOREMAL_EMAIL_ADDRESS if the GOREMAL_EMAL_CODE = UCX-CFG020 BANNER Email Code. If no match on the EMAL_CODE is found and the UCX-CFG020 BANNER Email Override = Y, the record with a GOREMAL_PREFERRED_IND = Y will be used for the email address (if found). Note that GOREMAL_STATUS_IND = A is typically in the sql used to select the GOREMAL records, but you can check what you have configured in integration.banner.extract.config.
rad_user_def1	The SPRIDEN_PIDM is loaded into this User Defined field for debugging purposes. The SPRIDEN_ID is stored as the rad_id for all student/staff oriented data. However, when retrieving Banner data in the Banner database the SPRIDEN_PIDM is used as the key to student and staff data. Thus, it is stored here for reference purposes.
rad_user_def2	Not extracted. Loaded with BLANKS.
rad_user_def3	Not extracted. Loaded with BLANKS.
rad_user_def4	Not extracted. Loaded with BLANKS.
rad_user_def5	Not extracted. Loaded with BLANKS.
rad_user_def6	Not extracted. Loaded with BLANKS.
rad_user_def7	Not extracted. Loaded with BLANKS.
rad_user_def8	Not extracted. Loaded with BLANKS.
rad_user_def9	Not extracted. Loaded with BLANKS.
rad_user_def10	Not extracted. Loaded with BLANKS.

R027GOAL - Goal Record

Updated: September 29, 2023

Several Banner tables are used to obtain the data in this record.

Only the pieces of data obtained from Banner tables are defined below. Degree information is obtained from Banner using one or more of five different paths:

1. Student Goal Data may be loaded from LEARNER SORLCUR/SORLFOS curriculum records.
2. Student Goal Data may be loaded from SGBSTDN curriculum data.
3. Student Goal Data may be loaded from SGBSTDN DUAL degree data.
4. Applicant Goal Data may be loaded from ADMISSIONS SORLCUR/SORLFOS curriculum records.
5. Applicant Goal Data may be loaded from SARADAP curriculum data.

The rad_goal_dtl contains a School code (Level) and Degree code combination that makes the record unique for a student and applicant.

Refer to the Applicant Processing Special Topic in the Banner Considerations document for a more detailed explanation of how the student and applicant data is extracted from the Banner database and imported into the Degree Works database.

The following five sections of the Goal Record documentation outline the columns extracted from Banner for each retrieval path for the rad_goal_dtl.

PATH 1: Student LEARNER SORLCUR/SORLFOS records

Student Goal Data will be loaded from LEARNER SORLCUR/SORLFOS curriculum records if at least one valid pair of records (one SORLCUR record and at least one related SORLFOS record) are found based on the SQL for these two Banner tables defined in the integration.banner.extract.config Shepherd setting.

The following SQL statements are executed in an attempt to retrieve the Concurrent Curriculum records from Banner contained in the SORLCUR and SORLFOS tables for a given STUDENT (the SORLCUR_PIDM in the SQL below is replaced by the actual Banner SPRIDEN_PIDM in the rBannerStuSPRIDEN.zPidm field when executed):

Make sure to customize the SORLCUR and SORLFOS entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

In the SQL samples included below, the SELECT and ORDER BY clauses are hardwired in the Banner extract program. The FROM and WHERE clauses come from the integration.banner.extract.config Shepherd setting that should be reviewed and customized for your site. The SQL as defined below is NOT executable because the <students-pidm> reference will be replaced with the SPRIDEN_PIDM being processed by the extract.

SQL used to read the SORLCUR table:

```
SELECT a.SORLCUR_PIDM,
       a.SORLCUR_SEQNO,
       a.SORLCUR_LMOD_CODE,
       a.SORLCUR_TERM_CODE,
       a.SORLCUR_KEY_SEQNO,
       a.SORLCUR_PRIORITY_NO,
       a.SORLCUR_CACT_CODE,
       a.SORLCUR_LEVL_CODE,
       a.SORLCUR_COLL_CODE,
       a.SORLCUR_DEGC_CODE,
       a.SORLCUR_TERM_CODE_CTLG,
       a.SORLCUR_PROGRAM,
       a.SORLCUR_STYP_CODE,
       a.SORLCUR_TERM_CODE_ADMIT,
       a.SORLCUR_PROGRAM

  FROM <SORLCUR-from>
 WHERE <SORLCUR-where>
       a.SORLCUR_PIDM = <students-pidm>

 ORDER BY a.SORLCUR_PRIORITY_NO;
```

SQL used to read the corresponding SORLFOS table:

```

SELECT a.SORLFOS_PIDM,
       a.SORLFOS_LCUR_SEQNO,
       a.SORLFOS_LFST_CODE,
       a.SORLFOS_TERM_CODE,
       a.SORLFOS_PRIORITY_NO,
       a.SORLFOS_CSTS_CODE,
       a.SORLFOS_MAJR_CODE,
       a.SORLFOS_TERM_CODE_CTLG,
       a.SORLFOS_DEPT_CODE,
       a.SORLFOS_MAJR_CODE_ATTACH

  FROM <SORLFOS-from>
 WHERE <SORLFOS-where>
       a.SORLFOS_PIDM = <students-pidm>

 ORDER BY a.SORLCUR_PRIORITY_NO,
          a.SORLFOS_PRIORITY_NO;

```

The ORDER BY example in the above SORLFOS SQL section is a default.

- If there is not an ORDER BY in the SORLFOS section, a default ORDER BY is added. The following determines what that ORDER BY will default to:
 - If there is a SORLCUR in the where-clause, the default is:
`ORDER BY b.SORLCUR_PRIORITY_NO,
 a.SORLFOS_PRIORITY_NO`
 - If there is not a SORLCUR in the where-clause, the default is:
`ORDER BY a.SORLFOS_PRIORITY_NO`

Note: The SORLCUR must be in caps to be recognized by the program.

- To add an ORDER BY to the SORLFOS section, the entry would look something like:
`ORDER BY a.SORLFOS_LCUR_SEQNO,
 a.SORLFOS_PRIORITY_NO`

If data is returned from these SQL calls, then the following columns will be used to populate an unlimited number of rad_goal_dtl records. Each rad_goal_dtl created will have a unique School (Level)/Degree code combination.

These records are bridged into the RAD_GOAL_DTL table in Degree Works.

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goal_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R027GOAL for the rad_goal_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R027GOAL200810..A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.

Field Name	Comments
rad_school (level)	SORLCUR_LEVL_CODE This element defines the School (Level) in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. This code must match the bridged School code on the rad_class_dtl.
rad_degree_code	The SORLCUR_DEGC_CODE Multiple rad_goal_dtl records may be created if multiple SORLCUR records are retrieved with different School/Degree combinations using the SQL above. If SORLCUR records containing the same School and Degree are retrieved only one rad_goal_dtl record will be generated. If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goaldata_dtl.rad_program. This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.
rad_catalog_yr	SORLCUR_TERM_CODE_CTLG If this catalog term is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the SORLCUR_TERM_CODE is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. This element defines the catalog year in effect for the student's degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the student's progress towards completing the degree.
rad_stu_level	A special Banner function call, F_CLASS_CALC_FCN, is made using the PIDM, LEVL_CODE and TERM_CODE specified in the integration.banner.extract.config Shepherd setting to generate the student's Class Standing code. A set of special records with a key of CALCFCN are included in this configuration file. The default set of CALCFCNrecords includes the FROM and WHERE clauses from the SGBSTDN default entry. Change this set of CALCFCN records as appropriate for your site. The Class Standing calculated from Banner will be loaded into this field. This element is the class level of the student within the university. Examples: 01 for Freshman, 04 for Senior, 05 for Graduate Student, 06 for PhD Student.
rad_term	Obsolete – loaded with spaces.
rad_degree_src	Degree Source

Field Name	Comments
	Set to S for SORLCUR LEARNER records

PATH 2: Student SGBSTDN record

Student Goal Data will be loaded from SGBSTDN curriculum data if NO Student and NO Applicant curriculum records are found in the SORLCUR/SORLFOS tables for the ID code being processed. If SORLCUR and SORLFOS records are NOT found for the given student the SGBSTDN table is read to obtain the goal data.

Make sure to customize the SGBSTDN entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

SQL used to read the SGBSTDN table:

```
SELECT a.SGBSTDN_PIDM,
       a.SGBSTDN_TERM_CODE_EFF ,
       a.SGBSTDN_STST_CODE ,
       a.SGBSTDN_LEVL_CODE ,
       a.SGBSTDN_STYP_CODE ,
       a.SGBSTDN_COLL_CODE_1 ,
       a.SGBSTDN_DEGC_CODE_1 ,
       a.SGBSTDN_MAJR_CODE_1 ,
       a.SGBSTDN_MAJR_CODE_MINR_1 ,
       a.SGBSTDN_MAJR_CODE_MINR_1_2 ,
       a.SGBSTDN_MAJR_CODE_CONC_1 ,
       a.SGBSTDN_MAJR_CODE_CONC_1_2 ,
       a.SGBSTDN_MAJR_CODE_CONC_1_3 ,
       a.SGBSTDN_COLL_CODE_2 ,
       a.SGBSTDN_DEGC_CODE_2 ,
       a.SGBSTDN_MAJR_CODE_2 ,
       a.SGBSTDN_MAJR_CODE_MINR_2 ,
       a.SGBSTDN_MAJR_CODE_MINR_2_2 ,
       a.SGBSTDN_MAJR_CODE_CONC_2 ,
       a.SGBSTDN_MAJR_CODE_CONC_2_2 ,
       a.SGBSTDN_MAJR_CODE_CONC_2_3 ,
       a.SGBSTDN_ADV_R_PIDM ,
       a.SGBSTDN_MAJR_CODE_1_2 ,
       a.SGBSTDN_MAJR_CODE_2_2 ,
       a.SGBSTDN_ACYR_CODE ,
       a.SGBSTDN_DEPT_CODE ,
       a.SGBSTDN_DEPT_CODE_2 ,
       a.SGBSTDN_DEGC_CODE_DUAL ,
       a.SGBSTDN_LEVL_CODE_DUAL ,
       a.SGBSTDN_DEPT_CODE_DUAL ,
       a.SGBSTDN_COLL_CODE_DUAL ,
       a.SGBSTDN_MAJR_CODE_DUAL ,
       a.SGBSTDN_TERM_CODE_CTLG_1 ,
       a.SGBSTDN_DEPT_CODE_1_2 ,
       a.SGBSTDN_MAJR_CODE_CONC_121 ,
       a.SGBSTDN_MAJR_CODE_CONC_122 ,
```

```

a.SGBSTDN_MAJR_CODE_CONC_123,
a.SGBSTDN_TERM_CODE_CTLG_2,
a.SGBSTDN_LEVL_CODE_2,
a.SGBSTDN_DEPT_CODE_2_2,
a.SGBSTDN_MAJR_CODE_CONC_221,
a.SGBSTDN_MAJR_CODE_CONC_222,
a.SGBSTDN_MAJR_CODE_CONC_223

FROM SGBSTDN a
WHERE a.SGBSTDN_TERM_CODE_EFF =
      (SELECT MAX(b.SGBSTDN_TERM_CODE_EFF)
       FROM SGBSTDN b WHERE b.SGBSTDN_PIDM = a.SGBSTDN_PIDM)
AND
a.SGBSTDN_PIDM = <students-pidm>

```

If data is returned from these SQL calls, the following columns will be used to populate an unlimited number of rad_goal_dtl records. Each rad_goal_dtl created will have a unique School (Level)/Degree code combination.

The order of the goal records for students with multiple degree goals determines the order in which they will display in the drop-down at the top of the dashboard. For example, if a student has a goal for both BA and MBA, and the BA record is bridged before the MBA goal record, the BA degree will appear first in the degree drop-down. Because the Banner extract is sorting the SORLCUR records by the PRIORITY_NO column, the degrees will appear in the drop-down sorted by this column. The degree with PRIORITY_NO of 1 will come first, the degree with PRIORITY_NO of 2 will come second, and so on.

This record contains information for the rad_goal_dtl table. Total length = 1000 bytes.

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goal_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R027GOAL for the rad_goal_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R027GOAL200810..A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
rad_school (level)	SGBSTDN_LEVL_CODE
	This element defines the school in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. This code must match the bridged School code on the rad_class_dtl.
rad_degree_code	The SGBSTDN_DEGC_CODE_1 is used. If the SGBSTDN_DEGC_CODE_2 is NOT blank and it does NOT match SGBSTDN_DEGC_CODE_1 then a second rad_goal_dtl will be created.

Field Name	Comments
	<p>If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goaldata_dtl.rad_program.</p>
	<p>This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.</p>
rad_catalog_yr	<p>The SGBSTDN_TERM_CODE_CTLG1 is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If this code is blank then the SGBSTDN_TERM_CODE_CTL2 will be used to lookup UCX-STU016. If both CTLG codes are blank the SGBSTDN_TERM_CODE_EFF will be used to lookup UCX-STU016.</p> <p>This element defines the catalog year in effect for the student's degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the student's progress towards completing the degree.</p>
rad_stu_level	<p>A special Banner function call, F_CLASS_CALC_FCN, is made using the PIDM, LEVL_CODE and TERM_CODE specified in the integration.banner.extract.config Shepherd setting to generate the student's Class Standing code. A set of special records with a key of CALCFCN are included in this configuration file. The default set of CALCFCN records includes the FROM and WHERE clauses from the SGBSTDN default entry. Change this set of CALCFCN records as appropriate for your site. The Class Standing calculated from Banner will be loaded into this field.</p> <p>This element is the class level of the student within the university. Examples: 01 for Freshman, 04 for Senior, 05 for Graduate Student, 06 for PhD Student.</p>
rad_term	<p>Active Term</p> <p>This element defines the Active Term, which is used by Degree Works. It is particularly important for the Planner and Financial Aid audits as the Active Term is really the Current Term for processing purposes.</p> <p>Refer to the rad_term definition in the R011PRIM record for a discussion on how the Active Term is calculated.</p>
rad_degree_src	<p>Degree Source</p> <p>Set to S for SGBSTDN student records</p>

PATH 3: Student SGBSTDN DUAL Degree Record

Student Goal may be loaded from SGBSTDN DUAL degree data. If SORLCUR and SORLFOS records are found for the given student the SGBSTDN table is still read if DUAL Degree data is desired.

If the UCX-CFG020 BANNER Check Dual Degree flag is set to Y and Dual Degree information exists on the Student Base table, SGBSTDN, it will be used to create a ad_goal_dtl data.

Regardless of whether the rest of the degree data is found in the SORLCUR/SORLFOS tables or the SGBSTDN table for a given student the Dual Degree data will be checked and imported if found.

If the SGBSTDN_LEVL_CODE_DUAL and SGBSTDN_DEGC_CODE_DUAL are unique and do not exist in the internal degree table calculated from the PATH #1 or PATH #2 scenarios above and SGBSTDN_MAJR_CODE_DUAL is NOT blank, then the LEVL_CODE_DUAL and DEGC_CODE_DUAL will be used in creating the rad_goal_dtl data for a student. The Catalog Year will be loaded using SGBSTDN_TERM_CODE_CTLG_2 first if not blank or null. If the Catalog Year is blank SGBSTDN_TERM_CODE_CTLG_1 will be used to obtain the Catalog Year. If neither of the SGBSTDN_TERM_CODE_CTLG_## years is loaded, the SGBSTDN_TERM_CODE_EFF will be used to get the Catalog Year.

If the SGBSTDN_LEVL_CODE_DUAL and SGBSTDN_DEGC_CODE_DUAL are NOT unique and are found in the internal degree table the DUAL degree data on the SGBSTDN record will be SKIPPED and NOT imported into Degree Works.

Make sure to customize the SGBSTDN entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

SQL used to read the SGBSTDN table:

```
SELECT a.SGBSTDN_PIDM,
       a.SGBSTDN_TERM_CODE_EFF,
       a.SGBSTDN_STST_CODE,
       a.SGBSTDN_LEVL_CODE,
       a.SGBSTDN_STYP_CODE,
       a.SGBSTDN_COLL_CODE_1,
       a.SGBSTDN_DEGC_CODE_1,
       a.SGBSTDN_MAJR_CODE_1,
       a.SGBSTDN_MAJR_CODE_MINR_1,
       a.SGBSTDN_MAJR_CODE_MINR_1_2,
       a.SGBSTDN_MAJR_CODE_CONC_1,
       a.SGBSTDN_MAJR_CODE_CONC_1_2,
       a.SGBSTDN_MAJR_CODE_CONC_1_3,
       a.SGBSTDN_COLL_CODE_2,
       a.SGBSTDN_DEGC_CODE_2,
       a.SGBSTDN_MAJR_CODE_2,
       a.SGBSTDN_MAJR_CODE_MINR_2,
       a.SGBSTDN_MAJR_CODE_MINR_2_2,
       a.SGBSTDN_MAJR_CODE_CONC_2,
       a.SGBSTDN_MAJR_CODE_CONC_2_2,
       a.SGBSTDN_MAJR_CODE_CONC_2_3,
       a.SGBSTDN_ADVR_PIDM,
       a.SGBSTDN_MAJR_CODE_1_2,
       a.SGBSTDN_MAJR_CODE_2_2,
       a.SGBSTDN_ACYR_CODE,
       a.SGBSTDN_DEPT_CODE,
       a.SGBSTDN_DEPT_CODE_2,
       a.SGBSTDN_DEGC_CODE_DUAL,
       a.SGBSTDN_LEVL_CODE_DUAL,
```

```

a.SGBSTDN_DEPT_CODE_DUAL,
a.SGBSTDN_COLL_CODE_DUAL,
a.SGBSTDN_MAJR_CODE_DUAL,
a.SGBSTDN_TERM_CODE_CTLG_1,
a.SGBSTDN_DEPT_CODE_1_2,
a.SGBSTDN_MAJR_CODE_CONC_121,
a.SGBSTDN_MAJR_CODE_CONC_122,
a.SGBSTDN_MAJR_CODE_CONC_123,
a.SGBSTDN_TERM_CODE_CTLG_2,
a.SGBSTDN_LEVL_CODE_2,
a.SGBSTDN_DEPT_CODE_2_2,
a.SGBSTDN_MAJR_CODE_CONC_221,
a.SGBSTDN_MAJR_CODE_CONC_222,
a.SGBSTDN_MAJR_CODE_CONC_223

FROM SGBSTDN a
WHERE a.SGBSTDN_TERM_CODE_EFF =
      (SELECT MAX(b.SGBSTDN_TERM_CODE_EFF)
       FROM SGBSTDN b WHERE b.SGBSTDN_PIDM = a.SGBSTDN_PIDM)
AND
a.SGBSTDN_PIDM = <students-pidm>

```

If data is returned from these SQL calls, then the following columns will be used to populate a DUAL Degree rad_goal_dtl record. Each rad_goal_dtl created will have a unique School (Level)/Degree code combination.

This record contains information for the rad_goal_dtl table. Total length = 1000 bytes.

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goal_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R027GOAL for the rad_goal_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R027GOAL200810..A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
rad_school (level)	SGBSTDN_LEVL_CODE_DUAL This element defines the school in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. This code must match the bridged School code on the rad_class_dtl.
rad_degree_code	SGBSTDN_DEGC_CODE_DUAL If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goaldtdata_dtl.rad_program.

Field Name	Comments
	<p>This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.</p>
rad_catalog_yr	<p>The SGBSTDN_TERM_CODE_CTLG2 is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year if this code is NOT blank. Otherwise the SGBSTDN_TERM_CODE_CTL1 will be used to lookup UCX-STU016. If both CTLG codes are blank the SGBSTDN_TERM_CODE_EFF will be used to lookup UCX-STU016.</p>
	<p>This element defines the catalog year in effect for the student's degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the student's progress towards completing the degree.</p>
rad_stu_level	<p>A special Banner function call, F_CLASS_CALC_FCN, is made using the PIDM, LEVL_CODE and TERM_CODE specified in the integration.banner.extract.config Shepherd setting to generate the student's Class Standing code. A set of special records with a key of CALCFCN are included in this configuration file. The default set of CALCFCN records includes the FROM and WHERE clauses from the SGBSTDN default entry. Change this set of CALCFCN records as appropriate for your site. The Class Standing calculated from Banner will be loaded into this field.</p>
	<p>This element is the class level of the student within the university. Examples: 01 for Freshman, 04 for Senior, 05 for Graduate Student, 06 for PhD Student.</p>
rad_term	<p>Active Term</p>
	<p>This element defines the Active Term, which is used by Degree Works. It is particularly important for the Planner and Financial Aid audits as the Active Term is really the Current Term for processing purposes.</p>
	<p>Refer to the rad_term definition in the R011PRIM record for a discussion on how the Active Term is calculated.</p>
rad_degree_src	<p>Degree Source</p>
	<p>Set to S for SGBSTDN student records</p>

PATH 4: Applicant ADMISSIONS SORLCUR/SORLFOS Records

Applicant Goal Data may be loaded from ADMISSIONS SORLCUR/SORLFOS curriculum records if applicant processing is set appropriately for your site and the ID code being processed has valid applicant data. Review the three Applicant oriented flags in the UCX-CFG020 BANNER record and set them appropriately for your site.

The following SQL statements are executed in an attempt to retrieve the Concurrent Curriculum records from Banner contained in the SORLCUR and SORLFOS tables for a given APPLICANT (the :rBannerStuSPRIDEN.zPidm field is replaced by the actual SORLCUR_PIDM when executed):

Make sure to customize the SORLCUR2 and SORLFAS2 entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

SQL used to read the SORLCUR table:

```
SELECT a.SORLCUR_PIDM,
       a.SORLCUR_SEQNO,
       a.SORLCUR_LMOD_CODE,
       a.SORLCUR_TERM_CODE,
       a.SORLCUR_KEY_SEQNO,
       a.SORLCUR_PRIORITY_NO,
       a.SORLCUR_CACT_CODE,
       a.SORLCUR_LEVL_CODE,
       a.SORLCUR_COLL_CODE,
       a.SORLCUR_DEGC_CODE,
       a.SORLCUR_TERM_CODE_CTLG,
       a.SORLCUR_PROGRAM,
       a.SORLCUR_STYP_CODE

  FROM SORLCUR a, STVTERM t, SARADAP c
 WHERE (SELECT COUNT(*) FROM SHTRCR
       WHERE SHTRCR_PIDM = c.SARADAP_PIDM) > 0
       AND t.STVTERM_START_DATE > SYSDATE
       AND ((SELECT COUNT(*) FROM SGBSTDN
             WHERE SGBSTDN_PIDM = c.SARADAP_PIDM) < 1
             OR (SELECT COUNT(*) FROM STVSTST, SGBSTDN p
                  WHERE STVSTST_CODE = p.SGBSTDN_STST_CODE
                  AND STVSTST_REG_IND = 'Y'
                  AND p.SGBSTDN_TERM_CODE_EFF = (SELECT MAX
                        (o.SGBSTDN_TERM_CODE_EFF) FROM SGBSTDN o
                     WHERE o.SGBSTDN_PIDM = p.SGBSTDN_PIDM
                     AND o.SGBSTDN_TERM_CODE_EFF <
                         c.SARADAP_TERM_CODE_ENTRY)) < 1)
             AND a.SORLCUR_CACT_CODE = 'ACTIVE'
             AND a.SORLCUR_LMOD_CODE = 'ADMISSIONS'
             AND a.SORLCUR_SEQNO = (SELECT MAX(b.SORLCUR_SEQNO)
                   FROM SORLCUR b
                   WHERE b.SORLCUR_PIDM = a.SORLCUR_PIDM
                   AND b.SORLCUR_PRIORITY_NO = a.SORLCUR_PRIORITY_NO
                   AND b.SORLCUR_LMOD_CODE = 'ADMISSIONS')
             AND a.SORLCUR_PIDM = c.SARADAP_PIDM
             AND a.SORLCUR_TERM_CODE = c.SARADAP_TERM_CODE_ENTRY
             AND a.SORLCUR_KEY_SEQNO = c.SARADAP_APPL_NO
             AND t.STVTERM_CODE = c.SARADAP_TERM_CODE_ENTRY
       AND
           a.SORLCUR_PIDM = <applicants-pidm>

  ORDER BY a.SORLCUR_SEQNO
```

SQL used to read the SORLFOS table:

```

SELECT a.SORLFOS_PIDM,
       a.SORLFOS_LCUR_SEQNO,
       a.SORLFOS_LFST_CODE,
       a.SORLFOS_TERM_CODE,
       a.SORLFOS_PRIORITY_NO,
       a.SORLFOS_CSTS_CODE,
       a.SORLFOS_MAJR_CODE,
       a.SORLFOS_TERM_CODE_CTLG,
       a.SORLFOS_DEPT_CODE,
       a.SORLFOS_MAJR_CODE_ATTACH

  FROM SORLFOS a, SORLCUR b, STVTERM t, SARADAP d
 WHERE (SELECT COUNT(*) FROM SHRTRCR
       WHERE SHRTRCR_PIDM = d.SARADAP_PIDM) > 0
       AND t.STVTERM_START_DATE > SYSDATE
       AND ((SELECT COUNT(*) FROM SGBSTDN
             WHERE SGBSTDN_PIDM = d.SARADAP_PIDM) < 1
             OR (SELECT COUNT(*) FROM STVSTST, SGBSTDN p
                  WHERE STVSTST_CODE = p.SGBSTDN_STST_CODE
                  AND STVSTST_REG_IND = 'Y'
                  AND p.SGBSTDN_TERM_CODE_EFF = (SELECT MAX
                        (o.SGBSTDN_TERM_CODE_EFF) FROM SGBSTDN o
                     WHERE o.SGBSTDN_PIDM = p.SGBSTDN_PIDM
                     AND o.SGBSTDN_TERM_CODE_EFF <
                         d.SARADAP_TERM_CODE_ENTRY)) < 1)
             AND b.SORLCUR_CACT_CODE = 'ACTIVE'
             AND b.SORLCUR_LMOD_CODE = 'ADMISSIONS'
             AND b.SORLCUR_SEQNO = (SELECT MAX(f.SORLCUR_SEQNO)
                   FROM SORLCUR f
                   WHERE f.SORLCUR_PIDM = b.SORLCUR_PIDM
                     AND f.SORLCUR_PRIORITY_NO = b.SORLCUR_PRIORITY_NO
                     AND f.SORLCUR_LMOD_CODE = 'ADMISSIONS')
                     AND b.SORLCUR_PIDM = d.SARADAP_PIDM
                     AND b.SORLCUR_TERM_CODE = d.SARADAP_TERM_CODE_ENTR
                     AND b.SORLCUR_KEY_SEQNO = d.SARADAP_APPL_NO
                     AND t.STVTERM_CODE = d.SARADAP_TERM_CODE_ENTRY
                     AND a.SORLFOS_CSTS_CODE = 'INPROGRESS'
                     AND a.SORLFOS_CACT_CODE = 'ACTIVE'
                     AND a.SORLFOS_PIDM = b.SORLCUR_PIDM
                     AND a.SORLFOS_LCUR_SEQNO = b.SORLCUR_SEQNO
                     AND a.SORLFOS_SEQNO =
                         (SELECT MAX(1.SORLFOS_SEQNO) FROM SORLFOS 1
                          WHERE 1.SORLFOS_PIDM = b.SORLCUR_PIDM
                            AND 1.SORLFOS_PRIORITY_NO = a.SORLFOS_PRIORITY_NO
                            AND 1.sorlfos_csts_code = 'INPROGRESS'
                            AND 1.SORLFOS_LCUR_SEQNO = b.SORLCUR_SEQNO)
                     AND
                     a.SORLFOS_PIDM = <applicants-pidm>

 ORDER BY a.SORLFOS_LCUR_SEQNO, a.SORLFOS_PRIORITY_NO

```

This record contains information for the rad_goal_dtl table. Total length = 1000 bytes.

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goal_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R027GOAL for the rad_goal_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R027GOAL201020..A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a applicant at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
rad_school (level)	SORLCUR_LEVEL_CODE This element defines the School (Level) in which the applicant is applying. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool.
rad_degree_code	SORLCUR_DEGC_CODE Multiple rad_goal_dtl records may be created if multiple SORLCUR records are retrieved with different School/Degree combinations using the SQL above. If SORLCUR records containing the same School and Degree are retrieved only one rad_goal_dtl record will be generated. If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goaldata_dtl.rad_program. This element contains the code that identifies the applicant's intended degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.
rad_catalog_yr	SORLCUR_TERM_CODE_CTLG If this catalog term is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the SORLCUR_TERM_CODE is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. This element defines the catalog year in effect for the applicant's intended degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the applicant's progress towards completing the degree (for example, transfer classes, test scores, custom data, and so on will be evaluated when audits are run for admissions applicants).
rad_stu_level	A special Banner function call, F_CLASS_CALC_FCN, is made using the PIDM, LEVL_CODE and TERM_CODE specified in the integration.banner.extract.config Shepherd setting to generate the applicant's Class Standing code. A set of special records with a key of

Field Name	Comments
	CALCFCN are included in this configuration file. The default set of CALCFCN records includes the FROM and WHERE clauses from the SGBSTDN default entry. Change this set of CALCFCN records as appropriate for your site. The Class Standing calculated from Banner will be loaded into this field.
	This element is the class level of the applicant within the university. Examples: 01 for Freshman, 04 for Senior, 05 for Graduate Applicant, 06 for PhD Applicant.
rad_term	<p>Active Term</p> <p>This element defines the Active Term, which is used by Degree Works. It is particularly important for the Planner and Financial Aid audits as the Active Term is really the Current Term for processing purposes.</p> <p>Refer to the rad_term definition in the R011PRIM record for a discussion on how the Active Term is calculated.</p>
rad_degree_src	<p>Degree Source</p> <p>Set to A for SORLCUR ADMISSIONS records</p>

PATH 5: Applicant SARADAP Record

Applicant Goal Data may be loaded from SARADAP curriculum data if applicant processing is set appropriately for your site and the ID code being processed has valid applicant data. Review the three Applicant oriented flags in the UCX-CFG020 BANNER record and make sure they are set appropriately for your site.

If SORLCUR and SORLFOS records are NOT found for the given student or applicant and the UCX-CFG020 BANNER Load SARADAP Goals = Y the SARADAP table is read to obtain the goal data. If one or more valid SARADAP records are found for an admissions applicant the appropriate rad_goal_dtl records will be generated.

Make sure to customize the SARADAP entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

SQL used to read the SARADAP table:

```
SELECT a.SARADAP_PIDM,
       a.SARADAP_TERM_CODE_EFF ,
       a.SARADAP_STST_CODE ,
       a.SARADAP_LEVL_CODE ,
       a.SARADAP_STYP_CODE ,
       a.SARADAP_COLL_CODE_1 ,
       a.SARADAP_DEGC_CODE_1 ,
       a.SARADAP_MAJR_CODE_1 ,
       a.SARADAP_MAJR_CODE_MINR_1 ,
       a.SARADAP_MAJR_CODE_MINR_1_2 ,
```

```

a.SARADAP_MAJR_CODE_CONC_1,
a.SARADAP_MAJR_CODE_CONC_1_2,
a.SARADAP_MAJR_CODE_CONC_1_3,
a.SARADAP_COLL_CODE_2,
a.SARADAP_DEGC_CODE_2,
a.SARADAP_MAJR_CODE_2,
a.SARADAP_MAJR_CODE_MINR_2,
a.SARADAP_MAJR_CODE_MINR_2_2,
a.SARADAP_MAJR_CODE_CONC_2,
a.SARADAP_MAJR_CODE_CONC_2_2,
a.SARADAP_MAJR_CODE_CONC_2_3,
a.SARADAP_ADVR_PIDM,
a.SARADAP_MAJR_CODE_1_2,
a.SARADAP_MAJR_CODE_2_2,
a.SARADAP_ACYR_CODE,
a.SARADAP_DEPT_CODE,
a.SARADAP_DEPT_CODE_2,
a.SARADAP_DEGC_CODE_DUAL,
a.SARADAP_LEVL_CODE_DUAL,
a.SARADAP_DEPT_CODE_DUAL,
a.SARADAP_COLL_CODE_DUAL,
a.SARADAP_MAJR_CODE_DUAL,
a.SARADAP_TERM_CODE_CTLG_1,
a.SARADAP_DEPT_CODE_1_2,
a.SARADAP_MAJR_CODE_CONC_121,
a.SARADAP_MAJR_CODE_CONC_122,
a.SARADAP_MAJR_CODE_CONC_123,
a.SARADAP_TERM_CODE_CTLG_2,
a.SARADAP_LEVL_CODE_2,
a.SARADAP_DEPT_CODE_2_2,
a.SARADAP_MAJR_CODE_CONC_221,
a.SARADAP_MAJR_CODE_CONC_222,
a.SARADAP_MAJR_CODE_CONC_223

FROM SARADAP a
WHERE a.SARADAP_TERM_CODE_ENTRY =
(SELECT MAX(b.SARADAP_TERM_CODE_ENTRY)
 FROM SARADAP b
 WHERE b.SARADAP_PIDM = a.SARADAP_PIDM)
AND
a.SARADAP_PIDM = <applicant's-pidm>

```

If data is returned from this SQL call, then the following columns will be used to populate an unlimited number of rad_goal_dtl records. Each rad_goal_dtl created will have a unique School (Level)/Degree code combination.

This record contains information for the rad_goal_dtl table. Total length = 1000 bytes.

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goal_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R027GOAL for the rad_goal_dtl), an 8-byte term and a 2-byte ACTION-

Field Name	Comments
	FLAG (A=Add, D=Delete). Example: 00129918..R027GOAL201020..A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
rad_school (level)	SARADAP_LEVEL_CODE This element defines the school in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. This code must match the bridged School code on the rad_class_dtl.
rad_degree_code	The SARADAP_DEGC_CODE This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.
rad_catalog_yr	SARADAP_TERM_CODE_CTLG_1 SARADAP_TERM_CODE_CTLG_2 If Degree #1(SARADAP_DEGC_CODE_1) is being processed and SARADAP_TERM_CODE_CTLG_1 is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the SARADAP_TERM_CODE_ENTRY is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If Degree #2 (SARADAP_DEGC_CODE_2) is being processed and SARADAP_TERM_CODE_CTLG_2 is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the Catalog Year from SARADAP_TERM_CODE_CTLG_1 is used. If still blank, SARADAP_TERM_CODE_ENTRY is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. This element defines the catalog year in effect for the student's degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the student's progress towards completing the degree.
rad_stu_level	A special Banner function call, F_CLASS_CALC_FCN, is made using the PIDM, LEVEL_CODE and TERM_CODE specified in the integration.banner.extract.config Shepherd setting to generate the student's Class Standing code. A set of special records with a key of CALCFCN are included in this configuration file. The default set of CALCFCN records includes the FROM and WHERE clauses from the SARADAP default entry. Change this set of CALCFCN records as appropriate for your site. The Class Standing calculated from Banner will be loaded into this field.

Field Name	Comments
	This element is the class level of the student within the university. Examples: 01 for Freshman, 04 for Senior, 05 for Graduate Student, 06 for PhD Student.
rad_term	<p>Active Term</p> <p>This element defines the Active Term, which is used by Degree Works. It is particularly important for the Planner and Financial Aid audits as the Active Term is really the Current Term for processing purposes.</p> <p>Refer to the rad_term definition in the R011PRIM record for a discussion on how the Active Term is calculated.</p>
rad_degree_src	<p>Degree Source</p> <p>Set to A for SARADAP Applicant records</p>

R033GDTA - Goal Data Record

Updated: September 29, 2023

Several Banner tables are used to obtain the data in this record.

Only the pieces of data obtained from Banner tables are defined below. Degree information is obtained from Banner using one or more of the following five paths:

1. Student Goal Data may be loaded from LEARNER SORLCUR/SORLFOS curriculum records.
2. Student Goal Data may be loaded from SGBSTDN curriculum data.
3. Student Goal Data may be loaded from SGBSTDN DUAL degree data.
4. Applicant Goal Data may be loaded from ADMISSIONS SORLCUR/SORLFOS curriculum records.
5. Applicant Goal Data may be loaded from SARADAP curriculum data.

The rad_goaldata_dtl contains a School code (Level) and Degree code combination that makes the record unique for a student. Advisors will be loaded based on the UCX-CFG020 BANNER Advisor Method, and the same advisors will be loaded onto every rad_goaldata_dtl created:

A Allow 4 rad_advr# fields to be loaded based on three SGRADVR_ADVR_CODES for each advisor. The valid advisor ID codes will be written to the rad_goaldata_dtl rad_goal_value with a rad_goal_code of ADVISOR.

C Two major advisor ID codes may be loaded in addition to one minor advisor ID. The manner in which the data is gathered is slightly different, but the ultimate goal is the same: load major data first, followed by minor data for a given School/Degree combination. All valid advisors found will be written to the rad_goaldata_dtl rad_goal_value with a rad_goal_code of ADVISOR.

S The Primary Advisor will be loaded first (SGRADVR_PRIM_IND = Y). Then all additional advisors selected through the SQL specified in the integration.banner.extract.config Shepherd setting will be written to the rad_goaldata_dtl rad_goal_value with a rad_goal_code of ADVISOR. However, if the primary advisor is not found, no secondary advisors will be loaded resulting in no advisors being extracted for the student.

Refer to the Applicant Processing Special Topic in the Banner Considerations document for a more detailed explanation of how the student and applicant data is extracted from the Banner database and imported into the Degree Works database.

The following five sections of the GoalData Record documentation outline the columns extracted from Banner for each retrieval path for the rad_goaldata_dtl:

PATH 1: Student LEARNER SORLCUR/SORLFOS Records

Student Goal Data will be loaded from LEARNER SORLCUR/SORLFOS curriculum records if at least one valid pair of records (one SORLCUR record and at least one related SORLFOS record) are found based on the SQL specified in the integration.banner.extract.config Shepherd setting for these two Banner tables. The following SQL statements are executed in an attempt to retrieve the Concurrent Curriculum records from Banner contained in the SORLCUR/SORLFOS tables for a given STUDENT (the SORLCUR_PIDM in the SQL below is replaced by the actual Banner SPRIDEN_PIDM in the rBannerStusPRIDEN.zPidm field when executed):

Make sure to customize the SORLCUR/SORLFOS entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

See the SORLCUR SQL and SORLFOS SQL in the [R027GOAL - Goal Record](#) topic.

This record contains information for the rad_goaldata_dtl table. This is a required table for Degree Works but not for Transfer Equivalency. The rad_goaldata_dtl records that are bridged must have unique ID, term, school and degree combinations. They must have a corresponding rad_goal_dtl record with the same ID, term, school and degree.

These records are bridged into the RAD_GOALDATA_DTL table in Degree Works.

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goaldata_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R033GDTA for the rad_goaldata_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R033GDTA200810..A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
rad_school	SORLCUR_LEVL_CODE This element defines the School (Level) in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool.

Field Name	Comments
	This code must match the bridged School code on the rad_class_dtl.
rad_degree_code	SORLCUR_DEGC_CODE
	<p>Multiple rad_goal_dtl records may be created if multiple SORLCUR records are retrieved with different School/Degree combinations using the SQL above. If SORLCUR records containing the same School and Degree are retrieved only one rad_goal_dtl record will be generated.</p>
	<p>If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goaldata_dtl.rad_program.</p>
	<p>This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.</p>
rad_catalog_yr	SORLFOS_TERM_CODE_CTLG
	<p>If this catalog term is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the SORLCUR_TERM_CODE is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year.</p>
	<p>This element defines the catalog year in effect for the student's goaldata record: COLLEGE, CONC, MAJOR, MINOR, PROGRAM, or SPEC. The catalog year determines which set of requirement definitions (for example, which requirement block) should be used when evaluating the student's progress towards completing the degree.</p>
rad_goal_code	<p>This element is the code associated with the type of record involved in a degree/goal. Valid values are:</p> <ul style="list-style-type: none"> ADVISOR – Advisors from SGRADVR COLLEGE – SORLCUR_COLL_CODE CONC – Concentration Codes from SORLFOS MAJOR – Major Codes from SORLFOS MINOR – Minor Codes from SORLFOS PROGRAM – Program Codes from SORLCUR SPEC – Certification Codes (CERTIFICATION or CERTIFICATE) from SORLFOS LBL – Option Codes (OPTION) from SORLFOS STUSTATUS – STYP Code from SORLCUR

Field Name	Comments
rad_goal_value	<p>This element is the actual rad_goal_value recorded for a given student for this goal code. Leading spaces are removed from any value placed here. Example: " PE" becomes "PE "</p> <p>See the table below for description of Banner Location of data used to populate this item.</p>
rad_goal_seq	<p>This element is the sequence number associated with the Goal Code and Goal Value. They do NOT have to be unique. For example, if there are 1 DEGREE, 1 MAJOR, 1 MINOR, 1 CONC and 1 ADVISOR that are all associated they would have the same sequence number (for example, 0001). If there are 1 DEGREE, 2 MAJORS, 2 MINORS and 1 ADVISOR associated with MAJOR #1 this is how they would look:</p> <p>Goal Code=DEGREE, Goal Value=BS, Goal Seq=0001</p> <p>Goal Code=MAJOR, Goal Value=MATH, Goal Seq=0001</p> <p>Goal Code=MAJOR, Goal Value=CS, Goal Seq = 0002</p> <p>Goal Code=MINOR, Goal Value=PHYS, Goal Seq = 0001</p> <p>Goal Code=MINOR, Goal Value=MIS, Goal Seq = 0002</p> <p>Goal Code=ADVISOR, Goal Value=12345, Goal Seq = 0001</p>
rad_attach_code	This element contains the type of data in the rad_attach_value field. The MAJR_CODE_ATTACH value is found in the list of SORLFOS records and this code is derived from its LFST_CODE. This will be MAJOR, MINOR, CONC or SPEC – though normally it is set to MAJOR as concentrations are normally attached to majors (and not the reverse).
rad_attach_value	The element contains the SORLFOS_MAJR_CODE_ATTACH value. Usually this is populated when the goal_code/value is for a concentration and this field is filled with the attached major code. This is saying that this concentration is attached/associated with this particular major.

Table providing detail of rad_goal_value:

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
ADVISOR	See the ADVISOR Processing discussion at the end of this document.
COLLEGE	SORLCUR_COLL_CODE
CONC	SORLFOS_MAJR_CODE where SORLFOS_LFST_CODE = CONCENTRATION.
MAJOR	The SORLFOS_MAJR_CODE when SORLFOS_LFST_CODE = MAJOR

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
MINOR	SORLFOS_MAJR_CODE when SORLFOS_LFST_CODE = MINOR.
PROGRAM	The SORLCUR_PROGRAM will be loaded if it exists for the given student. If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found in the rad_degree_code on the rad_goal_dtl and rad_goaldata_dtl records.
SPEC	Loaded with the SORLFOS_MAJR_CODE if the SORLFOS_LFST_CODE is equal to CERTIFICATION or CERTIFICATE. The ban40.ec student extract looks for this CERTIFICATION or CERTIFICATE code for a SORLFOS record that is being processed for a corresponding SORLCUR record. If a CERTIFICATION or CERTIFICATE record is found the SORLFOS_MAJR_CODE will be written to the rad_goal_value.
LIBL	Loaded with the SORLFOS_MAJR_CODE if the SORLFOS_LFST_CODE is equal to OPTION. The ban40.ec student extract looks for this OPTION code for a SORLFOS record that is being processed for a corresponding SORLCUR record. If a OPTION record is found the SORLFOS_MAJR_CODE will be written to the rad_goal_value.
STUSTATUS (STYP)	The SORLCUR_STYP_CODE will be loaded if it exists for the given student.

PATH 2: Student SGBSTDN Record

Student Goal Data will be loaded from SGBSTDN curriculum data if NO Student and NO Applicant curriculum records are found in the SORLCUR/SORLFOS tables for the ID code being processed.

If SORLCUR and SORLFOS records are NOT found for the given student the SGBSTDN table is read to obtain the goal data. Make sure to customize the SGBSTDN entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

SQL used to read the SGBSTDN table:

```
SELECT a.SGBSTDN_PIDM,
       a.SGBSTDN_TERM_CODE_EFF,
       a.SGBSTDN_STST_CODE,
       a.SGBSTDN_LEVL_CODE,
```

```

        a.SGBSTDN_STYP_CODE,
        a.SGBSTDN_COLL_CODE_1,
        a.SGBSTDN_DEGC_CODE_1,
        a.SGBSTDN_MAJR_CODE_1,
        a.SGBSTDN_MAJR_CODE_MINR_1,
        a.SGBSTDN_MAJR_CODE_MINR_1_2,
        a.SGBSTDN_MAJR_CODE_CONC_1,
        a.SGBSTDN_MAJR_CODE_CONC_1_2,
        a.SGBSTDN_MAJR_CODE_CONC_1_3,
        a.SGBSTDN_COLL_CODE_2,
        a.SGBSTDN_DEGC_CODE_2,
        a.SGBSTDN_MAJR_CODE_2,
        a.SGBSTDN_MAJR_CODE_MINR_2,
        a.SGBSTDN_MAJR_CODE_MINR_2_2,
        a.SGBSTDN_MAJR_CODE_CONC_2,
        a.SGBSTDN_MAJR_CODE_CONC_2_2,
        a.SGBSTDN_MAJR_CODE_CONC_2_3,
        a.SGBSTDN_AdVR_PIDM,
        a.SGBSTDN_MAJR_CODE_1_2,
        a.SGBSTDN_MAJR_CODE_2_2,
        a.SGBSTDN_ACYR_CODE,
        a.SGBSTDN_DEPT_CODE,
        a.SGBSTDN_DEPT_CODE_2,
        a.SGBSTDN_DEGC_CODE_DUAL,
        a.SGBSTDN_LEVL_CODE_DUAL,
        a.SGBSTDN_DEPT_CODE_DUAL,
        a.SGBSTDN_COLL_CODE_DUAL,
        a.SGBSTDN_MAJR_CODE_DUAL,
        a.SGBSTDN_TERM_CODE_CTLG_1,
        a.SGBSTDN_DEPT_CODE_1_2,
        a.SGBSTDN_MAJR_CODE_CONC_121,
        a.SGBSTDN_MAJR_CODE_CONC_122,
        a.SGBSTDN_MAJR_CODE_CONC_123,
        a.SGBSTDN_TERM_CODE_CTLG_2,
        a.SGBSTDN_LEVL_CODE_2,
        a.SGBSTDN_DEPT_CODE_2_2,
        a.SGBSTDN_MAJR_CODE_CONC_221,
        a.SGBSTDN_MAJR_CODE_CONC_222,
        a.SGBSTDN_MAJR_CODE_CONC_223
      FROM SGBSTDN a
      WHERE a.SGBSTDN_TERM_CODE_EFF =
        (SELECT MAX(b.SGBSTDN_TERM_CODE_EFF)
         FROM SGBSTDN b WHERE b.SGBSTDN_PIDM = a.SGBSTDN_PIDM)
      AND
        a.SGBSTDN_PIDM = <students-pidm>
    
```

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goaldata_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R033GDTA for the rad_goaldata_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R033GDTA200810..A. (periods represent spaces).

Field Name	Comments
rad_id	<p>This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.</p>
rad_school	<p>SGBSTDN_LEVL_CODE</p>
	<p>SGBSTDN_LEVL_CODE_2</p> <p>Two different rad_goal_dtl records will be created if these two LEVL_CODEs are different and degree/major data exists. The SGBSTDN curriculum data will be loaded into the appropriate rad_goaldata_dtl records for each School(Level)/Degree combination.</p>
	<p>This element defines the School (Level) in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. This code must match the bridged School code on the rad_class_dtl.</p>
rad_degree_code	<p>SGBSTDN_DEGC_CODE_1 is used if it exists.</p>
	<p>SGBSTDN_DEGC_CODE_2 is used if it exists and if the SGBSTDN_LEVL_CODE_2/SGBSTDN_DEGC_CODE_2 combination is unique.</p>
	<p>If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goaldata_dtl.rad_program.</p>
	<p>This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.</p>
rad_catalog_yr	<p>SGBSTDN_TERM_CODE_CTLG_1</p>
	<p>SGBSTDN_TERM_CODE_CTLG_2</p>
	<p>If Degree #1(SGBSTDN_DEGC_CODE_1) is being processed and SGBSTDN_TERM_CODE_CTLG_1 is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the SGBSTDN_TERM_CODE_EFF is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year.</p>
	<p>If Degree #2 (SGBSTDN_DEGC_CODE_2) is being processed and SGBSTDN_TERM_CODE_CTLG_2 is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the Catalog Year from SGBSTDN_TERM_CODE_CTLG_1 is used. If still blank, SGBSTDN_TERM_CODE_EFF is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year.</p>
	<p>This element defines the catalog year in effect for the student's goaldata record: COLLEGE, CONC, MAJOR, MINOR, or PROGRAM. The catalog</p>

Field Name	Comments
	year determines which set of requirement definitions (for example, which requirement block) should be used when evaluating the student's progress towards completing the degree.
rad_goal_code	<p>This element is the code associated with the type of record involved in a degree. Valid values are:</p> <p>ADVISOR – Advisors from SGRADVR</p> <p>COLLEGE – College Codes from SGBSTDN</p> <p>CONC – Concentration Codes from SGBSTDN</p> <p>MAJOR – Major Codes from SGBSTDN</p> <p>MINOR – Minor Codes from SGBSTDN</p> <p>PROGRAM – Program Codes from SGBSTDN</p> <p>STUSTATUS – STYP Code from SGBSTDN</p>
rad_goal_value	<p>This element is the actual rad_goal_value recorded for a given student for this goal code. Leading spaces are removed from any value placed here. Example: " PE" becomes "PE"</p> <p>See the table below for description of Banner Location of data used to populate this item.</p>
rad_goal_seq	<p>This element is the sequence number associated with the Goal Code and Goal Value. They do NOT have to be unique. For example, if you have 1 DEGREE, 1 MAJOR, 1 MINOR, 1 CONC and 1 ADVISOR that are all associated they would have the same sequence number (for example, 0001). If you have 1 DEGREE, 2 MAJORS, 2 MINORS and 1 ADVISOR associated with MAJOR #1 this is how they would look:</p> <p>Goal Code=DEGREE, Goal Value=BS, Goal Seq=0001</p> <p>Goal Code=MAJOR, Goal Value=MATH, Goal Seq=0001</p> <p>Goal Code=MAJOR, Goal Value=CS, Goal Seq = 0002</p> <p>Goal Code=MINOR, Goal Value=PHYS, Goal Seq = 0001</p> <p>Goal Code=MINOR, Goal Value=MIS, Goal Seq = 0002</p> <p>Goal Code=ADVISOR, Goal Value=12345, Goal Seq = 0001</p>

Table providing detail of rad_goal_value:

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
ADVISOR	See the ADVISOR Processing discussion at the end of this document.
COLLEGE	SGBSTDN_COLL_CODE_1 SGBSTDN_COLL_CODE_2
CONC	SGBSTDN_MAJR_CODE_CONC_1 SGBSTDN_MAJR_CODE_CONC_1_2 SGBSTDN_MAJR_CODE_CONC_1_3 SGBSTDN_MAJR_CODE_CONC_121 SGBSTDN_MAJR_CODE_CONC_122 SGBSTDN_MAJR_CODE_CONC_123 SGBSTDN_MAJR_CODE_CONC_2 SGBSTDN_MAJR_CODE_CONC_2_2 SGBSTDN_MAJR_CODE_CONC_2_3 SGBSTDN_MAJR_CODE_CONC_221 SGBSTDN_MAJR_CODE_CONC_222 SGBSTDN_MAJR_CODE_CONC_223
MAJOR	SGBSTDN_MAJR_CODE_1 SGBSTDN_MAJR_CODE_1_2 SGBSTDN_MAJR_CODE_2 SGBSTDN_MAJR_CODE_2_2
MINOR	SGBSTDN_MAJR_CODE_MINR_1 SGBSTDN_MAJR_CODE_MINR_1_2 SGBSTDN_MAJR_CODE_MINR_2 SGBSTDN_MAJR_CODE_MINR_2_2
PROGRAM	SGBSTDN_PROGRAM_1

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
	SGBSTDN_PROGRAM_2 If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goal_dtl.rad_degree_code.
STUSTATUS	SGBSTDN_STYP_CODE

PATH 3: Student SGBSTDN DUAL Degree Record

Student Goal Data may be loaded from SGBSTDN DUAL degree data. If SORLCUR and SORLFOS records are found for the given student the SGBSTDN table is still read if DUAL Degree data is desired.

If the UCX-CFG020 BANNER Check Dual Degree flag is set to Y and Dual Degree information exists on the Student Base table, SGBSTDN, it will be used to create rad_goaldata_dtl data. Regardless of whether the rest of the degree data is found in the SORLCUR/SORLFOS tables or the SGBSTDN table for a given student the Dual Degree data will be checked and imported if found.

If the SGBSTDN_LEVL_CODE_DUAL and SGBSTDN_DEGC_CODE_DUAL are unique and do not exist in the internal degree table calculated from the PATH #1 or PATH #2 scenarios above and SGBSTDN_MAJR_CODE_DUAL is NOT blank, then all of the DUAL data (including the SGBSTDN_COLL_CODE_DUAL and SGBSTDN_DEPT_CODE_DUAL) will be used in creating the rad_goal_dtl data for a student. The Catalog Year will be loaded using SGBSTDN_TERM_CODE_CTLG_2 first if not blank or null. If the Catalog Year is blank SGBSTDN_TERM_CODE_CTLG_1 will be used to obtain the Catalog Year. If neither of the SGBSTDN_TERM_CODE_CTLG_# years is loaded, the SGBSTDN_TERM_CODE_EFF will be used to get the Catalog Year.

If the SGBSTDN_LEVL_CODE_DUAL and SGBSTDN_DEGC_CODE_DUAL are NOT unique and are found in the internal degree table the DUAL degree data on the SGBSTDN record will be SKIPPED and NOT imported into Degree Works.

Make sure to customize the SGBSTDN entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

SQL used to read the SGBSTDN table:

```
SELECT a.SGBSTDN_PIDM,
       a.SGBSTDN_TERM_CODE_EFF,
       a.SGBSTDN_STST_CODE,
       a.SGBSTDN_LEVL_CODE,
       a.SGBSTDN_STYP_CODE,
       a.SGBSTDN_COLL_CODE_1,
       a.SGBSTDN_DEGC_CODE_1,
       a.SGBSTDN_MAJR_CODE_1,
       a.SGBSTDN_MAJR_CODE_MINR_1,
       a.SGBSTDN_MAJR_CODE_MINR_1_2,
       a.SGBSTDN_MAJR_CODE_CONC_1,
```

```

a.SGBSTDN_MAJR_CODE_CONC_1_2,
a.SGBSTDN_MAJR_CODE_CONC_1_3,
a.SGBSTDN_COLL_CODE_2,
a.SGBSTDN_DEGC_CODE_2,
a.SGBSTDN_MAJR_CODE_2,
a.SGBSTDN_MAJR_CODE_MINR_2,
a.SGBSTDN_MAJR_CODE_MINR_2_2,
a.SGBSTDN_MAJR_CODE_CONC_2,
a.SGBSTDN_MAJR_CODE_CONC_2_2,
a.SGBSTDN_MAJR_CODE_CONC_2_3,
a.SGBSTDN_AdVR_PIDM,
a.SGBSTDN_MAJR_CODE_1_2,
a.SGBSTDN_MAJR_CODE_2_2,
a.SGBSTDN_ACYR_CODE,
a.SGBSTDN_DEPT_CODE,
a.SGBSTDN_DEPT_CODE_2,
a.SGBSTDN_DEGC_CODE_DUAL,
a.SGBSTDN_LEVL_CODE_DUAL,
a.SGBSTDN_DEPT_CODE_DUAL,
a.SGBSTDN_COLL_CODE_DUAL,
a.SGBSTDN_MAJR_CODE_DUAL,
a.SGBSTDN_TERM_CODE_CTLG_1,
a.SGBSTDN_DEPT_CODE_1_2,
a.SGBSTDN_MAJR_CODE_CONC_121,
a.SGBSTDN_MAJR_CODE_CONC_122,
a.SGBSTDN_MAJR_CODE_CONC_123,
a.SGBSTDN_TERM_CODE_CTLG_2,
a.SGBSTDN_LEVL_CODE_2,
a.SGBSTDN_DEPT_CODE_2_2,
a.SGBSTDN_MAJR_CODE_CONC_221,
a.SGBSTDN_MAJR_CODE_CONC_222,
a.SGBSTDN_MAJR_CODE_CONC_223

FROM SGBSTDN a
WHERE a.SGBSTDN_TERM_CODE_EFF =
(SELECT MAX(b.SGBSTDN_TERM_CODE_EFF)
 FROM SGBSTDN b WHERE b.SGBSTDN_PIDM = a.SGBSTDN_PIDM)
AND
a.SGBSTDN_PIDM = <students-pidm>

```

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goaldata_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R033GDTA for the rad_goaldata_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R033GDTA200810..A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
rad_school	SGBSTDN_LEVL_CODE_DUAL

Field Name	Comments
	<p>This element defines the School (Level) in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. This code must match the bridged School code on the rad_class_dtl.</p>
rad_degree_code	SGBSTDN_DEGC_CODE_DUAL.
	<p>This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.</p>
rad_catalog_yr	SGBSTDN_TERM_CODE_CTLG_1
	<p>If this term is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the SORLCUR_TERM_CODE is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year.</p>
	<p>This element defines the catalog year in effect for the student's goaldata record: COLLEGE or MAJOR. The catalog year determines which set of requirement definitions (for example, which requirement block) should be used when evaluating the student's progress towards completing the degree.</p>
rad_goal_code	<p>This element is the code associated with the type of record involved in a degree. Valid values are:</p>
	ADVISOR – Advisors from SGRADVR
	COLLEGE – DUALCollege Code from SGBSTDN
	MAJOR – DUAL Major Code from SGBSTDN
rad_goal_value	<p>This element is the actual rad_goal_value recorded for a given student for this goal code. Leading spaces are removed from any value placed here.</p>
	Example: " PE" becomes "PE "
	ADVISOR – Advisors from SGRADVR – see the Advisor Processing discussion at the end of this document for details.
	COLLEGE – College Code from SGBSTDN_COLL_CODE_DUAL
	MAJOR – Major Code from SGBSTDN_MAJR_CODE_DUAL
rad_goal_seq	<p>This element is the sequence number associated with the Goal Code and Goal Value. For a DUAL Degree this sequence number will always be set to 0001.</p>

PATH 4: Applicant ADMISSIONS SORLCUR/SORLFOS Records

Applicant Goal Data may be loaded from ADMISSIONS SORLCUR/SORLFOS curriculum records if applicant processing is set appropriately for your site and the ID code being processed has valid applicant data. Review the three Applicant oriented flags in the UCX-CFG020 BANNER record and set them appropriately for your site.

The following SQL statements are executed in an attempt to retrieve the Concurrent Curriculum records from Banner contained in the SORLCUR and SORLFOS tables for a given APPLICANT (the :rBannerStuSPRIDEN.zPidm field is replaced by the actual SORLCUR_PIDM when executed):

Make sure to customize the SORLCUR2 and SORLFOS2 entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

SQL used to read the SORLCUR table:

```
SELECT a.SORLCUR_PIDM,
       a.SORLCUR_SEQNO,
       a.SORLCUR_LMOD_CODE,
       a.SORLCUR_TERM_CODE,
       a.SORLCUR_KEY_SEQNO,
       a.SORLCUR_PRIORITY_NO,
       a.SORLCUR_CACT_CODE,
       a.SORLCUR_LEVL_CODE,
       a.SORLCUR_COLL_CODE,
       a.SORLCUR_DEGC_CODE,
       a.SORLCUR_TERM_CODE_CTLG,
       a.SORLCUR_PROGRAM,
       a.SORLCUR_STYP_CODE

  FROM SORLCUR a, STVTERM t, SARADAP c
 WHERE (SELECT COUNT(*) FROM SHRTRCR
 WHERE SHRTRCR_PIDM = c.SARADAP_PIDM) > 0
   AND t.STVTERM_START_DATE > SYSDATE
   AND ((SELECT COUNT(*) FROM SGBSTDN
 WHERE SGBSTDN_PIDM = c.SARADAP_PIDM) < 1
      OR (SELECT COUNT(*) FROM STVSTST, SGBSTDN p
 WHERE STVSTST_CODE = p.SGBSTDN_STST_CODE
   AND STVSTST_REG_IND = 'Y'
   AND p.SGBSTDN_TERM_CODE_EFF = (SELECT MAX
        (o.SGBSTDN_TERM_CODE_EFF) FROM SGBSTDN o
 WHERE o.SGBSTDN_PIDM = p.SGBSTDN_PIDM
   AND o.SGBSTDN_TERM_CODE_EFF <
        c.SARADAP_TERM_CODE_ENTRY)) < 1)
   AND a.SORLCUR_CACT_CODE = 'ACTIVE'
   AND a.SORLCUR_LMOD_CODE = 'ADMISSIONS'
   AND a.SORLCUR_SEQNO = (SELECT MAX(b.SORLCUR_SEQNO)
  FROM SORLCUR b
 WHERE b.SORLCUR_PIDM = a.SORLCUR_PIDM
   AND b.SORLCUR_PRIORITY_NO = a.SORLCUR_PRIORITY_NO
   AND b.SORLCUR_LMOD_CODE = 'ADMISSIONS')
   AND a.SORLCUR_PIDM = c.SARADAP_PIDM
```

```

        AND a.SORLCUR_TERM_CODE = c.SARADAP_TERM_CODE_ENTRY
        AND a.SORLCUR_KEY_SEQNO = c.SARADAP_APPL_NO
        AND t.STVTERM_CODE = c.SARADAP_TERM_CODE_ENTRY
    AND
        a.SORLCUR_PIDM = <applicants-pidm>

    ORDER BY a.SORLCUR_SEQNO

```

SQL used to read the corresponding SORLFOS table:

```

SELECT a.SORLFOS_PIDM,
       a.SORLFOS_LCUR_SEQNO,
       a.SORLFOS_LFST_CODE,
       a.SORLFOS_TERM_CODE,
       a.SORLFOS_PRIORITY_NO,
       a.SORLFOS_CSTS_CODE,
       a.SORLFOS_MAJR_CODE,
       a.SORLFOS_TERM_CODE_CTLG,
       a.SORLFOS_DEPT_CODE,
       a.SORLFOS_MAJR_CODE_ATTACH

FROM SORLFOS a, SORLCUR b, STVTERM t, SARADAP d
WHERE (SELECT COUNT(*) FROM SHRTRCR
WHERE SHRTRCR_PIDM = d.SARADAP_PIDM) > 0
      AND t.STVTERM_START_DATE > SYSDATE
      AND ((SELECT COUNT(*) FROM SGBSTDN
WHERE SGBSTDN_PIDM = d.SARADAP_PIDM) < 1
      OR (SELECT COUNT(*) FROM STVSTST, SGBSTDN p
WHERE STVSTST_CODE = p.SGBSTDN_STST_CODE
      AND STVSTST_REG_IND = 'Y'
      AND p.SGBSTDN_TERM_CODE_EFF = (SELECT MAX
(o.SGBSTDN_TERM_CODE_EFF) FROM SGBSTDN o
WHERE o.SGBSTDN_PIDM = p.SGBSTDN_PIDM
      AND o.SGBSTDN_TERM_CODE_EFF <
d.SARADAP_TERM_CODE_ENTRY)) < 1)
      AND b.SORLCUR_CACT_CODE = 'ACTIVE'
      AND b.SORLCUR_LMOD_CODE = 'ADMISSIONS'
      AND b.SORLCUR_SEQNO = (SELECT MAX(f.SORLCUR_SEQNO)
FROM SORLCUR f
WHERE f.SORLCUR_PIDM = b.SORLCUR_PIDM
      AND f.SORLCUR_PRIORITY_NO = b.SORLCUR_PRIORITY_NO
      AND f.SORLCUR_LMOD_CODE = 'ADMISSIONS')
      AND b.SORLCUR_PIDM = d.SARADAP_PIDM
      AND b.SORLCUR_TERM_CODE = d.SARADAP_TERM_CODE_ENTR
      AND b.SORLCUR_KEY_SEQNO = d.SARADAP_APPL_NO
      AND t.STVTERM_CODE = d.SARADAP_TERM_CODE_ENTRY
      AND a.SORLFOS_CSTS_CODE = 'INPROGRESS'
      AND a.SORLFOS_CACT_CODE = 'ACTIVE'
      AND a.SORLFOS_PIDM = b.SORLCUR_PIDM
      AND a.SORLFOS_LCUR_SEQNO = b.SORLCUR_SEQNO
      AND a.SORLFOS_SEQNO =
(SELECT MAX(1.SORLFOS_SEQNO) FROM SORLFOS 1

```

```

WHERE 1.SORLFOS_PIDM = b.SORLCUR_PIDM
AND 1.SORLFOS_PRIORITY_NO = a.SORLFOS_PRIORITY_NO
AND 1.sorlfos_csts_code = 'INPROGRESS'
AND 1.SORLFOS_LCUR_SEQNO = b.SORLCUR_SEQNO)
AND
a.SORLFOS_PIDM = <applicants-pidm>

ORDER BY a.SORLFOS_LCUR_SEQNO, a.SORLFOS_PRIORITY_NO

```

This record contains information for the rad_goaldata_dtl table. This is a required table for Degree Works but not for Transfer Equivalency. The rad_goaldata_dtl records that are bridged must have unique ID, term, school and degree combinations. They must have a corresponding rad_goal_dtl record with the same ID, term, school and degree.

Total length = 1000 bytes.

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goaldata_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R033GDTA for the rad_goaldata_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R033GDTA200810..A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
rad_school	SORLCUR_LEVL_CODE This element defines the School (Level) in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. This code must match the bridged School code on the rad_class_dtl.
rad_degree_code	The SORLCUR_DEGC_CODE is used if it exists. Multiple rad_goal_dtl records may be created if multiple SORLCUR records are retrieved with different School/Degree combinations using the SQL above. If SORLCUR records containing the same School and Degree are retrieved only one rad_goal_dtl record will be generated. If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goaldata_dtl.rad_program. This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.
rad_catalog_yr	The SORLFOS_TERM_CODE_CTLG, if it is not blank, is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank the SORLCUR_TERM_CODE is used to lookup UCX_STU016 to get the UCX0STU016 Catalog Year.

Field Name	Comments
	<p>This element defines the catalog year in effect for the student's goaldta record: COLLEGE, CONC, MAJOR, MINOR, PROGRAM, or SPEC. The catalog year determines which set of requirement definitions (for example, which requirement block) should be used when evaluating the student's progress towards completing the degree.</p>
rad_goal_code	<p>This element is the code associated with the type of record involved in a degree/goal. Valid values are:</p> <ul style="list-style-type: none"> ADVISOR – Advisors from SGRADVR COLLEGE – College Codes from SORLCUR CONC – Concentration Codes from SORLFOS MAJOR – Major Codes from SORLFOS MINOR – Minor Codes from SORLFOS PROGRAM – Program Codes from SORLCUR SPEC – Certification Codes (CERTIFICATION or CERTIFICATE) from SORLFOS LIBL – Option Codes from SORLFOS STUSTATUS – STYP Code from SORLCUR
rad_goal_value	<p>This element is the actual rad_goal_value recorded for a given student for this goal code. Leading spaces are removed from any value placed here. Example: " PE" becomes "PE"</p> <p>See the table below for description of Banner Location of data used to populate this item.</p>
rad_goal_seq	<p>This element is the sequence number associated with the Goal Code and Goal Value. They do NOT have to be unique. For example, if you have 1 DEGREE, 1 MAJOR, 1 MINOR, 1 CONC and 1 ADVISOR that are all associated they would have the same sequence number (for example, 0001). If you have 1 DEGREE, 2 MAJORS, 2 MINORS and 1 ADVISOR associated with MAJOR #1 this is how they would look:</p> <p>Goal Code=DEGREE, Goal Value=BS, Goal Seq=0001</p> <p>Goal Code=MAJOR, Goal Value=MATH, Goal Seq=0001</p> <p>Goal Code=MAJOR, Goal Value=CS, Goal Seq = 0002</p> <p>Goal Code=MINOR, Goal Value=PHYS, Goal Seq = 0001</p>

Field Name	Comments
	Goal Code=MINOR, Goal Value=MIS, Goal Seq = 0002
	Goal Code=ADVISOR, Goal Value=12345, Goal Seq = 0001
rad_attach_code	This element contains the type of data in the rad_attach_value field. The MAJR_CODE_ATTACH value is found in the list of SORLFOS records and this code is derived from its LFST_CODE. This will be MAJOR, MINOR, CONC or SPEC – though normally it is set to MAJOR as concentrations are normally attached to majors (and not the reverse).
rad_attach_value	The element contains the SORLFOS_MAJR_CODE_ATTACH value. Usually this is populated when the goal_code/value is for a concentration and this field is filled with the attached major code. This is saying that this concentration is attached/associated with this particular major.

Table providing detail of rad_goal_value:

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
ADVISOR	See the ADVISOR Processing at the end of this document.
COLLEGE	SORLCUR_COLL_CODE
CONC	SORLFOS_MAJR_CODE where SORLFOS_LFST_CODE = CONCENTRATION.
MAJOR	SORLFOS_MAJR_CODE when SORLFOS_LFST_CODE = MAJOR.
MINOR	SORLFOS_MAJR_CODE when SORLFOS_LFST_CODE = MINOR.
PROGRAM	The SORLCUR_PROGRAM will be loaded if it exists for the given student. If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found in the rad_degree_code on the rad_goal_dtl.and rad_goaladata_dtl records.
SPEC	Loaded with the SORLFOS_MAJR_CODE if the SORLFOS_LFST_CODE is equal to CERTIFICATION or CERTIFICATE. The ban40.ec student extract looks for this CERTIFICATION or CERTIFICATE code for a SORLFOS record that is being processed for a corresponding SORLCUR record. If a CERTIFICATION or CERTIFICATE record is found the SORLFOS_MAJR_CODE will be written to the rad_goal_value.

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
LIBL	Loaded with the SORLFOS_MAJR_CODE if the SORLFOS_LFST_CODE is equal to OPTION. The ban40.ec student extract looks for this OPTION code for a SORLFOS record that is being processed for a corresponding SORLCUR record. If a OPTION record is found the SORLFOS_MAJR_CODE will be written to the rad_goal_value.
STUSTATUS (STYP)	The SORLCUR_STYP_CODE will be loaded if it exists for the given student

PATH 5: Applicant SARADAP Record

Applicant Goal Data may be loaded from SARADAP curriculum data if applicant processing is set appropriately for your site and the ID code being processed has valid applicant data. Review the three Applicant oriented flags in the UCX-CFG020 BANNER record and make sure they are set appropriately for your site.

If SORLCUR and SORLFOS records are NOT found for the given student or applicant and the UCX-CFG020 BANNER Load SARADAP Goals = Y the SARADAP table is read to obtain the goal data. If one or more valid SARADAP records are found for an admissions applicant the appropriate rad_goaldata_dtl records will be generated.

Make sure to customize the SARADAP entries in the integration.banner.extract.config Shepherd setting to be appropriate for your site.

SQL used to read the SARADAP table:

```
SELECT a.SARADAP_PIDM,
       a.SARADAP_TERM_CODE_EFF,
       a.SARADAP_STST_CODE,
       a.SARADAP_LEVL_CODE,
       a.SARADAP_STYP_CODE,
       a.SARADAP_COLL_CODE_1,
       a.SARADAP_DEGC_CODE_1,
       a.SARADAP_MAJR_CODE_1,
       a.SARADAP_MAJR_CODE_MINR_1,
       a.SARADAP_MAJR_CODE_MINR_1_2,
       a.SARADAP_MAJR_CODE_CONC_1,
       a.SARADAP_MAJR_CODE_CONC_1_2,
       a.SARADAP_MAJR_CODE_CONC_1_3,
       a.SARADAP_COLL_CODE_2,
       a.SARADAP_DEGC_CODE_2,
       a.SARADAP_MAJR_CODE_2,
       a.SARADAP_MAJR_CODE_MINR_2,
       a.SARADAP_MAJR_CODE_MINR_2_2,
       a.SARADAP_MAJR_CODE_CONC_2,
```

```

a.SARADAP_MAJR_CODE_CONC_2_2,
a.SARADAP_MAJR_CODE_CONC_2_3,
a.SARADAP_ADVR_PIDM,
a.SARADAP_MAJR_CODE_1_2,
a.SARADAP_MAJR_CODE_2_2,
a.SARADAP_ACYR_CODE,
a.SARADAP_DEPT_CODE,
a.SARADAP_DEPT_CODE_2,
a.SARADAP_DEGC_CODE_DUAL,
a.SARADAP_LEVL_CODE_DUAL,
a.SARADAP_DEPT_CODE_DUAL,
a.SARADAP_COLL_CODE_DUAL,
a.SARADAP_MAJR_CODE_DUAL,
a.SARADAP_TERM_CODE_CTLG_1,
a.SARADAP_DEPT_CODE_1_2,
a.SARADAP_MAJR_CODE_CONC_121,
a.SARADAP_MAJR_CODE_CONC_122,
a.SARADAP_MAJR_CODE_CONC_123,
a.SARADAP_TERM_CODE_CTLG_2,
a.SARADAP_LEVL_CODE_2,
a.SARADAP_DEPT_CODE_2_2,
a.SARADAP_MAJR_CODE_CONC_221,
a.SARADAP_MAJR_CODE_CONC_222,
a.SARADAP_MAJR_CODE_CONC_223

FROM SARADAP a
WHERE a.SARADAP_TERM_CODE_ENTRY =
(SELECT MAX(b.SARADAP_TERM_CODE_ENTRY)
 FROM SARADAP b
 WHERE b.SARADAP_PIDM = a.SARADAP_PIDM)
AND
a.SARADAP_PIDM = <applicant's-pidm>

```

Field Name	Comments
HEADER	This is a 28-byte field identifying the record as one containing rad_goaldata_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R033GDTA for the rad_goaldata_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918.R033GDTA200810.A. (periods represent spaces).
rad_id	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
rad_school	SARADAP_LEVL_CODE This element defines the School (Level) in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. This code must match the bridged School code on the rad_class_dtl.

Field Name	Comments
rad_degree_code	SARADAP_DEGC_CODE
	This element contains the code that identifies the student's degree.
	Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for
	Bachelor of Fine Arts.
rad_catalog_yr	SARADAP_TERM_CODE_CTLG_1
	SARADAP_TERM_CODE_CTLG_2
	If Degree #1(SARADAP_DEGC_CODE_1) is being processed and SARADAP_TERM_CODE_CTLG_1 is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the SARADAP_TERM_CODE_ENTRY is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year.
	If Degree #2 (SARADAP_DEGC_CODE_2) is being processed and SARADAP_TERM_CODE_CTLG_2 is not blank, it is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year. If it is blank then the Catalog Year from SARADAP_TERM_CODE_CTLG_1 is used. If still blank, SARADAP_TERM_CODE_ENTRY is used to lookup UCX-STU016 to get the UCX-STU016 Catalog Year.
	This element defines the catalog year in effect for the student's goaldata record: COLLEGE, CONC, MAJOR, MINOR, or PROGRAM. The catalog year determines which set of requirement definitions (for example, which requirement block) should be used when evaluating the student's progress towards completing the degree.
rad_goal_code	This element is the code associated with the type of record involved in a degree. Valid values are:
	ADVISOR – Advisors from SGRADVR
	COLLEGE – College Codes from SARADAP
	CONC – Concentration Codes from SARADAP
	MAJOR – Major Codes from SARADAP
	MINOR – Minor Codes from SARADAP
	PROGRAM – Program Codes from SARADAP
	STUSTATUS – STYP Code from SARADAP
rad_goal_value	This element is the actual rad_goal_value recorded for a given student for this goal code. Leading spaces are removed from any value placed here. Example: " PE" becomes "PE "

Field Name	Comments
	See the table below for description of Banner Location of data used to populate this item.
rad_goal_seq	<p>This element is the sequence number associated with the Goal Code and Goal Value. They do NOT have to be unique. For example, if you have 1 DEGREE, 1 MAJOR, 1 MINOR, 1 CONC and 1 ADVISOR that are all associated they would have the same sequence number (for example, 0001). If you have 1 DEGREE, 2 MAJORS, 2 MINORS and 1 ADVISOR associated with MAJOR #1 this is how they would look:</p> <p>Goal Code=DEGREE, Goal Value=BS, Goal Seq=0001</p> <p>Goal Code=MAJOR, Goal Value=MATH, Goal Seq=0001</p> <p>Goal Code=MAJOR, Goal Value=CS, Goal Seq = 0002</p> <p>Goal Code=MINOR, Goal Value=PHYS, Goal Seq = 0001</p> <p>Goal Code=MINOR, Goal Value=MIS, Goal Seq = 0002</p> <p>Goal Code=ADVISOR, Goal Value=12345, Goal Seq = 0001</p>

Table providing detail of rad_goal_value:

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
ADVISOR	See the ADVISOR Processing discussion at the end of this document.
COLLEGE	SARADAP_COLL_CODE_1 SARADAP_COLL_CODE_2
CONC	SARADAP_MAJR_CODE_CONC_1 SARADAP_MAJR_CODE_CONC_1_2 SARADAP_MAJR_CODE_CONC_1_3 SARADAP_MAJR_CODE_CONC_121 SARADAP_MAJR_CODE_CONC_122 SARADAP_MAJR_CODE_CONC_123 SARADAP_MAJR_CODE_CONC_2 SARADAP_MAJR_CODE_CONC_211

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
	SARADAP_MAJR_CODE_CONC_212
	SARADAP_MAJR_CODE_CONC_213
	SARADAP_MAJR_CODE_CONC_221
	SARADAP_MAJR_CODE_CONC_222
	SARADAP_MAJR_CODE_CONC_223
MAJOR	SARADAP_MAJR_CODE_1 SARADAP_MAJR_CODE_1_2 SARADAP_MAJR_CODE_2 SARADAP_MAJR_CODE_2_2
MINOR	SARADAP_MAJR_CODE_MINR_1_1 SARADAP_MAJR_CODE_MINR_1_2 SARADAP_MAJR_CODE_MINR_2_1 SARADAP_MAJR_CODE_MINR_2_2
PROGRAM	SARADAP_PROGRAM_1 SARADAP_PROGRAM_2 If UCX-CFG020 BANNER Program as Degree is set to Y, then this value will be the value found on the rad_goal_dtl.rad_degree_code.
STUSTATUS	SARADAP_STYP_CODE
ADVISOR Processing	
If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
ADVISOR	The same ADVISOR rad_goaldata_dtl records will be loaded for each unique School(Level)/Degree combination found for a given student. All SGRADVR records are read for a given student based on the SQL specified in the integration.banner.extract.config Shepherd setting. The SGRADVR_Advr_PIDM is then

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
----------------------	---

used to lookup each advisor SPRIDEN record. The number of advisors loaded into Degree Works from Banner is controlled by the UCX-CFG020BANNER Advisor Method. Default SQL used to read Advisors:

```

SELECT a.SGRADVR_PIDM,
       a.SGRADVR_TERM_CODE_EFF,
       a.SGRADVR_ADVR_PIDM,
       a.SGRADVR_ADVR_CODE,
       a.SGRADVR_PRIM_IND
  FROM SGRADVR a
 WHERE a.SGRADVR_PIDM = :rBanne
 rStuSPRIDEN.zPidm
      AND a.SGRADVR_TERM_CODE_EFF
      =
      (SELECT MAX(b.SGRADVR_TER
 M_CODE_EFF)
        FROM SGRADVR b
       WHERE b.SGRADVR_PIDM = a
 .SGRADVR_PIDM)
 ORDER BY a.SGRADVR_ADVR_CODE;

```

If A allow 4 ADVISOR ID values to be loaded based on three SGRADVR_ADVR_CODES (four sets of three codes). Four new sets of 3 Advisor Codes each have been added to the UCX-CFG020 BANNER record for this feature. Three SGRADVR_ADVR_CODES can be specified for 4 rad_goal_values. Each of these 4 advisor codes is loaded independently based on these codes.

For example, if the UCX-CFG020 BANNER Advisor Code arrays contain the following SGRADVR_ADVR_CODES:

Academic Advisor 1: ESL, AADV, UNDC

Faculty Advisor 2: MAJR

Faculty Advisor 3: MAJ2

Career Advisor 4: CARE

For Advisor #1 it will look for a match on ESL and if found it will load the associated SPRIDEN_ID into the rad_goal_value. If not found, it will continue looking for a match on AADV and UNDC. The first match found will

If the Goal Code is:	Then the Data is extracted from here in Banner and populates the Goal Value
C	cause the rad_goal_value field to be loaded with the SPRIDEN_ID. If no match is found, then no rad_goal_value will be loaded. This pattern continues for the other three Advisor fields. Thus, a maximum of 12 Advisor codes (3 Advisors times 4) may be used for loading Advisors from Banner.
If C the following rules will be used: the Advisor SPRIDEN_ID is loaded into ADVISOR rad_goal_value with a rad_goal_seq of 1. If the SGRADVR_PRIM_IND = Y and the SGRADVR_Advr_CODE = the UCX-CFG020 BANNER Advisor Major. If the SGRADVR_Advr_CODE = UCX-CFG020 BANNER Advisor Major and the PRIM_IND = N then the Advisor SPRIDEN_ID is loaded into the second rad_goaldata_dtl record. If the SGRADVR_Advr_CODE = UCX-CFG020 BANNER Advisor Minor then the Advisor SPRIDEN_ID is loaded into another rad_goaldata_dtl.	
If C and the Load Extra Advisors flag = Y, then any extra advisor ID codes left over after the Advisor Major and Advisor Minor matching has been performed will be loaded into remaining blank RAD ADVR #1 – ADVR #4 codes with the Primary Advisor loaded into rad_advr1 if the SGRADVR_PRIM_IND = Y on any of the SGRADVR Advisor records.	
If S the selected SGRADVR records will be read for a given student based on the SQL SELECT statement specified in the integration.banner.extract.config Shepherd setting. A rad_goaldata_dtl with a rad_goal_code of ADVISOR and a rad_goal_value containing the Advisor SPRIDEN_ID will be created for each SGRADVR Advisor ID.	

R062TERM - Term Record

Updated: March 24, 2023

The Banner Overall GPA table, SHRLGPA, is used for most of the data in this record.

This table contains three types of records differentiated by the SHRLGPA_GPA_TYPE indicator:

I - Institutional Totals

O - Overall Totals

T - Transfer Totals

At a minimum, at least one Totals record should exist for a student for each unique School (SHRLGPA_LEVEL_CODE). The rad_term_dtl record will be loaded with only ONE set of totals for each Student/School. The UCX-CFG020 BANNER record contains the GPA Type that determines what Banner record to load into the rad_term_dtl. If loaded with an I the Institutional Totals will be loaded into the rad_term_dtl. If loaded with an O the Overall Totals will be loaded into the rad_term_dtl. However, if the student is a first-time transfer student with only a SHRLGPA record with a T GPA Type flag then a rad_term_dtl will be created with only transfer information. For more information, see the [UCX-CFG020 BANNER](#) topic.

In addition, if the student has transfer data in Banner then a Transfer Totals record should exist as well. In this case the SHRLGPA_HOURS_EARNED from this record will be written to the appropriate rad_term_dtl for the Student/School.

SQL used to read the SHRLGPA table:

```
SELECT SHRLGPA_PIDM,
       SHRLGPA_LEVEL_CODE,
       SHRLGPA_GPA_TYPE_IND,
       SHRLGPA_HOURS_ATTEMPTED,
       SHRLGPA_HOURS_EARNED,
       SHRLGPA_GPA_HOURS,
       SHRLGPA_QUALITY_POINTS,
       SHRLGPA_GPA,
       SHRLGPA_HOURS_PASSED,
       SHRLGPA_GPA_CALC
  FROM SHRLGPA
 WHERE SHRLGPA_PIDM = :rBannerStuSPRIDEN.zPidm
 ORDER BY SHRLGPA_LEVEL_CODE,
          SHRLGPA_GPA_TYPE_IND;
```

These records are bridged into the RAD_TERM_DTL table in Degree Works.

Field	Description
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_school	SHRLGPA_LEVEL_CODE
rad_deg_interest	Not extracted. Loaded with BLANKS.
rad_cum_tot_earn	SHRLGPA_HOURS_EARNED
	If the only SHRLGPA record existing for a student or applicant has a SHRLGPA_GPA_TYPE_IND = T for Transfer Totals then this field will be loaded with the rad_cum_tr_earn data defined below.

Field	Description
rad_cum_tr_earn	Loaded with the SHRLGPA_HOURS_EARNED from the record with the SHRLGPA_GPA_TYPE_IND = T for Transfer Totals for the matching rad_school (SHRLGPA_LEVL_CODE) for a given student.
rad_cum_cr_earn	Not extracted. Loaded with BLANKS.
rad_cum_gr_att	SHRLGPA_GPA_HOURS
rad_cum_gr_pts	SHRLGPA_QUALITY_POINTS
rad_cum_gpa	SHRLGPA_GPA
	For the format of the rad_cum_gpa, review the UCX-CFG020 DAP14 GPA Round flag.
rad_term	Obsolete – loaded with spaces.
rad_user_gpa1	Not extracted. Loaded with BLANKS.
rad_user_gpa2	Not extracted. Loaded with BLANKS.
rad_user_credit1	Not extracted. Loaded with BLANKS.
rad_user_credit2	Not extracted. Loaded with BLANKS.
rad_user_def1	Not extracted. Loaded with BLANKS.
rad_user_def2	Not extracted. Loaded with BLANKS.
rad_user_def3	Not extracted. Loaded with BLANKS.
rad_user_def4	Not extracted. Loaded with BLANKS.
rad_user_def5	Not extracted. Loaded with BLANKS.
rad_user_def6	Not extracted. Loaded with BLANKS.
rad_user_def7	Not extracted. Loaded with BLANKS.
rad_user_def8	Not extracted. Loaded with BLANKS.
rad_user_def9	Not extracted. Loaded with BLANKS.
rad_user_def10	Not extracted. Loaded with BLANKS.

R071CLAS - Class Record - CURRENT

Updated: March 24, 2023

Several Banner tables are used to create the data on the rad_class_dtl for current class records.

SFRSTCR	The Grade Component Student Audit table is read using the student's SPRIDEN_PIDM. This table is the driver for the current class extraction.
SSBSECT	The Course Information table is read using the SFRSTCR_TERM_CODE and SFRSTCR_CRN for the class.
SHRGRDE	The Grade Code Table is read using the SFRSTCR_GRDE_CODE and SFRSTCR_LEVL_CODE. The matching record with the highest SHRGRDE_TERM_CODE_EFFECTIVE that is LESS THAN or

EQUAL to the SFRSTCR_TERM_CODE is used to obtain the various grade flags.

SSRATTR	The Degree Program Attribute table is read using the SFRSTCR_TERM_CODE and SFRSTCR_CRN for the class. Refer to the Special Topic on Attributes for details on how the class attributes are recorded on the User Defined fields on the rad_class_dtl.
---------	--

Note: Several special edits may be made to determine if a current class should be excluded from Degree Works. The Banner fields from SSBSECT and SFRSTCR used in these special edits are as follows (refer to the Technical UCX documentation for more details on the UCX table used in these descriptions).

SSBSECT_SUBJ_CODE: Current class records will NOT be created if this code (rad_discipline) is Inactive. The Discipline Status on UCX-STU352 should be set to an I for Inactive discipline codes.

SSBSECT_SCHD_CODE: A special edit may be made using this section code for ZERO credit classes. If particular Schedule Types are to be skipped and NOT rolled to Degree Works, load these Schedule Types into UCX-BAN001. The banner extract will read UCX-BAN001 for each non-blank SSBSECT_SCHD_CODE. If a match is found AND the class has ZERO credits, the class will NOT be loaded into the rad_class_dtl.

SSBSECT_SEQ_NUMB: A special edit may be made using this section code for ZERO credit classes. If particular sections are to be skipped and NOT rolled to Degree Works, load these Section codes into UCX-BAN002. The banner extract will read UCX-BAN002 for each non-blank SSBSECT_SEQ_NUMB. If a match is found AND the class has ZERO credits, the class will NOT be loaded into the rad_class_dtl.

SFRSTCR_GMOD_CODE: A special edit may be made using this Gmod Code (Grade Type). If particular Gmod Codes are to be skipped and NOT rolled to Degree Works, load these Gmod Codes into UCX-BAN003. The banner extract will read UCX-BAN003 for each SFRSTCR_GMOD_CODE. If a match is found, the class will NOT be loaded into the rad_class_dtl.

SQL used to read the SFRSTCR table. The WHERE clause comes from the integration.banner.extract.config Shepherd setting, so you can change it if you need to.

```

SELECT SFRSTCR_TERM_CODE,
       SFRSTCR_PIDM,
       SFRSTCR_CRN,
       SFRSTCR_RSTS_CODE,
       SFRSTCR_CREDIT_HR,
       SFRSTCR_GMOD_CODE,
       SFRSTCR_GRDE_CODE,
       TO_CHAR(SFRSTCR_GRDE_DATE, 'YYYYMMDD'),
       SFRSTCR_LEVL_CODE
  FROM SFRSTCR
 WHERE SFRSTCR_PIDM = :rBannerStuSPRIDEN.zPidm
   AND SFRSTCR_RSTS_CODE IN
    (SELECT STVRSTS_CODE
      FROM STVRSTS WHERE STVRSTS_INCL_SECT_ENRL = 'Y')
   AND SFRSTCR_GRDE_DATE is NULL

```

```
ORDER BY SFRSTCR_TERM_CODE,
SFRSTCR_CRN;
```

SQL used to read the SSBSECT table based on the SFRSTCR_TERM_CODE and SFRSTCR_CRN for each current class for a given student:

```
SELECT SSBSECT_TERM_CODE,
SSBSECT_CRN,
SSBSECT_SUBJ_CODE,
SSBSECT_CRSE_NUMB,
SSBSECT_SEQ_NUMB,
SSBSECT_SCHD_CODE,
SSBSECT_CAMP_CODE,
SSBSECT_CRSE_TITLE,
SSBSECT_CREDIT_HRS,
SSBSECT_GMOD_CODE,
SSBSECT_GRADABLE_IND
```

These records are bridged into the RAD_CLASS_DTL table in Degree Works.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_course_key	The Course Key is a composite key composed of a 12-byte discipline code and a 12-byte course number.
rad_discipline	SSBSECT_SUBJ_CODE (explanation above)
rad_course_num	SSBSECT_CRSE_NUMB
rad_section	SSBSECT_SEQ_NUMB(explanation above)
rad_course_title	SSBSECT_CRSE_TITLE. If this course title is blank, the SCBCRSE table will be read by using the SSBSECT_SUBJ_CODE, SSBSECT_CRSE_NUMB and SSBSECT_TERM_CODE (LESS THAN or EQUAL TO) in DESCENDING sequence. The FIRST matching SCBCRSE record that is read will be the used to load the rad_course_title as it will be the most recent course title that has an Effective Term NOT GREATER than the SSBSECT_TERM_CODE.
rad_school	SFRSTCR_LEVL_CODE
rad_division	SCBCRSE_DIVS_CODE
rad_dept	SCBCRSE_DEPT_CODE
rad_deg_interest	Not extracted. Loaded with BLANKS.
rad_class_type	Not extracted. Loaded with BLANKS.
rad_acad_votech	Not extracted. Loaded with BLANKS.

Field Name	Banner Data
rad_audit_flag	If the SFRSTCR_GMOD_CODE = A, the audit flag is set to Y. Otherwise it is set to N.
rad_insuff_flag	Set to N.
rad_inprog_flag	This flag will be loaded with one of three values (P, Y, N): P: If the STVTERM start date for this class is in the future. This indicates that it is a Preregistered class. Be sure your UCX-STU385 In-progress override flags do not conflict with this P value. It is recommended that your In-progress override flag be set to <blank> for most grades. Y: If the rad_final_grade matches the UCX_CFG020 BANNER Default Grade or is BLANK. N: The class has been graded and not yet rolled to history (the rad_inprog_flag has not already been set to P or Y).
rad_withdr_flag	If the SFRSTCR_GRDE_CODE = W, the withdraw flag is set to Y. Otherwise it is set to N.
rad_incomp_flag	If the SHRGRDE_IMPCMP_IND = N, the incomplete flag is set to N.
rad_pass_flag	SHRGRDE_PASSED_IND
rad_credits	SFRSTCR_CREDIT_HR.
rad_credits_earn	The rad_credits are copied into this field even though the class is most likely in-progress. Degree Works considers in-progress classes passed for auditing purposes.
rad_gpa_credits	If the SHRGRDE_GPA_IND = Y, the rad_credits are copied into the GPA credits.
rad_grade_points	SHRGRDE_QUALITY_POINTS times the rad_credits. The Banner quality points are converted to the RAD format of 9999v999. For example, 12 becomes 0012000, 10.5 becomes 0010500 and 6 becomes 0006000.
rad_credit_type	Not extracted. Loaded with BLANKS.
rad_class_status	Loaded with a default of A. If the class has been repeated, one of the following will be loaded:

Field Name	Banner Data
	CH – the SFRSTCR current class is a repeat of a Historic class (SHRTCKN)
	CT – the SFRSTCR current class is a repeat of a Transfer class (SHRTRCE)
	CB - the SFRSTCR current class is a repeat of both a historic class and a transfer class.
	WD – if the SFRSTCR current class has been withdrawn (rad_withdr_flag = Y). However, if SHBRPTS_TITLE_IND=Y and the in-progress class is found as a match to a completed class but with a different title, the two classes are not considered repeats. When the flag is N, a match based on the subject and course number is sufficient to say that the two are repeats. The SHBRPTS_TITLE_IND flag is in Banner in SHARPTR and is the check box for Title Indicator.
	These repeat codes can be listed in the dash.audit.repeat.codes setting. After you enable the UCX-RPT036 Show Repeated flag you will see these repeated classes showing a repeated indicator in the Responsive Dashboard worksheet.
rad_grade_type	SFRSTCR_GMOD_CODE
rad_pass_fail	If SFRSTCR_GMOD_CODE = P, the pass fail flag is set to Y. Otherwise it is set to N.
rad_repeat_ptr	Loaded with the appropriate repeat_ptr if this in-progress class has been repeated. The rad_repeat_ptr should contain the rad_course_key of the course that is being counted in Degree Works and should match the rad_repeat_ptr loaded on the historic or transfer class.
rad_repeat_plcy	Loaded with the rad_course_key if this in-progress class has been repeated. The rad_class_status listed above indicates whether the class is a repeat of a historic (SHRTCKN) or transfer (SHRTRCE) class.
rad_session	Not extracted. Loaded with BLANKS.
rad_location	SSBSECT_CAMP_CODE.

Field Name	Banner Data
rad_final_grade	SFRSTCR_GRDE_CODE – but if it is blank the SFRSTCR_GRDE_CODE_MID is loaded – and if that is blank the default grade from the UCX-CFG020 BANNER Default Grade field is used.
rad_final_gr_num	SHRGRDE_QUALITY_POINTS. Multiple quality point values may be defined for a given SHRGRDE_LEVL_CODE (rad_school), SHRGRDE_CODE (rad_final_grade) and SHRGRDE_TERM_CODE_EFFECTIVE (rad_term). After a match on LEVL_CODE and grade CODE is made for a given SFRSTCR record the SHRGRDE records are read in descending term sequence. When the SFRSTCR term is GREATER THAN or EQUAL to the SHRGRDE term the associated SHRGRDE_QUALITY_POINTS will be loaded into the rad_final_gr_num. For example, a grade B+ in the undergraduate school UG starts out with Term 000000 as 3.5 quality points. In Term 200710 the quality points are changed to 3.3 quality points. If the SFRSTCR_TERM_CODE is GREATER THAN or EQUAL to the SHRGRDE TERM_CODE_EFFECTIVE of 200710, 3.3 quality points would be loaded. Otherwise the quality points would be loaded with 3.5 quality points. The Banner quality points are converted to the RAD format of 9999v999. For example, 3.575 becomes 0003575, 4.0 becomes 0004000 and .5 becomes 0000500.
rad_instr1_id	Not extracted. Loaded with BLANKS.
rad_term	SFRSTCR_TERM_CODE
rad_stu_level	Not extracted. Loaded with BLANKS.
rad_user_def1	Not extracted. Loaded with BLANKS.
rad_user_def2	Not extracted. Loaded with BLANKS.
rad_user_def3	Not extracted. Loaded with BLANKS.
rad_user_def4	Not extracted. Loaded with BLANKS.
rad_user_def5	Not extracted. Loaded with BLANKS.
rad_user_def6	Not extracted. Loaded with BLANKS.
rad_user_def7	Not extracted. Loaded with BLANKS.
rad_user_def8	Not extracted. Loaded with BLANKS.
rad_user_def9	Not extracted. Loaded with BLANKS.
rad_user_def10	Not extracted. Loaded with BLANKS.

Field Name	Banner Data
rad_attr_key	Course key plus a sequence number. This is used to get the associated rad_attr_dtl records.
rad_crn	Course Reference Number taken from SFRSTCR. This value together with the term is used to find the associated rad_signal_dtl obtained from Course Signals.

R071CLAS - Class Record - HISTORIC

Updated: March 24, 2023

Several Banner tables are used to create the data on the rad_class_dtl for historic class records.

SHRTCKN	The Institutional Course Term Maintenance Repeating Table is read using the student's SPRIDEN_PIDM. This table is the driver for the historic class extraction.
SHRTCKL	SHRTCKL
SHRTCKG	The Institutional Courses Grade Repeating Table is read using the student's SPRIDEN_PIDM for a match between the SHRTCKN_SEQ_NO / SHRTCKG_TCKN_SEQ_NO and SHRTCKN_TERM_CODE / SHRTCKG_TERM_CODE.
SHRGRDE	The Grade Code Table is read using the SHRTCKG_GRDE_CODE_FINAL and SHRTCKL_LEVL_CODE.
SHRATTR	The History Course Section Attribute table is read using the student's SPRIDEN_PIDM for a match on the SHRTCKN_SEQ_NO and the SHRTCKN_TERM_CODE. Refer to the Special Topic on Attributes for details on how the class attributes are recorded on the User Defined fields on the rad_class_dtl.
SHRATTC	The History Course Section by CRN Attribute table is read using the student's SPRIDEN_PIDM for a match on the SHRTCKN_SEQ_NO and the SHRTCKN_TERM_CODE. Refer to the Special Topic on Attributes for details on how the class attributes are recorded on the User Defined fields on the rad_class_dtl.

Note: Several special edits may be made to determine if a class should be excluded from Degree Works. The Banner fields from SHRTCKN used in special edits are as follows (refer to the Technical UCX documentation for more details on the UCX table used in these descriptions):SHRTCKN_SUBJ_CODE: Historic class records will NOT be created if this code (rad_discipline) is Inactive. The Discipline Status on UCX-STU352 should be set to an I for Inactive discipline codes.

SHRTCKN_SCHD_CODE: A special edit may be made using this section code for ZERO credit classes. If particular Schedule Types are to be skipped and NOT rolled to Degree Works then load these Schedule Types into UCX-BAN001. The banner extract will read UCX-BAN001 for each non-blank SHRTCKN_SCHD_CODE. If a match is found AND the class has ZERO credits then the class will NOT be loaded into the rad_class_dtl.

SHRTCKN_SEQ_NUMB: A special edit may be made using this section code for ZERO credit classes. If particular sections are to be skipped and NOT rolled to Degree Works then load these Section codes into UCX-BAN002. The banner extract will read UCX-BAN002 for each non-blank SHRTCKN_SEQ_NUMB. If a match is found AND the class has ZERO credits then the class will NOT be loaded into the rad_class_dtl.

SHRTCKG_GMOD_CODE: A special edit may be made using this Gmod Code (Grade Type). If particular Gmod Codes are to be skipped and NOT rolled to Degree Works then load these Gmod Codes into UCX-BAN003. The banner extract will read UCX-BAN003 for each SHRTCKG_GMOD_CODE associated with each SHRTCKN historic class. If a match is found then the class will NOT be loaded into the rad_class_dtl.

SQL used to read the SHRTCKN table:

```

SELECT SHRTCKN_PIDM,
       SHRTCKN_TERM_CODE,
       SHRTCKN_SEQ_NO,
       SHRTCKN_CRN,
       SHRTCKN_SUBJ_CODE,
       SHRTCKN_CRSE_NUMB,
       SHRTCKN_CAMP_CODE,
       SHRTCKN_DEPT_CODE,
       SHRTCKN_DIVS_CODE,
       SHRTCKN_CRSE_TITLE,
       SHRTCKN_REPEAT_COURSE_IND,
       SHRTCKN_SEQ_NUMB,
       SHRTCKN_SCHD_CODE
  FROM SHRTCKN
 WHERE SHRTCKN_PIDM = :rBannerStuSPRIDEN.zPidm
 ORDER BY SHRTCKN_TERM_CODE,
          SHRTCKN_SUBJ_CODE,
          SHRTCKN_CRSE_NUMB;

```

These records are bridged into the RAD_CLASS_DTL table in Degree Works.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_course_key	The Course Key is a composite key composed of a 12-byte discipline code and a 12-byte course number.
rad_discipline	SHRTCKN_SUBJ_CODE (explanation above)
rad_course_num	SHRTCKN_CRSE_NUMB
rad_section	SHRTCKN_SEQ_NUMB (explanation above)
rad_course_title	SHRTCKN_CRSE_TITLE
rad_school	SHRTCKL_LEVL_CODE. One rad_class_dtl is created for each school/level found on SHRTCKL. For example, if a one SHRTCKL record has a level of UG and another has a

Field Name	Banner Data
	level of GR then two rad_class_dtl records will be created – each with a different rad_school value.
rad_division	SHRTCKN_DIVS_CODE
rad_dept	SHRTCKN_DEPT_CODE
rad_deg_interest	Not extracted. Loaded with BLANKS.
rad_class_type	Not extracted. Loaded with BLANKS.
rad_acad_votech	Not extracted. Loaded with BLANKS.
rad_audit_flag	If the SHRTCKG_GMOD_CODE = A, the audit flag is set to Y. Otherwise it is set to N.
rad_insuff_flag	Set to N. However, if this class is coded with the Repeat-Course-Indicator = E or A and the UCX-CFG020 BANNER RepeatPolicyE setting is B the insuff-flag will be set to Y to force the class into the insufficient section. In addition, the RepeatPolicy and RepeatPointer fields will be blanked out.
	When the CFG020 BANNER Averaged Repeats Count flag is set to Y the Insufficient flag is left as-is and the rad_pass_flag is set to N. This way the class can apply to major/minor GPAs as needed.
	When the pre-audit stage sees that the class is withdrawn (see rad_withdr_flag below) this rad_insuff_flag is set to Y and the class status is set to WD.
rad_inprog_flag	Loaded with a default value of N. However, an In-Progress flag has been added to UCX-STU385. If this flag is set to Y for a given UCX-STU385 Key (School + Grade Type + Final Grade) then this rad_inprog_flag will be set to Y. This new flag is ONLY used for historical classes stored in SHRTCKN as current classes in SFRSTCR are always considered In-Progress (Y in the rad_inprog_flag).
rad_withdr_flag	If the SHRTCKG_GRDE_CODE_FINAL = W, the withdraw flag is set to Y. Otherwise it is set to N.
rad_incomp_flag	If the SHRGRDE_IMPCMP_IND = N, the incomplete flag is set to N.
rad_pass_flag	SHRGRDE_PASSED_IND

Field Name	Banner Data
	This will be set to N for averaged repeats when Averaged Repeats Count flag is set to Y.
rad_credits	SHRTCKG_CREDIT_HOURS.
rad_credits_earn	If the SHRGRDE_IMPCMP_IND = N, the SHRTCKG_CREDIT_HOURS are loaded into the credit hours earned. Otherwise the credit hours earned are loaded with all zeroes. In addition, if the SHRGRDE_COMPLETED_IND is N the credit hours earned are also loaded with zeroes – regardless of the IMPCMP_IND flag.
rad_gpa_credits	If the SHRGRDE_GPA_IND = Y, the SHRTCKG_CREDIT_HOURS are loaded into the GPA credits.
rad_grade_points	SHRGRDE_QUALITY_POINTS times the rad_credits. The Banner quality points are converted to the RAD format of 0999v999. for example, 12 becomes 0012000, 10.5 becomes 0010500 and 6 becomes 0006000.
rad_credit_type	Defaulted to AC for Academic Credit.
rad_class_status	Loaded with a default of A. If the class has been repeated with a SHRTCKN_REPEAT.Course_IND = E (bad grade), and HE is loaded. If the class has been repeated with a SHRTCKN_REPEAT.Course_IND = A (averaged grade), an HA is loaded. If the class has been retaken with a SHRTCKN_REPEAT.Course_IND = I (grade to be included), an HI is loaded. If a current in-progress class is being repeated with a class in SHRTCKN, the rad_class_status is loaded with a CH (Current and History).
	These repeat codes can be listed in the dash.audit.repeat.codes setting. After you enable the UCX-RPT036 Show Repeated flag you will see these repeated classes showing a repeated indicator in the Responsive Dashboard worksheet.
rad_grade_type	SHRTCKG_GMOD_CODE
rad_pass_fail	If SHRTCKG_GMOD_CODE = P, the pass fail flag is set to Y. Otherwise it is set to N.
rad_repeat_ptr	If the SHRTCKN_REPEAT.Course_IND is an A, E or I and the class is being repeated for

Field Name	Banner Data
	<p>a better grade (not repeatable) the Course Key (Subject + Course Number) is checked against the DAP equivalency table, dap_eqv_crs_mst (UCX-CFG070), to determine if the historic class changed Course Keys over time. If a new Course Key equivalent is found, it is loaded into the rad_repeat_ptr. Otherwise the rad_course_key from the historic class is loaded.</p>
	<p>Otherwise it is loaded with BLANKS.</p>
rad_repeat_plcy	<p>If the SHRTCKN_REPEAT_COURSE_IND is an A (Averaged) then the UCX-CFG020 BANNER Repeat Skip A flag is checked. If it is a Y, then the class will be skipped and will not be rolled to Degree Works. If it is an N, then the UCX-CFG020 BANNER Repeat Policy A value will be loaded into the rad_repeat_plcy.</p>
	<p>A Banner repeat policy of B may be used for these classes. If set to B the insuff-flag will be set to Y and the repeat_ptr and repeat_plcy will be blanked out. This allows these classes to be displayed in the insufficient section of the report, but they will still impact the GPA.</p>
	<p>If the SHRTCKN_REPEAT_COURSE_IND is an E (Excluded) then the UCX-CFG020 BANNER Repeat Skip E flag is checked. If it is a Y, then the class will be skipped and will not be rolled to Degree Works. If it is an N, then the UCX-CFG020 BANNER Repeat Policy E value will be loaded into the rad_repeat_plcy.</p>
	<p>A special repeat policy of ZERO 0 can be used for these classes. If set to ZERO the rad_credits_earn, rad_gpa_credits and rad_grade_points will be set to 000000. This allows these special classes to be displayed on degree audits, but with no impact to the credits earned or GPA.</p>
	<p>A Banner repeat policy of B can be used for these classes. If set to B the rad_credits_earn, rad_gpa_credits and rad_grade_points will be set to 000000, the insuff-flag will be set to Y and the repeat_ptr and repeat_plcy are blanked out. This allows these special classes to be displayed in the insufficient section of the</p>

Field Name	Banner Data
	report, but with no impact to the credits earned or GPA.
	<p>If the SHRTCKN_REPEAT_COURSE_IND is an I (Included) the Banner Student Extract will perform special edit checks to determine if the course is repeatable or is being repeated for a better grade. For details on these edits, review the UCX-CFG020 BANNER Repeatable Option flag. If based on these special rules the course is found to be NOT repeatable and instead is being taken for a better grade the UCX-CFG020 BANNER Repeat Policy I value will be loaded into the rad_repeat_plcy.</p>
	<p>A Banner repeat policy of B may be used for these classes. If set to B these included classes will apply to rules as normal classes. The repeat_ptr and repeat_plcy will be blanked out for these classes.</p>
	<p>For more information, see the UCX-AUD047 Repeat Policies topic.</p>
rad_session	Not extracted. Loaded with BLANKS.
rad_location	SHRTCKN_CAMP_CODE.
rad_final_grade	SHRTCKG_GRDE_CODE_FINAL. The rad_final_grade has been increased to 6-bytes to match the length of the Banner SHRGRDE_GRDE_CODE.
rad_final_gr_num	SHRGRDE_QUALITY_POINTS. Multiple quality point values may be defined for a given SHRGRDE_LEVL_CODE (rad_school), SHRGRDE_CODE (rad_final_grade) and SHRGRDE_TERM_CODE_EFFECTIVE (rad_term). After a match on LEVL_CODE and grade CODE is made for a given SHRTCKN record the SHRGRDE records are read in descending term sequence. When the SHRTCKN term is GREATER THAN or EQUAL to the SHRGRDE term the associated SHRGRDE_QUALITY_POINTS will be loaded into the rad_final_gr_num. For example, a grade B+ in the undergraduate school UG starts out with Term 000000 as 3.5 quality points. In Term 200710 the quality points are changed to 3.3 quality points. If the SHRTCKN_TERM_CODE is GREATER THAN or EQUAL to the SHRGRDE

Field Name	Banner Data
	TERM_CODE_EFFECTIVE of 200710 then 3.3 quality points would be loaded. Otherwise the quality points would be loaded with 3.5 quality points. The Banner quality points are converted to the RAD format of 9999v999. for example, 0003.575 becomes 0003575, 4.0 becomes 0004000 and .5 becomes 0000500.
rad_instr1_id	Not extracted. Loaded with BLANKS.
rad_term	SHRTCKN_TERM_CODE
rad_stu_level	Not extracted. Loaded with BLANKS.
rad_user_def1	Not extracted. Loaded with BLANKS.
rad_user_def2	Not extracted. Loaded with BLANKS.
rad_user_def3	Not extracted. Loaded with BLANKS.
rad_user_def4	Not extracted. Loaded with BLANKS.
rad_user_def5	Not extracted. Loaded with BLANKS.
rad_user_def6	Not extracted. Loaded with BLANKS.
rad_user_def7	Not extracted. Loaded with BLANKS.
rad_user_def8	Not extracted. Loaded with BLANKS.
rad_user_def9	Not extracted. Loaded with BLANKS.
rad_user_def10	Not extracted. Loaded with BLANKS.
rad_attr_key	Course key plus a sequence number. This is used to get the associated rad_attr_dtl records.
rad_crn	Course Reference Number taken from SHRTCKN. This value together with the term is used to find the associated rad_signal_dtl obtained from Course Signals.

R083TRAN - Transfer Record

Updated: March 24, 2023

Several Banner tables are used to create the data on the rad_transfer_dtl.

Field name	Banner data
SHRTRCE	The Grade Component Student Audit table is read using the student's SPRIDEN_PIDM. This table is the driver for the transfer class extraction.
SHRTRCR	The Course Information table is read to find a match on the SHRTRCE_TRIT_SEQ_NO and SHRTRCR_TRIT_SEQ_NO and the SHRTRCE_SEQ_NO and SHRTRCR_SEQ_NO.

Field name	Banner data
SHRTRIT	The Grade Component Definition table is read using the student's SPRIDEN_PIDM for a match between the SHRTRCE_TRIT_SEQ_NO and SHRTRIT_SEQ_NO.
STVSBGI	The ETS Validation table is read using the SHRTRIT_SBGI_CODE.
SHRGRDE	The Grade Code Table is read using the SHRTRCE_GRDE_CODE and SHRTRCE_LEVL_CODE.
SHRTATT	The Transfer Course Attribute table is read using the student's SPRIDEN_PIDM for a match on the SHRTRCE_SEQ_NO. Refer to the Special Topic on Attributes for details on how the class attributes are recorded on the User Defined fields on the rad_transfer_dtl.

Note: Transfer class records will NOT be created if the SHRTRCE_SUBJ_CODE (rad_discipline) is Inactive. The Discipline Status on UCX-STU352 should be set to an I for Inactive discipline codes.

SQL used to read the SHRTRCE table:

```

SELECT SHRTRCE_PIDM,
       SHRTRCE_TRIT_SEQ_NO,
       SHRTRCE_TRAM_SEQ_NO,
       SHRTRCE_SEQ_NO,
       SHRTRCE_TRCR_SEQ_NO,
       SHRTRCE_TERM_CODE_EFF,
       SHRTRCE_LEVL_CODE,
       SHRTRCE_SUBJ_CODE,
       SHRTRCE_CRSE_NUMB,
       SHRTRCE_CRSE_TITLE,
       SHRTRCE_CREDIT_HOURS,
       SHRTRCE_GRDE_CODE,
       SHRTRCE_GMOD_CODE,
       SHRTRCE_COUNT_IN_GPA_IND,
       SHRTRCE_REPEAT.Course
FROM SHRTRCE
WHERE SHRTRCE_PIDM = :rBannerStuSPRIDEN.zPidm
ORDER BY SHRTRCE_TRIT_SEQ_NO;
    
```

These records are bridged into the RAD_TRANSFER_DTL table in Degree Works.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_school	SHRTRCE_LEVL_CODE
rad_tr_ets	SHRTRIT_SBGI_CODE
rad_tr_name	STVSBGI_SBGI_DESC
rad_tr_crse_key	SHRTRCR_TRANS_COURSE_NAME + SHRTRCR_TRANS_COURSE_NUMBERS.

Field Name	Banner Data
rad_tr_course	SHRTRCR_TCRSE_TITLE.
rad_course_key	The Course Key is a composite key composed of a 12-byte discipline code and a 12-byte course number: <ul style="list-style-type: none">• rad_discipline• rad_course_num
	<ul style="list-style-type: none">• SHRTRCE_SUBJ_CODE• SHRTRCE_CRSE_NUMB
rad_section	Not extracted. Loaded with BLANKS.
rad_course_title	SHRTRCE_CRSE_TITLE
rad_division	Not extracted. Loaded with BLANKS.
rad_dept	Not extracted. Loaded with BLANKS.
rad_class_type	Not extracted. Loaded with BLANKS.
rad_acad_votech	Not extracted. Loaded with BLANKS.
rad_audit_flag	If the SHRTRCE_GMOD_CODE = A, the audit flag is set to Y. Otherwise it is set to N.
rad_insuff_flag	Set to N. However, if this class is coded with the Repeat-Course-Indicator = E or A and the UCX-CFG020 BANNER RepeatPolicyE setting is B the insuff-flag will be set to Y to force the class into the insufficient section. In addition, the RepeatPolicy and RepeatPointer fields will be blanked out.
	When the CFG020 BANNER Averaged Repeats Count flag is set to Y the Insufficient flag is left as-is and the rad_pass_flag is set to N. This way the class can apply to major/minor GPAs as needed.
rad_inprog_flag	Loaded with a default value of N.
rad_withdr_flag	If the SHRTRECG_GRDE_CODE = W, the withdraw flag is set to Y. Otherwise it is set to N'.
rad_incomp_flag	If the SHRGDE_IMPCMP_IND = N, the incomplete flag is set to N.
rad_cr_exam_flag	Not extracted. Loaded with BLANKS.
rad_pass_flag	SHRGDE_PASSED_IND This will be set to N for averaged repeats when Averaged Repeats Count flag is set to Y.

Field Name	Banner Data
rad_credits	SHRTRCE_CREDIT_HOURS.
rad_credits_earn	If the SHRGRDE_IMPCMP_IND = N, the rad_credits are loaded into the credit hours earned. Otherwise the credit hours earned are loaded with all zeroes. In addition, if the SHRGRDE_COMPLETED_IND is N the credit hours earned are also loaded with zeroes – regardless of the IMPCMP_IND flag.
rad_gpa_credits	If the SHRGRDE_GPA_IND = Y, the rad_credits are loaded into the GPA credits.
rad_grade_points	SHRGRDE_QUALITY_POINTS times the rad_credits. The Banner quality points are converted to the RAD format of 999v999. for example, 12 becomes 012000, 10.5 becomes 0010500 and 6 becomes 0006000.
rad_credit_type	Loaded with a default value of TR.
rad_class_status	Loaded with a default value of A. If the class has been repeated with a SFRSTCR_REPEAT.Course_IND = E (bad grade), a TE is loaded. If the class has been repeated with a SFRSTCR_REPEAT.Course_IND = A (averaged grade), a TA is loaded. If the class has been retaken with a SFRSTCR_REPEAT.Course_IND = I (grade to be included), a TI is loaded. If the UCX-STU385 Override Transfer Repeat Policy is NOT blank, then a TO class status is loaded. If a current in-progress class is being repeated with a class in SHRTRCE, the rad_class_status is loaded with a CT (Current and Transfer).
	These repeat codes can be listed in the dash.audit.repeat.codes setting. After you enable the UCX-RPT036 Show Repeated flag you will see these repeated classes showing a repeated indicator in the Responsive Dashboard worksheet.
rad_grade_type	SHRTRCE_GMOD_CODE
rad_pass_fail	If SHRTRCE_GMOD_CODE = P, the pass fail flag is set to Y. Otherwise it is set to N.
rad_repeat_ptr	If the SHRTRCE_REPEAT.Course indicator is an A, E, or I and the class is being repeated for a better grade the Course Key (Subject + Course Number) is checked against the

Field Name	Banner Data
	<p>equivalency table, dap_eqv_crs_mst (UCX-CFG070), to determine if the transfer class changed Course Keys over time. If a new Course Key equivalent is found, it is loaded into the rad_repeat_ptr. Otherwise the rad_course_key from the transfer class is loaded.</p>
	<p>If the UCX-STU385 Override Transfer Repeat Policy is NOT BLANK the Course Key (Subject + Course Number) is checked against the equivalency table, dap_eqv_crs_mst (UCX-CFG070), to determine if the transfer class changed Course Keys over time. If a new Course Key equivalent is found, it is loaded into the rad_repeat_ptr. Otherwise the rad_course_key from the transfer class is loaded.</p>
	<p>Otherwise it is loaded with BLANKS.</p>
rad_repeat_plcy	<p>If the SHTRCE_REPEAT_COURSE is an A (Averaged) then the UCX-CFG020 BANNER Transfer Repeat Skip A flag is checked. If it is a Y, then the class will be skipped and will not be rolled to Degree Works. If it is an N, then the UCX-CFG020 BANNER Transfer Repeat Policy A value will be loaded into the rad_repeat_plcy.</p>
	<p>If the SHTRCE_REPEAT_COURSE is an E (Excluded) then the UCX-CFG020 BANNER Transfer Repeat Skip E flag is checked. If it is a Y, then the class will be skipped and will not be rolled to Degree Works. If it is an N, then the UCX-CFG020 BANNER Transfer Repeat Policy E value will be loaded into the rad_repeat_plcy.</p>
	<p>A special repeat policy of ZERO 0 can be used for these classes. If set to ZERO, the rad_credits_earn, rad_gpa_credits, and rad_grade_points will be set to 000000. This allows these special classes to be displayed on degree audits, but with no impact to the credits earned or GPA.</p>
	<p>If the SHTRCE_REPEAT_COURSE is an I (Included) then the UCX-CFG020 BANNER Transfer Repeat Policy I value will be loaded into the rad_repeat_plcy.</p>

Field Name	Banner Data
	If the UCX-STU385 Override Transfer Repeat Policy is NOT BLANK, then this Override Repeat Policy will be loaded.
	For more information, see the UCX-AUD047 Repeat Policies topic.
rad_location	Not extracted. Loaded with BLANKS.
rad_final_grade	SHRTRCE_GRDE_CODE. The rad_final_grade has been increased to 6-bytes to match the length of the Banner SHRTRCE_GRDE_CODE.
rad_final_gr_num	SHRGRDE_QUALITY_POINTS. Multiple quality point values may be defined for a given SHRGRDE_LEVL_CODE (rad_school), SHRGRDE_CODE (rad_final_grade) and SHRGRDE_TERM_CODE_EFFECTIVE (rad_term). After a match on LEVL_CODE and grade CODE is made for a given SHRTRCE record the SHRGRDE records are read in descending term sequence. When the SHRTRCE term is GREATER THAN or EQUAL to the SHRGRDE term the associated SHRGRDE_QUALITY_POINTS will be loaded into the rad_final_gr_num. For example, a grade 'B+' in the undergraduate school 'UG' starts out with Term '000000' as 3.5 quality points. In Term '200710' the quality points are changed to 3.3 quality points. If the SHRTRCE_TERM_CODE_EFF is GREATER THAN or EQUAL to the SHRGRDE TERM_CODE_EFFECTIVE of '200710' then 3.3 quality points would be loaded. Otherwise the quality points would be loaded with 3.5 quality points. The Banner quality points are converted to the RAD format of 9999v999. for example, 3.575 becomes 0003575, 4.0 becomes 0004000, and .5 becomes 0000500.
rad_term	SHRTRCE_TERM_CODE_EFF
rad_user_def1	Not extracted. Loaded with BLANKS.
rad_user_def2	Not extracted. Loaded with BLANKS.
rad_user_def3	Not extracted. Loaded with BLANKS.
rad_user_def4	Not extracted. Loaded with BLANKS.
rad_user_def5	Not extracted. Loaded with BLANKS.
rad_user_def6	Not extracted. Loaded with BLANKS.
rad_user_def7	Not extracted. Loaded with BLANKS.

Field Name	Banner Data
rad_user_def8	Not extracted. Loaded with BLANKS.
rad_user_def9	Not extracted. Loaded with BLANKS.
rad_user_def10	Not extracted. Loaded with BLANKS.
rad_attr_key	Together with the rad_id this links to the rad_attr_dtl – for class attributes.
rad_sis_key	A unique value for each transfer class for this student. This ends up on the rad_crn field on the rad_result_dtl. This 10 bytes are comprised of these values from SHRTRCE: TRIT_SEQ_NO (bytes 1 - 2) TRAM_SEQ_NO (bytes 3 - 4) TRCR_SEQ_NO (bytes 5 - 7) SEQ_NO (bytes 8 - 10)

R085ATTR - Attribute Record

Updated: March 25, 2022

Several Banner Class Attribute tables are used for the data in this record.

SHRATTR	Historic Class Attributes by PIDM and Sequence Numbers – processed first for the historic classes contained in the SHRTCKN Banner table. If records are found then the SHRATTC table will NOT be checked.
SHRATTC	Historic Class Attribute by PIDM, CRN and TERM – only processed if NO SHRATTR records are found for the historic class found in the SHRTCKN Banner table.
SHRTATT	Transfer Class Attributes by PIDM and Sequence Numbers – the SHRTRCE transfer record data is compared against the following SHRTATT sequence numbers: TRIT_SEQ_NO, TRAM_SEQ_NO, TRCR_SEQ_NO and TRCE_SEQ_NO. If a match is found a class attribute BIF record will be created.
SSRATTR	Current Class Attributes by PIDM, CRN and TERM – processed for current classes found in the SFRSTCR table. For each SFRSTCR current class record the CRN and TERM are used to read the SSRATTR table. If a match is found then the attribute code on that record will be used to create a class attribute BIF record. Multiple BIF records will be created if multiple attributes exist for a given class.

The attribute data will be loaded into the rad_attr_dtl with a special hardwired key of ATTRIBUTE.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_attr_key	Discipline + Course Number + - + 3-digit Sequence Number that is unique for each student. For example, if a student has ECON111 with attribute BLHS, AMST115 with attributes AL02, BLHS and WRT1 and ENGL106 with attribute WRT1 the keys would be: ECON111-001 (1 rad_attr_dtl), AMST115-002 (3 rad_attr_dtls) and ENGL106-003 (1 rad_attr_dtl).
rad_attr_code	Hardwired with ATTRIBUTE
rad_attr_value	The particular ATTR_CODE (occurs 39 times) from one of the above tables. Examples of values listed in the attribute key definition above are BLHS, AL02 and WRT1.

R091TEST - Test Record

Updated: March 25, 2022

The Banner Test table, SORTEST, is used to retrieve test data from the Banner database using the student's ID code (SPRIDEN_ID).

SQL used to read the SORTEST table:

```
SELECT SORTEST_PIDM,
       SORTEST_TESC_CODE,
       TO_CHAR(SORTEST_TEST_DATE, 'YYYYMMDD'),
       SORTEST_TEST_SCORE
  FROM SORTEST
 WHERE SORTEST_PIDM = :rBannerStuSPRIDEN.zPidm
 ORDER BY SORTEST_TESC_CODE,
          SORTEST_TEST_DATE;
```

Test scores are stored in the rad_test_dtl.

The UCX-SCR002, SCR003, and RPT046 entries should use elements 1292 and 1291 to pull the test data from the rad_test_dtl.

This is an optional table.

These records are bridged into the RAD_TEST_DTL table in Degree Works.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.

Field Name	Banner Data
rad_test_code	SORTEST_TEST_CODE. This element is the code that represents the test taken. For example: A-ENGL, A-MATH, A-HIST, VERBAL, S-MATH, and so on.
rad_test_score	SORTEST_TEST_SCORE. This element is the score or value associated with the test code. Left justified. Not zero-filled. If the Banner test score length is greater than the length of the test score in the rad_test_dtl then the score will be truncated and an error message will appear in the log file.
rad_test_date	SORTEST_TEST_DATE. This element is the date the test was taken. Format = CCYYMMDD.
rad_user_def1	Not extracted. Loaded with BLANKS.
rad_user_def2	Not extracted. Loaded with BLANKS.
rad_user_def3	Not extracted. Loaded with BLANKS.
rad_school	Not extracted. Loaded with BLANKS.
rad_degree_code	Not extracted. Loaded with BLANKS.
rad_term	SORTEST_TEST_DATE is converted to a term using the date range in STVTERM.

R100PDEG - Previous Institution Record

Updated: March 25, 2022

The Banner Prior College Degree table, SORDEGR, is used for most of the data in this record.

SQL used to read the SORDEGR table:

```

SELECT SORDEGR_PIDM,
       SORDEGR_SBGI_CODE,
       SORDEGR_DEGC_CODE,
       TO_CHAR(SORDEGR_ATTEND_FROM, 'YYYYMMDD'),
       TO_CHAR(SORDEGR_ATTEND_TO, 'YYYYMMDD'),
       TO_CHAR(SORDEGR_DEGC_DATE, 'YYYYMMDD')
  FROM SORDEGR
 WHERE SORDEGR_PIDM = :rBannerStuSPRIDEN.zPidm
 ORDER BY SORDEGR_SBGI_CODE;

```

These records are bridged into the RAD_PREVINST_DTL table in Degree Works.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_prev_degree	SORDEGR_DEGC_CODE
rad_prev_date	SORDEGR_DEGC_DATE
rad_prev_ets	SORDEGR_SGBI_CODE
rad_prev_major	Not extracted. Loaded with BLANKS.

Field Name	Banner Data
rad_confer_flag	Not extracted. Loaded with BLANKS.
rad_tr_start	SORDEGR_ATTEND_FROM
rad_tr_stop	SORDEGR_ATTEND_TO
rad_term	rad_primary_mst Active Term
rad_user_def1	Not extracted. Loaded with BLANKS.
rad_user_def2	Not extracted. Loaded with BLANKS.
rad_user_def3	Not extracted. Loaded with BLANKS.
rad_user_def4	Not extracted. Loaded with BLANKS.
rad_user_def5	Not extracted. Loaded with BLANKS.
rad_user_def6	Not extracted. Loaded with BLANKS.
rad_user_def7	Not extracted. Loaded with BLANKS.
rad_user_def8	Not extracted. Loaded with BLANKS.
rad_user_def9	Not extracted. Loaded with BLANKS.
rad_user_def10	Not extracted. Loaded with BLANKS.

R111NCRS - Non Course Record

Updated: March 25, 2022

Several tables are used to obtain Non Course data from Banner.

Field	Description
SHRNCRS	Read using the student's PIDM to obtain Non Course data from Banner.
STVNCRQ	Read using the shrncrs_ncrq_code.
STVNCST	Read using the shrncrs_ncst_code.
SHRQPNM	Read using the student's PIDM to obtain Non-Course Test and Exam data from Banner.
STVQPTP	Read using the shrqptp_code.

SQL used to read the SHRNCRS table

```

SELECT SHRNCRS_PIDM,
       SHRNCRS_SEQ_NO,
       SHRNCRS_NCRQ_CODE,
       SHRNCRS_NCST_CODE
  FROM SHRNCRS
 WHERE SHRNCRS_PIDM = :rBannerStuSPRIDEN.zPidm
 ORDER BY SHRNCRS_NCRQ_CODE,
          SHRNCRS_SEQ_NO DESC;

```

These records are bridged into the RAD_NONCRSE_DTL table in Degree Works.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_non_course	SHRNCRS_NCRQ_CODE or SHRQPNM_QPTP_CODE depending on what values are found in the Banner database for a given student.
rad_non_score	SHRNCST_NCST_CODE if the UCX-CFG020 BANNER Non Course Score = C, or BLANK; STVNCST_SATISFIED_IND if the UCX-CFG020 BANNER Non Course Score = I.
	Nothing is loaded into this score field for the SHRQPNM table.
rad_non_title	STVNCRQ_DESC if the SHRNCRS_NCRQ_CODE is being loaded or STVQPTP_DESC if the SHRQPNM_QPTP_CODE is being loaded
rad_term	SHRNCRS_COMPLETE_DATE is converted to a term using the date range in STVTERM.
rad_school	Not extracted. Loaded with BLANKS.
rad_degree_code	Not extracted. Loaded with BLANKS.

R121CUST - Athletic Data - Custom Record

Updated: September 29, 2023

The Banner Student Athlete table SGRATHE is used for getting the student's first date of attendance with regard to athletic eligibility. This data is extracted automatically, but only when the SGRATHE section exists in integration.banner.extract.config.

The Banner bridge will load this date as a custom record with a code of AEAFIRSTDATE. The date is matched against the STVTERM table to find the corresponding term. This term is then bridged as AEAFIRSTTERM.

SQL used to read the SGRATHE table

```
SELECT SGRATHE_ATTEND_FROM_DATE
      FROM SGRATHE a
     WHERE SGRATHE_PIDM = :rBannerStuSPRIDEN.zPidm
   ORDER BY SGRATHE_ACTIVITY_DATE desc;
```

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_custom_code	Hardwired with AEAFIRSTDATE"

Field Name	Banner Data
rad_custom_value	SGRATHE_ATTEND_FROM_DATE.
rad_custom_title	Not extracted. Loaded with BLANKS.
rad_term	Not extracted. Loaded with BLANKS.
rad_school	Not extracted. Loaded with BLANKS.
rad_degree_code	Not extracted. Loaded with BLANKS.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_custom_code	Hardwired with AEAFIRSTTERM
rad_custom_value	STVTERM_CODE where the STVTERM_START_DATE and STVTERM_END_DATE correspond to the SGRTHE_ATTEND_FROM_DATE.
rad_custom_title	Not extracted. Loaded with BLANKS.
rad_term	Not extracted. Loaded with BLANKS.
rad_school	Not extracted. Loaded with BLANKS.
rad_degree_code	Not extracted. Loaded with BLANKS.

R121CUST - Dynamic Retrieval - Custom Record

Updated: September 30, 2022

If your institution has stored non-standard data in Banner that is required in Degree Works to accurately define the rules to complete a degree but is not being extracted through the Degree Works Banner Bridge program, UCX-BAN080 may be used to specify this data so it can be extracted into the rad_custom_dtl.

This table uses Dynamic SQL to obtain data from Banner tables from pre-defined values in UCX-BAN080. The table, column, and keyword are used to load the custom data to be extracted from Banner.

Currently numeric data cannot be retrieved dynamically. Only codes or character strings can be retrieved.

For a complete discussion and definition of the UCX-BAN080 table, see the [UCX-BAN080 Banner Custom Data Definitions](#) topic.

Only the pieces of data obtained from Banner tables are defined below.

Field Name	Banner Data
rad_custom_code	UCX-BAN080 Keyword: the portion of the UCX Key BEFORE the colon (:). For example, the UCX Key for the Core Requirements being Satisfied might be CORESAT. The UCX Key with the database column reference would then be CORESAT:COLUMN.

Field Name	Banner Data
rad_custom_value	Banner database value retrieved using the SQL select clause defined in UCX-BAN080. The UCX-BAN080 keys containing the database COLUMN, TABLE, WHERE, and ORDER BY clauses are used to form a valid SQL statement to retrieve the desired piece of data from the Banner database.
rad_custom_title	Not extracted. Loaded with BLANKS.
rad_term	Not extracted. Loaded with BLANKS.
rad_school	Optional depending on BAN080 setup.
rad_degree_code	Optional depending on BAN080 setup.

R121CUST - Student Attributes - Custom Record

Updated: March 25, 2022

The Banner Student Attribute table SGRSATT is used for the data in this record using the student's PIDM.

The attribute data will be loaded into the rad_custom_dtl with a special hardwired custom_code of ATTRIBUTE.

SQL used to read the SGRSATT table

```
SELECT SGRSATT_PIDM,
       SGRSATT_TERM_CODE_EFF ,
       SGRSATT_ATTS_CODE
  FROM SGRSATT a
 WHERE SGRSATT_PIDM = :rBannerStuSPRIDEN.zPidm and
       a.SGRSATT_TERM_CODE_EFF =
 (SELECT MAX(b.SGRSATT_TERM_CODE_EFF)
    FROM SGRSATT b
   WHERE b.SGRSATT_PIDM = a.SGRSATT_PIDM)
 ORDER BY SGRSATT_ATTS_CODE,
          SGRSATT_TERM_CODE_EFF;
```

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_custom_code	Hardwired with ATTRIBUTE
rad_custom_value	SGRSATT_ATTS_CODE.
rad_custom_title	Not extracted. Loaded with BLANKS.
rad_term	SGRSATT_TERM_CODE_EFF.
rad_school	Not extracted. Loaded with BLANKS.
rad_degree_code	Not extracted. Loaded with BLANKS.

R126REPT - Dynamic Retrieval - Report Record

Updated: March 25, 2022

If your institution has stored non-standard data in Banner that is desired to display in the Degree Works audits but is not being extracted through the Degree Works Banner Bridge program, UCX-BAN080 may be used to specify this data so that it may be extracted into the rad_report_dtl.

This table uses Dynamic SQL to obtain data from Banner tables from pre-defined values in UCX-BAN080. The table, column, and keyword are used to load the custom data to be extracted from Banner.

The UCX-BAN080 Banner Custom Data Definitions are used to generate the records in this table if REPORT is found in the UCX Key definition for a given Keyword (for example, KEYWORD:REPORT). Otherwise the custom data will be loaded into the rad_custom_dtl (see the previous definition above – R121CUST).

The Banner data will be loaded into the rad_report_dtl with the rad_report_code loaded with the UCX-BAN080 Keyword used in the UCX Key. The SQL Table, Column and Where Clause defined in that table are used to format a valid SQL statement used to retrieve the custom piece of data from the Banner database. If the UCX Value for the REPORT keyword is NOT BLANK and does NOT contain NONE then the custom piece of data will then be used in the STV????_CODE to lookup the appropriate STV???? Table to get the description contained in the STV????_DESC. The STV description will be loaded into the rad_report_value. If the UCX Value for the REPORT keyword IS BLANK or contains NONE, the custom piece of data will be loaded into the Report Value and NO STV table lookup will be performed.

For example, if Registration Holds are desired from the Banner database the UCX Value for the REPORT keyword should contain the STVHLDD table name. Then the STVHLDD_CODE retrieved from Banner using the UCX-BAN080 Table, Column and Where clause will then be used to look up the STVHLDD table to get the STVHLDD_DESC.

Currently numeric data CANNOT be retrieved dynamically. Only codes or character strings may be retrieved.

Refer to the DGW Technical Guide UCX documentation for a complete discussion and definition of the UCX-BAN080 table.

Only the pieces of data obtained from Banner tables are defined below.

Field Name	Banner Data
rad_id	The SPRIDEN_ID is used for all student/staff oriented data.
rad_report_code	UCX-BAN080 Keyword: the portion of the UCX Key BEFORE the colon (:). For example, the UCX Key for a Registration Hold the keyword might be STUDENTHOLD, for the First Term the key might be FIRSTTERM and for the Academic Standing the key might be ACADSTANDING, and so on.
rad_report_value	If the UCX Value for the REPORT keyword contains an STV validation table name, then this field will contain the STV????_DESC. The ??? are replaced by the particular CODE that is being retrieved from the Banner database. However, if the UCX Value for the REPORT keyword is BLANK or contains NONE then the CODE

Field Name	Banner Data
	that is retrieved from the Banner database will be loaded into the rad_report_value.
rad_term	Loaded with the STV???? Table used to retrieve the description.
rad_seq	Hardwired with 0001. However, if multiple records exist for a given keyword this sequence number will be incremented by 1 for each piece of data found.
rad_school	Optional depending on BAN080 setup.
rad_degree_code	Optional depending on BAN080 setup.

R171SHPU - SHP User Record

Updated: March 25, 2022

The Banner ID and Name table, SPRIDEN, is used for most of the data in this record.

Only the pieces of data obtained from Banner tables are defined below.

These records are bridged into the SHP_USER_MST table in Degree Works.

Field Name	Banner Data
shp_access_id	SPRIDEN_ID
shp_id	SPRIDEN_ID
shp_access_code	STUDENT access codes: (1) Custom SQL may be written for the PASSWORDSTU" entries and added to the integration.banner.extract.config Shepherd setting entries (Select, From, Where and Order By clauses). Multiple fields or subsets of fields may be concatenated to generate a password. The banner student extract will use this SQL to read the database and format the desired password. Warning! Make sure the appropriate users are given access to any new database tables that may be used for this custom SQL. (2) If the PASSWORDSTU SELECT clause is blank a random password will be automatically generated using uppercase letters and numbers. ADVISOR access codes: (1) Custom SQL may be written for the PASSWORDADV entries and added to the integration.banner.extract.config Shepherd setting (Select, From, Where and Order By clauses). Multiple fields or subsets of fields may be concatenated to generate a password.

Field Name	Banner Data
------------	-------------

Warning! Make sure the appropriate users are given access to any new database tables that may be used for this custom SQL.

(2) If the PASSWORDADV SELECT clause is blank a random password will be automatically generated using uppercase letters and numbers.

STAFF access codes:

(1) Custom SQL may be written for the PASSWORDSTF entries and added to the integration.banner.extract.config Shepherd setting (Select, From, Where and Order By clauses). Multiple fields or subsets of fields may be concatenated to generate a password.

Warning! Make sure the appropriate users are given access to any new database tables that may be used for this custom SQL.

(2) A password may be included in a /local/sql/ staff.ids file (the file name does not have to be staff – it can be any valid file name, but the .ids extension is required) that contains the ID codes of the staff members to be loaded into the rad_primary_mst and shp_user_mst. The format of the .ids file is ID (9) + BLANK (1) + PASSWORD (64). For example:

123456789 staff!pass

10000222 another-pass

9988443 yet,anotherpass

159524

1234567890123456789012345678901234567890123456789012345678901234

The first 9-bytes must be the Banner SPRIDEN_ID code and are REQUIRED. They will be into the individual's rad_id on the rad_primary_mst. If the SHP AccessCode is to be loaded from the .ids file, then the 10th byte MUST be BLANK, with the next 64-bytes considered the staff member's shp_access_code.

(3) If the PASSWORDSTF SELECT clause is blank and NO Access Code is supplied in a staff.ids file a RANDOM password will be automatically generated using uppercase letters and numbers.

A Change Password configuration flag exists in the UCX-CFG020 WEBPARAMS record. If this flag is set to N then the Access Code (shp_access_code on the shp_user_mst) will NOT be changed by

Field Name	Banner Data
	the Banner extracts, only added when the student, advisor or staff record is originally created.
shp_user_class	Hardwired with STU if loaded from bannerextract STUDENT (ban40), the UCX-CFG020 BANNER Advisor User Class if loaded from bannerextract ADVISOR (ban45) and the UCX-CFG020 BANNER Staff User Class if loaded from bannerextract STAFF (ban45). If left blank the Advisor User Class is loaded with ADV and the Staff User Class is loaded with REG.
shp_group_list	Not extracted. Loaded with BLANKS.
shp_key_list	Not extracted. Loaded with BLANKS.
shp_sso_id and shp_finder_id	If the GOBUMAP_UDC_ID is being used for CAS authentication instead of the SPRIDEN_PIDM/SPRIDEN_ID, then the Banner Student and Staff/Advisor extracts will load the UDC_ID into the shp_user_mst.shp_sso_id and shp_finder_id. If the GORADID_ADDITIONAL_ID is being extracted then the Banner Student and Staff/Advisor extracts will load the ADDITIONAL_ID into the shp_user_mst.shp_sso_id and shp_finder_id. If both GOBUMAP and GORADID are specified in the integration.banner.extract.config Shepherd setting then the GOBUMAP UDC_ID will be used if it is found. If not found then the ADDITIONAL_ID will be used if it is found. If IDs are still blank, then the SPRIDEN_ID is loaded.

R190DEQV - DAP Equivalent Course Record

Updated: September 30, 2022

Several Banner database tables are used in the building of the Degree Works course equivalent records.

SCREQIV	Course Equivalent records
SCBCRSE	Course Master records
SCBCRKY	Course Start/End dates
STVTERM	Term Codes
STVACYR	Academic Years – Catalog Years
STVCSTA	Validation table for CSTA Codes

A special UCX-CFG074, has been created for the Banner Equivalency extract to store Course Keys (Subject Code + Course Number) that have been reused over time and have been Active at different times for different courses. If reused Course Keys exist in your Banner database add them to UCX-CFG074 before running the Banner Equivalency extract. Use Controller to add the

reused Course Key (4-byte Subject Code and 5-byte Course Number) into the UCX Key in UCX-CFG074. The UCX Value (Description) may be left blank. This UCX-CFG074 table will be used in the processing of the SCREQIV equivalency records outlined below.

The Banner EQUIV extract, ban43, has been modified to follow the UCX-CFG020 BANNER Term As Catalog Year flag. If this flag is set to Y the Banner UCX Extract (ban44) will load STVTERM codes instead of STVACYR Academic Year codes into the STU035 Catalog Years. Then when the EQUIV extract is executed the actual Catalog Year references referenced below will contain Term Codes.

Note: The rules defined in Scribe must also reference term codes so that the equivalencies are processed correctly when audits are generated.

The logic flow for the Banner Equivalent extract program is as follows:

1. Read the entire SCREQIV table, sorted by the NEW Subject Code and Course Number:

SQL used to read the SCREQIV table	Degree Works Counterpart
SELECT SCREQIV_SUBJ_CODE,	New Discipline
SCREQIV_CRSE_NUMB,	New Course Number
SCREQIV_EFF_TERM,	New Catalog Year
SCREQIV_SUBJ_CODE_EQIV,	Old Discipline
SCREQIV_CRSE_NUMB_EQIV,	Old Course Number
SCREQIV_START_TERM,	Old Catalog Year
SCREQIV_END_TERM	Old Catalog Year
FROM SCREQIV ORDER BY SCREQIV_SUBJ_CODE,	SCREQIV_CRSE_NUMB;

For every equivalent SCREQIV record read, all three of the TERM codes (effective, start and end) are looked up on the STVTERM table to get the associated STVTERM_ACYR_CODES (Academic Year Codes - Catalog Years in Degree Works). The EQUIV extract processes one SCREQIV equivalent record at a time.

Note: As mentioned above if the UCX-CFG020 BANNER Term as Catalog Year flag is set to Y then the actual SCREQIV Term Codes will be used as Catalog Years.

2. Read the SCBCRSE record for the NEW Subject Code and Course Number to determine if it is Active. The record with the highest Effective Term is found and then the STVCSTA_CODE is looked up on the STVCSTA table to make sure the STVCSTA_ACTIVE_IND = A for Active.

The SQL is as follows (the New Subject Code is loaded into zHoldSubjCode and the New Course Number is loaded into the zHoldCrseNumb for the database lookup):

```
SELECT A.SCBCRSE_SUBJ_CODE,
       A.SCBCRSE_CRSE_NUMB,
       A.SCBCRSE_EFF_TERM,
       A.SCBCRSE_DIVS_CODE,
       A.SCBCRSE_DEPT_CODE,
       A.SCBCRSE_CSTA_CODE,
```

```

A.SCBCRSE_TITLE,
A.SCBCRSE_CREDIT_HR_IND,
A.SCBCRSE_CREDIT_HR_LOW,
A.SCBCRSE_CREDIT_HR_HIGH,
A.SCBCRSE_REPEAT_LIMIT,
A.SCBCRSE_MAX_RPT_UNITS
FROM SCBCRSE A, STVCSTA
WHERE A.SCBCRSE_SUBJ_CODE = :zHoldSubjCode AND
      A.SCBCRSE_CRSE_NUMB = :zHoldCrseNumb AND
      A.SCBCRSE_EFF_TERM =
      (SELECT MAX(B.SCBCRSE_EFF_TERM)
       FROM SCBCRSE B
       WHERE B.SCBCRSE_SUBJ_CODE = A.SCBCRSE_SUBJ_CODE AND
             B.SCBCRSE_CRSE_NUMB = A.SCBCRSE_CRSE_NUMB)
ORDER BY A.SCBCRSE_EFF_TERM DESC;

```

If the SCBCRSE record is found the SCBCRSE_CSTA_CODE will be looked up on STVCSTA. If the STVCSTA_ACTIVE_IN = A the course is Active. If the course is NOT Active the SCREQIV record will be skipped. Processing will stop for the SCREQIV record and a new SCREQIV record will be read (return to Step #1).

3. If the UCX-CFG020 BANNER Inactive in SCBCRKY flag = N then go to Step #4. If a SCBCRKY record is found for the NEW Course Key (Subject Code + Course Number) and a SCBCRKY_TERM_CODE_END of '999999' is NOT found, processing will stop for the SCREQIV record and a new SCREQIV record will be read (return to Step #1).

If the UCX-CFG020 BANNER Cross List in SCREQIV flag = N then go to Step #5. If the SCREQIV_END_TERM is 999999 or if the SCREQIV_END_TERM is not 999999 and integration.banner.extract.equiv.crosslistedRange is enabled then check for cross listed references in the SCBCRKY table for both the OLD and NEW Course Keys (Subject Code + Course Number). Otherwise go to Step #5.

If a SCBCRKY record is found for the OLD Course Key (Subject Code + Course Number) and a Term of 999999 then check the NEW course. Otherwise go to Step #5.

If a SCBCRKY record is found for the NEW Course Key (Subject Code + Course Number) and a Term of 999999, the course is cross-listed and is NOT a course equivalent.

The OLD and NEW Course Keys will then be looked up on UCX-CFG074 – Reused Course Keys. If either the OLD or the NEW Course Key is found on UCX-CFG074 it is not considered a cross-listed course and will be thrown out (return to Step #4).

The cross-listed course is then written to the UCX-CFG073 Cross Listed Course table for use by the parser as additional rules are automatically added (not visible in Scribe). If SCREQIV_END_TERM is not 999999 then the start and end terms define a range of terms. This range of terms is written as a pair of CFG073 cross-listed records with the start and end terms. The auditor also uses these enhanced rules as it processes requirements.

Note: The NEW course (SUBJ_CODE and CRSE_NUMB) is loaded into the UCX-KEY while the OLD course (SUBJ_CODE_EQIV and CRSE_NUMB_EQIV) is loaded into the value area of the UCX-CFG073 record. This cross-listing record is only used by the parser if the UCX-KEY, the NEW course, is found in a scribed rule. If found, the parser then allows the OLD course to be used to satisfy the same requirement.

Thus, the SCREQIV will be skipped. Processing will stop for the SCREQIV record and a new SCREQIV record will be read (return to Step #1).

4. Read the SCBCRSE record for the OLD Course Key (Subject Code + Course Number) to determine if it is included in the Current Course Catalog. The record with the highest Effective Term is found. The SQL is as follows (the OLD Subject Code is loaded into zHoldSubjCode and the OLD Course Number is loaded into the zHoldCrseNumb for the database lookup):

```

SELECT A.SCBCRSE_SUBJ_CODE,
       A.SCBCRSE_CRSE_NUMB,
       A.SCBCRSE_EFF_TERM,
       A.SCBCRSE_DIVS_CODE,
       A.SCBCRSE_DEPT_CODE,
       A.SCBCRSE_CSTA_CODE,
       A.SCBCRSE_TITLE,
       A.SCBCRSE_CREDIT_HR_IND,
       A.SCBCRSE_CREDIT_HR_LOW,
       A.SCBCRSE_CREDIT_HR_HIGH,
       A.SCBCRSE_REPEAT_LIMIT,
       A.SCBCRSE_MAX_RPT_UNITS
  FROM SCBCRSE A, STVCSTA
 WHERE A.SCBCRSE_SUBJ_CODE = :zHoldSubjCode          AND
       A.SCBCRSE_CRSE_NUMB = :zHoldCrseNumb        AND
       A.SCBCRSE_EFF_TERM =
      (SELECT MAX(B.SCBCRSE_EFF_TERM)
        FROM SCBCRSE B
       WHERE B.SCBCRSE_SUBJ_CODE = A.SCBCRSE_SUBJ_CODE AND
             B.SCBCRSE_CRSE_NUMB = A.SCBCRSE_CRSE_NUMB)
 ORDER BY A.SCBCRSE_EFF_TERM DESC;

```

Depending on the UCX-CFG020 BANNER Current Course setting one of the rules below will be followed:

- A - the SCBCRSE_CSTA_CODE is used to lookup the STVCSTA record. If the STVCSTA_ACTIVE_IND = A for Active the OLD course is considered a Current Course.
- C – if the SCBCRSE_CSTA_CODE = C the OLD course is considered a Current Course.
- K – lookup the SCBCRKY record using the OLD Course Key. If the SCBCRKY_TERM_CODE_END = 999999 the course is considered a Current Course.

If the OLD Course is considered a Current Course the OLD Course Key will be looked up on UCX-CFG074 – Reused Course Keys. If the OLD Course Key is NOT found on UCX-CFG074 it is considered a circular or reversal course. The SCREQIV record will be written to a logdebug/BAN43_REVERSAL flat file and then will be skipped. Processing will stop for the SCREQIV record and a new SCREQIV record will be read (return to Step #1).

5. If the SCREQIV_START_TERM is GREATER THAN the SCREQIV_EFF_TERM then SKIP the SCREQIV record. This means that the OLD equivalent is more current than the NEW equivalent which is NOT valid for Degree Works. Processing will stop for the SCREQIV record and a new SCREQIV record will be read (return to Step #1).
6. Read the STVACYR table to find the Lowest Academic Year and the Highest Academic Year for comparison purposes with the Starting and Ending Academic Years on the SCREQIV records. Typical values for those fields might be 0000 for the Lowest Academic Year (beginning of time) and 9999 for the Highest Academic Year (end of time).

These two values, Lowest/Highest Academic Years, are used to determine if the @ can be used when creating the BIF R190DEQV records for the dap_eqv_crs_mst. If the @ can be

used for the Old Catalog Year it would avoid the creation of a large number of dap_eqv_crs_mst records that would be identical except for the Old Catalog Year.

To use the @ in the R190DEQV equivalency records the OLD Course Key (Subject Code + Course Number) must NOT be REUSED. Reused Course Keys are checked above in Step #4 and if the OLD Course Key is found in UCX-CFG074 the @ will NOT be used.

7. Load the Old and New Course Equivalent values into the following BIF record:

Field Name	Banner Data
Old Catalog Year	See Rules below in Steps #8 #10.
Old Discipline	SCREQIV_SUBJ_CODE_EQIV
Old Course Number	SCREQIV_CRSE_CODE_EQIV
New Catalog Year	Loaded with a @ so the equivalencies can be used for students with various catalog years.
New Discipline	SCREQIV_SUBJ_CODE
New Course Number	SCREQIV_CRSE_NUMB

8. Load the BIF Old Catalog Year using the following rules:

If the SCREQIV Start Academic Year matches the Lowest Academic Year (calculated in Step #6) set the Old Catalog Year to an @. Normally the Lowest Academic Year (beginning of time) would be something like 0000.

Note: As mentioned above if the UCX-CFG020 BANNER Term as Catalog Year flag is set to Y then the actual SCREQIV Term Codes will be used as Catalog Years. Thus, the Academic Year references below will actually contain Starting and Ending Term Codes.

For example #1, an SCREQIV record contains the following:

New Course Key	MATH123
New Academic Year	2003
Old Course Key	MATH111
Starting Academic Year	0000
Ending Academic Year	2002

If there are 23 records in the STVACYR table (from 1990 thru 2010 with 0000 and 9999 then 13 R190DEQV records would be created (which means 13 dap_eqv_crs_mst records) for catalog years: 1990, 1991, 1992, 1993, 1994, 1995, 1996, 1997, 1998, 1999, 2000, 2001, 2002):

Old Catalog Year	1990
Old Course Key	MATH111
New Catalog Year	@
New Course Key	MATH123

Old Catalog Year	1991
Old Course Key	MATH111
New Catalog Year	@
New Course Key	MATH123
Old Catalog Year	1992
Old Course Key	MATH111
New Catalog Year	@
New Course Key	MATH123
Old Catalog Year	1993
Old Course Key	MATH111
New Catalog Year	@
New Course Key	MATH123
... continue through an Ending Old Catalog Year of 2002.	

Whenever Degree Works sees MATH111 it would convert it to MATH123. In this case (Starting Academic Year of 0000) this course started at the beginning of time so a @ could be used for the Old Catalog Year which would drastically reduce the number of records required for this equivalency from 13 to 1.

Old Catalog Year	@
Old Course Key	MATH111
New Catalog Year	@
New Course Key	MATH123

If the SCREQIV Ending Academic Year matches the Highest Academic Year (calculated in Step #6) set the Old Catalog Year to an @. Normally the Highest Academic Year (end of time) would be something like 9999.

For example #2, an SCREQIV record contains the following:

New Course Key	ENGL222
New Academic Year	2004
Old Course Key	ENGL303
Starting Academic Year	2000
Ending Academic Year	9999

If there are 23 records in the STVACYR table (from 1990 thru 2010 with 0000 and 9999 then 11 R190DEQV records would be created from 2000 thru 9999 (which means 11 dap_eqv_crs_mst records): 2000, 2001, 2002, 2003, 2004, 2005, 2006, 2007, 2008, 2009, 2010.

Old Catalog Year	2000
------------------	------

Old Course Key	ENGL222
New Catalog Year	@
New Course Key	ENGL303
Old Catalog Year	2001
Old Course Key	ENGL222
New Catalog Year	@
New Course Key	ENGL303
Old Catalog Year	2002
Old Course Key	ENGL222
New Catalog Year	@
New Course Key	ENGL303
... continue through an Ending Old Catalog Year of 2010.	

Whenever Degree Works sees ENGL222 it will convert it to ENGL303. In this case (Ending Academic Year of 9999) it doesn't matter to Degree Works when ENGL222 started as this equivalency holds throughout the end of time. Thus, a @ would be used for the Old Catalog Year.

Old Catalog Year	@
Old Course Key	ENGL222
New Catalog Year	@
New Course Key	ENGL303

9. If the Old Catalog Year is equal to an @ then write the BIF record and return to Step #1 to process another SCREQIV record.
10. Write a BIF record for each academic year starting with the Start Academic Year through the Ending Academic Year. Load the Old Catalog Year with each Academic Year processed. For example, if a course has a Starting Academic Year of 2001 and an Ending Academic Year of 2005 five BIF records would be created with Old Catalog Years of 2001, 2002, 2003, 2004 and 2005.

Note: As mentioned above, if the UCX-CFG020 BANNER Term as Catalog Year flag is set to Y then the actual SCREQIV Term Codes will be used as Catalog Years. Thus, the Academic Year references below will actually contain Starting and Ending Term Codes.

For example #3, an SCREQIV record contains the following:

New Course Key	HIST222
New Academic Year	2004
Old Course Key	HIST246
Starting Academic Year	2001
Ending Academic Year	2005

If there are 23 records in the STVACYR table (from 1990 thru 2010 with 0000 and 9999 then 5 R190DEQV records would be created from 2001 thru 2005 (which means 5 dap_eqv_crs_mst records): 2001, 2002, 2003, 2004, 2005. Thus, if HIST222 is found in Catalog Years 2001 thru 2005 it will be converted to HIST246 for Degree Works purposes.

Old Catalog Year	2001
Old Course Key	HIST222
New Catalog Year	@
New Course Key	HIST246
Old Catalog Year	2002
Old Course Key	HIST222
New Catalog Year	@
New Course Key	HIST246
Old Catalog Year	2003
Old Course Key	HIST222
New Catalog Year	@
New Course Key	HIST246
Old Catalog Year	2004
Old Course Key	HIST222
New Catalog Year	@
New Course Key	HIST246
Old Catalog Year	2005
Old Course Key	HIST222
New Catalog Year	@
New Course Key	HIST246

Note: Remember that the Start and End Academic Years were calculated above in Step #1 from the STVTERM table. The SCREQIV_START_TERM was used to lookup the STVTERM table to get the STVTERM_ACYR_CODE and the SCREQIV_END_TERM was used to lookup the STVTERM table to get the STVTERM_ACYR_CODE.

11. Return to Step #1 and continue processing in this manner until all Banner equivalent records have been loaded into the BIF records and written to the dap_eqv_crs_mst.
12. Review the entries in UCX-CFG070 after the Banner equivalencies have been loaded. Make sure to remove any records that should not have been rolled to Degree Works. Make any other changes necessary. The contents of this table will be reloaded into the dap_eqv_crs_mst table when you run dapucx2eqv under ADMIN in Transit. RAD clients can update the dap_eqv_crs_mst table directly by using the RAD11 bridge.

Depending on the data in the SCREQIV Banner table it is possible that duplicate BIF (Bridge Interface Format) records could be created that map to the same dap_eqv_crs_key. Because the dap_eqv_crs_mst is a master record all dap_eqv_crs_key values must be unique. If duplicate EQVCRS BIF records are created they will NOT be added to the dap_eqv_crs_mst. Instead they will be written to the rad_log_dtl where they can be reviewed. If you see the following message in the extract log then duplicate records were found:

```
***→ DUPLICATE DAP_EQV_CRS_MST RECORDS ----- 8 ←***
```

Below is some simple sql used to display a few of the 8 duplicate dap_eqv_crs_mst records found for the rad_log_key = EQVCRS which is used for ban43 equivalency errors and the create date of 8/27/2009:

```
SQL> r
```

```
 1 select rad_error_msg1,rad_log_data from rad_log_dtl
 2* where rad_log_key = 'EQVCRS' and rad_create_date = '20090827
 ,
```

Error eqv crs add						
200420	ENGL	3001	✉	ENGL	301	
200420	COU	0070	✉	COU	0050	Error eqv crs add
199920	ECON	272	✉	ECON	1272	Error eqv crs add

The actual UNIQUE dap_eqv_crs_key for these records is the first 4 concatenated fields:

```
200420 ENGL 3001 ✉
```

This means that multiple OLD SCREQIV records generate this same concatenated key were found. Only one of these composite keys can be written to the dap_eqv_crs_mst and CFG070. If the wrong NEW course key (for example, ENGL 301 above) is created in CFG070 and the dap_eqv_crs_mst and a different NEW equivalent course key is desired then the SCREQIV records for both the OLD and NEW course keys should be reviewed for ENGL3001 and ENGL301. Two options exist if this situation occurs:

- Make changes to the Banner SCREQIV records so that when the Degree Works Banner extract is rerun it produces the desired results in CFG070 and the dap_eqv_crs_mst.
- If changing the SCREQIV data is not possible then CFG070 may be manually updated using Controller. After CFG070 is manually updated the script dapucx2eqv must be run to update the dap_eqv_crs_mst. Remember that every time the EQUIV Banner extract is run this same manual change must be made.

Note: You cannot add multiple entries with the same key to CFG070. These will cause errors when the dapucx2eqv is run to load UCX entries into the dap_eqv_crs_mst.

R602CRSE - COURSE Record

Updated: March 25, 2022

The Course Master table, SCBCRSE, is used for all of the data in this record.

Note: The course bridge adds new courses to, updates existing courses in, but does not delete courses from the rad_course_mst. To remove old entries from the rad_course_mst, you must use sql to delete them manually.

SQL used to read the SCBCRSE table:

```

SELECT a.SCBCRSE_SUBJ_CODE,
       a.SCBCRSE_CRSE_NUMB,
       a.SCBCRSE_EFF_TERM,
       a.SCBCRSE_DIVS_CODE,
       a.SCBCRSE_DEPT_CODE,
       a.SCBCRSE_TITLE,
       a.SCBCRSE_CREDIT_HR_IND,
       a.SCBCRSE_CREDIT_HR_LOW,
       a.SCBCRSE_CREDIT_HR_HIGH,
       a.SCBCRSE_REPEAT_LIMIT
  FROM SCBCRSE a, STVCSTA
 WHERE a.SCBCRSE_CSTA_CODE = STVCSTA_CODE AND
       STVCSTA_ACTIVE_IND = 'A'
 AND a.SCBCRSE_EFF_TERM =
       (SELECT MAX(b.SCBCRSE_EFF_TERM)
          FROM SCBCRSE b
         WHERE b.SCBCRSE_SUBJ_CODE = a.SCBCRSE_SUBJ_CODE
           AND b.SCBCRSE_CRSE_NUMB = a.SCBCRSE_CRSE_NUMB)
 ORDER BY SCBCRSE_SUBJ_CODE,
       SCBCRSE_CRSE_NUMB;

```

SQL used to read the SCRRTST table

```

SELECT SCRRTST_SUBJ_CODE,
       SCRRTST_CRSE_NUMB,           SCRRTST_TERM_CODE_EFF,           SCRRTST_SU
       BJ_CODE_PREQ,
       SCRRTST_CRSE_NUMB_PREQ,
       SCRRTST_LEVL_CODE  FROM SCRRTST a  WHERE a.SCRRTST_TERM_CODE_EFF
       =
       (SELECT
           MAX(b.SCRRTST_TERM_CODE_EFF)           FROM SCRRTST b
          WHERE b.SCRRTST_SUBJ_CODE = a.SCRRTST_SUBJ_CODE           AND b.
SCRRTST_CRSE_NUMB =
           a.SCRRTST_CRSE_NUMB) ORDER BY
           SCRRTST_SUBJ_CODE,
           SCRRTST_CRSE_NUMB,
           SCRRTST_TERM_CODE_EFF DESC;

```

These records are bridged into the RAD COURSE MST table in Degree Works.

Field Name	Banner Data
rad_course_key	Subj-code (12) + Crse-numb(12). For example, if the Subj-code is ENGL and the Crse-numb is 123 the course-key will be "ENGL 123".
rad_school	Not extracted. Loaded with BLANKS.

Field Name	Banner Data
	The SCRLEVL record(s) for the MAX Effective Term with a matching subject and course number to the SCBCRSE_SUBJ_CODE and SCBCRSE_CRSE_NUMB are written to the rad_crs_attr_dtl with the special rad_attr_code = DW-SCHOOL. The rad_attr_value is loaded with the actual School Code (SCRLEVL_LEVEL_CODE).
	These School attributes are used by the Course List in the Planner to filter courses by School.
rad_division	SCBCRSE_DIVS_CODE.
rad_dept	SCBCRSE_DEPT_CODE.
rad_course_title	SCBCRSE_TITLE.
rad_credits	Defaulted to SCBCRSE_CREDIT_HR_LOW. If the SCBCRSE_CREDIT_HR_IND = TO or OR then SCBCRSE_CREDIT_HR_HIGH is loaded into the credits.
rad_credit_ind	SCBCRSE_CREDIT_HR_IND
rad_credits_low	SCBCRSE_CREDIT_HR_LOW
rad_credits_high	SCBCRSE_CREDIT_HR_HIGH
rad_repeat_max	SCBCRSE_REPEAT_LIMIT is loaded as is unless it is NULL. A new Y/N flag, Force Null Repeatable has been added to UCX_CFG020 BANNER. If set to Y and the SCBCRSE_REPEAT_LIMIT is NULL then the Banner Course extract, ban41.ec, will load the RAD.Course_Mst.RAD_REPEAT_MAX with 99 (infinitely repeatable). If set to N or BLANK and the SCBCRSE_REPEAT_LIMIT is NULL then the RAD.Course_Mst.RAD_REPEAT_MAX will be loaded with 00 (NOT repeatable). Note: Other potential SCBCRSE_REPEAT_LIMIT values such as 1, 2, 3, and so on. are not currently being used by Transfer Equivalency Admin. Only the 0, 00, and 99 values are being used.
rad_acad_votech	Not extracted. Loaded with BLANKS.
rad_class_type	Not extracted. Loaded with BLANKS.
rad_user_def1	Loaded with PREREQ if SCRRTST record found for this course key.
rad_user_def2 - 10	Not extracted. Loaded with BLANKS.
rad_cat_yr_start	Currently, only used by the Planner to filter courses (Degree Works version 4.1.1). If UCX_CFG020 BANNER Term As Catalog Year = N, the SCBCRKY record is looked up using the SCBCRSE_SUBJ_CODE and SCBCRSE_CRSE_NUMB. The SCBCRKY_TERM_CODE_START is looked up on UCX_STU016 to get the associated Catalog Year to be loaded into the

Field Name	Banner Data
rad_cat_yr_start	<p>rad_cat_yr_start. If the rad_cat_yr_start is BLANK after this lookup 0000 will be loaded into the rad_cat_yr_start.</p> <p>If UCX_CFG020 BANNER Term As Catalog Year = Y, the SCBCRKY record is looked up using the SCBCRSE_SUBJ_CODE and SCBCRSE_CRSE_NUMB. The SCBCRKY_TERM_CODE_START is loaded directly into the rad_cat_yr_start. If the rad_cat_yr_start is BLANK after this process 000000 will be loaded into the rad_cat_yr_start.</p>
rad_cat_yr_stop	<p>Currently, only used by the Planner to filter courses (Degree Works version 4.1.1).</p> <p>If UCX_CFG020 BANNER Term As Catalog Year = N, the SCBCRKY record is looked up using the SCBCRSE_SUBJ_CODE and SCBCRSE_CRSE_NUMB. The SCBCRKY_TERM_CODE_END is looked up on UCX_STU016 to get the associated Catalog Year to be loaded into the rad_cat_yr_stop. If the rad_cat_yr_stop is BLANK after this lookup 999999 will be loaded into the rad_cat_yr_stop.</p> <p>If UCX_CFG020 BANNER Term As Catalog Year = Y, the SCBCRKY record is looked up using the SCBCRSE_SUBJ_CODE and SCBCRSE_CRSE_NUMB. The SCBCRKY_TERM_CODE_END is loaded directly into the rad_cat_yr_stop. If the rad_cat_yr_stop is BLANK after this process 999999 will be loaded into the rad_cat_yr_stop.</p>

R605CRSA - Course Attribute Record

Updated: March 25, 2022

The Course Attribute table SCRATTR and the School Attribute table SCRLEVL are used for the data in this record.

SCRATTR – Course Attributes – processed for courses found in the SCBCRSE table. For each SCBCRSE course master record the SUBJ_CODE, CRSE_NUMB and EFF_TERM are used to read the SCRATTR table. If a match is found then the attribute code (SCRATTR_ATTR_CODE) on that record will be used to create a course attribute BIF record. Multiple BIF records will be created if multiple attributes exist for a given course. A BIF course attribute record will be written for each course attribute found in the SCRATTR table.

The attribute data will be loaded into the rad_crs_attr_dtl with a special hardwired attribute code of ATTRIBUTE.

SQL used to read the SCRATTR table

```
SELECT SCRATTR_SUBJ_CODE,
       SCRATTR_CRSE_NUMB,
       SCRATTR_EFF_TERM,
       SCRATTR_ATTR_CODE
```

```

FROM SCRATTR a
WHERE a.SCRATTR_EFFECTIVE_TERM =
  (SELECT MAX(b.SCRATTR_EFFECTIVE_TERM)
   FROM SCRATTR b
   WHERE b.SCRATTR_SUBJECT_CODE = a.SCRATTR_SUBJECT_CODE
     AND b.SCRATTR_COURSE_NUMBER = a.SCRATTR_COURSE_NUMBER)
ORDER BY SCRATTR_SUBJECT_CODE,
         SCRATTR_COURSE_NUMBER,
         SCRATTR_EFFECTIVE_TERM DESC;

```

Field Name	Banner Data
rad_course_key	Subj-code (12) + Crse-numb(12). For example, if the Subj-code is ENGL and the Crse-numb is 123 the course-key will be ENGL 123.
rad_attr_code	Hardwired with ATTRIBUTE.
rad_attr_value	The particular ATTR_CODE from the SCRATTR table. This element gives a name to the value; used in a Scribe rule as (WITH Attribute = attr_value). For example, the attr_value might be WRITING which might result in the following statement: 5 Credits in ENGL @ (WITH Attribute = WRITING)

SCRLEVL - Course School Attributes - records have been added to the rad_crs_attr_dtl in Release 4.1.1.

The SCRLEVL record(s) for the MAX Effective Term with a matching subject and course number to the SCBCRSE_SUBJECT_CODE and SCBCRSE_COURSE_NUMBER are written to the rad_crs_attr_dtl with the special rad_attr_code = DW-SCHOOL. The rad_attr_value is loaded with the actual School Code (SCRLEVL_LEVEL_CODE).

These School attributes are used by the Course List in the Planner to filter courses by School.

The School Course attribute data will be loaded into the rad_crs_attr_dtl with a special hardwired attribute code of DW-SCHOOL.

SQL used to read the SCRLEVL table

```

SELECT SCRLEVL_SUBJECT_CODE,
       SCRLEVL_COURSE_NUMBER,
       SCRLEVL_EFFECTIVE_TERM,
       SCRLEVL_LEVEL_CODE
  FROM SCRLEVL a
 WHERE a.SCRLEVL_EFFECTIVE_TERM =
  (SELECT MAX(b.SCRLEVL_EFFECTIVE_TERM)
   FROM SCRLEVL b
   WHERE b.SCRLEVL_SUBJECT_CODE = a.SCRLEVL_SUBJECT_CODE
     AND b.SCRLEVL_COURSE_NUMBER = a.SCRLEVL_COURSE_NUMBER)
ORDER BY SCRLEVL_SUBJECT_CODE,
         SCRLEVL_COURSE_NUMBER,
         SCRLEVL_EFFECTIVE_TERM DESC;

```

Field Name	Banner Data
rad_course_key	Subj-code (12) + Crse-numb(12). For example, if the Subj-code is ENGL and the Crse-numb is 123 the course-key will be ENGL 123.
rad_attr_code	Hardwired with DW-SCHOOL.
rad_attr_value	The particular LEVL_CODE from the SCRLEVL table.
	These School attributes are used by the Course List in the Planner to filter courses by School.

R652MAPD - DAP Mapping Record

Updated: March 24, 2023

The dap_mapping_dtl, required for Transfer Equivalency and Transfer Equivalency Self-Service, is generated from Banner tables SHBTATC, SHRTATC and SHRICMTC.

Note that an associated dap_map_cond_dtl will be created if the SHBTATC_PROGRAM column is populated.

Added a WARN message to the mapping extract to identify mappings that have mixed AND and OR connectors without the use of parentheses. This condition would cause the mappings to be created incorrectly in Degree Works.

```
***** WARN: Mixed ANDs/ORs with NO Parentheses for Transfer Course
[MUSC100 ]
and School ID [1001 ] ; SHBTATC/SHRTACT Mapping records SKIPPED! *****
```

Example: (MUSC101 AND MUSC110) OR MUSC5A AND MUSC5B AND MUSC5C **should be:**
(MUSC101 AND MUSC110) OR (MUSC5A AND MUSC5B AND MUSC5C)

Mappings which receive this WARN message will be SKIPPED and not processed. Parenthesis should be added to the Banner mappings in the appropriate places and then the Banner mapping extract rerun. Then the mapping extract should generate the appropriate mapping records for these more complex scenarios in Degree Works.

Added a WARN message to the mapping extract to identify mappings that have different Primary Connectors and mixed AND/OR connectors with parentheses. This condition causes the Mapping Extract to quit prematurely.

```
***** WARN: Two different Primary Connectors for Transfer Course  
[COMM1000 ]  
and School ID [1011 ]; SHBTATC/SHRTATC Mapping record SKIPPED! *****
```

Example: COMM0911 **OR** (COMM0912 OR COMM0913) **AND** COMM0914 **should be:**
COMM0911 **AND** (COMM0912 OR COMM0913) **AND** COMM0914

Mappings which receive this WARN message will be SKIPPED and not processed. The Banner mappings should be fixed to use the same Primary Connectors in the appropriate places and then the Banner mapping extract rerun. Then the mapping extract should generate the appropriate mapping records for this scenario in Degree Works.

Added a WARN message to the mapping extract to identify duplicate mappings. The duplicate mapping will be skipped and not written to the dap_mapping_dtl.

```
***** WARN: DUPLICATE Mapping N0000321 R650MAPD      A N00003211001      ENGL  
1111      Introduction to Literature    201240      9999      003.000003.000  
ELIT      411          Studies in Literature     0000      9999  
003.000  
1TO1N]; SHBTATC Mapping record SKIPPED! *****
```

Mappings which receive this WARN message will be SKIPPED and not processed. The Banner mappings that cause duplicate warnings should be fixed, but the duplicate mappings were not written to Degree Works so the mapping extract does NOT have to be rerun.

Example mapping scenarios that include both AND/OR connectors and parentheses are included after the mapping definitions below. The primary connector between the set of parentheses is defined. If a primary OR mapping is found, the single OR mappings are listed followed by a list of courses included with AND connectors. If a primary AND mapping is found, the single AND mappings are listed followed by a list of courses included with OR connectors. Finally some of the columns from the dap_mapping_dtl are listed showing how the mapping scenario was processed.

SQL used to read the SHBTATC table

```
SELECT a.SHBTATC_SBGI_CODE,  
       a.SHBTATC_PROGRAM,  
       a.SHBTATC_TLVL_CODE,  
       a.SHBTATC_SUBJ_CODE_TRNS,  
       a.SHBTATC_CRSE_NUMB_TRNS,  
       a.SHBTATC_TERM_CODE_EFF_TRNS,  
       a.SHBTATC_TRNS_TITLE,  
       a.SHBTATC_TRNS_LOW_HRS,  
       a.SHBTATC_TRNS_HIGH_HRS,  
       a.SHBTATC_TRNS_REVIEW_IND,  
       a.SHBTATC_TAST_CODE,  
       a.SHBTATC_TRNS_CATALOG,
```

```

        a.SHTATC_TGRD_CODE_MIN,
        a.SHTATC_GROUP,
        a.SHTATC_GROUP_PRIMARY_IND"
    FROM SHTATC a
    WHERE a.SHTATC_TRNS REVIEW_IND = 'Y' AND
        a.SHTATC_TAST_CODE IN
            (SELECT STVTAST_CODE FROM STVTAST WHERE STVTAST_STATUS_IND
            = 'A')

```

SQL used to read the SHRTATC table

```

SELECT a.SHRTATC_SBGI_CODE,
       a.SHRTATC_PROGRAM,
       a.SHRTATC_TLVL_CODE,
       a.SHRTATC_SUBJ_CODE_TRNS,
       a.SHRTATC_CRSE_NUMB_TRNS,
       a.SHRTATC_TERM_CODE_EFF_TRNS,
       a.SHRTATC_SEQNO,
       a.SHRTATC_CONNECTOR,
       a.SHRTATC_SUBJ_CODE_INST,
       a.SHRTATC_CRSE_NUMB_INST,
       a.SHRTATC_INST_TITLE,
       a.SHRTATC_INST_CREDITS_USED,
       a.SHRTATC_GROUP
    FROM SHRTATC a
   WHERE a.SHRTATC_SBGI_CODE= SHTATC_SBGI_CODE AND
         a.SHRTATC_PROGRAM = SHTATC_PROGRAM AND
         a.SHRTATC_TLVL_CODE = SHTATC_TLEVL_CODE AND
         a.SHRTATC_SUBJ_CODE_TRNS = SHTATC_SUBJ_CODE_TRNS AND
         a.SHRTATC_CRSE_NUMB_TRNS = SHTATC_CRSE_NUMB_TRNS AND
         a.SHRTATC_TERM_CODE_EFF_TRNS = SHTATC_TERM_CODE_EFF_TRNS

```

SQL used to read the SHICMT table

```

SELECT a.SRICMTC_SBGI_CODE,
       a.SRICMTC_PROGRAM,
       a.SRICMTC_TLVL_CODE,
       a.SRICMTC_SUBJ_CODE_TRNS,
       a.SRICMTC_CRSE_NUMB_TRNS,
       a.SRICMTC_TERM_CODE_EFF,
       a.SRICMTC_SUBJ_CODE_INST,
       a.SRICMTC_CRSE_NUMB_INST,
       a.SRICMTC_SEQNO,
       a.SRICMTC_SHRTATC_SEQNO,
       a.SRICMTC_TEXT,
       a.SRICMTC_GROUP
    FROM SRICMTC a
   WHERE a.SRICMTC_SEQNO IN (1,2) AND
         a.SRICMTC_SHRTATC_SEQNO = SHRTATC_SEQ_NO AND
         a.SRICMTC_SBGI_CODE= SHRTATC_SBGI_CODE AND
         a.SRICMTC_PROGRAM = SHRTATC_PROGRAM AND
         a.SRICMTC_TLVL_CODE = SHRTATC_TLEVL_CODE AND
         a.SRICMTC_SUBJ_CODE_TRNS = SHRTATC_SUBJ_CODE_TRNS AND

```

a.SHRICMT_CRSE_NUMB_TRNS = SHRTATC_CRSE_NUMB_TRNS AND
 a.SHRICMT_TERM_CODE_EFF = SHRTATC_TERM_CODE_EFF_TRNS

These records are bridged into the DAP_MAPPING_DTL table in Degree Works.

Field Name	Banner Data
dap_map_id	Nnnnnnn where nnnnnnn is a number representing the mapping set, for example N0000001. Each dap_map_cond_dtl associated with the mapping has the same dap_map_id as the dap_mapping_dtl.
dap_school_id	SHBTATC_SBGI_CODE
dap_tr_disc	SHBTATC_SUBJ_CODE_TRNS
dap_tr_crse_num	SHBTATC_CRSE_NUMB_TRNS
dap_tr_title	SHBTATC_TRNS_TITLE
dap_tr_catyr_beg	The STVACYR value associated with SHBTATC_TERM_CODE_EFF_TRNS
dap_tr_catyr_end	The prior STVACYR value associated with SHBTATC_TERM_CODE_EFF_TRNS of a future entry in SHBTATC with the same key (SHBTATC_SBGI_CODE, SHBTATC_SUBJ_CODE_TNRS and SHBTATC_CRSE_NUMB_TRNS). If no future entry exists, set to 9999.
dap_tr_cr_min	SHBTATC_TRNS_LOW_HOURS converted to a format of 9999.999.
dap_tr_cr_max	SHBTATC_TRNS_HIGH_HOURS converted to a format of 9999.999.
dap_tr_gr_min	SHBTATC_TGRD_CODE_MIN
dap_tr_gr_max	Not extracted. Loaded with BLANKS.
dap_tr_yrs_ago	Not extracted. Loaded with BLANKS.
dap_discipline	SHRTATC_SUBJ_CODE_INST
dap_course_num	SHRTATC_CRSE_NUMB_INST
dap_section	Not extracted. Loaded with BLANKS.
dap_course_title	SHRTATC_INST_TITLE
dap_cat_yr_start	The earliest Catalog Year this course was valid at your institution. Use value from STVACYR.
dap_cat_yr_stop	The Catalog Year this course is no longer valid at your institution. Use value from STVACYR, and 9999 if course is currently valid.
dap_cr_earn	SHRTATC_INST_CREDITS_USED converted to a format of 9999.999.
dap_comment1	SHRICMT_TEXT where SHRICMT_SEQNO = 1
dap_comment2	SHRICMT_TEXT where SHRICMT_SEQNO = 2
dap_authorizer	Not extracted. Loaded with BLANKS.
dap_auth_date	Not extracted. Loaded with BLANKS.

Field Name	Banner Data
dap_articulation	<p>The articulation code (1/M transfer courses from SHBTATC on the left, TO, 1/M local courses from SHRTATC on the right):</p> <p>1TO1 = One-to-One: One transfer SHBTATC course to one local SHRTATC course.</p> <p>1TOM = One-to-Many: One transfer SHBTATC course matches the key in two or more local SHRTATC course records.</p> <p>MTO1 = Many-to-One: Two or more transfer SHBTATC records exist with the same Group code and map to one local SHRTATC course record.</p> <p>MTOM = Many-to-Many: Two or more transfer SHBTATC course records are found with the same Group code and map to two or more local SHRTATC course records.</p>
dap_cond_flag	<p>Y/N flag indicating that mapping conditions exist. If SHBTATC_PROGRAM is populated, a dap_map_cond_dtl will be created and this flag will be set to Y. The associated dap_map_cond_dtl must contain the same dap_map_id.</p>

Examples containing both AND/OR connectors and a pair of parentheses:

(1) ENGL1000 => (ENGL100 AND ENGL101) OR ENGL102

```
DisplayOrMappings; **** OR ****
DisplayOrMappings; ***** Single Courses ****
[0] CourseIndex=[2]; Inst CourseKey=[ENGL102 ]
DisplayOrMappings; **** And Courses List ****
[0] [0] CourseIndex=[0]; Inst CourseKey=[ENGL100 ]
[0] [1] CourseIndex=[1]; Inst CourseKey=[ENGL101 ]
DisplayOrMappings; **** END ****
```

MapId	school	seq	subjtr	numbtr	subjin	numbin	catyr	S	catyr	E	Artic
MA006355	1591	1	ENGL	1000	ENGL	100	1994	9999			1TOM
MA006355	1591	2			ENGL	101					1TOM
MA006356	1591	1	ENGL	1000	ENGL	102	1994	9999			1TO1

(2) ENGL1002 => ENGL103 AND (ENGL101 OR ENGL102)

```
DisplayAndMappings; ***** AND *****
DisplayAndMappings; ***** Single Courses *****
[0] CourseIndex=[0]; Inst CourseKey=[ENGL103 ]
DisplayAndMappings; ***** Or Courses List *****
[0] [0] CourseIndex=[1]; Inst CourseKey=[ENGL101 ]
[0] [1] CourseIndex=[2]; Inst CourseKey=[ENGL102 ]
DisplayAndMappings; ***** END *****
```

MapId	school	seq	subjtr	numbtr	subjin	numbin	catyr	<u>S</u>	<u>catyr</u>	E	Artic
MA006357	1591	1	ENGL	1002	ENGL	101	1994	9999			1TOM
MA006357	1591	2			ENGL	103					1TOM
MA006358	1591	1	ENGL	1002	ENGL	102	1994	9999			1TOM
MA006358	1591	2			ENGL	103					1TOM

(3) BIOL1001 => BIO100 AND (BIO120 OR BIO121) AND BIO110

```
DisplayAndMappings; ***** AND *****
DisplayAndMappings; ***** Single Courses *****
[0] CourseIndex=[0]; Inst CourseKey=[BIO 100 ]
[0] CourseIndex=[3]; Inst CourseKey=[BIO 110 ]
DisplayAndMappings; ***** Or Courses List *****
[0] [0] CourseIndex=[1]; Inst CourseKey=[BIO 120 ]
[0] [1] CourseIndex=[2]; Inst CourseKey=[BIO 121 ]
DisplayAndMappings; ***** END *****
```

MapId	school	seq	subjtr	numbtr	subjin	numbin	catyr	<u>S</u>	<u>catyr</u>	E	Artic
MA006359	2125	1	BIOL	1001	BIO	120	1994	9999			1TOM
MA006359	2125	2			BIO	100					1TOM
MA006359	2125	3			BIO	110					1TOM
MA006360	2125	1	BIOL	1001	BIO	121	1994	9999			1TOM
MA006360	2125	2			BIO	100					1TOM
MA006360	2125	3			BIO	110					1TOM

(4) MUSC100 AND MUSIC 118 => (MUSC101 AND MUSC110) OR (MUSC5A AND MUSC5B AND MUSC5C)

```
DisplayOrMappings; ***** OR *****
DisplayOrMappings; ***** Single Courses *****
DisplayOrMappings; **** And Courses List *****
[0] [0] CourseIndex=[0]; Inst CourseKey=[MUSC101 ]
[0] [1] CourseIndex=[1]; Inst CourseKey=[MUSC110 ]
[1] [0] CourseIndex=[2]; Inst CourseKey=[MUSC5A ]
[1] [1] CourseIndex=[3]; Inst CourseKey=[MUSC5B ]
[1] [2] CourseIndex=[4]; Inst CourseKey=[MUSC5C ]
DisplayOrMappings; ***** END *****
```

MapId	school	seq	subjtr	numbtr	subjin	numbin	catyr	S	catyr	E	Artic
MA006351	1001	1	MUSC	100	MUSC	101	2004		9999		MTOM
MA006351	1001	2	MUSC	118	MUSC	110	2004		9999		MTOM
MA006352	1001	1	MUSC	100	MUSC	5A	2004		9999		MTOM
MA006352	1001	2	MUSC	118	MUSC	5B	2004		9999		MTOM
MA006352	1001	3			MUSC	5C					MTOM

R655MAPA - DAP Map Attributes Record

Updated: March 25, 2022

A dap_map_attr_dtl record is created for each SHRTRAT found linked to the mapping.

Degree Works only needs to know the institutional course. It does not need to record the transfer course. After an articulation is complete the local course will be looked up along with the mapping-id that was used for the articulation to find the transfer attributes.

Field Name	Banner Data
dap_map_id	Use the same dap_map_id as the associated dap_mapping_dtl.
dap_discipline	The institutional course discipline: SHRTRAT_SUBJ_CODE_INST
dap_course_num	The institutional course number: SHRTRAT_CRSE_NUMB_INST
dap_course_title	The institutional course title: SHRTATC_INST_TITLE
dap_attr_key	For Banner schools this is always ATTRIBUTE.
dap_attr_code	The SHRTRAT_ATTR_CODE

R658MAPC - DAP Map Conditions Record

Updated: March 24, 2023

This record contains information for the dap_map_cond_dtl and is an optional table.

A dap_map_cond_dtl record is created for PROGRAM when the SHBTATC_PROGRAM column of the SHBTATC record is populated (other than with the default value) and when UCX-CFG020 BANNER ProgramAsDegree=Y.

A dap_map_cond_dtl record is created for SCHOOL when the SHBTATC_TLVL_CODE column of the SHBTATC record is populated.

When ProgramAsDegree=N the PROGRAM condition is not bridged because in Transfer Equivalency Self-Service, the program the student enters is not available to checked when the articulation occurs.

When ProgramAsDegree=Y the PROGRAM condition is bridged but the UCX-TRQ061 PROGRAM entry should point to the DEGREE value on the dap_applicant_mst.

These records are bridged into the DAP_MAPCOND_DTL table in Degree Works.

Field Name	Banner Data
dap_map_id	Use the same dap_map_id as the associated dap_mapping_dtl.
dap_school_id	SHBTATC_SBGI_CODE, or the same dap_school_id as the associated dap_mapping_dtl.
dap_cond_name	PROGRAM
dap_cond_op	Use the equals sign =
dap_cond_value	SHBTATC_PROGRAM from the SHBTATC record which populated the associated dap_mapping_dtl.

Field Name	Banner Data
dap_map_id	Use the same dap_map_id as the associated dap_mapping_dtl.
dap_school_id	SHBTATC_SBGI_CODE, or the same dap_school_id as the associated dap_mapping_dtl.
dap_cond_name	SCHOOL
dap_cond_op	Use the equals sign =
dap_cond_value	SHBTATC_TLVL_CODE from the SHBTATC record which populated the associated dap_mapping_dtl.

R702ETSM - ETS Record

Updated: March 24, 2023

The ETS Master data is obtained from three tables: STVSBGI - contains the ETS Code, School Name, and Type and is the main driver to the other two tables linked by ETS Code. SOBSBGI - contains the city and state for each school. SORBTAG - contains the Calendar code.

SQL used to read the STVSBGI table

```
SELECT STVSBGI_CODE,
       STVSBGI_TYPE_IND,
       STVSBGI_DESC
  FROM STVSBGI
 ORDER BY STVSBGI_CODE;
```

SAMPLE Data

STVSBGI_CODE	STVSBGI_SBGI_CODE
CCCG1000	University of Memphis
CCCG1001	University of Maryland
CCCG1002	University of Florida
CCCG1003	Universite de Paris
CCCG1004	Universite de Montreal
CCCG1005	University of Los Angeles
CCCG1006	University of Fresno
CCCG1007	University of Fulton
CCCG1008	University of Chicago
CCCG1009	University of Boston
CCCG1010	University of Vermont

These records are bridged into the RAD_ETS_MST table in Degree Works.

Field Name	Banner Data
HEADER	<p>This is a 28-byte field identifying the record as one containing rad_ets_mst data. It is composed of an ETS plus 7 bytes of spaces, an 8-byte IDENTIFIER (R702ETSM for the rad_ets_mst), an 8-byte rad_term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete, M=Modify).</p> <p>Example: 4854.....R702ETSM.....A. (periods represent spaces). .</p>
rad_ets_code	<p>STVSBGI_CODE</p> <p>This element is the school code your system uses. If your institution uses ETS, it is the ETS code assigned by the Education Testing Service. If your school does not ETS then it whatever you have stored in STVSBGI – we simply refer to it as ETS in Degree Works.</p>
rad_ets_school	<p>STVSBGI_SBGI_CODE</p> <p>This element is the name of the school.</p> <p>Example: University of Washington</p>
rad_ets_city	SOBSBGI_CITY
rad_ets_state	SOBSBGI_STATE
rad_ets_type	SORBTAG_HLWK_CODE
	<p>This element defines the highest degree offered or college level indicating a 4-year, 2-year or other school. Examples: 4YR, 2YR.</p>
rad_calendar	<p>SORBTAG_CALD_CODE</p> <p>This element defines the calendar structure for the institution's academic year. Examples: S = Semester, Q = Quarter.</p>

R800UCXT - UCX Record

Updated: September 29, 2023

The UCX tables included in this document are those included for STV data tables that will be bulk loaded from the Banner student system into Degree Works UCX tables.

Only UCX tables that are being sent in bulk from the Banner database are listed here. For a complete list of UCX tables used by Degree Works, see [Bridge Interface Format](#). Ellucian will supply its standard UCX tables and train your site in Controller for maintenance of the codes. Use the Controller application for maintaining the UCX records after they have been bridged from Banner to Degree Works.

The UCX tables may be rolled from Banner in bulk (all tables in one run of the RAD36 UCX job in Transit) or one-by-one using a bannerextract ucx file containing a list of Degree Works UCX tables to be reloaded from Banner. For details on how to run the ucx bannerextract for all and selected Degree Works UCX tables, see the [Extract scheduling](#) topic.

Warning! Due to the existence of many Degree Works only flags/values that must be manually loaded into several of the UCX tables listed below using Controller, it is recommended that the UCX_CFG020 RADBRIDGE Add UCX Entries Only flag be set to Y after the UCX extract has been run when in bulk. This flag will allow only new values that have been created in Banner to be extracted and loaded into one of the UCX tables listed below. Existing UCX records will remain untouched and will not be reloaded from Banner.

UCX	Key	Description	Banner Table
UCX-AUD027	12	Major What-If Picklist	STVMAJR
UCX-AUD029	12	Minor What-If Picklist	STVMAJR
UCX-STU016	8	Term Codes	STVTERM
UCX-STU023	12	Major Codes	STVMAJR
UCX-STU024	12	Minor Codes	STVMAJR
UCX-STU035	4	Catalog Years	STVTERM
UCX-STU050	4	Attributes	STVATTR
UCX-STU305	6	Student Level Codes	STVCLAS
UCX-STU306	2	Student Status Codes	STVSTYP
UCX-STU307	12	Degree Code	STVDEGC
UCX-STU316	12	Program Codes	SMRPRLE
UCX-STU346	2	Calendar Codes	STVACCL
UCX-STU350	2	School Codes	STVLEVL
UCX-STU352	12	Discipline Code	STVSUBJ

UCX	Key	Description	Banner Table
UCX-STU356	18	Grade Type Code	STVGMOD, SHRGRDE, SHRGRDO
UCX-STU385	24	Grade Information Table	STVGMOD, SHRGRDE, SHRGRDO
UCX-STU560	2	College Codes	STVCOLL
UCX-STU563	12	Concentration Codes	STVMAJR

UCX-AUD027 - STVMAJR: What-If Picklist Major Codes

Field Name	Banner Data
HEADER	UCX-AUD027.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-AUD027
UCX CODE	STVMAJR_CODE (4) + 26 blanks if the STVMAJR_VALID_MAJOR_IND = Y. Leading spaces are removed.
UCX VALUE	
Description	STVMAJR_DESC (30)

UCX-AUD029 - STVMAJR: What-If Picklist Minor Codes

Field Name	Banner Data
HEADER	UCX-AUD029.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-AUD029
UCX CODE	STVMAJR_CODE (4) + 26 blanks if the STVMAJR_VALID_MINOR_IND = Y. Leading spaces are removed.
UCX VALUE	
Description	STVMAJR_DESC (30)

UCX-STU016 - STVTERM: Term Codes

Field Name	Banner Data
HEADER	UCX-STU016.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU016
UCX CODE	STVTERM_CODE (6) + 24 blanks.
UCX VALUE	
Description	STVTERM_DESC (truncated from 30-bytes to 12-bytes). Some of these term descriptions may need to be fixed using Controller after the data has been rolled from Banner.
Long Description	STVTERM_DESC – the full 30-byte description
Filler	The full STVTERM_DESC (30) is being loaded into this FILLER field for reference purposes.

Field Name	Banner Data
Catalog Year	STVTERM_ACYR_CODE (4)
Web Planner	Set to Y if term starts in the future; set to N if term started in the past.
Term Type	Filled with FALL, WINTER, SPRING or SUMMER if description contains any of those values.
Show In Web TreQer	Set to Q if term starts in the future; set to Y if term started in the past.
Financial Aid Year	STVTERM_FA_PROC_YR
Show in SEP	Set to Y if term starts in the future; set to N if term started in the past.

UCX-STU023 - STVMAJR: Major Codes

Field Name	Banner Data
HEADER	UCX-STU023.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU023
UCX CODE	STVMAJR_CODE (4) + 26 blanks if the STVMAJR_VALID_MAJOR_IND = Y. Leading spaces are removed.
UCX VALUE	
Description	STVMAJR_DESC (30)

UCX-STU024 - STVMAJR: Minor Codes

Field Name	Banner Data
HEADER	UCX-STU024.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU024
UCX CODE	STVMAJR_CODE (4) + 26 blanks if the STVMAJR_VALID_MINOR_IND = Y. Leading spaces are removed.
UCX VALUE	
Description	STVMAJR_DESC (30)

UCX-STU035 - STVTERM: Catalog Years

Field Name	Banner Data
HEADER	UCX-STU035.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU035
UCX CODE	STVTERM_ACYR (4) + 26 blanks
UCX VALUE	
Description	STVTERM_DESC (30)

UCX-STU050 - STVATTR: Attribute Codes

Field Name	Banner Data
HEADER	UCX-STU050.....R800UCXT.....A. (periods represent spaces).

Field Name	Banner Data
UCX TABLE	UCX-STU050
UCX CODE	STVATTR_CODE (4) + 26 blanks.
UCX VALUE	
Description	STVATTR_DESC (30)

UCX-STU305 - STVCLAS: Student Level Codes

Field Name	Banner Data
HEADER	UCX-STU305...R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU305
UCX CODE	STVCLAS_CODE (2) + 28 blanks.
UCX VALUE	
Description	STVCLAS_DESC (30)
Short Description	STVCLAS_DESC (truncated to 6-bytes: fix after loaded from Banner)

UCX-STU306 - STVSTYP: Student Status Codes

Field Name	Banner Data
HEADER	UCX-STU306...R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU306
UCX CODE	STVSTYP_CODE (1) + 29 blanks.
UCX VALUE	
Description	STVSTYP_DESC (30)

UCX-STU307 - STVDEGC: Degree Codes

Field Name	Banner Data
HEADER	UCX-STU307....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU307
UCX CODE	STVDEGC_CODE (4) + 26 blanks.
UCX VALUE	
Description	STVDEGC_DESC (30)
Short Description	STVDEGC_CODE

UCX-STU316 - SMRPRLE: Program Codes

Field Name	Banner Data
HEADER	UCX-STU316....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU316
UCX CODE	SMRPRLE_PROGRAM (12) + 18 blanks.

Field Name	Banner Data
UCX VALUE	
Description	SMRPRLE_PROGRAM_DESC (30)

UCX-STU346 - STVACCL: Calendar Codes

Field Name	Banner Data
HEADER	UCX-STU346.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU346
UCX CODE	STVACCL_CODE (2) + 28 blanks.
UCX VALUE	
Description	STVACCL_DESC (30)

UCX-STU350 - STVLEVL: School Codes

Field Name	Banner Data
HEADER	UCX-STU350.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU350
UCX CODE	STVLEVL_CODE (2) + 28 blanks.
UCX VALUE	
Description	STVLEVL_DESC (30)

UCX-STU352 - STVSUBJ: Discipline Codes

Field Name	Banner Data
HEADER	UCX-STU352.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU352
UCX CODE	STVSUBJ_CODE (4) + 26 blanks.

Additional Rules:

1. If the STVSUBJ_CODE contains IMBEDDED SPACES they will be replaced with underscores automatically by the extract programs. For example, AU B would become AU_B, A C would become A__C, A BC would become A_BC while ART would remain ART as the space is trailing.
2. If new Discipline codes are being input manually into UCXDW352 using Controller or some other tool then the same rule must be followed: replace imbedded spaces with underscores. Otherwise Degree Works will not process the discipline codes properly.

UCX VALUE	
Description	STVSUBJ_DESC (30)

Field Name	Banner Data
New Discipline	The dap69 conversion program translates the old value to this value if the course is not found in UCX-CFG069.
Discipline Status	A or Blank=Active, I=Inactive. The BannerBridge will SKIP classes with an Inactive discipline and will NOT load them into the rad_class_dtl (current and history classes) or the rad_transfer_dtl.

UCX-STU356 - STVGMOD: Grade Types

The key to UCX-STU356 is a composite School Key (LEVL_Code) of 12-bytes plus the Grade Type code (GMOD_Code) of 2-bytes. The SHRGRDO table containing the valid grade combinations for each School (Level) is used as the driver for the loading of the Grade Type data.

However, both the SHRGRDO and SHRGRDE tables are used in the standard bannerextract SQL:

```

SELECT b.SHRGRDE_LEVL_CODE,
a.SHRGRDO_GMOD_CODE,
b.SHRGRDE_CODE,
b.SHRGRDE_ABBREV,
b.SHRGRDE_TERM_CODE_EFFECTIVE,
b.SHRGRDE_QUALITY_POINTS,
b.SHRGRDE_ATTEMPTED_IND,
b.SHRGRDE_COMPLETED_IND,
b.SHRGRDE_PASSED_IND,
b.SHRGRDE_GPA_IND,
b.SHRGRDE_GRDE_STATUS_IND,
b.SHRGRDE_NUMERIC_VALUE,
b.SHRGRDE_REPEAT_INCLUDE_IND)
FROM SHRGRDO a, SHRGRDE b
WHERE a.SHRGRDO_GRDE_CODE = b.SHRGRDE_CODE
AND a.SHRGRDO_LEVL_CODE = b.SHRGRDE_LEVL_CODE
AND a.SHRGRDO_TERM_CODE_EFFECTIVE = b.SHRGRDE_TERM_CODE_EFFECTIVE
AND b.SHRGRDE_TERM_CODE_EFFECTIVE =
    (SELECT MAX(SHRGRDE.SHRGRDE_TERM_CODE_EFFECTIVE)
     FROM SHRGRDE
     WHERE SHRGRDE.SHRGRDE_CODE = b.SHRGRDE_CODE
       AND SHRGRDE.SHRGRDE_LEVL_CODE = b.SHRGRDE_LEVL_CODE
       AND SHRGRDE.SHRGRDE_GRDE_STATUS_IND = 'A')
AND b.SHRGRDE_GRDE_STATUS_IND = 'A'
ORDER BY b.SHRGRDE_LEVL_CODE,
a.SHRGRDO_GMOD_CODE,
b.SHRGRDE_CODE,
b.SHRGRDE_TERM_CODE_EFFECTIVE DESC

```

Invalid combinations should be deleted from UCX_STU356 using Controller.

Field Name	Banner Data
HEADER	UCX-STU356.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU356
UCX CODE	STVLEVEL_CODE (12) + STVGMOD_CODE (2) + 16 blanks.

Field Name	Banner Data
School Code (12) plus the Grade Type Code (2) = 14-byte key	
UCX VALUE	
Range Code	Hardwired to F for A-F grade.
Description	STVGMOD_DESC
Audit Flag	Hardwired to N. For Audit Grade Type codes this flag should be manually set to Y using Controller after UCX-STU356 is loaded from Banner.
Repeat Policy	Hardwired to 1. Depending on your institution's Repeat Policy this value may need to be changed on the appropriate records using Controller after UCX-STU356 is loaded from Banner. Valid values are 1-6: 1 - Keep most recent repeat; 2 - Keep repeat with highest grade; 3 - Keep all repeats; 4 - Keep most recent credits, all count in the GPA; 5 - Keep credits of highest grade, all count in the GPA; 6 - Keep this repeat
DAP Process Flag	Hardwired to Y indicating all Grade Types will be processed by Degree Works. If any School/Grade Types are NOT to be processed by Degree Works set this flag to N using Controller after UCX-STU356 is loaded from Banner.

UCX-STU385 - SHRGRDE: Grade Table

The key to UCX-STU385 is a composite School Key (LEVL_Code) of 12-bytes plus the Grade Type code (GMOD_Code) of 2-bytes plus the Grade (GRDE Code) of 6-bytes. The SHRGRDO table containing the valid grade combinations for each School (Level) is used as the driver for the loading of the Grade data.

However, both the SHRGRDO and SHRGRDE tables are used in the standard bannerextract SQL:

```
SELECT b.SHRGRDE_LEVL_CODE,
a.SHRGRDO_GMOD_CODE,
b.SHRGRDE_CODE,
b.SHRGRDE_ABBREV,
b.SHRGRDE_TERM_CODE_EFFECTIVE,
b.SHRGRDE_QUALITY_POINTS,
b.SHRGRDE_ATTEMPTED_IND,
b.SHRGRDE_COMPLETED_IND,
b.SHRGRDE_PASSED_IND,
b.SHRGRDE_GPA_IND,
b.SHRGRDE_GRDE_STATUS_IND,
b.SHRGRDE_NUMERIC_VALUE,
```

```

b.SHRGRDE_REPEAT_INCLUDE_IND)
FROM SHRGRDO a, SHRGRDE b
WHERE a.SHRGRDO_GRDE_CODE = b.SHRGRDE_CODE
    AND a.SHRGRDO_LEVL_CODE = b.SHRGRDE_LEVL_CODE
    AND a.SHRGRDO_TERM_CODE_EFFECTIVE = b.SHRGRDE_TERM_CODE_EFFECTIVE
    AND b.SHRGRDE_TERM_CODE_EFFECTIVE = (
        SELECT MAX(SHRGRDE.SHRGRDE_TERM_CODE_EFFECTIVE)
        FROM SHRGRDE
        WHERE SHRGRDE.SHRGRDE_CODE = b.SHRGRDE_CODE
            AND SHRGRDE.SHRGRDE_LEVL_CODE = b.SHRGRDE_LEVL_CODE
            AND SHRGRDE.SHRGRDE_GRDE_STATUS_IND = 'A')
        AND b.SHRGRDE_GRDE_STATUS_IND = 'A'
ORDER BY b.SHRGRDE_LEVL_CODE,
a.SHRGRDO_GMOD_CODE,
b.SHRGRDE_CODE,
b.SHRGRDE_TERM_CODE_EFFECTIVE DESC

```

Invalid combinations should be deleted from UCX_STU385 using Controller.

After this grade information has been extracted from Banner and loaded into UCX_STU385 many additional settings may be manually made for Degree Works using Controller (e.g., several Override flags, Transfer Repeat flags, Transfer Grade value and whether or not to include the UCX_STU385 record in the SEP picklist).

Warning! Due to the existence of many Degree Works only flags/values that must be manually loaded into UCX_STU385 using Controller it is recommended that the UCX_CFG020 RADBRIDGE Add UCX Entries Only flag be set to Y if the UCX extract, ban44, is used to extract new values that have been created in SHRGRDO or any other UCX table originally created from Banner data. In this case only new SHRGRDO records will be loaded into UCX_STU385. Existing UCX_STU385 records will remain untouched and will not be reloaded from Banner. For more information, see the [UCX-STU385 Grade Table](#) topic.

Field Name	Banner Data
HEADER	UCX-STU385.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU385
UCX CODE	STVLEVEL_CODE (12) + STVGMOD_CODE (2) + 4 blanks + SHRGRDO_GRDE (6) + 10 blanks. School Code (12) plus Grade Type (2) plus Grade (6) = 20-byte key.
<hr/>	
UCX VALUE	
Numeric Grade	SHRGRDE_QUALITY_POINTS. The Banner quality point float value is converted to a 9999v999 value with the decimal point removed. For example, a value of 3.5 is converted to 0003500.
Graded Attempted	Y/N. SHRGRDE_GPA_IND
Incomplete Flag	Y/N. If the SHRGRDE_COMPLETED_IND = Y set this Incomplete flag to N. Otherwise set this Incomplete flag to Y.

Field Name	Banner Data
Use in DW GPA Calculator	Y/N. SHRGRDE_GPA_IND

UCX-STU560 - STVCOLL: College Codes

Field Name	Banner Data
HEADER	UCX-STU560.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU560
UCX CODE	STVCOLL_CODE (2) + 28 blanks.
UCX VALUE	
Description	STVCOLL_DESC (30)

UCX-STU563 - STVMAJR: Concentration Codes

Field Name	Banner Data
HEADER	UCX-STU563...R800UCXT.....A. (periods represent spaces).
UCX TABLE	UCX-STU563
UCX CODE	STVMAJR_CODE (4) + 26 blanks if the STVMAJR_VALID_CONCENTRATN_IND = Y. Leading spaces are removed.
UCX VALUE	
Description	STVMAJR_DESC (30)

R900CURR - Curriculum Rules Record

Updated: September 29, 2023

Several Banner tables are used to obtain the data for this Curriculum Rules record.

SOBCURR	Curriculum Rule Table
SORMCRL	Curriculum Control Table
SORCMJR	Curriculum Major Table
SORCMNR	Curriculum Minor Table
SORCCON	Curriculum Concentration Table

The following documentation outlines the columns extracted from Banner for the rad_currrule_dtl. The data in this table will be used to control the drop-down lists on the What-If audit page if the UCX-CFG020 WHATIF Obey Curriculum Rules flag is Y. If this flag is N or not set the standard What-If logic will be used that follows various configurations on several UCX tables to determine what majors, minors, concentrations, etc. are to be used in the drop-down lists for What-If audits.

The following SQL statements are executed to retrieve the Curriculum Rule records from Banner:

Customize your entries in the integration.banner.extract.config Shepherd setting as required – the columns listed within <> will be replaced with the actual data for the record being processed:

SQL used to read the SOBCURR table:

```
SELECT a.SOBCURR_CURR_RULE,
       a.SOBCURR_TERM_CODE_INIT,
       a.SOBCURR_LEVL_CODE,
       a.SOBCURR_CAMP_CODE,
       a.SOBCURR_COLL_CODE,
       a.SOBCURR_DEGC_CODE,
       a.SOBCURR_PROGRAM
  FROM SOBCURR a
 ORDER BY a.SOBCURR_CURR_RULE;
```

SQL used to read the Control records from the SORMCRL table for the SOBCURR_CURR_RULE:

```
SELECT a.SORMCRL_CURR_RULE,
       a.SORMCRL_TERM_CODE_EFF,
       a.SORMCRL_REC_IND,
       a.SORMCRL_ADM_IND,
       a.SORMCRL_STU_IND,
       a.SORMCRL_HIS_IND,
       a.SORMCRL_DAU_IND
  FROM SORMCRL a
 WHERE a.SORMCRL_CURR_RULE = <sobcurr_curr_rule>
   AND a.SORMCRL_TERM_CODE_EFF >= <sobcurr_term_code_init>
 ORDER BY a.SORMCRL_CURR_RULE,
          a.SORMCRL_TERM_CODE_EFF;
```

SQL used to read the Major Rules from the SORCMJR table for the SOBCURR_CURR_RULE:

```
SELECT a.SORCMJR_CURR_RULE,
       a.SORCMJR_CMJR_RULE,
       a.SORCMJR_TERM_CODE_EFF,
       a.SORCMJR_ADM_IND,
       a.SORCMJR_STU_IND,
       a.SORCMJR_HIS_IND,
       a.SORCMJR_REC_IND,
       a.SORCMJR_DAU_IND,
       a.SORCMJR_MAJR_CODE
  FROM SORCMJR a
 WHERE a.SORCMJR_DAU_IND = 'Y'
   AND a.SORCMJR_CURR_RULE = <sobcurr_curr_rule>
 ORDER BY a.SORCMJR_CURR_RULE,
          a.SORCMJR_MAJR_CODE,
          a.SORCMJR_TERM_CODE_EFF DESC,
          a.SORCMJR_CMJR_RULE;
```

SQL used to read the Minor Rules from the SORCMNR table for the SOBCURR_CURR_RULE:

```

SELECT a.SORCMNR_CURR_RULE,
       a.SORCMNR_CMNR_RULE,
       a.SORCMNR_TERM_CODE_EFF,
       a.SORCMNR_ADM_IND,
       a.SORCMNR_STU_IND,
       a.SORCMNR_HIS_IND,
       a.SORCMNR_REC_IND,
       a.SORCMNR_DAU_IND,
       a.SORCMNR_MAJR_CODE_MINR
  FROM SORCMNR a
 WHERE a.SORCMNR_DAU_IND = 'Y'
   AND a.SORCMNR_CURR_RULE = <sobcurr_curr_rule>
 ORDER BY a.SORCMNR_CURR_RULE,
          a.SORCMNR_MAJR_CODE_MINR,
          a.SORCMNR_TERM_CODE_EFF DESC;

```

SQL used to read the Concentration Rules from the SORCCON table for the SOBCURR_CURR_RULE:

```

SELECT a.SORCCON_CURR_RULE,
       a.SORCCON_CCON_RULE,
       a.SORCCON_TERM_CODE_EFF,
       a.SORCCON_ADM_IND,
       a.SORCCON_STU_IND,
       a.SORCCON_HIS_IND,
       a.SORCCON_REC_IND,
       a.SORCCON_DAU_IND,
       a.SORCCON_CMJR_RULE,
       a.SORCCON_MAJR_CODE_CONC
  FROM SORCCON a
 WHERE a.SORCCON_DAU_IND = 'Y'
   AND a.SORCCON_CURR_RULE = <sobcurr_curr_rule>
 ORDER BY a.SORCCON_CURR_RULE,
          a.SORCCON_MAJR_CODE_CONC,
          a.SORCCON_TERM_CODE_EFF DESC,
          a.SORCCON_CMJR_RULE;

```

SQL used to get the Attach Value from the SORCMJR table for the SORCCON_CMJR_RULE:

```

SELECT a.SORCMJR_CURR_RULE,
       a.SORCMJR_CMJR_RULE,
       a.SORCMJR_TERM_CODE_EFF,
       a.SORCMJR_ADM_IND,
       a.SORCMJR_STU_IND,
       a.SORCMJR_HIS_IND,
       a.SORCMJR_REC_IND,
       a.SORCMJR_DAU_IND,
       a.SORCMJR_MAJR_CODE
  FROM SORCMJR a
 WHERE a.SORCMJR_DAU_IND = 'Y'
   AND a.SORCMJR_CMJR_RULE = <sobcurr_curr_rule>
 ORDER BY a.SORCMJR_CMJR_RULE,
          a.SORCMJR_MAJR_CODE,
          a.SORCMJR_TERM_CODE_EFF DESC;

```

a.SORCMJR_TERM_CODE_EFF DESC,
 a.SORCMJR_CMJR_RULE

These records are bridged into the RAD_CurrRule_DTL table in Degree Works. This is an optional table for Degree Works.

Field Name	UCX	Comments
HEADER		This is a 28-byte field identifying the record as one containing rad_currrule_dtl data. It is composed of a 10-byte code containing the 8-byte CURR_RULE_ID plus 2-bytes of FILLER, an 8-byte IDENTIFIER (R900CURR for the rad_currrule_dtl), an 8-byte term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete).
		Example:
		70.....R900CURR.....A.. Spaces are filled with periods (.).
rad_curr_rule_id		This element is the curriculum rule ID code found in the SOBCURR CURR_RULE. It is a number that is left justified and space-filled. For example, a curr_rule of 70 would be loaded as 70..... where each (.) represents a space (8-byte number + 2-bytes of filler).
rad_cat_yr_start	STU016	Calculated using the SORMCRL control table. Refer to the Catalog Start/Stop Processing discussion below for details.
rad_cat_yr_stop	STU035	Calculated using the SORMCRL control table. Refer to the Catalog Start/Stop Processing discussion below for details.
rad_goal_code	R	This element is the code associated with the type of record involved in a degree/goal. Valid values are: CAMPUS: Campus Codes SOBCURR_CAMP_COD E COLLEGE: College Codes SOBCURR_COLL_COD E CONC: Concentrations SORCCON_MAJR_COD E_CONC DEGREE: Degree Codes SOBCURR_DEGC_COD E MAJOR: Major Codes SORCMJR_MAJR_COD E MINOR: Minor Codes SORCMNR_MAJR_COD E_MINR PROGRAM: Program Codes SOBCURR_PROGRAM SCHOOL: School Codes SOBCURR_LEVL_COD E
rad_goal_value		This element is the actual rad_goal_value recorded for a given curriculum rule for this goal code. Leading spaces are removed from any value placed here. Example: PE becomes PE

Field Name	UCX	Comments
rad_attach_code		<p>The SORCCON_CMJR_RULE for a given SORCCON concentration rule is used to lookup the associated SORCMJR record through the SORCMJR_CMJR_RULE. If a record is found this field will be loaded with MAJOR.</p> <p>This code is saying that this concentration is attached or associated with this particular major.</p>
rad_attach_value		<p>The SORCCON_CMJR_RULE for a given SORCCON concentration rule is used to lookup the associated SORCMJR record through the SORCMJR_CMJR_RULE. If a record is found this field will be loaded with the SORCMJR_MAJR_CODE.</p> <p>So if a major is selected on the What-if page that matches the major in an Attached Value then the concentration picklist will be loaded with the associated concentration rad_goal_value above. In the example data below, the 70 rule has an ECON Major.</p> <p>Furthermore, the Concentration of EUTL has an Attach Major Code of ECON. So, if the student selects an ECON Major on the What-If page the EUTL would be automatically loaded into the Concentration picklist (assuming all configurations are set correctly in the UCX_CFG020 WHATIF record).</p>

Catalog Year Start/Stop Processing

The SOBCURR table is the driver. For each SOBCURR record read, the SORMCRL records for that sobcurr_curr_rule are read and used to calculate a Catalog Year Start and a Catalog Year Stop. This Catalog Year Start/Stop range is used to filter rad_currrule_dtl records against the Catalog Year specified on the What-If page. There may be multiple records for a given SORMCRL_CURR_RULE. The SORMCRL_DAU_IND flag (indicates the curriculum rule is used for Degree Auditing purposes if Y) is being used in the standard integration.banner.extract.config Shepherd setting SQL to filter records from the major, minor, and concentration tables.

The following is an example of how the SORMCRL records are processed for a given curr_rule. First, the SORMCRL table is read using the SOBCURR_CURR_RULE of 70. Four SORMCRL records are found.

SORMCRL_CURR_RULE	SORMCRL_TERM_CODE_EFF	SORMCRL_DAU_IND
70	000000	Y
70	200910	N
70	201710	Y
70	201910	N

When integration.banner.extract.curriculumRules.multipleRanges.enabled = false

The first SORMCRL_DAU_IND = Y so the SORMCRL_TERM_CODE_EFF of 000000 is looked up on STU016 to get the catalog year, which becomes the rad_cat_yr_start of 0000. The next SORMCRL_DAU_IND value is N, which ends the curriculum rule. The term of 200910 is looked up in STU016 to get a catalog year of 2009 for academic years 2008-2009. That is when the rule ended. It was valid the year before that so go back one year and use 2008 as the rad_cat_yr_stop. The last two SORMCRL records are ignored because we stopped as soon the first N record was seen.

70	0000	2008	SCHOOL	UG
70	0000	2008	DEGREE	BS
70	0000	2008	PROGRAM	VAT
70	0000	2008	CAMPUS	M
70	0000	2008	COLLEGE	M
70	0000	2008	MAJOR	ACCT
70	0000	2008	MAJOR	ECON
70	0000	2008	MAJOR	FIN
70	0000	2008	MINOR	ECOM
70	0000	2008	CONC	EUTL MAJOR ECON

When integration.banner.extract.curriculumRules.multipleRanges.enabled = true

The first SORMCRL_DAU_IND = Y so the SORMCRL_TERM_CODE_EFF of 000000 is looked up on STU016 to get the catalog year, which becomes the rad_cat_yr_start of 0000. The next SORMCRL_DAU_IND value is N, which ends the curriculum rule. The term of 200910 is looked up in STU016 to get a catalog year value of 2009 for academic years 2008-2009. That is when the rule ended. It was valid the year before that so go back one year and use 2008 as the rad_cat_yr_stop. Because the **multipleRanges** setting is true, the extract continues and analyzes the rest of the SORMCRL records. The next record is a Y starting in term 201710, which means the curriculum rule was reactivated. The last record has a flag of N, which ends the rule in term 201910 and means that the rule is also active for years 2017 through 2018 – the catalog years found by looking up those terms in STU016.

70	0000	2008	SCHOOL	UG
70	0000	2008	DEGREE	BS
70	0000	2008	PROGRAM	VAT
70	0000	2008	CAMPUS	M
70	0000	2008	COLLEGE	M
70	2017	2018	SCHOOL	UG
70	2017	2018	DEGREE	BS
70	2017	2018	PROGRAM	VAT
70	2017	2018	CAMPUS	M
70	2017	2018	COLLEGE	M
70	0000	2008	MAJOR	ACCT
70	0000	2008	MAJOR	ECON
70	0000	2008	MAJOR	FIN
70	0000	2008	MINOR	ECOM
70	0000	2008	CONC	EUTL MAJOR ECON

These two sets of ranges tell Degree Works that the curriculum rule is valid between 0000 and 2008 and again between 2017 and 2018.

Majors, Minors, and Concentrations

For the majors, minors, and concentrations attached to curriculum rules, the same logic applies. Here is an example of some majors associated with curriculum rule 31.

CURR_RULE	CMJR_RULE	TERM_CODE_EFF	DAU_IND	MAJR_CODE
31	139	199010	Y	ENGL
31	256	199010	Y	ACCT
31	12381	199010	Y	CHEM
31	255	199010	Y	HIST
31	51	199510	Y	ENGL
31	52	199510	Y	ACCT
31	132	199510	Y	BIOL
31	12378	200010	N	ENGL
31	12379	200010	N	ACCT
31	12380	200010	Y	CHEM

When integration.banner.extract.curriculumRules.multipleRanges.enabled = false

Major ENGL should be active from 1990 to 1999 because there is an N for term 200010.

Major ACCT should be active from 1990 to 1999 because there is an N for term 200010.

Major CHEM should be active from 1990 to 9999 because there is no N record found.

Major HIST should be active from 1990 to 9999 because there is no N record found.

Major BIOL should be active from 1995 to 9999 because there is no N record found.

These records should then be sent to Degree Works and stored in the rad_currrule_dtl.

RAD_CURR_RULE	RAD_CAT_YR_ST	RAD_CAT_YR_ST	RAD_GOAL_CODE	RAD_GOAL_VALUE
31	1990	1999	MAJOR	ENGL
31	1990	1999	MAJOR	ACCT
31	1990	9999	MAJOR	CHEM
31	1990	9999	MAJOR	HIST
31	1995	9999	MAJOR	BIOL

When integration.banner.extract.curriculumRules.multipleRanges.enabled = true

Major ENGL should be active from 1990 to 1999 because there is an N for term 200010.

Major ACCT should be active from 1990 to 1999 because there is an N for term 200010.

Major CHEM should be active from 1990 to 1994 because in 199510 there is no CHEM record. However, it should then be active again from 2000 to 9999.

Major HIST should be active from 1990 to 1994 because in 199510 there is no HIST record.

Major BIOL should be active from 1995 to 1999 because in 200010 there is no BIOL record.

These records should then be sent to Degree Works and stored in the rad_currrule_dtl.

RAD_CURR_RULE	RAD_CAT_YR_ST	RAD_CAT_YR_ST	RAD_GOAL_CODE	RAD_GOAL_VALUE
31	1990	1999	MAJOR	ENGL
31	1990	1999	MAJOR	ACCT
31	1990	1994	MAJOR	CHEM
31	1990	1994	MAJOR	HIST
31	1995	1999	MAJOR	BIOL
31	2000	9999	MAJOR	CHEM

Class repeats/multiple occurrences

Updated: September 30, 2022

When bridging Banner class data into Degree Works, it is necessary to ensure that decisions made about classes taken multiple times are included in the bridged data.

Degree Works will treat repeated coursework based upon the repeat policy information included in the bridged class records. This data is bridged in the rad_repeat_plcy and rad_repeat_ptr columns on the rad_class_dtl and rad_transfer_dtl records. The valid repeat policies that can be used in Degree Works are defined in UCX-AUD047. For each class record which represents a repeat instance, the Repeat Plcy and the Repeat Ptr must be filled in appropriately.

The [Repeat policy](#) topic describes how Degree Works treats repeated course work in general along with examples. This section is devoted specifically to the options available to Banner sites in determining how classes that have been taken multiples times in Banner are extracted into Degree Works. Several issues involving these classes are discussed below:

- Repeated Classes versus Repeatable Classes
- Identifying In-Progress Repeats
- Identifying Repeats for Renumbered Courses
- Identifying In-Progress Repeats for Courses that have been Renumbered

Repeated classes versus repeatable classes

Updated: September 30, 2022

A repeated class is a class that has been repeated more than one time for a better grade. A repeatable class is a class that may be taken multiple times for credit (for example, many music, physical education, and art classes).

To identify a repeated class in Banner for classes that have been completed, the Repeat Course Indicator is interrogated by the Banner extract: historic class - SHRTCKN_REPEAT_COURSE_IND; transfer class - the SHRTRCE_REPEAT_COURSE. If the repeat course indicator field is BLANK, the class will be processed as a "normal" class for credit. If this field contains an "A"- (Averaged) or "E" (Excluded) value, the class rules defined in the UCX-CFG020 BANNER record for these values will be followed. There is a Historic Skip flag/Repeat Policy and a Transfer Skip flag/Repeat Policy defined for each of these values. For more information, see the [UCX-CFG020 BANNER](#) topic. If, however, this field contains an "I" (Included) one of the Repeatable Options defined in the UCX-CFG020 BANNER record is used to determine how to process the class. If it is a repeatable class the rad_repeat_plcy and rad_repeat_ptr will not be loaded. Otherwise it will be treated as a repeated class for a better grade and these two repeat fields will be loaded.

The SHRTCKN_SUBJ_CODE and SHRTCKN_CRSE_NUMB are used to lookup the associated SCBCRSE record in descending sequence by SCBCRSE_EFF_TERM. The first SCBCRSE record returned with a SCBCRSE_EFF_TERM less than or equal to the SHRTCKN_TERM_CODE will be used in the rules defined below.

The valid UCX-CFG020 BANNER Repeatable Options are:

If the Repeatable Option is BLANK, Option "L" will be used as the baseline (default) value.

Option "N" - Do NOT check the SCBCRSE record at all

Load the UCX-CFG020 BANNER "Repeat Policy I" that contains the appropriate "Include" rad_repeat_plcy value. Each historic class with an "I" SHRTCKN_REPEAT_COURSE_IND will be considered as a **repeat for a better grade**. Each transfer class with an "I" in SHRTRCE_REPEAT_COURSE will be considered as a **repeat for a better grade**. The rules defined in "UCX-AUD047 - Repeat Policies" will be applied to each of the historic/transfer "I" classes.

Option "L" - Check the SCBCRSE_REPEAT_LIMIT Only

If the REPEAT_LIMIT = ZERO or the REPEAT_LIMIT = NULL, the class is processed as a **repeat for a better grade**. Otherwise the class is considered **repeatable**.

Option "U" - Check the SCBCRSE_MAX_RPT_UNITS

If the MAX_RPT_UNITS = NULL, the class is considered **repeatable**. If the MAX_RPT_UNITS > 0, the class is considered **repeatable**. Otherwise the class is processed as a **repeat for a better grade**.

Option "B" - Check both the SCBCRSE_REPEAT_LIMIT and SCBCRSE_MAX_RPT_UNITS

If the REPEAT_LIMIT = NULL and the MAX_RPT_UNITS = NULL, the class is processed as a **repeat for a better grade**. If the REPEAT_LIMIT = ZERO and the MAX_RPT_UNITS = ZERO, the class is processed as a **repeat for a better grade**. If the REPEAT_LIMIT > ZERO and the MAX_RPT_UNITS = ZERO, the class is processed as a **repeat for a better grade**. If the REPEAT_LIMIT = ZERO and the MAX_RPT_UNITS > ZERO, the class is processed as a **repeat for a better grade**. If the REPEAT_LIMIT = NULL and the MAX_RPT_UNITS > ZERO, the class is **repeatable**. If the REPEAT_LIMIT > ZERO and the MAX_RPT_UNITS = NULL, the class is **repeatable**. If the REPEAT_LIMIT > ZERO and the MAX_RPT_UNITS > ZERO, the class is **repeatable**. Otherwise process the class as a **repeat for a better grade**.

Option "I" - The Credits are Included in the REPEAT_LIMIT and SCBCRSE_MAX_RPT_UNITS

If the REPEAT_LIMIT = 1 and the MAX_RPT_UNITS >= CREDIT_HR_LOW, the class is processed as a **repeat for a better grade**. If the REPEAT_LIMIT = NULL and the MAX_RPT_UNITS = CREDIT_HR_LOW, the class is processed as a **repeat for a better grade**. If the REPEAT_LIMIT = NULL and the MAX_RPT_UNITS > CREDIT_HR_LOW, the class is **repeatable**. If the REPEAT_LIMIT > 1, the class is **repeatable**. Otherwise process the class as a **repeat for a better grade**.

In-progress repeats identification

Updated: March 25, 2022

In Banner in-progress classes are stored in the SFRSTCR table, but do not have a repeat indicator defined.

To determine if a current class is repeatable or is being repeated for a better grade, the Repeatable Option edit is performed. If the class is repeatable, the current course is not an in-progress repeat and the rest of the repeat processing is skipped. Otherwise, the current class may have a repeated class counterpart in the historic or transfer tables in which case processing continues with the following In-Progress searches.

Review the UCX-CFG020 BANNER **In-Progress Equiv** configuration flag. If set to Y, the dap_eqv_crs_mst will be searched for an equivalent course key that matches the current In-Progress class Course Key (Subject + Course Number). This allows courses that have been renumbered over time to still be identified as In-Progress repeats. However, this does require that the Equivalency mappings stored in the dap_eqv_crs_mst be loaded by the Banner Equivalent extract (ban43) or be loaded into CFG070 manually using Controller and then launching the dapucx2eqv script to update the dap_eqv_crs_mst with manual changes.

In-progress classes are handled when the UCX_CFG020 BANNER Repeat Policy is B.

For each SFRSTCR current class (linked to SSBSECT through Term Code and Crn) for a given student, the historic SHRTCKN table is read looking for an exact match on Course Key (Subject + Course Number) and Levl Code while NOT matching the current Term Code. If an exact match is found, the UCX_CFG020 BANNER Repeat Policy A will be checked.

If B, the rad_repeat_plcy and rad_repeat_ptr will be spaced out as the class will not be treated as a repeat by the auditor. The rad_class_status will be loaded with CH for Current-History. This flag is only used for informational purposes and shows that the class is part of a repeat scenario. An additional edit will then be made using UCX_CFG020 BANNER Averaged Count in Major Minor GPA.

If Y, the rad_pass_flag will be set to N, which marks the class as not passed (failed) so that this class will count in the major/minor if it could fit, and if not, it will end up in insufficient.

If N, the rad_insuff_flag (insufficient) will be set to Y, which will force the class into the insufficient section of the audit so it will not count in the overall credits. However, it will still affect the GPA.

If not B, the historic BIF rad_repeat_plcy will be loaded with the UCX-CFG020 BANNER I Repeat Policy, and the rad_repeat_ptr record will be loaded with the current Course Key. The current rad_class_dtl BIF record will be loaded with the same rad_repeat_plcy and rad_repeat_ptr. The

rad_class_status will be loaded with CH for Current-History. This flag is used only for informational purposes and shows that the class is part of an in-progress repeat scenario.

If an exact match is not found and the UCX-CFG020 BANNER In-Progress Equiv is set to Y, the dap_eqv_crs_mst will be searched for an equivalent Course Key match. If an equivalent Course Key is found for the historic class, the equivalent Course Key will be compared to the current Course Key. If an exact match is found, the UCX_CFG020 BANNER Repeat Policy A will be checked.

If B, the rad_repeat_plcy and rad_repeat_ptr will be spaced out as the class will not be treated as a repeat by the auditor. The rad_class_status will be loaded with CH for Current-History. This flag is used only for informational purposes and shows that the class is part of a repeat scenario. An additional edit will then be made using UCX_CFG020 BANNER Averaged Count in Major Minor GPA.

If Y, the rad_pass_flag will be set to N, which marks the class as not passed (failed) so that this class will count in the major/minor if it could fit, and if not, it will end up in insufficient.

If N, the rad_insuff_flag (insufficient) will be set to Y, which will force the class into the insufficient section of the audit so it will not count in the overall credits. However, it will still affect the GPA.

If not B, the historic BIF rad_repeat_plcy will be loaded with the UCX-CFG020 BANNER I Repeat Policy, and the rad_repeat_ptr record will be loaded with the dap_eqv_crs_mst equivalent Course Key. The current rad_class_dtl BIF record will be loaded with the UCX-CFG020 BANNER I Repeat Policy, and the rad_repeat_ptr record will be loaded with the historic SHRTCKN Course Key. The rad_class_status will be loaded with CH for Current-History. This flag is used only for informational purposes and shows that the class is part of an in-progress repeat scenario.

For each SFRSTCR current class (linked to SSBSECT through Term Code and Crn) for a given student, the transfer SHTRCE table is read looking for an exact match on Course Key (Subject + Course Number) and Levl Code while NOT matching the Term Code. If a match is found, the UCX_CFG020 BANNER Repeat Policy A will be checked.

If B, the rad_repeat_plcy and rad_repeat_ptr will be spaced out as the class will not be treated as a repeat by the auditor. The rad_class_status will be loaded with CT for Current-Transfer. This flag is used only for informational purposes and shows that the class is part of a repeat scenario. An additional edit will then be made using UCX_CFG020 BANNER Averaged Count in Major Minor GPA.

If Y, the rad_pass_flag will be set to N, which marks the class as not passed (failed) so that this class will count in the major/minor if it could fit, and if not, it will end up in insufficient.

If N, the rad_insuff_flag (insufficient) will be set to Y, which will force the class into the insufficient section of the audit so it will not count in the overall credits. However, it will still affect the GPA.

If not B, the transfer BIF rad_repeat_plcy will be loaded with the UCX-CFG020 BANNER I Transfer Repeat Policy, and the rad_repeat_ptr record will be loaded with the current Course Key. The current rad_class_dtl BIF record will be loaded with the UCX-CFG020 BANNER I Repeat Policy (note the difference between this and the transfer rad_repeat_ptr loaded), and the same rad_repeat_ptr. The rad_class_status will be loaded with CT for Current-Transfer. This flag is used only for informational purposes and shows that the class is part of an in-progress repeat scenario.

If an exact match is not found and the UCX-CFG020 BANNER In-Progress Equiv is set to Y, the dap_eqv_crs_mst will be searched for an equivalent Course Key match. If an equivalent Course Key is found for the transfer class, the equivalent Course Key will be compared to the current Course Key. If an exact match is found, the UCX_CFG020 BANNER Repeat Policy A will be checked.

If B, the rad_repeat_plcy and rad_repeat_ptr will be spaced out as the class will not be treated as a repeat by the auditor. The rad_class_status will be loaded with CT for Current-Transfer. This flag is used only for informational purposes and shows that the class is part of a repeat scenario. An additional edit will then be made using UCX_CFG020 BANNER Averaged Count in Major Minor GPA.

If Y, the rad_pass_flag will be set to N, which marks the class as not passed (failed) so that this class will count in the major/minor if it could fit, and if not, it will end up in insufficient.

If N, the rad_insuff_flag (insufficient) will be set to Y, which will force the class into the insufficient section of the audit so it will not count in the overall credits. However, it will still affect the GPA.

If not B, the transfer BIF rad_repeat_plcy will be loaded with the UCX-CFG020 BANNER I Transfer Repeat Policy, and the rad_repeat_ptr record will be loaded with the dap_eqv_crs_mst equivalent Course Key. The current rad_class_dtl BIF record will be loaded with the UCX-CFG020 BANNER I Repeat Policy (note the difference between this and the transfer rad_repeat_ptr loaded), and the rad_repeat_ptr record will be loaded with the transfer SHRTRCE Course Key. The rad_class_status will be loaded with CT for Current-Transfer. This flag is used only for informational purposes and shows that the class is part of an in-progress repeat scenario.

If no exact matches and no equivalency matches are found in the historic SHRTCKN or transfer SHRTRCE class records, the current class is not considered as a repeated class so the rad_repeat_plcy and rad_repeat_ptr are not loaded.

If the UCX_CFG020 BANNER Repeat Policy A is B, the rad_repeat_plcy and rad_repeat_ptr will always be BLANK as the class flags (pass flag, insufficient flag, and so on) will be used to control how the repeated class is treated by the auditor.

Example for In-Progress Repeat Matches and Equivalents

Student A has 4 Current classes. Relevant History and Transfer classes are also displayed.

Courses were renumbered for MATH115 and HIST105 starting in 2010.

SFRSTCR/SSBSECT Current Classes (Term = 200940, Catalog Year = 2010):

			RepeatPtr
ENGL101	Composition	ENGL101	history exact match
HIST121	US History I	HIST105	transfer equivalent match
MATH130	Geometry	MATH115	history equivalent match
PHYS141	Astronomy	PHYS141	transfer exact match

SHRTCKN History Classes (Term = 200620, Catalog Year = 2006):

ENGL101	Composition	ENGL101
MATH115	Geometry	MATH130

SHRTRCE Transfer Classes (Term = 200730, Catalog Year = 2007):

HIST105	US History I	HIST121
PHYS141	Astronomy	PHYS141

dap_eqv_crs_mst (partial – only showing pertinent Catalog Years):

Old Cat Year	Old Course Key	New Cat Year	New Course Key
2006	MATH 115	2010	MATH 130
2007	HIST 105	2010	HIST 121

Repeats for renumbered courses identification

Updated: March 25, 2022

If a Banner class record has a Repeat Course Indicator of 'I' (historic - SHRTCKN_REPEAT.Course_IND, transfer - SHRTRCE_REPEAT.Course) and the course is not repeatable (class is being repeated for a better grade) then the equivalency table (dap_eqv_crs_mst) will be checked for a course equivalent.

If a new course equivalent is found for the given history or transfer class, the equivalent Course Key (Subject + Course Number) will be loaded into the rad_repeat_ptr on the appropriate RAD table (rad_class_dtl for historic classes and rad_transfer_dtl for transfer classes). Thus, when audits are processed the rad_repeat_ptr will contain the new Course Key that should be used in the requirement blocks defined using Scribe.

Bridge Interface Format

Updated: March 25, 2022

Review the guidelines for bridging the data in client student records systems with Degree Works.

Note: If you are a Banner school, you should instead see [Banner Integration](#).

Degree Works data storage

Updated: March 25, 2022

Degree Works has two distinctively different data storage needs. The first is the need to store the results of degree audits in a proprietary format for use by the Auditor Engine. The second is the need to store basic data about a student, both academic and demographic.

The source of information about the student comes from an institution's student records system, usually located on a computer platform separate from the Degree Works platform.

Simply stated, the client must unload portions of the data in the student records system into a series of flat files that can then be imported through a bridge into the Degree Works data structure, thereby replicating the data. The biggest advantage to this approach is that the bridge is a single point of communication with Degree Works, thereby making updates to Degree Works easy. Only if Ellucian was to change the Degree Works data structures or if the student records system were to change would the bridge program have to be modified.

Degree Works components

Updated: March 25, 2022

For the purposes of discussion, think of Degree Works data as collections of columns that reside in "data sets".

Much of the information stored in the columns is in the form of "coded values". The coded values are validated against a set of "approved" values located in something called the UCX. The UCX is a data storage construct where each code needing validation is referenced against a "named" table. For instance, the student level column must have a coded value validated against UCX-STU305, where codes have been created for values such as "Freshman", "Sophomore" and so on.

Some of the data sets are mandatory and are required to be sent from the Student Records System to Degree Works. If any of the required data sets are missing then either a Degree Works audit or a Transfer Equivalency transfer articulation will fail. In addition to the mandatory data sets, there are optional data sets that may be used as batch reporting criteria, for non-course requirements, or for "IF" conditions in degree requirements.

The following table contains the names of the “tables” in the Repository for Academic Data (RAD), with a description of each data set. The “Degree Works” and “Transfer Equivalency” columns indicate that the data set is required (R), optional (O), or is not used (X). The “Indicator” defines the type of bridge interface record created for each database table and is used in the “Header” for every record bridged to Degree Works.

Indicator	Database table	Degree Works	Transfer Equivalency	Description
R011PRIM	rad_primary_mst	R	R	ID, name, gender, address, SSN, birthdate (of students, advisors, administrators). Active term for students.
R011PRIM	rad_student_mst	R	O	The “Active Term” loaded from the R011PRIM record above is actually stored on raddb in this table.
R026APPL	rad_applicant_dtl	X	R	Applicant data: degree, major(s), college(s), and term.
R027GOAL	rad_goal_dtl	R	O	By term: degree, school, level, catalog year
R033GDTA	rad_goaldata_dtl	O	O	By term, degree and school: major(s), minor(s), concentrations(s), specializations (s), advisor(s), program, student status.
R062TERM	rad_term_dtl	O	O	By term: credits earned, attempted, graded, grade points, GPA.
R071CLAS	rad_class_dtl	R	O	Course, grade, credits, term, repeat status, credit-type, grade-type, section, miscellaneous.
R082TRAN	rad_transfer_dtl	O	O	Transfer class data: course, grade, credits, term, repeat status, credit-type, grade-type, advanced placement scores, transfer school, data from transfer school.
R085ATTR	rad_attr_dtl	O	O	Class Attribute data; the rad_attr_key ties this data to either a class or transfer record. Contains a name-value pair.
R091TEST	rad_test_dtl	O	O	Test codes, scores, and dates.

Indicator	Database table	Degree Works	Transfer Equivalency	Description
R100PDEG	rad_previnst_dtl	O	O	Previous college and degree data, used primarily by Transfer Equivalency.
R111NCRS	rad_noncrse_dtl	O	O	Information about non-course requirements, used with NONCOURSE rule.
R121CUST	rad_custom_dtl	O	O	Information about custom test codes, scores, and dates; used with IF rule.
R126REPT	rad_report_dtl	O	O	Extra information to be printed on audit reports that is not otherwise used by Degree Works.
R128AID	rad_aid_dtl	O	O	Data needed for Financial Aid audits.
R130NOTE	dap_note_dtl	O	O	Information about the notes in the dap_note_txt_dtl.
R140NTXT	dap_note_txt_dtl	O	O	Notes regarding a student.
R171SHPU	shp_user_mst	O	O	Security data indicating user's Access-ID, Access-Code (password), and access to Degree Works services.
R180SWAP	rad_swap_id_dtl	O	O	Information about IDs that have changed.
R190DEQV	dap_eqv_crs_mst	O	O	Information about equivalent courses.
R200TREQ	dap_transfer_dtl	O	R	One record for each transfer transcript course to be evaluated by Transfer Equivalency for transfer equivalence.
R602CRSE	rad_course_mst	R	R	Course catalog; required if courses in requirements must be validated.
R605CRSA	rad_crs_attr_dtl	O	O	Course Attribute data; the rad_attr_key contains a rad_course_key which ties this data to a rad_course_mst record.
R651MAPD	dap_mapping_dtl	O	O	Transfer Equivalency transfer equivalence data mapping courses at transfer schools to courses at your institution.

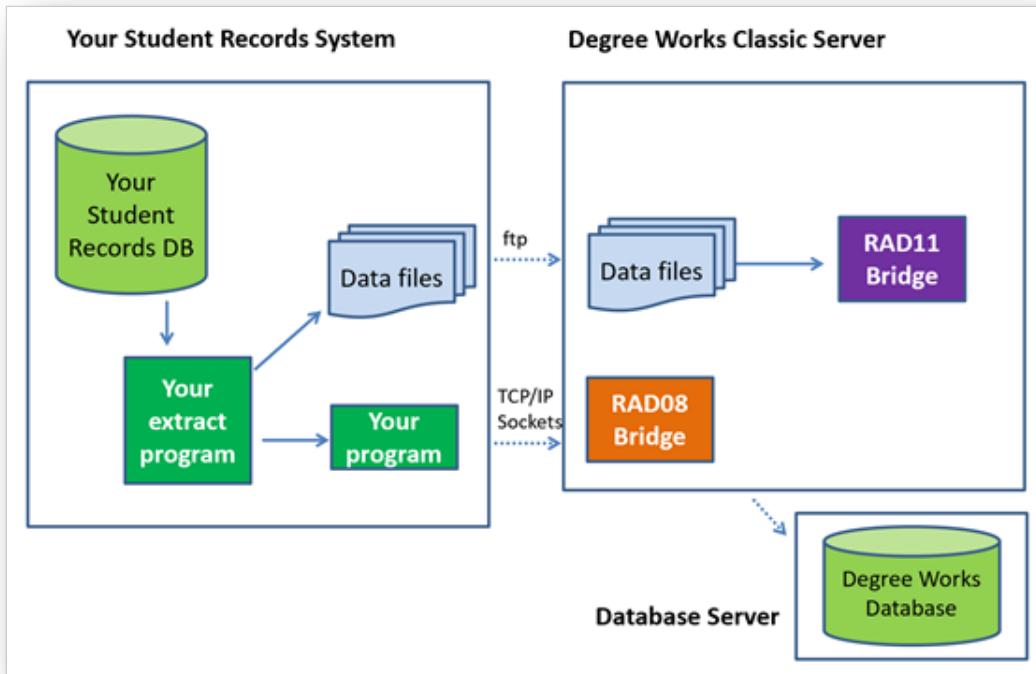
Indicator	Database table	Degree Works	Transfer Equivalency	Description
R655MAPC	dap_map_cond_dtl	O	O	Transfer Equivalency transfer equivalence data indicating special conditions the must be met before a mapping will be applied during transfer articulation.
R701ETSM	rad_ets_mst	O	R	HS and Transfer school codes, names, calendar, and identification data.

A determination will be made during the installation analysis with Ellucian as to which, if any, optional tables will be needed for the bridge. The name of the program that runs on the host computer, which provides Degree Works bridging services, is RAD11. This program operates in “batch” mode, updating the Degree Works data structure periodically. In addition to batch, the bridge is available for on-demand refresh when an audit executes.

Degree Works bridge specifications

Updated: September 30, 2022

There is a structural flow of information between the customer’s existing Student Records System and Degree Works, a process known as the bridge.



The customer must create an extract process to move data from the Student Records System to one or more flat files that will be transferred to the classic server or uploaded as part of the RAD11 Transit job. These flat files must be formatted according to the Bridge Interface Formats presented on the following pages. Ellucian staff will work with the customer to determine the proper source of information for each field. File Transfer Protocol (FTP) may be used to transfer the files to the classic server. When transferred, the RAD11 Bridge Batch Processor reads the data files and loads data into the Degree Works Repository for Academic Data (RAD). The disadvantage of the bridge approach is the timeliness of the data. Student data needed by Degree Works, such as currently enrolled classes and grades, is only as up-to-date as the last time the bridge program executed. To avoid such timeliness issues, but with a potential impact on network traffic, Degree Works also supports on-demand refresh of data from the Student Records System.

After the student data resides in the RAD database, Degree Works can use the data with Degree Works on the Web, Transfer Equivalency, and Transit. The data is used for degree audits, student searches, ID-specific requirements, notes, and exceptions. Transfer Equivalency, an optional add-on to Degree Works, uses the data for transfer articulation.

Minimum data

Updated: September 30, 2022

Minimum data is required for auditing a student in Degree Works and transferring mappings and transcripts in Transfer Equivalency.

The major and minor information on the rad_goaldata_dtl are not required (and thus not listed here) unless the University has major/minor requirements and the student has elected majors or minors. The same is true for Concentration data. The rad_goal_dtl records that are bridged must have unique ID, school, and degree combinations.

Degree Works record	Degree Works field
rad_primary_mst	HEADER
	rad_id
	rad_name
rad_student_mst	ACTIVE TERM
rad_goal_dtl	HEADER
	rad_id
	rad_school
	rad_degree_code
	rad_catalog_yr
	rad_stu_level
rad_class_dtl	HEADER
	rad_id
	COURSE-DISC
	COURSE-NUM
	rad_course_title
	rad_school
	rad_audit_flag
	rad_insuff_flag
	rad_inprog_flag
	rad_withdw_flag
	rad_incomp_flag
	rad_pass_flag
	rad_credits
	rad_credits_earn
	rad_gpa_credits
	rad_grade_points
	rad_pass_fail
	rad_final_grade
	rad_final_gr_num

Degree Works record	Degree Works field
	rad_term

The following table identifies the minimum data required for transfer mappings in Transfer Equivalency.

Degree Works record	Degree Works field
rad_primary_mst	HEADER
	rad_id
	rad_name
rad_ets_mst	HEADER
	rad_ets_code
	rad_ets_school
rad_course_mst	HEADER
	COURSE-DISC
	COURSE-NUM
	rad_course_title

The following table identifies the minimum data required for transfer transcripts in Transfer Equivalency.

DGW record	DGW field
rad_primary_mst	HEADER
	rad_id
	rad_name
dap_transfer_dtl	HEADER
	dap_stu_id
	dap_school_id
	dap_tr_disc
	dap_tr_crse_num
	dap_tr_credits
	CALENDAR
	dap_tr_grade
	dap_tr_trm_sort

Setup considerations

Updated: September 30, 2022

There are a number of items that must be considered when setting up Degree Works.

Where can Ellucian get the access ID for administrators and staff who will be using Degree Works? It is recommended that you start with the Registrar and Registrar's staff. Ellucian will need the password for each user who will access Scribe or the dashboard. In production, it is recommended that you use single sign-on and thus Degree Works does not need to store your users' passwords.

The GPA calculated by Degree Works accommodates 3 digits after the decimal, for example, 1.999. Should the GPA be truncated or rounded? In other words, is $1.9999 = 1.999$ or 2.000 ?

Should transfer classes be included in the GPA calculated by Degree Works?

Degree Works needs to know how to handle classes that have been repeated. How are repeated classes designated in your student record system? If possible, provide a copy of your repeat policy.

In Degree Works, catalog year is an essential value on a student's record. This value links each student to a particular set of requirements. Does your student record system provide a catalog year? If not, can you default the catalog year in the data sent from the Student Records System?

Transfer Equivalency setup considerations

Updated: September 30, 2022

There are a number of items that must be considered when setting up Transfer Equivalency.

Does your student system allow multiple simultaneous applications from the same student? Is each application for a different degree or term?

Do transfer equivalents vary by degree? In other words, does ENGL110 articulate to a different course depending on the intended degree of the applicant?

When test scores are sent to the bridge as part of a transfer transcript, which of the following settings describes the action to be taken by the bridge if a record for the test score already exists? (UCX-CFG020 TREQ TEST-SCHEME)

- A = Add a new test record.
- M = Modify the existing test record, overwriting it with the incoming test data.
- H = keep the Highest test score. Compare the incoming test score to the existing score. If the incoming score is higher, then overwrite with the incoming test data.
- L = keep the Latest test date. Compare the incoming test date to the existing date. If the incoming date is more recent, then overwrite with the incoming test data.

Bridge program

Updated: September 30, 2022

The Bridge batch program (RAD11) is the processor used to update the Degree Works data repository.

The bridge takes as input a data file. The data file is an ASCII file with fixed length records of 1000 bytes. The data file contains records as described in this document. A data file can contain the data for multiple students, ETS schools, courses, UCX tables and course equivalents. It is very important that the data for an ID is together in a file. Data for an ID must not be split across multiple files. The same is true for UCX tables. Data for a UCX table must not be split across multiple files. Transfer transcript data destined for Transfer Equivalency should be in a separate data file.

It is essential for a student to have at least two records: a rad_primary_mst (R011PRIM) and rad_goal_dtl (R027GOAL). If a student is missing one of these records a fatal error will be recorded in the rad_log_dtl and processing will continue with the next ID. The rad_primary_mst record (R011PRIM) MUST BE FIRST in the set of records for each student. After the primary record the data records may occur in any sequence desired as the data file is NOT sorted. However, the note logic requires that the R130NOTE records come BEFORE the R140NTXT note text detail records for a given student. All of the R130NOTE records may come first for a given student followed by all of the R140NTXT records or each R130NOTE record may be followed by its associated R140NTXT records. Also, to be considered a student the “active term” on the R011PRIM record must NOT be blank. The “active term” will ultimately be written to the rad_student_mst, but the data is included in the rad_primary_mst (R011PRIM) record to reduce the number of data files required for the bridge. Instructors and other staff members can be added by RAD11 and they require only one record: a rad_primary_mst – and the term field must be blank. In addition, the shp_user_mst is a required table for any user who wishes to access Degree Works. The shp_user_mst records may be included (recommended) with the student data or included in a separate file that is processed AFTER the student data has been bridged.

All records for an ID must be included in a single data file. If the bridge detects that no data has changed for the student as of the last time the student was bridged then no records will be updated in Degree Works. This is controlled through “hash” records. There is a hash table for each type of record: one for classes, one for goals, and so on. When a grade on a class comes in with a different value from the previous time the hash for classes will contain a new value, for example. In doing this the bridge knows that the class data has changed and thus the old classes are removed and the new classes are inserted. If the goals have not changed then the goal hash will not change and the old goal records in Degree Works will remain as before. When data changes do occur for a student a new audit for each of the student’s goals is processed.

For ETS, Course and dap_eqv_crs_mst data only records that need to be added or updated need to be included when processing these types of records. It is also not true when processing transfer transcripts for Transfer Equivalency (see below).

In addition to fatal errors, RAD11 records warnings about missing or incorrect data and success to the rad_log_dtl. At the completion of processing RAD11 extracts all of the records in the rad_log_dtl and writes them to a file in the “data” sub-directory named RADLOG. This is an ASCII file that can be reviewed using most text editors. In addition to student, ETS and course data, RAD11 can process transfer transcripts into Transfer Equivalency, UCX table updates, delete all records for an ID, change the ID of a student and add course equivalence records. Information on

how to format these types of records is included in this document. The modify flag is currently only used in the shp_user_mst and in other records when bridging Transfer Transcripts as described in the section that follows.

When the bridge finishes loading student data a list of students with changed data, based on the hashes mentioned above, is created. This list of students is then passed onto DAP22 to have audits run to ensure the latest audits reflect the data changes. You can control how these audits are created by modifying the integration.bridge.audits settings in Controller. For more information, see the [integration.bridge.audits](#) topic. A separate dap22 log file is created for this process and can be accessed through Transit. You can review the log to see how many students had data changes and to check the status of those audits.

The simplest and best way to run RAD11 is to run it through Transit. In the questions section you simply supply the name of the BIF data file that exists in the \$ADMIN_HOME/data directory on the classic server. However, if you want to schedule RAD11 to run inside of cron, you can use the launchjob script. For more information on copying the sample rad11 json file into \$ADMIN_HOME/myjobs and editing, see the [The launchjob script](#) topic. Here is what the RAD11 JSON file you place in the myjobs directory (that you create) should contain:

```
{
  "name": "RAD11",
  "parameters": [
    {
      "type": "QUESTIONS",
      "answers": [
        {
          "id": "rad11.bifFilename",
          "value": "bifFilenameIn_admin/data"
        }
      ]
    }
  ]
}
```

The value should be changed to the name of the BIF file in your admin/data directory. For example:

```
"value": "activeStudents.bif"
```

You then run RAD11 using launchjob pointing to your JSON file:

```
$ launchjob $ADMIN_HOME/myjobs/rad11.active.json
```

You can review the results of the job using the Manage Jobs tab in Transit.

It is recommended that you create a rad11.xxxx.json file for each type of extract in the myjobs directory. Perhaps names like this:

rad11.students.json

rad11.applicants.json

rad11.advisors.json

rad11.staff.json

rad11.course.json

rad11.ets.json

rad11.mappings.json

rad11.equivs.json

Each JSON file would have the “value” pointing to the corresponding BIF filename.

Transfer transcripts

Updated: March 25, 2022

When a transfer transcript is sent to the bridge, only certain tables are affected.

Transfer transcripts should be sent in a file separate from the student system data. If student system data and transfer transcript data from a source other than the student system need to be processed together, put the student system data in one BIF file and the transfer transcript file into a separate BIF file.

rad_primary_mst: Set the ACTION-FLAG to T to indicate that a transfer transcript is being processed for the student. This will prevent the bridge from deleting all records for the student. Instead, RAD11 will verify that the rad_primary_mst exists for the student ID. If it exists, then blank data columns will be updated with non-blank incoming data. If a rad_primary_mst does not exist then it will be added. If the student does not yet exist in the student system then the student ID sent with the transcript may be a unique “temporary” ID recognizable to the student system as such. The student system should then create a “real” student ID when the transfer data is rolled from Transfer Equivalency.

rad_applicant_dtl: The Bridge will automatically create a rad_applicant_dtl, if one does not already exist for the student ID. It is not necessary to send this data as part of the transfer transcript. If a rad_applicant_dtl is sent as part of the transfer transcript and a rad_applicant_dtl already exists, all data on the existing rad_applicant_dtl will be overwritten with data from the bridge file.

rad_previnst_dtl: The Bridge will automatically create a rad_previnst_dtl for the transfer school, if one does not already exist for the student. It is not necessary to send this data as part of the transfer transcript but, if sent, the bridge will overwrite any existing data.

rad_test_dtl: The Bridge will add or update a rad_test_dtl for each test code sent as part of the transcript. The bridge will also create a dap_transfer_dtl with a SCHOOL-ID of “TESTS” for use by Transfer Equivalency.

dap_transfer_dtl: The Bridge will delete unarticulated transfer courses for the rad_school_id and rad_school code (UG or GR) from the student’s dap_transfer_dtl. Each incoming dap_transfer_dtl record sent as part of the transcript will be evaluated to determine if it has already been articulated in Transfer Equivalency. If the course has not been articulated then it is added. To be considered a match to an existing articulated record, the incoming transfer record must have the same rad_id, rad_school_id, rad_school and start/stop date or term. The rad_school is used to differentiate undergraduate from graduate transcripts for the same student

from the same transfer institution. If neither the start/stop date nor the term match then the course is added. Any transfer records with errors (for example, invalid grade or school_id) are not added but are reported in the log file created by RAD11. Transfer courses in error can then be added using the Transcript function in Transfer Equivalency.

No other data is processed as part of a transfer transcript.

Repeat policy

Updated: September 30, 2022

When bridging class data into Degree Works, it is necessary to ensure that decisions made about repeated course work are included in the bridged data.

Degree Works will treat repeated coursework based upon the repeat policy information included in the bridged course records. This data is bridged in the rad_repeat_plcy and rad_repeat_ptr columns on the rad_class_dtl record. There are six (6) valid repeat policies that can be used in Degree Works. These policies are defined in UCX-AUD047. For a description of repeat policies, see the [UCX-AUD047 Repeat Policies](#) topic. For each class record which represents a repeat instance, the REPEAT-PLCY and the REPEAT-PTR must be filled in appropriately.

For example, a student took ECON 100 in term 20021 and received a grade of D. The student retook ECON 100 in term 20023 and received a grade of B+. If your institution only counts the most recent instance of a repeated class, the rad_class_dtl records would be filled in as follows:

Course key	Term	Repeat ptr	Repeat plcy
ECON 100	20021	ECON 100	1
ECON 100	20023	ECON 100	1

In the next example, a student took ECON 100 in term 20021 and received a grade of D. The student repeated the course in term 20023 and received a grade of B, but the course key for ECON 100 has changed in that term and is now called ECON 105. If the same rules regarding repeats were followed, the rad_class_dtl records would be filled in as follows:

Course key	Term	Repeat ptr	Repeat plcy
ECON 100	20021	ECON 105	1
ECON 105	20023	ECON 100	1

As you can see in this example, each set of repeated classes must reference each of the members of the set. In this case, ECON 100 references ECON 105 as the repeat instance and ECON 105 references ECON 100 as the repeat instance. If a student repeated the class a third time, the same structure would apply to the third repeat instance.

In the next example, a student took ECON 100 in term 20021 and received a grade of D. The student then repeated the same class in term 20022 under the Business Division as BUS 105 and received a grade of B. The student repeated the class a third time in term 20023 and received a grade of C. The institution has a forgiveness policy that allows the highest grade to be used and all

other instances not counted against the student. The rad_class_dtl records would be filled in as follows:

Course key	Term	Repeat ptr	Repeat plc
ECON 100	20021	BUS 105	(blank)
BUS 105	20022	ECON 100	6
ECON 100	20023	BUS 105	(blank)

In this example, the two ECON 100 courses are marked as repeats of the BUS 105 class. The rad_repeat_plcy for the ECON 100 classes is left blank. BUS 105 is marked as a repeat of ECON 100 and the rad_repeat_plcy is filled in as "6". The BUS 105 class will be used in the audits.

Record layout

Updated: September 30, 2022

The following data from the institution's student records system can be used by Degree Works as part of its audits or by Transfer Equivalency as part of transfer articulation.

Only those records marked as required must be supplied. If optional records are supplied then make sure the required fields in those records are valid. All fields are fixed length. Other fields may be supplied or filled with spaces. The trailing spaces on the end of a record are not needed but each field within the record must be filled with spaces before the next field begins.

Note the following notation used in this document:

POS	is the character position within the file,
LEN	is the length of the field,
R, RS, O, T	R=Required, RS=Required for a Student, O=Optional, T=Transfer Equivalency,
UCX	is the code definition table that governs the field. The comments are intended to assist with understanding important considerations regarding the field.

Each record in the data file contains a 28-byte header. The header is composed of a 10-byte rad_id code assigned by the student system, an 8-byte record type IDENTIFIER (for example, R011PRIM for the rad_primary_mst, R027GOAL for the rad_goal_dtl) and a 2-byte Action-Flag (A=Add, D=Delete, M=Modify). The Modify action is included in the header but is currently only being used for "master" records (for example, rad_primary_mst). The record types are listed in order by header IDENTIFIER (defined at the beginning of this document in the Degree Works Components section).

The lower case field names in each bridge record layout correspond directly to database column names on the associated database table. The upper case field names are most often composite field names that do not correspond directly to database column names, but contain components that are on the database. In some cases the upper case field names are flags or other pieces of data required for the bridge.

Each record has a max size of 1,000 characters. There is no need to add spaces at the end of each record to ensure a length of 1,000. Records smaller than 1,000 characters are valid.

R011PRIM - Primary Record

Updated: March 24, 2023

This record contains information for the rad_primary_mst and rad_student_mst.

The rad_primary_mst is a required table for individuals: students and staff members. However, the rad_student_mst will only be built for students.

Total length = 1000 bytes.

When processing a transfer transcript, the ACTION-FLAG must be set to T. If the rad_primary_mst does not exist for the student ID one will be added. If a rad_primary_mst exists, then it will be updated with the incoming data. Existing data will be overwritten. If a rad_term is included in the Primary Record a rad_student_mst record will be created. If a rad_student_mst already exists the ACTIVE rad_term will be updated regardless of whether it exists or not. Existing data will be overwritten.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		<p>This is a 28-byte field identifying the record as one containing rad_primary_mst data. It is composed of a 10-byte rad_id, an 8-byte IDENTIFIER (R011PRIM for the rad_primary_mst), an 8-byte Active rad_term (set to spaces for individuals who are employees only) and a 2-byte ACTION-FLAG (A=Add, M=Modify, T=Transcript). Example: “00129918..R011PRIM.....A.” (periods represent spaces)</p>
rad_id	29	10	R		<p>This element is the identification number of the individual. Each individual has a unique rad_id code that is attached to all records. This rad_id code is attached to each associated database table. All rad_id codes are validated against the rad_id on the rad_primary_mst.</p>
FILLER	39	4	O		Reserved for future use. Fill with spaces.

Field name	Pos	Len		UCX	Comments
rad_name	43	180	R		This element is the individual's full name. The name should be entered as follows: Last name (comma) (space) First name (space) Middle Name OR Middle Initial. Example: "Smith, John David" or "Smith, John D."
rad_sex	223	2	OT		This element identifies the gender of the individual. Examples: F for Female, M for Male and U for Unknown. Used by Transfer Equivalency only.
rad_birthdate	225	8	OT		Obsolete – fill with spaces.
rad_soc_sec_nbr	233	15	OT		Obsolete – fill with spaces.
rad_term	248	8	RS	UCX-STU016	This element defines the last ACTIVE term for a student and is stored on the rad_student_mst. It should be left BLANK for staff members who are NOT also students. For active (currently enrolled) students, this is the current term. For inactive (graduated, withdrawn, etc.) students, this is the last term for which there was registration activity. Example: 20072 for Spring 2007. This term must match the term on rad-term-dtl.
rad_address1	256	75	OT		Obsolete – fill with spaces.
rad_address2	331	75	OT		Obsolete – fill with spaces.
rad_city	406	50	OT		This element is the name of the city that is part of the address, but this city is only used for searching. Examples: Malvern and Seattle. Used by Transfer Equivalency only.
rad_state	456	3	OT		Obsolete – fill with spaces.
rad_zip	459	30	OT		Obsolete – fill with spaces.
rad_country	489	2	OT	UCX-CFG020	Obsolete – fill with spaces.
rad_phone	491	28	OT		Obsolete – fill with spaces.
rad_email	519	128	O		This element is the individual's email address.

Field name	Pos	Len		UCX	Comments
rad_user_def1	647	12	O		Elements 1-10 store additional data as defined by your site but we recommend using the rad-custom-dtl for special data needs.
rad_user_def2	659	12	O		
rad_user_def3	671	12	O		
rad_user_def4	683	12	O		
rad_user_def5	695	12	O		
rad_user_def6	707	12	O		
rad_user_def7	719	12	O		
rad_user_def8	731	12	O		
rad_user_def9	743	12	O		
rad_user_def10	755	12	O		
FILLER	767	233	O		Reserved for future use.

R026APPL Applicant Record

Updated: March 25, 2022

This record contains information for the rad_applicant_dtl.

This table is optional for Transfer Equivalency. It should be omitted if Transfer Equivalency is not in use. It should be supplied if you want Transfer Equivalency to use the applicant data from your student system instead of asking users to enter the data in Transfer Equivalency. The Bridge may eventually support one record per applicant per degree, but currently only supports one rad_applicant_dtl per applicant.

Total length = 1000 bytes.

When processing a transfer transcript (rad_primary_mst ACTION-FLAG = T), rad_applicant_dtl data may optionally be sent as part of the transcript. Whether or not sent to the bridge, the rad_applicant_dtl will be added if one does not already exist. If a rad_applicant_dtl exists and data is sent to the bridge, then it is updated with the incoming data only if the existing fields are blank. For example, MJMN1 is updated from the incoming transcript data only if MJMN1 on the existing rad_applicant_dtl is blank.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing rad_applicant_dtl data. It is composed of 10-byte rad_id number, an 8-byte IDENTIFIER (R026APPL for

Field name	Pos	Len		UCX	Comments
					the rad_applcnt_dtl), an 8-byte rad_term and a 2-byte ACTION-FLAG (A=Add, D=Delete, M=Modify). Example: “00129918..R026APPL20072...A.” (periods represent spaces).
rad_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
FILLER	39	4	O		Reserved for future use. Fill with spaces.
rad_school	43	12	R	UCX-STU350	This element defines the school (a.k.a. campus) to which the student has applied. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool).
rad_deg_interest	55	2	O	UCX-STU564	This element defines the type of degree interest manifested by the applicant. Examples: DS for Degree Seeking, ND for Non-Degree Seeking.
rad_degree_code	57	12	R	UCX-STU307	This element contains the code that identifies the student's intended degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.
rad_mjmn1	69	12	O	UCX-STU023	This element is used to store the first intended major for an applicant. Examples: ENGL for English Major, MATH for Math Major, PHYS for Physics Major.
rad_mjmn1_clg	81	6	O	UCX-STU560	This element defines the college that offers the first major. The college is a further sub-division of the school. Examples: EG for College of Engineering, LA for College of Liberal Arts.

Field name	Pos	Len		UCX	Comments
rad_mjmn2	87	12	O	UCX-STU023	This element is used to store the second intended major for an applicant. Examples: ENGL for English Major, MATH for Math Major, PHYS for Physics Major.
rad_mjmn2_clg	99	6	O	UCX-STU560	This element defines the college that offers the second major. The college is a further sub-division of the school. Examples: EG for College of Engineering, LA for College of Liberal Arts.
rad_matric_term	105	8	R	UCX-STU016	This element indicates the matriculation term, the term of first enrollment by the applicant.
rad_catalog_yr	113	12	R	UCX-STU035	This element defines the catalog year in effect for the student's degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the student's progress towards completing the degree. Examples: 20042006, 20062008.
rad_user_def1	125	12	O		This element stores additional data as defined by your site. The user-defined fields may be used as additional conditions for transfer articulation.
rad_user_def2	137	12	O		This element stores additional data as defined by your site.
rad_user_def3	149	12	O		This element stores additional data as defined by your site.
rad_user_def4	161	12	O		This element stores additional data as defined by your site.
rad_user_def5	173	12	O		This element stores additional data as defined by your site.
rad_user_def6	185	12	O		This element stores additional data as defined by your site.
rad_user_def7	197	12	O		This element stores additional data as defined by your site.
rad_user_def8	209	12	O		This element stores additional data as defined by your site.

Field name	Pos	Len		UCX	Comments
rad_user_def9	221	12	O		This element stores additional data as defined by your site.
rad_user_def10	233	12	O		This element stores additional data as defined by your site.
FILLER	245	756	O		Reserved for future use. Fill with spaces.

R027GOAL - Goal Record

Updated: September 29, 2023

The goal record records the degrees the student is currently working on.

This record contains Degree related information for the rad_goal_dtl table. Each unique School/Degree combination required in Degree Works for a given student must have a rad_goal_dtl created. Also, at least one corresponding rad_goaldata_dtl record with the same ID, school, and degree must be created. For Degree Works, an unlimited number of rad_goal_dtls may be created for a given student ID code, but the School/Degree must be unique.

The order of the goal records for students with multiple degree goals determines the order in which they will display in the drop-down at the top of the dashboard. For example, if a student has a goal for both BA and MBA, and the BA record is bridged before the MBA goal record, the BA degree will appear first in the degree drop-down.

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing rad_goal_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R027GOAL for the rad_goal_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R027GOAL20072...A." (periods represent spaces).
rad_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.

Field name	Pos	Len		UCX	Comments
FILLER	39	4	O		Reserved for future use. Fill with spaces.
rad_school	43	12	R	UCX-STU350	This element defines the school in which the student is enrolled. Examples: UG for Undergraduate School, GR for Graduate School, LW for Law School. This code must match the bridged School code on the rad_class_dtl. The rad_school and rad_degree_code below are also used to match the rad_school and rad_degree_code on the rad_goaldata_dtls (R033GDTA) to retrieve the associated Field of Study records (Majors, Minors, Concentrations, etc.).
rad_degree_code	55	12	R	UCX-STU307	This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts. The rad_degree_code and rad_school above are also used to match the rad_school and rad_degree_code on the rad_goaldata_dtls (R033GDTA) to retrieve the associated Field of Study records (Majors, Minors, Concentrations, etc.).
rad_catalog_yr	67	12	R	UCX-STU035	This element defines the catalog year in effect for the student's degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the student's progress towards completing the degree.
rad_stu_level	79	6	O	UCX-STU305	This element is the class level of the student within the university. Examples: "01" or "FR" for Freshman, "04" or "SR" for Senior, "05" for Graduate Student, "06" for PhD Student.

Field name	Pos	Len		UCX	Comments
rad_term	85	8	R	UCX-STU016	Obsolete – you can load with spaces.
rad_degree_src	93	2	O		“S” or “A” with a trailing space. A = applicant. S = non-applicant student. If left blank, the default is student. When set to “A”, the degree is considered to be an applicant’s degree.

R033GDTA - Goal Data Record

Updated: March 24, 2023

This record contains information for the rad_goaldata_dtl table. This is a required table for Degree Works but not for Transfer Equivalency.

The rad_goaldata_dtl records must have a corresponding rad_goal_dtl record with the same ID, school, and degree. An unlimited number of rad_goaldata_dtls may be created for a given Degree/Goal (stored on the associated rad_goal_dtl record).

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing rad_goaldata_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R033GDTA for the rad_goaldata_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: 00129918..R033GDTA20072...A. (periods represent spaces).
rad_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
FILLER	39	4	O		Reserved for future use. Fill with spaces.

Field name	Pos	Len		UCX	Comments
rad_school	43	12	R	UCX-STU350	This element defines the school in which the student is enrolled. Examples: UG for UndergraduateSchool, GR for GraduateSchool, LW for LawSchool. The rad_school and rad_degree_code below are also used to match the rad_school and rad_degree_code on the rad_goaldata_dtls (R027GOAL) to retrieve the associated Degree record.
rad_degree_code	55	12	R	UCX-STU307	This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts. The rad_degree_code and rad_school above are also used to match the rad_school and rad_degree_code on the rad_goal_dtls (R027GOAL) to retrieve the associated Degree record.
rad_catalog_yr	67	12	R	UCX-STU035	This element defines the catalog year in effect for the student's degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the student's progress towards completing the degree.
rad_goal_code	79	12	R		<p>This element is the code associated with the type of record involved in a degree. The standard values are:</p> <p>ADVISOR – Advisor ID Codes</p> <p>COLLEGE - College Codes (UCX_STU560)</p> <p>CONC - Concentration Codes (UCX_STU564)</p>

Field name	Pos	Len	UCX	Comments
				LIBL - Liberal Learning Codes (UCX_STU324)
				MAJOR - Major Codes (UCX_STU023)
				MINOR - Minor Codes (UCX_STU024)
				PROGRAM - Program Codes (UCX_STU3167)
				SPEC – Specialization Codes (UCX_STU323)
				STUSTATUS – Student Status Codes (UCX_STU306)
				If the Degree to Goal conversion is used to convert data from the R031DEGR record to the R027GOAL and R033GDTA records then these are the standard rad_goal_codes that will be created for the associated data.
rad_goal_value	91	12	R	This element is the actual value recorded for a given student for this goal code. If the goal code is MAJOR, the goal value might be ENGL for English, HIST for History, and so on.
rad_goal_seq	103	4	R	This element is the sequence number associated with the Goal Code and Goal Value. They do NOT have to be unique. For example, if you have 1 MAJOR, 1 MINOR, 1 CONC, and 1 ADVISOR that are all associated, they would have the same sequence number (for example, 0001). If you have 2 MAJORS, 2 MINORS, and 1 ADVISOR associated with a Degree Goal, this is how they would look:

Field name	Pos	Len		UCX	Comments
					Goal Code=ADVISOR, Goal Value=12345, Goal Seq = 0001
					Goal Code=MAJOR, Goal Value=MATH, Goal Seq=0001
					Goal Code=MAJOR, Goal Value=CS, Goal Seq = 0002
					Goal Code=MINOR, Goal Value=PHYS, Goal Seq = 0001
					Goal Code=MINOR, Goal Value=MIS, Goal Seq = 0002
rad_attach_code	107	12	R		This element is the code to which this goal record is attached. The standard values are: ADVISOR, COLLEGE, CONC, LIBL, MAJOR, MINOR, PROGRAM, SPEC, and STUSTATUS. For example, a goaldatal record for CONC=GRAPH may be attached to a MAJOR=ART goaldatal record. In this example, this field will contain MAJOR, which means this concentration is tied to this specific major. When block selection occurs, the MAJOR=ART block that contains a CONC secondary tag will be used only if the right concentration is attached to this major. For block selection, you are saying that it is not enough that the student has this particular concentration, the concentration must be attached to this particular major.
rad_attach_value	119	12	R		This element is the value to which this goal record is attached. For example, a goaldatal record for CONC=GRAPH may be attached to a MAJOR=ART goaldatal record. In this

Field name	Pos	Len		UCX	Comments
	example, this field will contain ART.				
FILLER	131	894	O	Reserved for future use.	

R062TERM - Term Record

Updated: September 30, 2022

The term record contains cumulative data such as the GPA and total credits.

Although this record has TERM in its name you must create only one record for each of the student's schools. For example, create one record for their undergraduate degree and another for their graduate degree. The term value included in this header is ignored.

This record contains information for the rad_term_dtl. This is an optional table. The values bridged here are used in the GPA Calculator and can be shown in the dashboard's student context area to help with advising.

The "user" fields below are deprecated. It is better to bridge other student data in the CUST record.

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R	This is a 28-byte field identifying the record as one containing rad_term_dtl data. It is composed of a 10-byte rad_id, an 8-byte IDENTIFIER (R062TERM for the rad_term_dtl), an 8-byte rad_term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R062TERM20072...A." (periods represent spaces)..	
rad_id	29	10	R	This element is the universal rad_id code assigned to a student at the time of admission. The same rad_id will remain with an individual throughout association with the institution. This rad_id is validated on the rad_primary_mst.	
FILLER	39	4	O	Reserved for future use. Fill with spaces.	

Field name	Pos	Len		UCX	Comments
rad_school	43	12	R	UCX-STU350	This element defines the school in which the student is enrolled. Examples: UG for Undergraduate School, GR for Graduate School, LW for Law School. This code must match the bridged School code on the rad_class_dtl.
rad_deg_interest	55	2	O		Obsolete.
rad_cum_tot_earn	57	7	O		This element is the cumulative total credits earned. It is stored in the format 9999v999. This data should be from the student's last graded term. It is equal to CUM-TR-EARN + CUM-CR-EARN.
rad_cum_tr_earn	64	7	O		This element is the cumulative transfer credits earned. It is stored in the format 9999v999. This data should be from the student's last graded term.
rad_cum_cr_earn	71	7	O		This element is the cumulative credits earned. It is stored in the format 9999v999. This data should be from the student's last graded term.
rad_cum_gr_att	78	7	O		This element is the cumulative graded credits attempted. It is stored in the format 9999v999. This data should be from the student's last graded term.
rad_cum_gr_pts	85	8	O		This element is the cumulative number of grade points earned. It is stored in the format 99999v999. This data should be from the student's last graded term.
rad_cum_gpa	93	6	O		This element is the cumulative grade point average. It is stored in the format 999v999. This data should be from the student's last graded term.
rad_term	99	8	R	UCX-STU016	Obsolete – you can load with spaces.
rad_user_gpa1	107	6	O		This element stores additional data as defined by your site

Field name	Pos	Len	UCX	Comments
				(must be numeric, format 999v999).
rad_user_gpa2	113	6	O	This element stores additional data as defined by your site (must be numeric, format 999v999).
rad_user_credit1	119	6	O	This element stores additional data as defined by your site (must be numeric, format 999v999).
rad_user_credit2	125	6	O	This element stores additional data as defined by your site (must be numeric, format 999v999).
rad_user_def1	131	12	O	This element stores additional data as defined by your site.
rad_user_def2	143	12	O	This element stores additional data as defined by your site.
rad_user_def3	155	12	O	This element stores additional data as defined by your site.
rad_user_def4	167	12	O	This element stores additional data as defined by your site.
rad_user_def5	179	12	O	This element stores additional data as defined by your site.
rad_user_def6	191	12	O	This element stores additional data as defined by your site.

R071CLAS - Class Record

Updated: September 30, 2022

The class record houses the classes the student has taken, is taking, or plans to take at your institution.

This record contains information for the rad_class_dtl. This is a required table for Degree Works, but it is optional for Transfer Equivalency.

The user fields below are deprecated. It is best to bridge additional class information on the ATTR record.

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing rad_class_dtl data. It is composed of a 10-byte rad_id, an 8-byte IDENTIFIER (R071CLAS for the rad_class_dtl), an 8-byte rad_term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R071CLAS20072...A." (periods represent spaces).
rad_id	29	10	R		This element is the universal rad_id code assigned to a student at the time of admission. The same rad_id will remain with an individual throughout association with the institution. This rad_id is validated on the rad_primary_mst.
FILLER	39	4	O		Reserved for future use. Fill with spaces.
COURSE DISCIPLINE	43	12	R	UCX-STU352	This element is the Course Discipline Code that is validated against UCX-STU352. It is the Course Discipline portion of the rad_course_key, bytes 1-12. Example: MATH for course key MATH 351.
COURSE NUMBER	55	12	R		This element is the Course Number portion of the rad_course_key, bytes 13-24. Example: 351 for course key MATH 351.
rad_section	67	12	O		This element is used to define multiple classes with the same course key. Examples: "ART 101C" for "ART 101", Section "C", "ART 101F" for ART 101, Section "F".
rad_course_title	79	50	R		This element contains the course title that will display on the Degree Works audit. (it may differ from that in the rad_course_mst). Double-quotes "...." in the title should

Field name	Pos	Len		UCX	Comments
					be removed as they will cause problems when the audit is displayed on the web. Single-quotes are allowed.
rad_school	129	12	R	UCX-STU350	This element is the school code within which the student registered for the class. Examples: UG for Undergraduate School, LW for Law School, GR for Graduate School. This code must match the bridged School code on the rad_goal_dtl.
rad_division	141	12	O		This element is the division code associated with a course. Examples: EG for Engineering, LA for Liberal Arts, HU for Humanities.
rad_dept	153	12	O		This element contains a code that defines the department of study. Examples: ENGL for the English Department, MATH for the Math Department, PHYS for the Physics Department.
rad_deg_interest	165	2	O		Not used
rad_class_type	167	2	O		This element contains codes that identify the type of class. It can be treated as an additional user defined field that stores additional data for a class as defined by your site.
rad_acad_votech	169	6	O		Not used.
rad_audit_flag	175	2	R		This element is a Y/N flag used to identify classes that have been audited. Set to Y if the class was taken as an audit. Otherwise set to N.
rad_insuff_flag	177	2	R		This element is a Y/N flag used to identify classes that were given an insufficient grade. Set to Y if you want to force the class to the Insufficient section of the audit report.
rad_inprog_flag	179	2	R		This element is a Y/N flag used to identify classes that are currently "In-Progress" (term has not yet been completed

Field name	Pos	Len		UCX	Comments
					and graded). Set to Y if the class is still “In-Progress”. Preregistered classes are determined by comparing the class term with the active-term on the student record; terms in the future are considered preregistered. However, you may load “P” into this flag to indicate a preregistered class in case for some students the term comparison logic will not work (like for incoming freshmen). Otherwise set to N.
rad_withdr_flag	181	2		R	This element is a Y/N flag used to identify classes that were withdrawn. Set to Y if the student withdrew from the class. Otherwise set to N.
rad_incomp_flag	183	2		R	This element is a Y/N flag used to identify classes that were not completed. Examples: Grades of I or X. Set to Y if the class was not successfully completed. Otherwise set to N.
rad_pass_flag	185	2		R	This element is a Y/N flag used to indicate that FINAL-GRADE is a passing grade. Set to Y if the grade was “Passing”. Otherwise set to N. This field must be set to Y for In-Progress and Pre-registered classes. (If set to N, the INCOMP-FLAG must be set to N for failed classes to go to the Insufficient section of the audit report; setting the INCOMP-FLAG to Y will result in the class will applying normally to rules.)
rad_credits	187	7		R	This element contains the number of credits the class was worth when taken by the student. The format is 9999v999. Example: 4 credits would be “0004000”.
rad_credits_earn	194	7		R	This element is used to store the number of credits the student earned for successfully

Field name	Pos	Len		UCX	Comments
					completing the class. The format is 9999v999. If the student successfully passed a 3-credit class, then this field should contain "0003000". If the credits were not earned, then load with "0000000".
rad_gpa_credits	201	7	R		This element is used to store the number of graded credits attempted to be used in GPA calculations. If 3 graded credits were attempted, then load with "0003000". If the class credits are not graded and are not to be used in GPA calculations, then load with "0000000".
rad_grade_points	208	7	R		This element is the number of grade points earned for a class. It is stored in the format 9999v999.
rad_credit_type	215	2	O	UCX-STU355	This element is used to define the type of credit a course will earn. Examples: AC for Academic Credit, AP for Advanced Placement Credit, CE for Credit By Exam Credit.
rad_class_status	217	2	O		This element is used to indicate the status of the class. A for Added, WD for Withdrawn. You may also want to populate a status indicating that the class was repeated. This code can then be listed in the dash.audit.repeat.codes setting. When you then enable the UCX-RPT036 Show Repeated flag you will see these repeated classes showing a repeated indicator in the Responsive Dashboard worksheet.
rad_grade_type	219	6	O	UCX-STU356	This element is used to define the type of grading used for the class for the particular student. Examples: AF for A-F Grading, PF for Pass/Fail Grading, SU for Satisfactory/Unsatisfactory.

Field name	Pos	Len		UCX	Comments
rad_pass_fail	225	2	R		This element is a Y/N flag indicating whether or not this class was taken as Pass/Fail. Set to Y if this class was taken as Pass/Fail. Otherwise set to N.
rad_repeat_ptr	227	24	O		This field contains the COURSE-KEY (Discipline + Course Number) of the class being repeated (usually from an earlier term or semester). If the class is not being repeated, it should be left blank. If the class is being repeated, this field should be filled in for all instances of the class. See COURSE DISCIPLINE and COURSE NUMBER (rad_course_key) for guidance on formatting this field.
rad_repeat_plcy	251	2	O	UCX-AUD047	If the class is involved in a repeat transaction, then load the appropriate repeat policy from UCX-AUD047. See the UCX-AUD047 documentation for information on the repeat policies.
rad_session	253	12	O		This element may be used to subdivide terms into smaller sessions. Examples: S1 might be summer session 1, S2 might be summer session 2.
rad_location	265	12	O		This element defines the location where the class is being taught. Examples: MC Main Campus, SA Satellite Campus.
rad_final_grade	277	6	R	UCX-STU385	This element stores the final grade received for the class. Examples: A, B+, C-, F, I, CR, WF. This grade value is not used in any MINGRADE calculations. It is for display purposes only.
rad_final_gr_num	283	7	R		This element is the final grade represented as a numeric value. The format is 9999v999. If the class is worth 3 grade

Field name	Pos	Len		UCX	Comments
					points, then load "0003000". If the class is worth 2.5 grade points, then load "0002500".
rad_instr1_id	290	10	O		Not used.
rad_term	300	8	R	UCX-STU016	This element stores the term associated with the class. Example: 20242 for Spring 2024.
rad_stu_level	308	6	O		Not used.
rad_user_def1	314	12	O		This element stores additional data as defined by your site.
rad_user_def2	326	12	O		This element stores additional data as defined by your site.
rad_user_def3	340	12	O		This element stores additional data as defined by your site.
rad_user_def4	350	12	O		This element stores additional data as defined by your site.
rad_user_def5	362	12	O		This element stores additional data as defined by your site.
rad_user_def6	374	12	O		This element stores additional data as defined by your site.
rad_user_def7	386	12	O		This element stores additional data as defined by your site.
rad_user_def8	398	12	O		This element stores additional data as defined by your site.
rad_user_def9	410	12	O		This element stores additional data as defined by your site.
rad_user_def10	422	12	O		This element stores additional data as defined by your site.
rad_attr_key	434	20	O		This together with the rad_id links to the rad_attr_dtl – for class attributes
rad_crn	454	10	O		Course Reference Number. This ends up on the rad_crn field on the rad_result_dtl.

Classes falling into the Insufficient Section

Here are the flag combinations that will affect a class falling into the Insufficient Section of an audit:

INSUFF FLAG	INCOMP FLAG	PASS FLAG	Translation	Result
N	N	N	The class is complete and the student did not receive a passing grade.	It will fall into the Insufficient Section.
N	Y	N	The class is incomplete and the student is not receiving a passing grade.	If UCX-CFG020 DAP14 INCOMPLETE-APPLY-FLAG is set to N the class will fall into the Insufficient Section. If UCX-CFG020 DAP14 INCOMPLETE-APPLY-FLAG is set to Y, the class will apply normally to rules and not fall into the Insufficient Section.
N	N	Y	The class is complete and the student received a passing grade.	It will apply to rules and not fall into the Insufficient Section.
N	Y	Y	The class is incomplete and the student is receiving a passing grade.	If UCX-CFG020 DAP14 INCOMPLETE-APPLY-FLAG is set to N the class will fall into the Insufficient Section. If UCX-CFG020 DAP14 INCOMPLETE-APPLY-FLAG is set to Y, the class will apply normally to rules and not fall into the Insufficient Section.
Y	Y or N	Y or N	Regardless of any other factors, this class is intended to be forced into the Insufficient Section.	Degree Works will not consider the INCOMP-FLAG or PASS-FLAG. When INSUFF-FLAG is set to Y it will be immediately forced into the Insufficient Section.

R083TRAN - Transfer Class Record

Updated: September 29, 2023

The transfer record houses classes transferred to your institution from another school.

This record contains information for the rad_transfer_dtl. This is an optional table. If “credit by exam” is mapped to a specific class send a rad_transfer_dtl record for that class and put the exam code in the TR-CRSE-KEY. If your student system does not track individual transfer classes but does track total transfer credits, you can send one transfer record with a fake rad_tr_crse_key and the total number of credits earned. Doing this will let Degree Works display the total transfer credits in the Fallthrough section of the audit report. Total length = 1000 bytes.

Do not use this record for transfer transcripts. Send transfer transcript courses to Transfer Equivalency using dap_transfer_dtl (R200TREQ).

The user fields below are deprecated. It is best to bridge additional class information on the ATTR record.

Field name	Pos	Len	UCX	Comments
HEADER	1	28	R	This is a 28-byte field identifying the record as one containing rad_transfer_dtl data. It is composed of a 10-byte rad_id, an 8-byte IDENTIFIER (R083TRAN for the rad_transfer_dtl), an 8-byte rad_term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R083TRAN20072...A." (the periods represent spaces).
rad_id	29	10	R	This element is the universal rad_id code assigned to a student at the time of admission. The same rad_id will remain with an individual throughout association with the institution. This rad_id is validated on the rad_primary_mst.
FILLER	39	4	O	Reserved for future use. Fill with spaces.
rad_school	43	12	R	UCX-STU350 This element is the school code within which the transfer class is to be associated. Examples: UG for Undergraduate School, LW for Law School, GR for Graduate School). This code must match the bridged School code on the rad_goal_dtl.
rad_tr_ets	55	20	O	This element identifies the transfer college where this transfer class was taken.
rad_tr_name	75	100	O	This element identifies the school name of the transfer college. If this school name is required to print on audit worksheets, then load this data.

Field name	Pos	Len		UCX	Comments
rad_tr_crse_key	175	24	O		<p>This element is the TR-COURSE-KEY taken at the transfer institution. It can be in any format although most institutions use a discipline code followed by some numeric scheme.</p> <p>Examples: "MATH 210", "ART 115", "SOC 211".</p> <p>It appears on audit worksheets.</p>
rad_tr_course	199	50	O		<p>This element stores the title of the course as it was identified at the transfer institution. It is used to print on audit worksheets.</p>
COURSE DISCIPLINE	249	12	R	UCX-STU352	<p>This element defines the COURSE-DISC to be used by Degree Works, validated against UCX-STU352. It should be the COURSE-DISC used at your institution (not the one used at a transfer institution). It is the Course Discipline of the rad_course_key, bytes 1-12.</p>
COURSE NUMBER	261	12	R		<p>This element defines the COURSE NUMBER to be used by Degree Works. It should be the COURSE NUMBER used at your institution (not the one used at a transfer institution). If a direct articulation cannot be made from the rad_tr_course_key to your rad_course_key, then an "@" sign may be used in place of any part of the Course Number. Examples: "@", "3@". It is the Course Number of the rad_course_key, bytes 13-24.</p>
rad_section	273	12	O		<p>This element is used to define multiple classes with the same course key. Examples: "ART 101C" for ART 101, Section "C", "ART 101F" for ART 101, Section "F").</p>

Field name	Pos	Len		UCX	Comments
rad_course_title	285	50	R		This element stores the course title at your institution (not the transfer institution). Double-quotes in the title should be removed as they will cause problems when the audit is displayed on the web, single-quotes are allowed.
rad_division	335	12	O		This element is the division code associated with a course. Examples: EG for Engineering, LA for Liberal Arts, HU for Humanities).
rad_dept	347	12	O		This element contains a code that defines the department of study. Examples: ENGL for the English Department, MATH for the Math Department, PHYS for the Physics Department.
rad_class_type	359	2	O		This element contains codes that identify the type of class. It can be treated as an additional user defined field that stores additional data for a class as defined by your site.
rad_acad_votech	361	6	O		Obsolete – fill with spaces.
rad_audit_flag	367	2	R		This element is a Y/N flag used to identify classes that have been audited. Set to Y if the class was taken as an audit. Otherwise set to N.
rad_insuff_flag	369	2	R		This element is a Y/N flag used to identify classes that were given an insufficient grade. Set to Y if the grade received in the class was insufficient. Examples: F, UN. Otherwise set to N.
rad_inprog_flag	371	2	R		This element is a Y/N flag used to identify classes that are currently “In-Progress” (term has not yet been completed and graded). Set to Y if the class is still “In-Progress”. Otherwise set to N.
rad_withdr_flag	373	2	R		This element is a Y/N flag used to identify classes that were

Field name	Pos	Len	UCX	Comments
				withdrawn. Set to Y if the student withdrew from the class. Otherwise set to N.
rad_incomp_flag	375	2	R	This element is a Y/N flag used to identify classes that were not completed. Examples: Grades of "I" or "X" often indicate classes that were not completed and would have this flag set to Y. Set to Y if the class was not successfully completed. Otherwise set to N.
rad_cr_exam_flag	377	2	O	This element is a Y/N flag used to indicate that this class represents a Credit by Exam. Set to Y if this transfer class represents a Credit by Exam. Otherwise set to N.
rad_pass_flag	379	2	R	This element is a Y/N flag used to indicate that rad_final_grade is a passing grade. Set to Y if the grade was "Passing". Otherwise set to N.
rad_credits	381	7	R	This element contains the number of institutional credits given to this transfer class. If the calendar of the transfer institution does not match your institution's calendar, then make the appropriate conversion specified in the CALENDAR field above (quarter to semester hours or semester to quarter credits). The format is 9999v999. Example: 4 credits would be "0004000".
rad_credits_earn	388	7	R	This element is used to store the number of credits the student earned for successfully completing the transfer class. The format is 9999v999. If the student successfully passed a 3-credit class, then this field should contain "0003000". If the credits were not earned, then load with "0000000".

Field name	Pos	Len		UCX	Comments
rad_gpa_credits	395	7	R		This element is used to store the number of graded credits attempted to be used in GPA calculations. If 3 graded credits were attempted, then load with "0003000". If the class credits are not graded and are not to be used in GPA calculations, then load with "0000000".
rad_grade_points	402	7	R		This element is the number of grade points earned for a class. It is stored in the format 9999v999. If the student earned 12 grade points then "0012000" is in this field.
rad_credit_type	409	2	O	UCX-STU355	This element is used to define the type of credit a course will earn. Examples: AC for Academic Credit, AP for Advanced Placement Credit, CE for Credit By Exam Credit, TR for Transfer Credit).
rad_class_status	411	2	O	UCX-STU380	This element is used to indicate the status of the class. "A" for Added, "WD" for Withdrawn. You may also want to populate a status indicating that the class was repeated. This code can then be listed in the dash.audit.repeat.codes setting. When you then enable the UCX-RPT036 Show Repeated flag you will see these repeated classes showing a repeated indicator in the Responsive Dashboard worksheet.
rad_grade_type	413	6	O	UCX-STU356	This element is used to define the type of grading used for the class for the particular student. Examples: AF for A-F Grading, PF for Pass/Fail Grading, SU for Satisfactory/Unsatisfactory.
rad_pass_fail	419	2	R		This element is a Y/N flag indicating whether or not this class was taken as Pass/Fail. Set to Y if this class was taken

Field name	Pos	Len		UCX	Comments
					as Pass/Fail. Otherwise set to N.
rad_repeat_ptr	421	24	O		This field contains the COURSE-KEY (Discipline + Course Number) of the class being repeated (usually from an earlier term or semester). If the class is not being repeated, it should be left blank. If the class is being repeated, this field should be filled in for all instances of the class. See COURSE DISCIPLINE and COURSE NUMBER (rad_course_key) for guidance on formatting this field.
rad_repeat_plcy	445	2	O	UCX-AUD047	If the class is involved in a repeat transaction, then load the appropriate value from UCX-AUD047. See the UCX-AUD047 documentation for information on the repeat policies.
rad_location	447	12	O	UCX-STU576	This element defines the location where the class was taught. Examples: MC for Main Campus, SA for Satellite Campus, TA for Term Abroad).
rad_final_grade	459	6	R		This element stores the final grade received for the class. Examples: A, B+, C-, F, I, CR, WF. This grade value is not used in any MINGRADE calculations. It is for display purposes only.
rad_final_gr_num	465	7	R		This element is the final grade represented as a numeric value. The format is 9v999. If the class is worth 3 grade points, then load "3000". If the class is worth 2.5 grade points, then load "2500". This field is used in the MINGRADE calculations in the requirement blocks.
rad_term	472	8	R	UCX-STU016	This element stores the term associated with the transfer

Field name	Pos	Len	UCX	Comments
				class. Example: 20212 for Spring 2021.
rad_user_def1	480	12	O	This element stores additional data as defined by your site.
rad_user_def2	492	12	O	This element stores additional data as defined by your site.
rad_user_def3	504	12	O	This element stores additional data as defined by your site.
rad_user_def4	516	12	O	This element stores additional data as defined by your site.
rad_user_def5	528	12	O	This element stores additional data as defined by your site.
rad_user_def6	540	12	O	This element stores additional data as defined by your site.
rad_user_def7	552	12	O	This element stores additional data as defined by your site.
rad_user_def8	564	12	O	This element stores additional data as defined by your site.
rad_user_def9	576	12	O	This element stores additional data as defined by your site.
rad_user_def10	588	12	O	This element stores additional data as defined by your site.
rad_attr_key	600	20	O	This together with the rad_id links to the rad_attr_dtl – for class attributes
rad_sis_key	620	10	O	A unique value for each transfer class for this student. This ends up on the rad_crn field on the rad_result_dtl.

R085ATTR - Class Attribute Record

Updated: March 25, 2022

This record contains information for the rad_class_dtl and rad_transfer_dtl. This is an optional table.

You can associate an unlimited number of attributes to a class using this record, the rad_attr_dtl. The rad_id and the rad_attr_key are used to tie these records to a particular class. This same attr_key value must be placed on the rad_class_dtl or rad_transfer_dtl. The attr_key may be any value as long as it is unique for a particular class_dtl/transfer_dtl.

You can bridge a list of attributes or traits associated with a class, such as it was taken as honors, is a writing intensive class, and so on.

You can refer to these attributes in Scribe, using code similar to the following:

```
5 Credits in HIST @ (With Trait = WRITING)
```

In this case TRAIT would be stored in the attr_code and WRITING would be stored in the attr_value.

For Banner schools, all class attributes are stored here with ATTRIBUTE as the attr_code and the 4-byte value in the attr_value field. Banner student attributes are housed in the rad_custom_dtl – not here.

Any code that is loaded here and is to be used in Scribe must be setup in UCX-SCR044 with the Element field set to “ATTR” – telling Degree Works to look for the value in this table.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing rad_attr_dtl data. It is composed of a 10-byte rad_id, an 8-byte IDENTIFIER (R085ATTR for the rad_attr_dtl), an 8-byte rad_term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R085ATTR20072...A." (the periods represent spaces).
rad_id	29	10	R		This element is the universal rad_id code assigned to a student at the time of admission. The same rad_id will remain with an individual throughout association with the institution. This rad_id is validated on the rad_primary_mst.
rad_attr_key	39	20	R		This element ties this record to a particular rad_class_dtl or rad_transfer_dtl record; this key must be on the parent record.
rad_attr_code	59	12	O		This element gives a name to the value; used in a Scribe as (WITH attr_code = attr_value)
rad_attr_value	71	12	O		This element gives a value to the named item; used in a Scribe as (WITH attr_code = attr_value)

The rad_attr_code and rad_attr_value fields can be repeated in pairs up to 39 times on a single BIF record. This means you would use 24 bytes for the first attribute, 24 bytes for the second

attribute and so on. The attributes found on this BIF record are all associated with the same rad_attr_key so this is only useful if you have multiple attributes for the same course. If a course has more than 39 attributes additional BIF records with the same rad_attr_key can be used. Bridging multiple attributes on a single BIF record simply reduces the number of BIF records that need to be processed and will make the process a little faster. Using one BIF record for each attribute on a course is also fine; the use of multiple attributes on a single BIF record is purely optional.

Additional comments for the creation of the rad_attr_key

The important point to remember in the linking of these records (rad_class_dtl/rad_transfer_dtl and rad_attr_dtl) is that the rad_attr_keys in each linked record must match for a given student. They do not have to conform to any set column offset as the full 20-byte rad_attr_keys will be compared and not any of the pieces. Thus, the key components do not have to match the full field sizes (the rad_discipline and rad_course_number are both 12).

In our Banner extract spaces are removed from the rad_attr_key components instead of using their actual size. For example, if the Course Key is “ENGL 210”, the rad_attr_key would be composed of the following:

4-byte Discipline (ENGL) +

3-byte Course number (210) (the max in Banner is a 5-byte Course Number) +

a dash (-) +

a unique sequence number (003).

The rad_attr_key in this case would be “ENGL210-003” which is 11-bytes. The sequence number “003” means that the student has at a minimum two other class attributes (for example, “001” and “002” for other classes (or this same class)).

If the actual Course Number was 5 digits (12345) the rad_attr_key would be ENGL12345-003 which is 13-bytes. So the maximum rad_attr_key for the Banner extract is 13-bytes.

Thus, the data fits within the 20-byte rad_attr_key (for example, ENGL210-003 or ENGL12345-003). The sequence number in the Banner extract is unique for the student for ALL of their class attributes (the max would be 999 attributes which is assumed would be more than enough for a given student). In Banner, a given class can have multiple attributes. Thus, the use of just the Section would not create a unique key. So Section is not used in the rad_attr_key and instead is created using a Sequence Number which is initialized to 1 for each new student. Then that Sequence Number is incremented by 1 for EVERY class attribute found on the rad_class_dtl and rad_transfer_dtl for a given student. This methodology eliminates the possibility of duplicate rad_attr_keys for a given student.

R091TEST - Test Record

Updated: September 30, 2022

This record contains information for the rad_test_dtl table. This is an optional table.

Both Degree Works and Transfer Equivalency use this table. Degree Works can use it as a condition in an IF rule or as student search criteria and tests can also be used for tracking in the planner.

Review your entries in UCX SCR002, SCR003 and RPT046. If any involve test data change element numbers from rad_custom_dtl to rad_test_dtl values:

R323 --> 1292 Data Element – Used in Scribe IF statements

R322 --> 1291 Edit Element – Used to filter the rad_test_dtl

For example, in UCX_SCR002 Custom Codes of 'SAT-VERBAL' and 'SAT-MATH' could be created and used in an IF statement similar to the following:

```

BeginSub
if (SAT-VERBAL >= 550 AND SAT-MATH >= 510) then
    RuleComplete
        Label 1 "SAT Test requirements have been Satisfied"
else
    RuleIncomplete
        Label 2 "You have not completed SAT Test requirements";
EndSub
LABEL "SAT TEST Requirements";

```

Transfer Equivalency uses tests for granting credit by exam.

Total length = 1000 bytes.

Test scores from transfer transcripts should be sent in this format. The bridge will not delete all the tests for a student when processing a transfer transcript (rad_primary_mst ACTION-FLAG = T). The bridge will ascertain if the rad_test_dtl already exists for the rad_test_code. If a record for the rad_test_code does not exist then a rad_test_dtl will be added. If it does exist then the bridge will follow the UCX-STU314 LOAD-TEST-SCHEME. The UCX-STU314 LOAD-TEST-SCHEME sets the policy for updating test scores with transcript test scores. UCX-STU314 LOAD-TEST-SCHEME may be one of the following: A = Add the transcript test score even if it already exists; M = Modify (overwrite the existing record with the incoming data); H = Keep the one with Highest test score; L = Keep the one with Latest test date; N = Never load the transcript test score.

Field name	Pos	Len	UCX	Comments
HEADER	1	28	R	This is a 28-byte field identifying the record as one containing rad_test_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R091TEST for the rad_test_dtl), an 8-byte term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R091TEST.....A." (periods represent spaces). .

Field name	Pos	Len		UCX	Comments
rad_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
FILLER	39	4	O		Reserved for future use. Fill with spaces.
rad_test_code	43	12	R	UCX-STU314	This element is the type of test score recorded. Examples: ATPH for ATP Physics, SATV for SAT Verbal, G29 for GRE Computer Science, AP07 for AP American History.
rad_test_score	55	12	R		This element is the actual score associated with a given test. It is stored in the database as a left-justified value, regardless of it being a number or not.
rad_test_date	67	8	O		This element is the date the test was taken. Format CCYYMMDD
rad_user_def1	75	12	O		This element stores additional data as defined by your site.
rad_user_def2	87	12	O		This element stores additional data as defined by your site.
rad_user_def3	99	12	O		This element stores additional data as defined by your site.
rad_school	111	12	O	UCX-STU350	This element defines the school in which the student is enrolled. Examples: UG for Undergraduate School, GR for Graduate School, LW for Law School. This code must match the bridged School code on the rad_class_dtl.
rad_degree_code	123	12	O	UCX-STU307	This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.

Field name	Pos	Len		UCX	Comments
rad_test_term	135	8	O	UCX-STU016	Term the test was taken. This is used for tracking within the planner.

R100PDEG - Prev Inst Record

Updated: September 30, 2022

This record contains the information for the read_previnst_dtl. This is an optional table.

Degree Works uses this table to record previous attendance at transfer institutions. The data can be used by Degree Works in IF rules. Transfer Equivalency uses this table to know from which colleges a student has transferred.

Total length = 1000 bytes.

When processing a transfer transcript (rad_primary_mst ACTION-FLAG = T), rad_previnst_dtl data may optionally be sent as part of the transcript. Whether or not sent to the bridge, the rad_previnst_dtl will be added if one does not already exist for the transfer school. If a rad_previnst_dtl exists for the transfer school and data is sent to the bridge, then it is updated with the incoming data only if the existing fields are blank. For example, rad_prev_degree is updated from the incoming transcript data only if rad_prev_degree on the existing rad_previnst_dtl is blank.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing rad_previnst_dtl data. It is composed of a 10-byte rad_id, an 8-byte IDENTIFIER (R100PDEG for the rad_previnst_dtl), an 8-byte rad_term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R100PDEG20072...A." (periods represent spaces).
rad_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. The same rad_id will remain with an individual throughout association with the institution. This rad_id is validated on the rad_primary_mst.
FILLER	39	4	O		Reserved for future use. Fill with spaces.

Field name	Pos	Len		UCX	Comments
rad_prev_degree	43	12	R	UCX-STU379	This element contains the code that identifies the student's previous degree. Examples: BA for Bachelor of Arts, AA for Associate of Arts Degree, MS for Master of Science.
rad_prev_date	55	8	O		This element is the date the previous degree was conferred. If no degree was conferred then it is the date of the transcript from this institution. Formatted CCYYMMDD.
rad_prev_ets	63	10	O		This element identifies the previous college of the student. Validated against the rad_ets_mst.
rad_prev_major	73	12	O	UCX-STU382	This element is used to store the major associated with this degree. Examples: ART for Art Major, BUS for Business Major, ENGL for English Major.
rad_confer_flag	85	2	R		This element is a Y/N flag indicating if this institution conferred a degree.
rad_tr_start	87	8	O		This element is the date the student began attending this transfer institution. Formatted CCYYMMDD
rad_tr_stop	95	8	O		This element is the date the student stopped attending this transfer institution. Formatted CCYYMMDD
rad_term	103	8	R	UCX-STU016	This element indicates the term with which the previous degree data is associated.
rad_user_def1	111	12	O		This element stores additional data as defined by your site.
rad_user_def2	123	12	O		This element stores additional data as defined by your site.
rad_user_def3	135	12	O		This element stores additional data as defined by your site.
rad_user_def4	147	12	O		This element stores additional data as defined by your site.

Field name	Pos	Len		UCX	Comments
rad_user_def5	159	12	O		This element stores additional data as defined by your site.
rad_user_def6	171	12	O		This element stores additional data as defined by your site.
rad_user_def7	183	12	O		This element stores additional data as defined by your site.
rad_user_def8	195	12	O		This element stores additional data as defined by your site.
rad_user_def9	207	12	O		This element stores additional data as defined by your site.
rad_user_def10	219	12	O		This element stores additional data as defined by your site.
FILLER	231	770	O		Reserved for future use.

R111NCRS - Non Crse Record

Updated: September 30, 2022

This record contains information for the rad_noncrse_dtl table.

This table contains data used with the NONCOURSE rule such as for a thesis or recital. This is an optional table.

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		<p>This is a 28-byte field identifying the record as one containing rad_noncrse_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R111NCRS for the rad_noncrse_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: “00129918..R111NCRS20072...A.” (periods represent spaces).</p>
rad_id	29	10	R		<p>This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution.</p>

Field name	Pos	Len	UCX	Comments	
				This ID is validated on the rad_primary_mst.	
FILLER	39	4	O	Reserved for future use. Fill with spaces.	
rad_non_crse	43	12	R	This element is the code associated with this NONCOURSE. Examples: THESIS, RECITAL, COMMSERV. These NONCOURSE requirements are used in the NONCOURSE rule and are needed to graduate, but they do not involve an actual class. Sometimes the NONCOURSE rules may expect a certain score or value. If so, then the actual score/value must be stored below.	
rad_non_score	55	12	O	This element is the score or value associated with the NONCOURSE code.	
rad_non_title	67	30	O	This element is the name or title of this NONCOURSE code. Example: "Community Service".	
rad_term	97	8	O	UCS-STU016	This element stores the term associated with this non-course. Example: 20072 for Spring 2007.
rad_school	105	12	O	UCS-STU350	This element defines the school in which the student is enrolled. Examples: UG for Undergraduate School, GR for Graduate School, LW for Law School. This code must match the bridged School code on the rad_class_dtl.
rad_degree_code	117	12	O	UCS-STU307	This element contains the code that identifies the student's degree. Examples: BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.
FILLER	129	872	O	Reserved for future use.	

R121CUST - Custom Record

Updated: September 30, 2022

This record contains information for the rad_custom_dtl table. This table contains data used in IF conditions. This is an optional table.

Total length = 1000 bytes.

Field name	Pos	Len	UCX	Comments
HEADER	1	28	R	<p>This is a 28-byte field identifying the record as one containing rad_custom_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R121CUST for the rad_custom_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete).</p> <p>Example: “00129918..R121CUST20072...A.” (periods represent spaces).</p>
rad_id	29	10	R	<p>This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.</p>
FILLER	39	4	O	Reserved for future use. Fill with spaces.
rad_custom_code	43	12	R	<p>This element is the type of custom code recorded. Examples: ENGLEXAM, RELIGION, LANGCOMP, SWIMTEST. These custom codes are used in the “IF” rule.</p>
rad_custom_value	55	12	R	<p>This element is the actual value recorded for a given student for this custom code. If the score of an ENGLEXAM was a “3”, then an “IF” statement could be created to control what classes the student is required to take (e.g., If VALUE > 3, then the student only needs to take ENGL115, else the student must take ENGL101 and ENGL115).</p>

Field name	Pos	Len		UCX	Comments
Note: NOT correct syntax for "IF" rule.					
rad_custom_title	67	30	O		Not used.
rad_term	97	8	O		Not used.
rad_school	105	12	O	UCX-STU350	<p>This element defines the school in which the student is enrolled. For example, UG for Undergraduate School, GR for Graduate School, LW for LawSchool. This field is optional, but if used, the code must match the bridged school code on the rad_goal_dtl.</p> <p>If this custom data item is valid for all schools the student has, this field should be left empty. Otherwise, the rad_custom_dtl data will be filtered so the appropriate data is passed to the auditor for use with scribed if-statements. Additionally, the Responsive Dashboard and PDF audit student headers will filter this data based on the selected school.</p>
rad_degree_code	117	12	O	UCX-STU307	<p>This element contains the code that identifies the student's degree. For example, BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.</p> <p>If this custom data item is valid for all degrees the student has, this field should be left empty. Otherwise, the rad_custom_dtl data will be filtered so that the appropriate data is passed to the auditor for use with scribed if-statements. Additionally, Responsive Dashboard and PDF audit student headers will filter this data based on the selected degree.</p>

Field name	Pos	Len		UCX	Comments
Copy To User Attrib	129	1	O		Y=copy the custom code and value to the shp_user_attrib table.
FILLER	129	872	O		Reserved for future use.

R126REPT - Report Record

Updated: March 24, 2023

This record contains information for the rad_report_dtl table.

This table contains additional data that you may want to show on the dashboard for advising purposes. The data is not used in evaluating degree audits or by Transfer Equivalency. This is an optional table.

This record is similar to the CUST record, but it allows an 80-character value instead of a 12-character value.

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing rad_report_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R126REPT for the rad_report_dtl), an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R126REPT20072...A." (periods represent spaces).
rad_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
FILLER	39	4	O		Reserved for future use. Fill with spaces.
rad_report_code	43	12	R		This element is the type of report code recorded. Examples: MILBRANCH, MILRANK, COMMENT. These

Field name	Pos	Len		UCX	Comments
					report codes are used so Degree Works knows which rad_report_dtl data to print on the audit report.
rad_report_value	55	80	R		This element is the actual value recorded for a given student for this report code. Examples: Army, Corporal, "Graduated from High School with honors". This value can print on the report.
rad_term	135	8	O		Not used
rad_report_seq	143	4	R		A 4-byte sequential number, indicating the order the rad_report_dtl value should print. Default = "0001". This field is needed to keep free-text comments in order. It is not meaningful for rad_report_codes that have only one occurrence and can be set to "0001" in that case. However, if the same rad_report_code appears multiple times in the same term then the sequence number must be filled in with a number that will keep the data in the order it is to be printed. For example COMMENT occurrence one would have SEQ set to "0001", occurrence two would be "0002".
rad_school	147	12	O	UCX-STU350	This element defines the school in which the student is enrolled. For example, UG for Undergraduate School, GR for Graduate School, LW for LawSchool. This field is optional, but if used, the code must match the bridged school code on the rad_goal_dtl. If this custom data item is valid for all schools the student has, this field should be left empty. Otherwise, the rad_report_dtl data will be filtered so the appropriate data is passed to

Field name	Pos	Len		UCX	Comments
					the auditor for use with scribed if-statements. Additionally, the Responsive Dashboard and PDF audit student headers will filter this data based on the selected school.
rad_degree_code	159	12	O	UCX-STU307	<p>This element contains the code that identifies the student's degree. For example, BS for Bachelor of Science, MA for Master of Arts, BFA for Bachelor of Fine Arts.</p> <p>If this custom data item is valid for all degrees the student has, this field should be left empty. Otherwise, the rad_report_dtl data will be filtered so that the appropriate data is passed to the auditor for use with scribed if-statements. Additionally, Responsive Dashboard and PDF audit student headers will filter this data based on the selected degree.</p>
FILLER	171	830	O		Reserved for future use. Fill with spaces.

R128AID - Financial Aid Record

Updated: March 25, 2022

This record contains information for the rad_aid_dtl table.

This table contains data by the Financial Aid audit. This is an optional table.

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		<p>This is a 28-byte field identifying the record as one containing rad_aid_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R128AID for the rad_aid_dtl),</p>

Field name	Pos	Len		UCX	Comments
					an 8-byte term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R128AID 20072...A." (periods represent spaces).
rad_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
FILLER	39	4	O		Reserved for future use. Fill with spaces.
rad_aid_code	43	12	R		This element is the type of aid code recorded. Examples: AWARD, ENROLLSTATUS, AIDSTATUS. These aid codes are used in the financial aid audit.
rad_aid_value	55	12	R		This element is the actual value recorded for a given student for this aid code.
FILLER	67	934	O		Reserved for future use. Fill with spaces.

R130NOTE - DAP Note Record

Updated: September 29, 2023

This record contains information for the dap_note_dtl table. This is an optional table.

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing dap_note_dtl data. It is composed of a 10-digit ID number, an 8-byte IDENTIFIER (R130NOTE for the dap_note_dtl), an 8-byte term (set to spaces for this table) and a 2-byte ACTION-FLAG

Field name	Pos	Len		UCX	Comments
(A=Add, D=Delete). Example: “00129918..R130NOTE.....A.” (periods represent spaces).					
dap_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the PRIMARY-MST.
dap_school	39	12	O	UCX-STU350	The school associated with the student.
dap_degree	51	12	O	UCX-STU307	The degree code associated with the student.
dap_note_type	63	2	R	UCX-SCR008	The type of note. The type can indicate internal, public, advisor, etc.
dap_note_status	65	2	R		Status of a note. Valid values are: <ul style="list-style-type: none"> • PA - Petition Approved • PP - Petition Applied • PR - Petition Rejected • PW - Petition Awaiting • OK - Normal note
dap_note_who	67	10	O		The ID of who modified the note.
dap_mod_id	77	10	O		The note IDENTIFIER from the student record system. Use “SRS”.
dap_mod_date	87	8	O		Date the note was last modified in the SRS
dap_note_num	95	4	R		A number associated with the corresponding NOTE-TXT-DTL.
FILLER	99	902	O		Reserved for future use. Fill with spaces.

R140NTXT - DAP Note Text Record

Updated: March 25, 2022

This record contains information for the dap_note_txt_dtl table. This is an optional table.

Total length = 1000 bytes.

Field name	Pos	Len	UCX	Comments
HEADER	1	28	R	This is a 28-byte field identifying the record as one containing dap_note_txt_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R140NTXT for the RAD-NOTE-TXT-DTL), an 8-byte term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R140NTXT.....A." (periods represent spaces).
dap_id	29	10	R	This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
dapd_note_text	39	72	R	The written text of a note about a student.
dap_note_num	111	4	R	A number associated with the corresponding dap_note_dtl.
dap_note_seq	115	4	R	The sequence number of this particular dap_note_txt_dtl.
FILLER	119	882	O	Reserved for future use. Fill with spaces.

R171SHPU - SHP User Record

Updated: September 29, 2023

This is a required table for any user that wishes to access Degree Works.

However, it should only be included in the bridge data file if the access for an individual is being added or changed. The SHP User Records may be bridged by themselves as a separate data file

after the student has been bridged/created or be these records may be included with the student data.

This record is used to build a shp_user_mst record with an individual's access ID, RAD ID, access code (logon password), and user class. The SHPCFG file is normally used to define the assignment of valid groups and valid keys controlling access to various Degree Works functions. However, they can be included in this record if so desired. For more information, see the [Groups](#) topic.

Degree Works. Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing shp_user_mst data. It is composed of an individual's 10-byte ID code (validated on the rad_primary_mst), an 8-byte IDENTIFIER (R171SHPU for the shp_user_mst), an 8-byte term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete, M=Modify). For example: 123456....R171SHPU.....A.
shp_access_id	29	14	R		The access ID is equivalent to a user or campus ID. It may be the same as the rad_id or be an email name or the Degree Works administrator can assign it. This is the logon ID for Degree Works.
shp_id	43	10	R		This field contains the rad_id code from the student system (must be valid on the rad_primary_mst).
shp_access_code	53	64	R		This access code is the user's logon password to Degree Works.
shp_filter	117	4	R	UCX-AUD012	The user's classification that determines their level of access to Degree Works. The most common values are ADV for Advisors, ADVX for Advisors without access to Exceptions, REG for the Registrar, and STU for students. For a complete list of valid User Classes, see the

Field name	Pos	Len		UCX	Comments
UCX-AUD012 User Class Codes topic.					
shp_group_list	121	160	O		The core.security.rules.shpcfg setting is normally used to define the assignment of valid groups controlling access to various Degree Works functions. However, if included here, these groups will be added in addition to the standard groups assigned based on the User Class in the setting. If the shp_user_mst already exist and has one or more groups already assigned the groups on this record will be ignored. You cannot overwrite the existing groups. This is an array of 20 8-byte group codes. To review the valid groups for Degree Works, see the Groups topic.
shp_key_list	281	400	O		The core.security.rules.shpcfg setting is normally used to define the assignment of valid keys controlling access to various Degree Works functions. However, if included here, these keys will be added in addition to the standard keys assigned based on the User Class in the setting. If the shp_user_mst already exist and has one or more keys already assigned the keys on this record will be ignored. You cannot overwrite the existing keys. This is an array of 50 8-byte keys. To review the valid keys for Degree Works, see the Services topic.
shp_sso_id	681	128	O		Stores the user's ID for single sign-on authentication. For example, the ID used in CAS authentication.
shp_finder_id	809	128	O		Stores the common system ID used by Transfer Finder.

If you want to change a user's Access ID, you must send a SHP record with the old Access ID and set the modify flag to D (delete) followed by another record with the new Access ID and the modify flag set to A (add) in a separate file that does not include any other BIF records for that user. Including the delete R171SHPU record with other BIF records will cause a 4060 error in the bridge.

R180SWAP - Swap ID Record

Updated: March 25, 2022

This record contains information for the rad_swap_id_dtl table.

Send this record only when an ID code for a student, staff, or applicant must be changed to a new ID code.

Total length = 1000 bytes.

Field name	Pos	Len	UCX	Comments
HEADER	1	28	R	This is a 28-byte field identifying the record as one containing rad_swap_id_dtl data. It is composed of the old 10-byte ID number*, an 8-byte IDENTIFIER (R180SWAP for the rad_swap_id_dtl), an 8-byte term (set to spaces for this record) and a 2-byte ACTION-FLAG (set to spaces for this record). Example: "00129918..R180SWAP.....A." (periods represent spaces).
rad_id_new	29	10	R	This is the new ID code assigned to the student. This element is the universal ID code assigned to a student at the time of admission. The same ID will remain with an individual throughout association with the institution. This ID is validated on the rad_primary_mst.
FILLER	39	4	O	Reserved for future use. Fill with spaces.
rad_id_old	43	10	R	This is the previous ID for this individual.
FILLER	53	948	O	Reserved for future use. Fill with spaces.

Example – to swap the old-id of 9231972 to the new-id of 9441973 create a record like this:

9231972	R180SWAP	9441973	9231972
---------	----------	---------	---------

R190DEQV - DAP Equivalent Course Record

Updated: September 30, 2022

This record contains information for the dap_eqv_crs_mst. This is an optional table.

It maps a course from one catalog year to an equivalent course in another catalog year. It is used primarily when course-numbers are re-used over time, but it can be used to map one course-key to another. This mapping lets Degree Works use “ENGL110” from 10 years ago to satisfy the current “ENGL111” requirement.

Total length = 1000 bytes.

Note: All records must be included in the bridge file used to update the dap_eqv_crs_mst.

Field name	Pos	Len		UCX	Comments
HEADER	1	28		R	This is a 28-byte field identifying the record as one containing dap_eqv_crs_mst data. It is composed of a 10-byte code “EQVCRS”, an 8-byte IDENTIFIER (R190DEQV for the dap_eqv_crs_mst), an 8-byte term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: “EQVCRS....R190DEQV.....A.”
TERM CATALOG YEAR	29	12		R	The Degree Works catalog year (UCX-STU035) mapped from the term the class was taken. Mapped from term to catalog year through UCX D16. The wildcard character “@” may be used.
COURSE DISCIPLINE	41	12		R	The course discipline from the rad_course_key of the class taken.
COURSE NUMBER	53	12		R	The course number from the rad_course_key of the class taken. The wildcard character “@” may be used.

Field name	Pos	Len		UCX	Comments
TARGET CATALOG YEAR	65	12	R	UCX-STU035	The catalog year of the student, the catalog year in which the target course occurred. The wildcard character "@" may be used.
TARGET COURSE DISCIPLINE	77	12	R		The course discipline from the rad_course_key in the target catalog year, used in the requirements against which the student is being audited.
TARGET COURSE NUM	89	12	R		The course number from the rad_course_key in the target catalog year - this course number is the one used in requirements for the target catalog year. The wildcard character "@" may be used.
FILLER	101	900	O		Reserved for future use.

R201TREQ - DAP Transfer Record

Updated: March 24, 2023

This record contains information for the dap_transfer_dtl. This table is optional for Transfer Equivalency.

There should be one record for each transfer transcript course to be evaluated by Transfer Equivalency for transfer equivalence. Test scores from a transfer transcript should be sent to Transfer Equivalency using a rad_test_dtl record (R091TEST).

You would only bridge transfer records here if you want them to be used in the Transfer Equivalency Admin tool; you could articulate the classes without having to enter them manually.

Total length = 1000 bytes.

When processing a transfer transcript (rad_primary_mst ACTION-FLAG = T), the bridge deletes all of the student's dap_transfer_dtl records for the transfer school (SCHOOL-ID) before adding the incoming R201TREQ records.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing dap_transfer_dtl data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R201TREQ for the dap_transfer_dtl), an 8-byte

Field name	Pos	Len		UCX	Comments
					term and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R201TREQ20072...A." (the periods represent spaces).
dap_id	29	10	R		This element is the universal ID code assigned to a student at the time of admission. This ID is validated on the rad_primary_mst. ID is one of the pieces of data used to match incoming transfer courses to those already articulated by Transfer Equivalency. All unarticulated transfer courses for this ID, SCHOOL-ID, and School-code are deleted.
FILLER	39	4	O		Reserved for future use. Fill with spaces.
dap_school	43	12	O	UCX-STU350	This element is the school code within which the transfer class is to be associated. Examples: UG for UndergraduateSchool, LW for LawSchool, GR for GraduateSchool. School code is one of the pieces of data used to match incoming transfer courses to those already articulated by Transfer Equivalency. All unarticulated transfer courses for this ID, SCHOOL-ID, and School-code are deleted.
dap_deg_interest	55	2	O		Not used - obsolete
dap_school_id	57	20	R		This element identifies the transfer institution where this transfer class was taken. It is validated against the rad_ets_mst. SCHOOL-ID is one of the pieces of data used to match incoming transfer courses to those already articulated by Transfer Equivalency. All unarticulated transfer courses for this ID,

Field name	Pos	Len	UCX	Comments	
				SCHOOL-ID, and School-code are deleted.	
dap_tr_disc	77	12	R	This element is the discipline code from the transfer course.	
dap_tr_crse_num	89	12	R	This element is the course number from the transfer course.	
dap_tr_title	101	30	O	This element stores the title of the course as it was identified at the transfer institution.	
dap_tr_credits	131	6	R	This element is the number of credits taken at the transfer institution. It must be numeric and is stored in the database in the format 999v999. Example: a 2.5 credit class would be "002500"). Transfer Equivalency uses it in conjunction with the CALENDAR field to convert from Semester to Quarter or Quarter to Semester hours at your institution. Default = "000000".	
dap_tr_start	137	8	O	This element is the start date for a particular transfer class. Format CCYYMMDD. Start, Stop Date and Term are some of the pieces of data used to match incoming transfer courses to those already articulated by Transfer Equivalency.	
dap_tr_stop	145	8	O	This element is the stop date for a particular transfer class. Format CCYYMMDD. Start, Stop Date and Term are some of the pieces of data used to match incoming transfer courses to those already articulated by Transfer Equivalency.	
dap_calendar	153	2	R	UCX-STU346	This element defines the calendar structure for the transfer institution's academic year. Examples: S for Semester, Q for Quarter. If this calendar structure differs from

Field name	Pos	Len		UCX	Comments
					the home institution, then Transfer Equivalency converts rad_tr_credits above to the appropriate Degree Works credits. Default = RAD-ETS-CALENDAR or UCX-CFG020 "TREQ" DFLT-CALENDAR.
dap_tr_grade	155	4	R	UCX-STU398	Grade from the transfer institution. If UCX-CFG020 "TREQ" TR-GRD-OPTIONAL = Y then TR-GRADE may be blank. Otherwise, TR-GRADE must be valid in UCX-STU398. Default = UCX-CFG020 "TREQ" DFLT-GRADE.
dap_tr_grade_pts	159	6	O		This element is the number of grade points awarded at the transfer institution. It must be numeric and is stored in the database in the format 999v999. Example: 12.5 grade points would be "012500". Default = "000000".
dap_term	165	8	R	UCX-STU016	This element stores the term associated with the transfer class. Example: 20072 for Spring 2007. Start, Stop Date and Term are some of the pieces of data used to match incoming transfer courses to those already articulated by Transfer Equivalency.
dap_user_def1	173	12	O		This element stores additional data as defined by your site.
dap_user_def2	185	12	O		This element stores additional data as defined by your site.
dap_user_def3	197	12	O		This element stores additional data as defined by your site.
dap_user_def4	209	12	O		This element stores additional data as defined by your site.
dap_user_def5	221	12	O		This element stores additional data as defined by your site.
dap_user_def6	233	12	O		This element stores additional data as defined by your site.
dap_user_def7	245	12	O		This element stores additional data as defined by your site.

Field name	Pos	Len		UCX	Comments
dap_user_def8	257	12	O		This element stores additional data as defined by your site.
dap_user_def9	269	12	O		This element stores additional data as defined by your site.
dap_user_def10	281	12	O		This element stores additional data as defined by your site.
dap_condition1	293	12	O		This element stores additional data as defined by your site that can be used by Transfer Equivalency as a condition to be evaluated during transfer articulation.
dap_condition2	305	12	O		This element stores additional data as defined by your site that can be used by Transfer Equivalency as a condition to be evaluated during transfer articulation.
dap_condition3	317	12	O		This element stores additional data as defined by your site that can be used by Transfer Equivalency as a condition to be evaluated during transfer articulation.
dap_condition4	329	12	O		This element stores additional data as defined by your site that can be used by Transfer Equivalency as a condition to be evaluated during transfer articulation.

R500DELI Delete ID Record

Updated: March 25, 2022

Send this record to delete a student or staff ID from the Degree Works repository.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing DELETEID data. It is composed of a 10-byte ID number, an 8-byte IDENTIFIER (R500DELI for DELETEID), an 8-byte term (set to spaces for

Field name	Pos	Len	UCX	Comments
				this table) and a 2-byte ACTION-FLAG (D=Delete). Example: "00129918..R500DELI.....D." (periods represent spaces).
rad_id	29	10	R	This element is the universal ID code assigned to a student by the student system. The same ID will remain with an individual throughout association with the institution. This rad_id is validated on the rad_primary_mst. This is the rad_id code assigned to the student who is being deleted.
FILLER	39	962	O	Reserved for future use. Fill with spaces.

R602CRSE - Course Record

Updated: March 25, 2022

This record contains information for the rad_course_mst.

- It is a required table.
- It is used in SEP (Student Educational Planner) validation, for running planned courses in an audit, by Scribe, and by Transfer Equivalency.
- Its title and credits are used in displaying the hint that appears on the course advice on the audit worksheets.
- Its total length = 1000 bytes.

Note: The course bridge adds new courses to, updates existing courses in, but does not delete courses from the rad_course_mst. To remove old entries from the rad_course_mst, you must use sql to delete them manually.

Field name	Pos	Len	UCX	Comments
HEADER	1	28	R	This is a 28-byte field identifying the record as one containing rad_course_mst data. It is composed of the word COURSE (6 bytes + 4 spaces), an 8-byte IDENTIFIER (R602CRSE for the rad_course_mst), an 8-byte

Field name	Pos	Len		UCX	Comments
					rad_term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete, M=Modify).). Example: "COURSE....R602CRSE.....A." (periods represent spaces).
COURSE DISCIPLINE	29	12	R	UCX-STU352	This element is the Course Discipline Code that is validated against UCX-STU352. It is the Course Discipline portion of the rad_course_key, bytes 1-12. Example: MATH for course key MATH 351.
COURSE NUMBER	41	12	R		This element is the Course Number portion of the rad_course_key, bytes 13-24. Example: 351 for course key MATH 351.
FILLER	53	4	O		Reserved for future use. Fill with spaces.
rad_school	57	12	O	UCX-STU350	<p>This element is the school code associated with a given course. Examples: UG for UndergraduateSchool, LW for LawSchool, GR for GraduateSchool.</p> <p>Note: Due to the possibility of the same course being associated with two or more different schools, Course School Attribute records have been added to the rad_crs_attr_dtl in Release 4.1.1.</p> <p>The rad_school should now be written to the rad_crs_attr_dtl (R605CRSA BIF record) with the special Degree Works rad_attr_code = 'DW-SCHOOL'. The rad_attr_value should be loaded with the</p>

Field name	Pos	Len	UCX	Comments
				actual rad_school (e.g., 'UG' for Undergraduate School).
				These School attributes are used by the Course List in the Planner to filter courses by School.
rad_division	69	12	O	Not used - obsolete.
rad_dept	81	12	O	Not used - obsolete.
rad_course_title	93	50	R	This element is the actual title assigned to the course. It is used by Transfer Equivalency to display the course title in the context of mappings. It is NOT the course title that prints on the Degree Works audit.
rad_credits	143	7	R	This element contains the number of credits associated with the course. Format is 9999v999. Example: 3 credits would be stored as "0003000".
rad_credit_ind	150	2	O	This element contains a flag for indicating if this is a variable credit course. If it is not, set to "NA". If it can have a range of credits, set to "TO". If it can have a high credit and a low credit, set to "OR".
rad_credits_low	152	7	R	This element contains the low-end credit range of the variable credit course. Format is 9999v999. Example: 3 credits would be stored as "0003000". rad_credit_ind must be set to TO or OR for this to have meaning.
rad_credits_high	159	7	R	This element contains the high-end credit range of the variable credit course. Format is 9999v999. Example: 3 credits would be stored as "0003000". rad_credit_ind must be set to TO or OR for this to have meaning.
rad_repeat_max	166	2	O	This element contains the maximum number of times a course can be repeated.

Field name	Pos	Len	UCX	Comments
				Example: "00" if the course cannot be repeated, "1" or "Y" if it can be repeated. This field is used by Transfer Equivalency only.
rad_acad_votech	168	6	O	Not used - obsolete.
rad_class_type	174	2	O	Not used - obsolete.
rad_user_def1	176	12	O	This element stores additional data as defined by your site.
rad_user_def2	188	12	O	This element stores additional data as defined by your site.
rad_user_def3	200	12	O	This element stores additional data as defined by your site.
rad_user_def4	212	12	O	This element stores additional data as defined by your site.
rad_user_def5	224	12	O	This element stores additional data as defined by your site.
rad_user_def6	236	12	O	This element stores additional data as defined by your site.
rad_user_def7	248	12	O	This element stores additional data as defined by your site.
rad_user_def8	260	12	O	This element stores additional data as defined by your site.
rad_user_def9	273	12	O	This element stores additional data as defined by your site.
rad_user_def10	284	12	O	This element stores additional data as defined by your site.
rad_cat_yr_start	296	12	O	The starting catalog year for this course
rad_car_yr_stop	308	12	O	The ending catalog year for this course

R605CRSA - Course Attribute Record

Updated: March 25, 2022

This record contains information for the rad_crs_attr_dtl. This is an optional table.

You may associate an unlimited number of attributes to a course using this record. The rad_course_key contains the same rad_course_key used in the CRSE record and is used to tie these records together.

You may bridge a list of attributes or traits associated with a course, such as it is an honors course, is a writing intensive course, etc.

You may refer to these attributes in Scribe using code similar to the following:

```
5 Credits in ENGL @ (With Trait = WRITING)
```

In this case TRAIT would be stored in the rad_attr_code and WRITING would be stored in the rad_attr_value.

When a CRSE record is encountered all rad_crs_attr_dtl records are deleted for the specified rad_course_key and thus all CRSA records must be bridged after their corresponding CRSE records.

For Banner schools, all course attributes are stored here with ATTRIBUTE as the rad_attr_code and the 24-byte rad_course_key in the rad_attr_value field.

Any code that is loaded here and is to be used in Scribe must be setup in UCX-SCR044 with the Element field set to "ATTR" – telling Degree Works to look for the value in this table.

Do not use this record for anything except Course Attributes used by the Planner.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing rad_crs_attr_dtl data. It is composed of a 10-byte rad_id (COURSEATTR), an 8-byte IDENTIFIER (R605CRSA for the rad_crs_attr_dtl), 8-bytes of blanks and a 2-byte ACTION-FLAG (A=Add, D=Delete). Example: "00129918..R605CRSA.....A." (the periods represent spaces).
rad_course_key	29	24	R		This element ties this record to a particular rad_course_mst. It contains the rad_course_key: Discipline (12) + Course Number (12). For example, if the Discipline is "ENGL" and the Course Number is "123" the attribute key will be "ENGL 123".
rad_attr_code	53	12	O		This element gives a name to the value; used in a Scribe as (WITH attr_code = attr_value). For example, the attr_code might be "Trait" and the attr_value might be "WRITING"

Field name	Pos	Len	UCX	Comments
				<p>which might result in the following statement: 5 Credits in ENGL @ (WITH Trait = WRITING). For Banner sites, the rad_attr_code in most cases should be hardwired to "ATTRIBUTE" so a sample statement would be: 5 Credits in ENGL @ (WITH Attribute = WRITING).</p> <p>Note: A new special "DW-SCHOOL" course attribute has been created for the 4.1.1 Planner to allow it to filter courses by school. See the comments below this grid for more information.</p>
rad_attr_value	65	12	O	These attributes are stored in the rad_crs_attr_dtl linked from the rad_course_mst by the course key. When a planner audit is performed the attributes associated with each course in the plan is sent to the auditor in case they are needed to satisfy requirements using "WITH Attribute="

DW-SCHOOL: Special Course School Attributes records have been added to the rad_crs_attr_dtl in Release 4.1.1.

The rad_school code(s) associated with a given Course Key should be written to the rad_crs_attr_dtl with the special rad_attr_code = 'DW-SCHOOL'. The rad_attr_value should be loaded with the actual School Code (e.g., 'UG' for Undergraduate School). This functionality allows a single course to be associated with multiple schools.

As of Release 4.1.1 these special 'DW-SCHOOL' course attributes are used by the Course List in the Planner to filter courses by School.

R652MAPD - DAP Mapping Record

Updated: March 24, 2023

The R650MAPD record has been deprecated. Use the new R651MAPD layout with the longer course title and credits fields instead.

This table is required for Transfer Equivalency and Transfer Equivalency Self-Service, and also if Course Link is configured to show mapping information. This data is usually entered using the Mapping function in Transfer Equivalency but it may be entered using the Bridge if you have this data in your student system. The dap_mapping_dtl maps one or more courses at a transfer school to one or more courses at your institution. These mappings are used by Transfer Equivalency to articulate transfer courses for a student to course equivalents at your institution.

Total length = 1000 bytes.

Each dap_mapping_dtl in a one-to-one mapping has a unique dap_map_id that forms the first eight bytes of the Header. The dap_map_id can be any identifier, but is usually “Nnnnnnnn” where “nnnnnnnn” is a number. Each dap_map_cond_dtl associated with the mapping has the same dap_map_id as the dap_mapping_dtl. For a many-to-one mapping (many transfer courses to one course at your institution), there will be one dap_mapping_dtl for each of the transfer courses. The dap_map_id will be the same for all of them and for all associated dap_map_cond_dtl records. For a many-to-many mapping (many transfer courses to many courses at your institution), the higher number of courses will dictate how many dap_mapping_dtl records exist but they will all have the same dap_map_id. For example, a 2-to-4 mapping will have 4 dap_mapping_dtl records, but a 3-to-2 mapping will have 3 dap_mapping_dtl records. For a one-to-many mapping (one transfer course to many courses at your institution), there will be one dap_mapping_dtl for each of the “home” courses and they will all have the same dap_map_id, as will all associated dap_map_cond_dtl records.

A separate file of mapping data containing the dap_mapping_dtl and dap_map_cond_dtl records should be sent to the bridge. Mapping data should not be intermixed with Course, ETS, UCX, or Student data. The data file should be sorted by Header (bytes 1-28) in ascending order.

The Bridge for transfer mappings works on the premise that all mappings previously added by the Bridge will be deleted including mappings that were subsequently modified using Transfer Equivalency. The Bridge deletes all mappings from dapdb with dap_create_who equal to “BRIDGE”. Then all the mappings in the input file are added to dapdb with dap_create_who and dap_who set to “BRIDGE”. If Transfer Equivalency Admin is used to modify a mapping then dap_who will be set to the user’s shp_access_id but the dap_create_who remains as “BRIDGE” and therefore, changes made by Transfer Equivalency Admin will be lost the next time the bridge is run. This methodology precludes updating a single mapping using the bridge and also using Transfer Equivalency Admin to make changes to the mappings stored in your student system. However, if you simply load your mappings into Degree Works one time and only one time, you can then maintain them in Degree Works, making any changes you want. The file of transfer mappings for the Bridge should contain all mappings each time you want to load mappings through the Bridge.

Any mapping records that are rejected by the Bridge are written to an error file called R81E####, where #### is the process number of the Bridge. See the RAD11 execution report (STDLIST) to view the number of errors and the name of the file. Errors for mappings are not logged in rad_log_dtl. If any mapping or map-condition for a dap_map_id has an error then all records with that dap_map_id are rejected and written to the error file.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing dap_mapping_dtl data. It is composed of an 8-byte dap_map_id (Nnnnnnnn), 2 spaces, an 8-byte IDENTIFIER (R652MAPD for the dap_mapping_dtl), 8 spaces, and a 2-byte ACTION-FLAG (A=Add, D=Delete). Set it to "A". Example: "N0000001....R652MAPD.....A."
dap_map_id	29	8	R		"Nnnnnnnn" where "nnnnnnnn" is a number representing the mapping set. See explanation above.
dap_school_id	37	20	R		The transfer school ID. Must be valid in rad_ets_mst.
dap_tr_disc	57	12	R		The discipline code of the course taken at the transfer school Examples: CHEM for Chemistry, MATH for Mathematics, SOC for Sociology.
dap_tr_crse_num	69	12	R		The course number of the course taken at the transfer school. Examples: 115 if the Course Key is ENGL115, 246 if the Course Key is MATH246.
dap_tr_title	81	50	O		The course title of the course taken at the transfer school. Examples: "Freshman Algebra", "History of Europe", "Intro to Chemistry".
dap_tr_catyr_beg	131	12	R	UCX-STU035	The first catalog year for which the transfer course is accepted.
dap_tr_catyr_end	143	12	R	UCX-STU035	The last catalog year for which the transfer course is accepted.
dap_tr_cr_min	155	8	O		The minimum number of credits granted at the transfer school for which the transfer course is accepted. Formatted "9999.999".
dap_tr_cr_max	163	8	O		The maximum number of credits granted at the transfer school for which the transfer

Field name	Pos	Len		UCX	Comments
					course is accepted. Formatted “9999.999”.
dap_tr_gr_min	171	6	O	UCX-STU398	The minimum grade granted at the transfer school for which the transfer course is accepted.
dap_tr_gr_max	177	6	O	UCX-STU398	The maximum grade granted at the transfer school for which the transfer course is accepted.
dap_tr_yrs_ago	183	2	O		If the transfer course was taken more than this number of years ago, the course will not be accepted. Formatted “99”.
dap_discipline	185	12	R	UCX-STU352	The discipline code of the course equivalence at your school. Examples: ENGL for English courses, MATH for Mathematics courses, PHYS for Physics courses.
dap_course_num	197	12	R		The course number of the course equivalence at your school. Examples: 125 if the Course Key is ENGL125, 322 if the Course Key is MATH322, 421 if the Course Key is PHYS421.
dap_section	209	2	O		The section designator of the course equivalence at your school. Example: “C” if the full Course Key + Section is “ENGL125C”, “X” if the full Course Key + Section is “PHYS421X”.
dap_course_title	211	50	O		The course title of the course equivalence at your school. Examples: Examples: “Algebra I”, “European History”, “Chemistry I”.
dap_cat_yr_start	261	12	R	UCX-STU035	The first catalog year of the course equivalence at your school.
dap_cat_yr_stop	273	12	R	UCX-STU035	The last catalog year of the course equivalence at your school.
dap_cr_earn	285	8	O		The number of credits to be granted to the student who

Field name	Pos	Len	UCX	Comments
				took the transfer course(s). Formatted "9999.999".
dap_comment1	293	80	O	Free-text comment about the mapping.
dap_comment2	373	80	O	Free-text comment about the mapping.
dap_authorizer	453	30	O	The name of the person at your school who authorized the mapping.
dap_auth_date	483	8	O	The date the mapping was authorized by your school.
dap_articulation	491	4	R	UCX- XXX347 The articulation code: 1TO1 = One-to-One 1TOM = One-to-Many MTO1 = Many-to-One MTOM = Many-to-Many
dap_cond_flag	495	1	R	Y/N flag indicating that mapping conditions exist. If set to "Y" then there should be dap_map_cond_dtl records with the same dap_map_id.

R658MAPC - DAP Map Conditions Record

Updated: March 24, 2023

This record contains information for dap_map_cond_dtl and is an optional table.

The dap_map_cond_dtl contains optional additional conditions (unlimited) that extend the mapping to apply only when special conditions, such as Degree or Major, are met. The dap_map_cond_dtl cannot stand-alone. There must be at least one corresponding dap_mapping_dtl for each dap_map_cond_dtl. The dap_map_id on the dap_map_cond_dtl must be the same as the dap_map_id on the associated dap_mapping_dtl.

Total length = 1000 bytes.

A separate file of mapping data containing the dap_mapping_dtl and dap_map_cond_dtl records should be sent to the bridge. Mapping data should not be intermixed with Course, ETS, UCX, or Student data. The data file should be sorted by Header (bytes 1-28) in ascending order.

The Bridge for transfer mappings works on the premise that all mappings and map conditions and attributes previously added by the Bridge will be deleted except those that were subsequently

modified using Transfer Equivalency. The Bridge deletes all mappings and map conditions and attributes from dapdb with dap_who equal to "BRIDGE". Then all the mappings and map conditions in the input file are added to dapdb with dap_who set to "BRIDGE". If Transfer Equivalency is used to modify a mapping then dap_who will be "TREQ" and the mapping will not be deleted. This methodology precludes updating a single mapping or map condition or attribute using the Bridge. The file of transfer mappings for the Bridge should contain all mappings and map conditions each time you want to load mappings through the Bridge.

Any map condition records that are rejected by the Bridge are written to an error file called R81E####, where #### is the process number of the Bridge. See the log file to view the number of errors and the name of the file. Errors for mappings are not logged in rad_log_dtl. If any mapping or map-condition or attribute for a dap_map_id has an error then all records with that dap_map_id are rejected and written to the error file.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing dap_map_cond_dtl data. It is composed of an 8-byte dap_map_id (Nnnnnnnn), 2 spaces, an 8-byte IDENTIFIER (R658MAPC for the dap_map_cond_dtl), 8 spaces, and a 2-byte ACTION-FLAG (A=Add, D=Delete). The so set it to "A". Example: "N0000001....R658MAPC.....A."
dap_map_id	29	8	R		"Nnnnnnnn" where "nnnnnnnn" is a number representing the mapping set. See explanation above.
dap_school_id	37	20	R		The transfer school ID. Must be valid in rad_ets_mst.
dap_cond_name	57	30	R	UCX-TRQ061	The name of the piece of data to be evaluated from the student's record to determine if the mapping can be used for the student. Example: MAJOR1 = HIST.
dap_cond_op	87	2	R		The relational operator that connects CONDITION-NAME and CONDITION-VALUE. "=" is Equal. ">" is Greater Than. "<" is Less Than. "<>" is Not Equal. Example: MAJOR1 = HIST.
dap_cond_value	89	80	R		The target value of the data on the student's record. Example: MAJOR1 = HIST. The CONDITION-VALUE may

Field name	Pos	Len		UCX	Comments
					contain a list of comma-separated values. Each comma indicates "OR". Example: MAJOR1 = SPAN, FREN means major #1 must be equal to SPAN or FREN.

R656MAPA - DAP Map Attributes Record

Updated: March 24, 2023

This record contains information for the dap_mapping_dtl and is an optional table.

This data is usually entered using the Mapping function in Transfer Equivalency Admin, but it is allowable to use the Bridge if you have this data in your student system.

The dap_map_attr_dtl contains optional attributes (unlimited) that are applied to the articulation results and used in the running of a transfer audit. The dap_map_attr_dtl cannot stand-alone. There must be at least one corresponding dap_mapping_dtl for each dap_map_attr_dtl. The dap_map_id on the dap_map_attr_dtl must be the same as the dap_map_id on the associated dap_mapping_dtl. The course discipline, number and title fields on the attribute record must match the values on the parent mapping record. By having the title on the attribute it allows you to have the same course discipline and number on a mapping but with different title and attribute values.

Total length = 1000 bytes.

A separate file of mapping data containing the dap_mapping_dtl, dap_map_cond_dtl and dap_map_attr_dtl records should be sent to the bridge. Mapping data should not be intermixed with Course, ETS, UCX, or Student data. The data file should be sorted by Header (bytes 1-28) in ascending order.

The Bridge for transfer mappings works on the premise that all mappings and map conditions previously added by the Bridge will be deleted except those that were subsequently modified using Transfer Equivalency. The Bridge deletes all mappings and map conditions and attributes with dap_who equal to "BRIDGE". Then all the mappings and map conditions in the input file are added to dapdb with dap_who set to "BRIDGE". If Transfer Equivalency is used to modify a mapping then dap_who will be "TREQ" and the mapping will not be deleted. This methodology precludes updating a single mapping or map condition or attribute using the Bridge. The file of transfer mappings for the Bridge should contain all mappings and map conditions and attributes each time you want to load mappings through the Bridge.

Any map condition records that are rejected by the Bridge are written to an error file called R81E####, where #### is the process number of the Bridge. See the log file to view the number of errors and the name of the file. Errors for mappings are not logged in rad_log_dtl. If any mapping or map-condition or attribute for a dap_map_id has an error then all records with that dap_map_id are rejected and written to the error file.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing dap_map_attr_dtl data. It is composed of an 8-byte dap_map_id (Nnnnnnnn), 2 spaces, an 8-byte IDENTIFIER (R656MAPA for the dap_map_attr_dtl), 8 spaces, and a 2-byte ACTION-FLAG (A=Add). The so set it to "A". Example: "N0000001....R656MAPA.....A."
dap_map_id	29	8	R		"Nnnnnnnn" where "nnnnnnnn" is a number representing the mapping set. See explanation above.
dap_discipline	37	12	R		The local course discipline
dap_course_num	49	12	R		The local course number
dap_course_title	61	50	R		The local course title
dap_attr_key	111	12	R		The attribute key. Example, ATTRIBUTE
dap_attr_code	123	12	R		The attribute code. Example, JWC

R702ETSM - ETS Record

Updated: March 24, 2023

Information for the rad_ets_mst. This is required for Transfer Equivalency and is not used by Degree Works.

Total length = 1000 bytes.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	RT		This is a 28-byte field identifying the record as one containing rad_ets_mst data. It is composed of an ETS plus 7 bytes of spaces, an 8-byte IDENTIFIER (R702ETSM for the rad_ets_mst), an 8-byte rad_term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete, M=Modify). Example:

Field name	Pos	Len	UCX	Comments
ETS.....R702ETSM.....A. (periods represent spaces).				
rad_ets_code	29	20	RT	This element is the school code your system uses. If your institution uses ETS then it is the ETS code assigned by the Education Testing Service.
FILLER	49	4	O	Reserved for future use. Fill with spaces.
rad_ets_school	53	100	RT	This element is the name of the school. Example: "University of Washington"
rad_ets_city	153	50	T	This element is the name of the city that is part of the address. Examples: Malvern or Seattle.
rad_ets_state	203	3	T	This element is the U.S. or Canadian postal code for the state or province (uppercase). Examples are: PA for Pennsylvania and WA for Washington.
rad_ets_type	206	12	T	This provides a description of the type of school. Examples: CC for 2 year community college, HS for high school).
rad_calendar	218	2	T UCX-STU346	This element defines the calendar structure for the institution's academic year. Examples: S = Semester, Q = Quarter.

R800UCXT UCX Tables

Updated: September 30, 2022

The UCX tables are included in the Bridge for sites that want to bulk load codes from the student system into Degree Works UCX tables.

It is not necessary to bulk load any or all tables. You may be selective about which tables you choose to bulk load through the Bridge.

Only UCX tables that may need to be sent in bulk are listed here. Ellucian will supply its standard UCX tables and train your site in Controller for maintenance of the codes. If your UCX tables do

not change en masse then there is no need to send them through the Bridge. Only those tables that you want to replace in their entirety should be sent through the Bridge. Use Controller for addition, modification, and deletion of most UCX codes.

UCX key values containing an apostrophe will not work in Degree Works and when found the apostrophe will be replaced with an underscore character in certain situations.

UCX	Key	Description	Notes
UCX-RPT046	12	Custom Report Data	
UCX-SCR002	12	Custom Data	
UCX-SCR003	12	Noncourse Codes	
UCX-SCR044	12	WITH Custom Data	
UCX-SCR045	30	DECIDE Option	
UCX-STU016	8	Term Codes	
UCX-STU023	12	Major Codes	
UCX-STU024	12	Minor Codes	
UCX-STU035	12	Catalog Year Codes	
UCX-STU305	6	Student Level	Also known as Classification; for example: Junior, Senior, etc.
UCX-STU306	2	Student Status Codes	
UCX-STU307	12	Degree Codes	
UCX-STU314	12	Test Codes	
UCX-STU323	12	Specialization Codes	
UCX-STU324	12	Liberal Learning Codes	
UCX-STU350	12	School Codes	Also known as Level; for example: Graduate, Undergraduate, etc.
UCX-STU352	12	Discipline Codes	
UCX-STU355	2	Credit Type Codes	
UCX-STU356	2	Grade Type Codes	
UCX-STU379	12	Transfer Degree Codes	
UCX-STU385	24	Grade Information Table	
UCX-STU560	6	College Codes	
UCX-STU563	12	Concentration Codes	
UCX-TRQ002	2	State Code	Ellucian supplies, but

UCX	Key	Description	Notes
			you can overwrite.

If you want to send UCX tables through the Bridge, send the 28-byte header followed by the 30-byte key (UCX-TABLE plus UCX-CODE) and the 200-byte UCX-VALUE. Send all records for a table. The Bridge will delete all the existing records for the UCX table, and then add the Bridge records for the table. You do not need to send all tables, just all records for the tables you want to update. Remember that the contents of a UCX table must be all together in a single data file.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		This is a 28-byte field identifying the record as one containing UCX data. It is composed of a 6-byte UCX table, 4 spaces, 8-byte IDENTIFIER (R800UCXT), 8-byte term (set to spaces) and a 2-byte ACTION-FLAG (A=Add). Example: STU023.....R800UCXT.....A. (periods represent spaces).
UCX TABLE	29	6	R	UCX-SYS001	This element is the UCX table number. Example: UCX-STU023 is the Major Table. It must be valid in UCX-SYS001.
UCX CODE	35	30	R		This element is the UCX code. Examples: Major codes in UCX-STU023 could be things like ENGL for English Majors, MATH for Math Majors or PHYS for Physics Majors. A unique KEY into the UCX tables is formed when UCX-TABLE and UCX-CODE are concatenated. Example: STU023 MATH
UCX VALUE	65	200	R		This element contains the code translation and flags set for each code. Examples: "English" for the ENGL Major code, "Mathematics" for the "MATH" Major code, "Physics" for the "PHYS" Major code. The layout of UCX-VALUE varies from table to table. The layout is defined for each table in the Degree Works Documentation.

Field name	Pos	Len		UCX	Comments
FILLER	265	736	O		Reserved for future use. Fill with spaces.

R900CURR - Curriculum Rules

Updated: March 25, 2022

This record contains information for the rad_currrule_dtl. This is an optional table.

It contains all of the relationships between school, degree, major, etc to define a curriculum.

The data in this table will be used control the drop-down lists for What-If audits and Transfer Equivalency Self-Service when configured to use curriculum rule data for filtering.

Total length = 1000 bytes.

Note: All records must be included in the bridge file used to update the rad_currrule_dtl.

Field name	Pos	Len		UCX	Comments
HEADER	1	28	R		<p>This is a 28-byte field identifying the record as one containing rad_currrule_dtl data. It is composed of a 10-byte code containing the 8-byte CURR_RULE_ID plus 2-bytes of FILLER, an 8-byte IDENTIFIER (R900CURR for the rad_currrule_dtl), an 8-byte term (set to spaces for this table) and a 2-byte ACTION-FLAG (A=Add, D=Delete).</p> <p>Example: “123.....R900CURR.....A.”. Spaces are filled with periods “.”</p>
Curr-Rule-Id	29	8	R		<p>A number used to tie associated records for the same curriculum together. This number must be left-justified and space filled. Example: “123.....” (8-byte number).</p> <p>Spaces are filled with periods “.”</p>

Field name	Pos	Len		UCX	Comments
Catalog Year Start	37	12	R	UCX-STU035	The starting catalog year when this curriculum began.
Catalog Year Stop	49	12	R	UCX-STU035	The last catalog year when this curriculum was/is valid.
Goal Code	61	12	R	UCX-SCR004	The type of curriculum data this record represents. Examples: CAMPUS COLLEGE CONC DEGREE MAJOR MINOR PROGRAM SCHOOL
					The Goal Codes used must be defined in UCX-SCR004.
Goal Value	73	12	R		The value of the SCHOOL, DEGREE, MAJOR, etc. For example, SCHOOL of "UG", DEGREE of "BS" and MAJOR of "CHEM".
Attach Code	85	12	O	UCX-SCR004	If this record is tied to another piece of curriculum data then fill this field in. For example, if the Goal Code = CONC and Goal Value = PIANO the Attach Code might be MAJOR with an Attach Value of MUSIC. This is used to tie concentrations to majors or minors to majors etc.
Attach Value	97	12	O		See above.
FILLER	109	892	O		Reserved for future use. Fill with spaces.

Prerequisite Checking

Updated: March 25, 2022

Missing prerequisite requirements in the Degree Works Student Educational Planner and in Banner registration can be identified, and advice to the student for resolving the errors can be provided using REQUISITE blocks in Scribe.

Scribing for prerequisite checking

Updated: March 25, 2022

REQUISITE blocks are Scribe blocks that specify and check for course pre-requisites or co-requisites.

REQUISITE blocks

For example, an employee from a university registrar's office could create a REQUISITE block for an advanced mathematics course. The REQUISITE block might specify a pre-requisite basic mathematics course that must be completed before a student can enroll for the advanced course. A REQUISITE block can specify and check for credit and grade requirements, too.

To have access to create and update REQUISITE blocks, a user needs the SCRBLREQ Shepherd key.

Degree Works allows users to scribe REQUISITE blocks that can be used in two services.

- Requisite Check - This is used by the next generation Student Educational Planner (SEP) to verify that prerequisites have been met when creating a plan. Also, Banner can send a request to Degree Works through the Oracle advanced message queues during registration to check if the student has satisfied the requisite requirements. The student must already exist in Degree Works for both of these to work.
- Requisite Description - Banner contacts Degree Works through the Oracle advanced message queues for a description of the requisite requirements while users are reviewing catalog information.

Sample REQUISITE block in Scribe

In the following example, the 1 Class rule and the ProxyAdvice are used for the Requisite Check service, while the Remarks are used for the Requisite Description service.

```
##REQUISITE = 200610 10285
##ENGL 523

BEGIN
StandAloneBlock #share with all blocks
;
```

```
Remark "In addition, the student must take:";  
Remark "PSYC 1000 OR SOC 211.";  
  
1 Class in PSYC 1000, SOC 211  
    ProxyAdvice "You need to take either PSYC 1000 or SOC 211"  
    Label "This text is ignored because ProxyAdvice was used."  
  
END.
```

Elements of a REQUISITE block

Updated: March 24, 2023

A REQUISITE block can include many elements.

Header qualifiers

Specify the StandAloneBlock header qualifier for all REQUISITE blocks. This allows sharing with other REQUISITE blocks. Otherwise, the class will be used only one time.

```
BEGIN  
StandAloneBlock # share with all blocks  
;  
1 Class in MATH 100  
    ProxyAdvice "Before taking MATH 101 you must first take MATH 100."  
    Label "ignored";  
END.
```

When scribing header qualifiers you must use ProxyAdvice. Header qualifiers apply only to the courses found in the rules in a block. In the following example, the pre-requisite check for 10 credits at the 200 level is done only if there are rules in this block that require classes at the 200 level. The header qualifiers do not examine all the classes that the student has taken; they only examine the classes that fit the rules in this block.

```
BEGIN  
StandAloneBlock # share with all blocks  
MinCredits 10 in @ 2@  
    ProxyAdvice "You first need at least 10 credits at the 200 level."  
;  
15 Credits in @  
    ProxyAdvice "You must take 15 credits first."  
    Label "ignored";  
END.
```

Proxy advice

You should use ProxyAdvice on all rules. If ProxyAdvice is not found on a rule the Label text is displayed to the user when a prerequisite is not met. Consider using the <APPLIED> and <NEEDED> tags to display the classes or credits that the user has taken and those that are still required.

```
BEGIN
StandAloneBlock # share with all blocks
;
5 Credits in @ 2@ 
    ProxyAdvice "You must take 5 credits at the 200 level. You have taken "
    ProxyAdvice "<APPLIED> but still need <NEEDED> more."
    Label "ignored";
END.
```

When building a group rule place the ProxyAdvice at the top-most group level.

```
BEGIN
StandAloneBlock # share with all blocks
;
1 Group in
    (3 Classes in SPAN @ Label "spanish") or
    (3 Classes in FREN @ Label "french") or
    (3 Classes in IRISH @ Label "irish")

    ProxyAdvice "You must take 3 classes in either Spanish, French or
Irish "
    ProxyAdvice "before taking this upper division linguistics course.
"
    Label "ignored";
END.
```

References to OTHER blocks

REQUISITE blocks can refer to OTHER blocks in a way similar to academic audits. If the referenced OTHER block cannot be found for the given registration term the Label text on the Block rule is displayed to the user. If the OTHER block is found then the Label is ignored and the ProxyAdvice (or Labels) found in the OTHER block is displayed to the user when a prerequisite is not met, as illustrated in the following example.

```
BEGIN
StandAloneBlock # share with all blocks
;
1 Block (OTHER = ENGLTEST)
    Label "No ENLTEST block was found - please contact the registrar's
office. ";

    1 Class in ENGL @ (With Attribute = "WRIT")
        ProxyAdvice "You must first take an English writing intensive class
."
        Label "ignored";
END.
```

Co-requisite check

You can perform a co-requisite check on any course using this scribing.

```
1 Class in BIOL 123 (With DWTerm = REGTERM)
    ProxyAdvice "You must take Biology 123 at the same time as Biology
```

```
106.“  
Label “ignored”;
```

Only if BIOL 123 is part of the current registration process will it pass this check. This means that if BIOL 106, BIOL 123 and ENGL 147 are part of a student's registration you will be able to check to see that BIOL 123 is a co-requisite to BIOL 106 (or to ENGL 147 of course). You could also use DWPreregistered = Y, however this won't catch planned courses so cannot be used for SEP prerequisite checking. DWTerm = REGTERM can be used in both SEP and Banner Registration prerequisites.

Strict prerequisite check

On any course using this scribing, you can perform a strict prerequisite check, meaning that a course must be taken in a prior term and cannot be taken in the same term.

```
1 Class in BIOL 123 (With DWTerm < REGTERM)  
ProxyAdvice “You must take Biology 123 prior to taking Biology 106  
.“  
Label “ignored”;
```

Only if BIOL 123 was taken or planned for in a prior term will it pass this check. This means that if BIOL 106 is planned for in the term before BIOL 123, it will pass the prerequisite check. But if BIOL 106 is planned for in the same term as BIOL 123, it will NOT pass the prerequisite check. If BIOL123 can be taken as either a co- or a prerequisite for 106, you do not need to use DWTerm at all.

In-progress classes

Classes that a student is currently taking and classes for which the student is trying to register can be applied to the requisite rules. Consider using the **DWInProgress** qualifier to ensure that the student has already completed and passed a particular class. However, DWInProgress cannot be used for SEP prerequisite checking so for blocks that are used in both SEP and Banner, we recommend also using **DWGradeLetter = PLAN**. For example, a student who is taking ENGL 213 in Fall 2014 may try, in early December, to register for ENGL 325 for Spring 2015. If the student is required to complete ENGL 213 before registering for ENGL 325 you can use the DWInProgress qualifier to prevent them from registering for ENGL 325 as shown in the following example.

```
BEGIN  
StandAloneBlock # share with all blocks  
;  
1 Class in ENGL 2@ (With DWInProgress = “N” or DWGradeLetter = PLAN)  
ProxyAdvice “You must have completed an English 200 level course.”  
Label “ignored”;  
END .
```

Note that if a student has planned for both ENGL 213 and ENGL 325 in SEP, then a prerequisite error will not be generated because of the DWGradeLetter qualifier.

You can also place a qualifier at the top of your block to disallow in-progress classes for all of the rules. By adding the DWGradeLetter <> PLAN you are allowing planned classes in the block; you are telling the auditor to ignore planned classes for this qualifier.

```
BEGIN
StandAloneBlock # share with all blocks
MaxClasses 0 in @ (With DWInProgress = "Y" and DWGradeLetter <> PLAN)
;
1 Class in ENGL 2@ 
    ProxyAdvice "You must have completed an English 200 level course."
    Label "ignored";
1 Class in HIST 2@ 
    ProxyAdvice "You must have completed a History 200 level course."
    Label "ignored also";

END.
```

Alternatively, you could re-run the requisite check for all classes after the fall semester to check for passing grades.

Minimum grades

Rules that require a minimum grade can be handled with the DWGrade qualifier. If you're using prerequisite checking in SEP, planned courses will not have a grade. To prevent them from failing the prerequisite check, use the DWGradeLetter qualifier. If using prerequisite checking in both SEP and Banner registration, you can combine these qualifiers on the rule. When DWGrade or DWGradeNumber are used, both in-progress and passed pass-fail classes are given a grade of 4.0.

```
BEGIN
StandAloneBlock
;
1 Class in DAN 37058 (With DWGrade > 2.0 or DWGradeLetter = PLAN)
    ProxyAdvice "Prerequisite: DAN 37058 with a grade of C or "
    ProxyAdvice "better is required before enrolling in DAN 47156."
    Label "Not Used";

END.
```

Prerequisites by major

Prerequisite rules may differ by major. The student's curriculum already in Degree Works is used when the prerequisite check is performed. You can scribe if-statements based on this data. You do need to keep in mind that if the student does not exist in Degree Works (because the student was added to Banner and registered all in the same day) there will be no curriculum information available. You should scribe for this possibility. Determine what should show if the student's data is not in Degree Works.

```
BEGIN
StandAloneBlock
;
# Chemistry majors - or if the student has no majors (ie, doesn't exist in Degree Works)
If (Major = CHEM or NumMajors = 0) then
    1 Class in BIOL 134
        ProxyAdvice "Prerequisite: Chemistry majors must take BIOL 134"
        ProxyAdvice "before enrolling in BIOL 206."
        Label "Not Used";
```

```
END.
```

Another option is to give specific advice for students without a major:

```
BEGIN
StandAloneBlock
;
If (NumMajors = 0) then
    RuleIncomplete
        ProxyAdvice "Prerequisite: You don't have a major or your curriculum is not known;"
        ProxyAdvice "you cannot enroll in BIOL 206."
        Label "Not Used"
else If (Major = CHEM or NumMajors = 0) then
    1 Class in BIOL 134
        ProxyAdvice "Prerequisite: Chemistry majors must take BIOL 134"
        ProxyAdvice "before enrolling in BIOL 206."
        Label "Not Used 2";
END.
```

Cross-listings check

You may want to scribe part of your prerequisites to ensure the student does not enroll in the class a second time under another name when two classes are cross listed. For example, MATH 108 may be cross listed with BUSN 108. You may want Degree Works to ensure the student cannot enroll in MATH 108 if they have already taken BUSN 108. This is not something that you should be attempting to do in a requisite block in Degree Works. This needs to be handled in Banner as part of the registration setup.

Ranges

Ellucian does not recommend scribing ranges in a REQUISITE block. When you do a DESCR request, Degree Works sends across all courses referenced in the block to a Banner table called so_dw_desc_reference. The coursenumber field in this table allows only five characters, so when a range such as 1000:1010 is sent across, an error occurs.

Examples

The following example illustrates a block in which a student must take 24 credits in Math and cannot have more than 6 credits with a grade of C or worse.

```
BEGIN
StandAloneBlock # share with all blocks
;
18 Credits in MATH @ (With DWGrade > 3.0 or DWGradeLetter = PLAN)
    ProxyAdvice "You must take 18 credits with a grade of B or better.
    "
    ProxyAdvice "You have taken <APPLIED> but still need <NEEDED> more
    "
    Label "ignored 1";

6 Credits in MATH @ (With DWGrade > 2.0 or DWGradeLetter = PLAN)
    ProxyAdvice "You must take 6 credits with a grade of C or better.
```

```
“  
    ProxyAdvice “You have taken <APPLIED> but still need <NEEDED> more  
.”  
    Label “ignored 2”;  
END.
```

The following block uses Attributes and GPA qualifiers.

```
BEGIN  
StandAloneBlock # share with all blocks  
MinGPA 3.0 in MATH @ (With Attribute = "WRIT") ,  
                ENGL @ (With Attribute = "SCI")  
;  
3 Classes in MATH @ (With Attribute = "WRIT") ,  
                ENGL @ (With Attribute = "SCI")  
EXCEPT ENGL 148, 202  
ProxyAdvice “You must take 3 classes of writing intensive math coursework”  
ProxyAdvice “or English science related coursework.”  
ProxyAdvice “Engl 148 and 202 do not count.”  
Label “ignored 1”;  
  
If (BannerGPA < 2.0) then  
    RuleIncomplete  
        ProxyAdvice “Your GPA is below a 2.0 – you can’t take this class  
.”  
        Label “ignored 2”;  
END.
```

The following block illustrates how to scribe a prerequisite based on a placement test score.

```
BEGIN  
StandAloneBlock # share with all blocks  
;  
If (WRITEXAM < 68) then  
    RuleIncomplete  
        ProxyAdvice “You need to achieve a score of 68 or higher in your  
writing”  
        ProxyAdvice “exam to take this class.”  
        Label “ignored 1”  
Else # good score!  
    RuleComplete  
    Label “ignored 2”;  
END.
```

The following block can be used when a course must be taken in a prior term and a planned course should not be allowed.

```
BEGIN  
StandAloneBlock  
;  
    1 Class in ECON 100 (With DWTerm < REGTERM and DWGrade > PLAN)
```

```
Proxy-Advice "You must have completed ECON 100 before taking  
ECON 101"  
    Proxy-Advice "(and cannot be planning to take it)"  
    Label "Prerequisite";  
END.
```

If a course must be completed or planned for in a prior term, use the following. This example is saying that the student cannot be taking GEOG 2310 right now – it must have been completed. However, when performing the prerequisite check from SEP, it is okay for the course to not have been completed because many future terms are being checked in SEP and all are considered in-progress.

```
BEGIN  
StandAloneBlock  
:  
    1 Class in GEOG 2310 (With DWInProgress = "N" or DWGradeLetter = PLAN)  
        Proxy-Advice "You must complete GEOG 2310 before taking GEOG  
3340."  
        Label 2 "Prerequisite";  
END.
```

If you are using the **Requisite Description** service each of your blocks must contain Remarks that explain the requirements, as illustrated in the following example.

```
BEGIN  
StandAloneBlock # share with all blocks  
:  
    Remark "In order to satisfy a prereq for an internship a student mus  
t take: ";  
    Remark "(HIST 110, HIST 111, and HIST 114) OR (ANTH 100, ANTH 101 ";  
    Remark "and ANTH 102) In addition, the student must take: ";  
    Remark "PSYC 100 OR SOC 110.";  
  
    # For group rules you should put ProxyAdvice on the Group itself to  
    # cover all of the options  
    1 Group in  
        (3 Classes in HIST 110 + 111 + 114  
            Label " ignored 1") OR  
        (3 Classes in ANTH 100, 101, 102  
            Label " ignored 2")  
        ProxyAdvice "You either need 3 classes in xyz or 3 classes in ab  
c."  
        LABEL "ignored";  
  
    1 Class in PSYC 100, SOC 110  
        ProxyAdvice "You need to take either PSYC 100 or SOC 110";  
        Label "this label is ignored";  
END.
```

As in the following example, you can place your rules inside a subset to control the message displayed to a user. If any of the requirements in the subset are not met, the user sees the same message.

```
BEGIN
StandAloneBlock # share with all blocks
;
BeginSub
  3 Credits in MATH 100, 110, 120, 130
    ProxyAdvice "this advice is ignored - proxy-advice on subset is
used"
    Label "A";
  3 Credits in STAT 200, 210, 220, 230
    ProxyAdvice "this advice is ignored - proxy-advice on subset is
used"
    Label "B";
  3 Credits in MATH 100, 110, 120, 130, STAT 200, 210, 220, 230
    ProxyAdvice "this advice is ignored - proxy-advice on subset is
used"
    Label "C";
EndSub
  ProxyAdvice "You need 9 credits from the courses you are taking.
"
  Label "ignored";
END.
```

Block sharing considerations

Updated: March 25, 2022

Processing multiple REQUISITE blocks could cause a sharing issue.

When a Requisite Check request is sent to Degree Works from registration, the list of all classes for which the student is attempting to register are passed. When a Requisite Check request is sent from SEP, the list of all planned COURSE and selected CHOICE requirements in terms greater than or equal to the student's active term and classes from terms before the student's active term are passed. Degree Works processes all the requisite blocks together in one audit. In an academic audit, you might have DEGREE, MAJOR, and OTHER blocks comprising the set of requirements. For a Requisite Check request the audit consists of a list of REQUISITE blocks, each one for a different registration class.

Processing multiple REQUISITE blocks could cause a sharing issue. For example, a student may be registering for or planning to take two classes, both with the same prerequisite. To support this, sharing must be allowed so that the same class can be used to satisfy all requirement blocks. This is accomplished by using the StandAloneBlock header qualifier. The following example shows REQUISITE blocks for two courses, MATH 101 and MATH 105 that have the same pre-requisite course, MATH 100.

REQUISITE=MATH101

```
BEGIN
StandAloneBlock # share with all blocks
;
1 Class in MATH 100
ProxyAdvice "Before taking MATH 101 you must first take MATH 100."
```

```
    Label "ignored";
END.

REQUISITE=MATH105

Begin
StandAloneBlock # share with all blocks
;
1 Class in MATH 100;
ProxyAdvice "Before taking MATH 105 you must first take MATH 100."
Label "ignored";
END.
```

In some cases, using the same class may not be as visible, such as when rules are defined in OTHER blocks. The same sharing considerations exist in the following example where ENGL 100 could be used twice to satisfy the prerequisite requirements for two different classes.

REQUISITE=ENGL101

```
Begin
StandAloneBlock # share with all blocks
;
1 Block (Other=ENGLTEST);
    Label "Label ignored - Labels/ProxyAdvice in OTHER block are used"
;
# But if the OTHER=ENGLTEST block is found that Label text is used

END.
```

OTHER=ENGLTEST

```
Begin
StandAloneBlock # share with all blocks
;
if (ENGLTEST < 100) then
1 Class in ENGL 100
    ProxyAdvice "You did not pass the English test "
    ProxyAdvice "so you must first take ENGL 100."
    Label "ignored";
else # Engl test was passed
    RuleComplete
    Label "this too is ignored";
END.
```

REQUISITE=ENGL102

```
Begin
StandAloneBlock # share with all blocks
;
1 Class in ENGL 100
    ProxyAdvice "Before taking ENGL 102 you must first take ENGL 100."
    Label "ignored";
END.
```

Block naming considerations

Updated: March 25, 2022

Creating REQUISITE blocks using a term range instead of a catalog year range gives you the flexibility to change requirements in the middle of a catalog year.

When saving a REQUISITE block in Scribe, you assign it a range of terms instead of catalog years. As soon as you select REQUISITE as the Block Type the form changes from Start Catalog Year to Start Term and from Stop Catalog Year to Stop Term. This allows you to make changes to requisite blocks in the middle of a catalog year – though normally your term range will go from the start of one catalog year to the end of another catalog year. For Banner Prerequisite Checking, the Block Value field can be populated with either the course key or course key and section or with a term and CRN value. For example, the block can be set up for any MATH123 course for the term range specified or with MATH123-A (for section A) or it could be set up for 20152012345 where 201520 is the term and 12345 is the CRN (Course Reference Number). If term+CRN is being used, be sure to set the Start Term and Stop Term to be the same term used in the value. The term in the value only has to be in the range of the Start Term and Stop Term, but it makes sense to make sure all three term values match. Note that for SEP Prerequisite Checking, only the course key can be used as the Block Value because planner courses do not have CRNs. Creating REQUISITE blocks using a term range instead of a catalog year range gives you the flexibility to change requirements in the middle of a catalog year. The simplest way to save a REQUISITE block is to use the entire term range. In the following example, the course's prerequisite rules are constant from term to term.

```
REQUISITE=ENGL123
Start Term=000000
Stop Term=999999
```

In the next example, the course has a certain prerequisite up to and through the Spring 2015 term, but the prerequisites change after that term. In this case, two different blocks are needed, one with the old term range and one with the new term range.

```
REQUISITE=MATH122
Start Term=000000
Stop Term=201530
```

```
REQUISITE=MATH122
Start Term=201610
Stop Term=999999
```

In this example, the requirements for section B are different so we create a special block with the course key, a hyphen followed by the section. The normal MATH122 requirement block will be used for all non-B sections.

```
REQUISITE=MATH122-B
Start Term=000000
Stop Term=201530
```

However, it is also possible that only one block may be needed. During academic audits students may be audited against old catalogs but with the prerequisite check students should never register for an old term. For this reason, the range may remain as 000000-999999 and you can change the

rules in the single block. A prerequisite that is specific to a term can be structured in two different ways. The first is to use the term range values to limit the scope to one single term by setting both the Start Term and Stop Term to the same term value. If you do this ensure that you don't have other blocks for the same course that include this term in their term-range. The following example shows this approach.

```
REQUISITE=CHEM312
Start Term=201030
Stop Term=201030
```

The second way to create a REQUISITE block specific to a term is to use the term+crn when saving the block. In the following example the term is Spring 2015 (201530) and the CourseReferenceNumber is 98765. When creating a block specific to this term you should set the Start Term and Stop Term values to the applicable term. In the example, the general block for CHEM 312 is in place for all term values. This is possible because Degree Works always looks for the term+crn block before attempting a search on a block with the course name specified. With these two blocks set up for CHEM 312 the first block will only be used for the Spring 2015 term for the course reference number 98765 while the second block will be used in all other cases.

```
REQUISITE=20153098765
Start Term=201530
Stop Term=201530
```

```
REQUISITE=CHEM312
Start Term=000000
Stop Term=999999
```

Scribing for the Requisite Check service

Updated: March 25, 2022

When creating a Scribe block for the Requisite Check service, review the list of considerations.

- ProxyAdvice comments are displayed to the user if the conditions of the rule are not met.
- If ProxyAdvice is not specified for a rule, the Label text is used instead.
- You should always use ProxyAdvice and not rely on Labels. ProxyAdvice allows for longer text and for the <APPLIED> and <NEEDED> dynamic values to be inserted.
- The Block rule label is used if the referenced OTHER block is not available. For example, 1 Block (Other=LANGTEST). The Block rule label is ignored if the block is available. The ProxyAdvice or Labels from the referenced OTHER block are used.
- ProxyAdvice is required on Header Min Qualifiers, although this is not enforced by the parser.
- While several lines of ProxyAdvice can be defined, the maximum number of characters that will display is 200.
- To indicate a course is a co-requisite and must be taken in the same term, use DWTerm = REGTERM.

- To indicate that a course must be taken in a prior term, use DWTerm < REGTERM.
- To specify that all classes in the block must have been completed you can use this in the header: MaxClasses 0 in @ (With DWTerm >= REGTERM).
- Because the student's degree is not a part of the Requisite Check service request, you should not scribe rules based on the degree. For example, If (Degree = BA) then 1 Class in HUMAN 136.
- The student's school (Banner level) is not a part of the Requisite Check service request. When a student has degrees from multiple schools, Degree Works obtains a GPA from the rad_term_dtl table, but it may not be the one that you want to use in your Scribe code. For example, when scribing IF (SSGPA > 2.0) THEN, or IF (BannerGPA > 2.0) THEN, you cannot be certain if the required GPA is being used because it could be the GPA for either the undergraduate or graduate school.

Scribing for the Requisite Description service

Updated: March 25, 2022

When creating a Scribe block for the Requisite Description service, review the list of considerations.

- Remarks are displayed to the user as the description of the REQUISITE block.
- You should place all remarks at the top of the block after the first semicolon. All Remarks found throughout the block are displayed as the complete description.
- All courses listed in the rules are also displayed to the user. You can display a list of these courses and provide links to their descriptions in the catalog.
- Remarks should note all courses listed in the rules within the block. For example, if you list "ACCT 101" in a rule you must list the course as "ACCT 101" in the remark as well, and not "Acct 101" or "Accounting 101". As in Banner, your system may match the list of classes from the rules against the remarks and insert a hyperlink, so you should write the remarks text to make it easy to find courses.
- If you are listing classes, repeat the discipline. For example, "HIST 110, HIST 111" instead of "HIST 110, 111".

Student Educational Planner prerequisite checking

Updated: September 29, 2023

A student's plan can be checked to ensure that all prerequisite courses have been planned for using requirements defined in Scribe REQUISITE blocks.

Plan changes are automatically saved as changes are made. When the application attempts to save a plan, planned COURSE requirements and selected CHOICE requirements in terms greater

than the student's active term and any actual classes in terms prior or equal to the student's active term will be audited against matching REQUISITE blocks. If a prerequisite is found to be missing for planned requirements in future terms, a localized message advising the student about these missing requisites will be displayed on the requirement. Requisite warnings for planned requirements in terms prior or equal to the student's active term will not be returned. Adjustments to the plan must be made to address the errors before the plan can be successfully saved; however, advisors or other power users may be given access to override these warnings.

Configuration

To enable prerequisite checking in SEP, set core.requisite.validate.enable = true. When enabled, if the application attempts a save or save as of any plan, the prerequisite service will be executed. The default action will be that plans cannot be saved until all prerequisite errors have been corrected. Users with the SEPPRQTO key are given the option to save plans without resolving prerequisite errors.

Scribing considerations

Prerequisite checking in SEP follows the guidelines for the Requisite Check service. For more information, see the [Scribing for the Requisite Check service](#) topic. There are a few additional considerations to keep in mind when scribing REQUISITE blocks for SEP.

- Scribe allows the Block Value in REQUISITE blocks to be either the Course Key or the Term plus CRN. Because SEP requirements do not have CRNs, the Block Value must be the Course Key for SEP prerequisites.
- Because planner requirements do not have all the data that actual classes do, some Scribe qualifiers might not work as well with SEP as they do with Banner.
- If a planned selected CHOICE requirement has attributes, only the course(s) are sent to the prerequisite service. The attributes are not sent, which means that these courses may not satisfy a prerequisite with class attributes.

For prerequisites that include a minimum course grade, use (DWGradeLetter = PLAN) along with the DWGrade qualifier you have in a WITH statement. This is necessary because planned courses do not yet have a real grade value and have a grade of PLAN instead.

Banner prerequisite checking

Updated: September 29, 2023

You have two options for complex prerequisite checking with Banner, Degree Works or CAPP (in Banner).

Both systems have capabilities to help you with your Banner prerequisite needs. To decide whether Degree Works or CAPP should be used with registration, determine which system is being used as your academic auditing tool. Because the setup of prerequisites is so similar to the setup needed for degree compliance and academic advising, it makes sense to use one tool for both instead of using two different tools for similar purposes.

Degree Works process configuration

There are two pl/sql packages that must be compiled in the Banner database, one is provided by Degree Works and the other by Banner. The Banner package is called sfkdwaq.

The Degree Works package is called dw_prerequisite_pkg. The source for this package is in the bannerprefunctions.sql file in the app/sql directory. This file should be executed in sqlplus while connected as the Banner user.

```
cd $DGWHOME/sql  
dbb  
start bannerprefunctions
```

If you are an STP (formerly MEP) institution, there are a couple of extra steps required for setup. The main package should be installed by the main (root) Banner user after which each other entity schema must be granted access to the package. In addition, synonyms must be set up for the package for each of the entities. This can be done by either creating a public synonym by the main schema or creating a synonym for each of the individual entities.

```
grant execute on DW_PREREQUISITE_PKG to entity1;  
grant execute on DW_PREREQUISITE_PKG to entity2;  
grant execute on DW_PREREQUISITE_PKG to ...;  
create or replace public synonym dw_prerequisite_pkg for dwmgr.dw_prerequisite_pkg;
```

To start the daemons processes that service Banner registration, you can run the preqrestart script. The preqrestart script simply issues a preqstop followed by a preqstart.

The preqstart script first starts up the four queues in the Banner database by running the following:

```
exec sfkdwaq.p_start_dw_preq_request_q;  
exec sfkdwaq.p_start_dw_preq_response_q;  
exec sfkdwaq.p_start_dw_desc_request_q;  
exec sfkdwaq.p_start_dw_desc_response_q;
```

The preqstart script runs utl01 to start dap61 and dap62. The number of each of these is controlled by these Shepherd settings:

```
classic.daemons.dap61.count  
classic.daemons.dap62.count
```

Usually you want more dap61 daemons than dap62. Most requests from Banner will be sent to dap61 because it handles the actual prerequisite requests for classes. dap62 handles getting a description of the prerequisite.

The preqstart script also issues a preqshow command to show the number and state of daemon processes running in addition to showing the number of Banner prerequisite requests that are currently waiting on the Oracle message queue. Here we see the preqshow output showing the utl01 daemon that created the dap61 and dap62 child processes. We also see how many requests are currently on each of the queues.

```
$ preqshow
```

OWNER	PID	PPID	STARTED	CPUTIME	COMMAND
dwadmin	16417	1	14:32	00:00:00	utl01x db=dwadmin_proda scope=preq
dwadmin	16420	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16421	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16422	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16423	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16424	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16425	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16426	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16427	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16428	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16429	16417	14:32	00:00:00	dap61 db=dwadmin_proda
dwadmin	16430	16417	14:32	00:00:00	dap62 db=dwadmin_proda
dwadmin	16431	16417	14:32	00:00:00	dap62 db=dwadmin_proda

The number of messages currently waiting on the CHECK REQUEST queue
= 0

The number of messages currently waiting on the CHECK RESPONSE queue
= 0

The number of messages currently waiting on the DESCR REQUEST queue
= 0

The number of messages currently waiting on the DESCR RESPONSE queue
= 0

You can use the following scripts:

- preqrestart – issues a preqstop followed by a preqstart.
- preqstart – starts utl01 and dap61 and dap62, issues a preqshow, and starts the queues in the Banner database.
- preqstop – stops all prereq daemon jobs running.
- preqshow – shows the daemon process currently running and shows the number of messages on the queues.
- preqpurge – empties the contents of the queues; may be useful during testing/setup.

The dap61 listener is for processing Banner registration prerequisite requests, known as CHECK requests.

The dap62 listener is for processing requests when a user clicks on a course in the catalog to get a description (pulled from the Remarks in the block), known as DESCR or description requests.

You need to monitor the number of requests waiting on the message queue. If the number of waiting messages remains high and increases in count, it means that Banner registration is doing a lot of waiting while Degree Works processes the requests. You may need to increase the number of dap61/dap62 processes that are serving your registration prerequisite requests to keep the number of waiting requests low. You can increase the number of processes with the

classic.daemons.dap61.count and classic.daemons.dap62.count Shepherd settings. After changing these values, run a prerestart. However, it is best to do this during times of low or no activity, like during the night. During slow registration periods, such as the drop-add period, it is okay to have many dap61 processes running on your system. Most of them will be waiting for work to do and will not be consuming CPU resources.

You should also monitor the \$ADMIN_HOME/logdebug/preq.log file. You can review this log file to see how many requests are being processed and get a feel for how long each request is taking. However, this log file shows only the time it takes Degree Works to process each request. It does not show how long the requests have been sitting on the Oracle message queue waiting to be processed. This log file will also contain messages about errors that may be occurring so it is good to keep an eye on this file, especially if your users complain of problems with registration.

Also note that increased activity in Banner prerequisite checking will degrade performance in normal Degree Works activities. That is, with limited CPU/memory resources, your web users running what-if audits may see a slowdown when many Banner prerequisite requests are being processed by these daemons.

For the description service, when a prerequisite has no block defined in Degree Works, or a block exists but no remarks are found, the system returns error messages from UCX-CFG098, errors 6201 and 6202.

For additional information about Oracle advanced message queues for prerequisite checking, see the [Using Oracle Advanced Queue Processing to Connect Banner and Degree Works for prerequisite checking](#) article.

Banner access configuration

Special grants are needed to allow your Banner DB Degree Works user to access the advanced queues located within the Banner database. This user is listed in \$DB_LOGIN_BANNER on the Degree Works classic server. You must assign the USR_DEGREE_WORKS role to this user and then compile the Degree Works software.

Scribe Banner prerequisite checking reports

The SCR9X reports help with the management of REQUISITE requirement blocks created with Scribe for use with Banner prerequisite checking.

The following reports can be run in Transit to help with testing and configuring REQUISITE blocks for Banner prerequisite checking:

- [SCR91 - Test Banner Prerequisite Checker Service](#)
- [SCR92 - Test Banner Prerequisite Description Service](#)
- [SCR93 - Report by CATALOG](#)
- [SCR94 - Report by SCHEDULE](#)
- [SCR95 - Report by REQUISITE Block](#)

Related concepts

- [Prerequisite service](#)
- [Registration Prerequisite Checking using DegreeWorks](#)

Technical Configuration Reference

Updated: March 25, 2022

Review configuration reference information as you work with Degree Works.

Shepherd settings

Updated: March 25, 2022

The Shepherd (SHP) Settings define various configurations for the java-based Degree Works software. To manage these settings, you can use the Controller application.

The Shepherd Settings contained in the shp_settings_mst table follow a naming structure that is in the format of module.submodule.name. All of the modules and submodules follow camel case standard notation with no underscores.

Some entries contained in the shp_settings_mst are for client modification, and others should not be modified. Entries that should not be modified by the client are designated with "Do not modify" in the Description field.

Spec value

Updated: March 24, 2023

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

It may be desirable, in certain situations, to specify a different value for these general settings to different applications. The specification field allows you to create or modify configurations based on the application. Each application has a unique spec value. The application will first use the setting that contains that specification. If it is not found, the specification with the default value will be used. Most settings, even the general ones, are not application specific, and so have only the default specification. There are actually very few settings that require separate specifications. The possible specification values are:

Spec	Application
default	All applications. Used if an application specific specification is not found.
composer	Composer
controller	Controller
dashboard	Responsive Dashboard
services	Web services

Spec	Application
transit	Transit
transitbe	Transit Batch Executor
treqadmin	Transfer Equivalency Admin
treqss	Transfer Equivalency Self-Service

Value types

Updated: March 25, 2022

There are three types of values that are used to distinguish the value format for each Shepherd setting.

These types are defined next to the description for each entry. Note that valid Text values are case-sensitive and must be entered exactly as suggested in the setting description. Currently the Shepherd settings support the following value types:

Value type	Accepted format
Block	Free text that may span more than one row (may include carriage returns).
Boolean	true or false
Choice	The field must contain one of the choices provided in the contained "choices" list.
Number	Numeric value only
Text	Free text that does not span more than one row (should not include carriage return).

Module settings

articulation

Updated: March 25, 2022

The articulation.* settings define behavior for the articulation engine used in Transfer Equivalency Admin, Transfer Equivalency Self-Service, and Transfer Finder.

Key	Value Type	Description
articulation.leftoverCourse.courseDiscipline	Text	The course discipline of the default course to use in those instances when no other course can be

Key	Value Type	Description
		identified. E.g. "MATH". This setting is used together with the .courseNumber setting to form the complete course key (e.g. MATH 101).
articulation.leftoverCourse.courseNumber	Text	The course number of the default course to use in those instances when no other course can be identified. E.g. "101". This setting is used together with the .courseDiscipline setting to form the complete course key (e.g. MATH 101).
articulation.resolutionRules.duplicate	Text	The rules to invoke when more than one transfer class maps to a single local class. Valid values for this setting are 'NONE', 'DEFAULT', 'CREDITMAX', 'LOCALHIGH', 'RECENT' and 'TRANSFERHIGH'.
articulation.resolutionRules.undecided	Text	The rules to invoke when a class can be used in multiple possible mappings. Valid values for this setting are 'NONE', 'DEFAULT', 'CREDITMAX', 'LOCALHIGH', 'RECENT' and 'TRANSFERHIGH'.
articulation.ruleOfLastResort	Text	The rule to be used if the list provided in "duplicate" or "undecided" does not result in a definitive resolution (i.e. one mapping chosen). This can be either "RANDOM" or "NONE".

classicConnector

Updated: March 25, 2022

The classicConnector.* settings control the Degree Works Java software interaction with the classic Degree Works interface.

Key	Value Type	Description
classicConnector.amqp.channelCacheSize	Number	The number of channels that are cached. Default=300.
classicConnector.amqp.exchange	Text	The name of the RabbitMQ exchange for requests going to web07. This must be unique for each environment.

Key	Value Type	Description
classicConnector.amqp.timeout	Number	The amount of time in milliseconds that the classic connector will wait for a web07 response. Default=60000.
classicConnector.bufferSize	Number	This setting defines the buffer size read by the classic connector. Do not modify.
classicConnector.dap08.port	Number	This setting controls the Degree Works port number which the Degree Works dap08 application uses for connectivity.
classicConnector.serverCharset	Text	This setting controls the Degree Works communication standard character set. Do not modify.
classicConnector.serverNameOrIp	Text	This setting controls the Degree Works application servername or IP address of the classic server.
classicConnector.timeout	Number	The amount of time in milliseconds that the classic connector will wait to establish the dap08 socket connection. Default=999.

classic.daemons

Updated: March 24, 2023

The classic.* settings control the Degree Works software running on the classic Degree Works server. You must restart to see changes to these settings.

Key	Value Type	Description
classic.daemons.dap25.auditMaximum	Number	The maximum number of audits to read per sleep cycle.
classic.daemons.dap25.auditsPerChild	Number	The number of audits to assign to each dap25 child process.
classic.daemons.dap25.loopMaximum	Number	The maximum number of times to loop. A value of 0 (run forever) is typical. A non-zero value is used only for testing.

Key	Value Type	Description
classic.daemons.dap25.sleepSeconds	Number	The number of seconds to sleep before checking for more new audits.
classic.daemons.dap61.count	Number	The number of dap61 daemons to run to support Banner prereq checking requests.
classic.daemons.dap62.count	Number	The number of dap62 daemons to run to support Banner prereq description requests.
classic.daemons.rad08.count	Number	The number of rad08 daemons to run to support dynamic bridge requests (non-Banner).
classic.daemons.web07.count	Number	The number of web07 daemons to run to support web requests.

core

Updated: September 29, 2023

The core.* settings are general system setting for the Degree Works Java software, controlling database and performance configuration.

Key	Value Type	Description
client.staleDataCheckInterval	Number	This setting defines the time interval, in seconds, after which the application will check that its data is still current. If it finds that the data in the database has changed, it will notify the user and allow them to reload.
core.apiClient.connectTimeout	Number	This setting is used by Degree Works when acting as a client to access a web service. It is the maximum time, in milliseconds, that it will wait to connect to a service before reporting an error. Default value = 5000.
core.apiClient.readTimeout	Number	This setting is used by Degree Works when acting as a client to access a web service. It is the maximum time, in milliseconds, that it will wait on a read from a service before reporting an error.

Key	Value Type	Description
		Default value = 30000.
core.authorization.adviseeFiltering.college.enabled	Boolean	This setting controls whether the option to add college filters for advisee filtering to a user's access record in Controller is enabled. When true, the College option in the Add Filter dialog will be active if the user does not have any filters defined. When false, the College option will still display but will be inactive.
core.authorization.adviseeFiltering.department.enabled	Boolean	This setting controls whether the option to add department filters for advisee filtering to a user's access record in Controller is enabled. When true, the Department option in the Add Filter dialog will be active if the user does not have any filters defined. When false, the Department option will still display but will be inactive.
core.authorization.adviseeFiltering.school.enabled	Boolean	This setting controls whether the option to add school filters for advisee filtering to a user's access record in Controller is enabled. When true, the School option in the Add Filter dialog will be active if the user does not have any filters defined. When false, the School option will still display but will be inactive.
core.credit.format	Text	This setting defines the display format for credits. The # characters can only be integers, but they are optional. 0 characters can only be integers. If left blank they will be replaced with zeroes (this controls the leading/trailing zeroes).
core.gpa.format	Text	This setting defines the display format for the GPA. The # characters can only be integers, but they are optional. 0 characters can only be integers. If left blank they will be replaced with zeroes (this controls the leading/trailing zeroes).

Key	Value Type	Description
core.hibernate.dialect	Text	This setting specifies the database dialect that hibernate will use (for example, Oracle, Mysql, and so on). Currently, Degree Works only accepts the default value. Do not modify.
core.idFactory.idPrefix	Text	This setting defines the prefix string used when creating IDs for Transfer Equivalency Self-Service.
core.institution.calendarType	Text	This setting defines the calendar at the institution (that is, Semester, Quarter, and so on).
core.performance.thresholdMilliseconds	Number	This setting specifies the max number of seconds a java method can run before printing a warning message in the logs. This value is used as a performance measurement.
core.requisite.validate.enable	Boolean	Indicates whether requisite checking should be enabled. If set to true, requisite checking will be enabled and the user will receive a warning if any pre- or co-requisite classes are missing.
core.shpSetting.encryptionKey	Text	Global Encryption Key for SHP settings. The value should have a minimum length of 8 characters.
core.treq.selfService.auditArticulate.report.default	Text	The default value of UCX- RPT036 report type key for Transfer Equivalency Self-Service Audit web service.
core.week.startingDay	Text	Used in Transit recurring scheduling to determine the day that starts a week for a weekly schedule. This may affect how the next date for a recurring schedule is calculated. The default is MONDAY, the ISO standard.

core.amqp

Updated: September 30, 2022

The core.amqp* settings control the behavior of RabbitMQ with Degree Works applications.

Key	Value Type	Description
core.amqp.broadcast.heartbeatSeconds	Number	The number of heartbeat seconds that RabbitMQ ucx-clean and settings-clean connections should be kept alive. This is useful for environments using networking tools or equipment that terminate idle TCP connections between the applications and RabbitMQ, and the recommended value is 55 seconds for most situations. For environments without such timeouts, the recommended value is 0 seconds.
core.amqp.broker.host	Text	The host name or IP address where RabbitMQ server is installed.
core.amqp.broker.port	Number	The port number on which the RabbitMQ server is running. This is usually 5672 for TCP and 5671 for SSL. When this is set to the SSL port core.amqp.useSsl must be set to true; when this is set to the TCP port that setting must be false. Default value = 5672.
core.amqp.exchange.shpSettings	Text	The name of the RabbitMQ exchange used to signal a change in Shep settings. This must be unique for each environment.
core.amqp.exchange.transit	Text	The name of the RabbitMQ exchange used to launch jobs from Transit. This must be unique for each environment.
core.amqp.exchange.ucx	Text	The name of the RabbitMQ exchange used to signal a change in UCX records. This must be unique for each environment.
core.amqp.password	Text	The password for connecting to the RabbitMQ server (use 'rabbitmqctl change_password' to change a user's password)
core.amqp.request.heartbeatSeconds	Number	The number of heartbeat seconds that RabbitMQ dashboard request connections should be kept alive.

Key	Value Type	Description
		This is useful for environments using networking tools or equipment that terminate idle TCP connections between the applications and RabbitMQ, and the recommended value is 55 seconds for most situations. For environments without such timeouts, the recommended value is 0 seconds.
core.amqp.request.timeoutSeconds	Number	The number of seconds web07 should wait for a new request to be placed on the queue. If no request is received before the timeout seconds is hit, web07 will check for a ucx-clean or settings-clean request and then go back to checking for another request using the same timeout value. For most environments, a value of 600 seconds is suggested. For environments using a load balancer, this value should be less than core.amqp.request.heartbeatSeconds.
core.amqp.username	Text	The username for connecting to the RabbitMQ server (use 'rabbitmqctl list_users' to see the list of available users)
core.amqp.useSsl	Boolean	Use SSL connectivity with RabbitMQ. The port must be changed to the SSL port - which is typically 5671. Default=false. See Enable SSL/TLS for Degree Works and RabbitMQ before enabling this setting.
core.amqp.virtualHost	Text	The virtual host for RabbitMQ server; usually this is '/'

core.articulation

Updated: March 24, 2023

The core.articulation* settings define default values for the transfer articulation engine.

Key	Value Type	Description
core.articulation.default.catalogYear	Text	This is the default value for the catalog year that will be used when running an audit. This setting applies to the audit web service for TransferFinder.
core.articulation.default.gradeType	Text	This is the Grade Type value used in conjunction with the STU398 Transfer Grade and core.articulation.default.school to determine which STU385 entry to use when evaluating a minimum grade on a mapping during articulation.
core.articulation.default.school	Text	This is the School value used in conjunction with the STU398 Transfer Grade and core.articulation.default.gradeType or core.articulation.gradeType to determine which STU385 entry to use when evaluating a minimum grade during articulation or generating a transfer audit.
core.articulation.gradeType	Text	This is the Grade Type used in conjunction with the STU398 Transfer Grade and core.articulation.default.school to determine which STU385 entry to use when evaluating minimum grade rules on a transfer audit.
core.articulation.translateGlobalSchoolIds	Boolean	Defines if the School Code will be translated to Local.

core.audit

Updated: September 29, 2023

The core.audit* settings define behaviors for audit generation in various applications such as Transfer Equivalency Self-Service and the PDF audit output generated in Responsive Dashboard and Transit.

Key	Value Type	Description
core.audit.api.report.default	Text	The default report type used in an API Services GET Audit Worksheet request when an HTML response is requested. This must be valid in RPT036 and can be overwritten by

Key	Value Type	Description
		specifying a reportType in the request.
core.audit.api.url	Text	Defines the base URL for your API-Services deployment. This is used by external audit requests to get the image and css resources used in an HTML audit/articulation worksheet.
core.audit.batch.progress.interval.seconds	Number	The number of seconds between checking to see how many audits have been processed so far. Used by DAP22 when running audits created as part of the RAD11 bridge or the Banner extract script and also batch audits run on the classic server. This allows those viewing the audit log to determine approximately when the batch will finish.
core.audit.cpa.latestTermOnly	Boolean	<p>When set to false, all data for previous terms is retained.</p> <p>When set to true, CPA records for older terms will be deleted when new CPA records are built for the student's active term. That is, there will only ever be one term's worth of CPA data for each student's school/degree audit.</p> <p>If this setting is changed from false to true, Ellucian strongly recommends removing the old data before the next time you run DAP22 to create CPA data. Doing so prevents this process from taking a long time to complete while it deletes the records from the database. Your team should consider truncating the dap_result_dtl, dap_resClass_dtl, and dap_resNoncr_dtl tables.</p>
core.audit.printView.dimensions.defaultValue	Text	The default selected key for the print dimensions dropdown in Responsive Dashboard and Transit. Default is LETTER_PORTRAIT.

Key	Value Type	Description
core.audit.printView.repeat.codes	Text	For Responsive Dashboard PDF output generated in the UI and through Transit, when the Show Repeats flag in UCX-RPT036 is enabled the classes with one of these repeat codes will appear with a repeat indicator in the worksheet. The class status on each class is matched against the codes in this comma separated list. This is usually set to the same value as dash.audit.repeat.codes.
core.audit.printView.showNotesInternal	Boolean	For the Responsive Dashboard PDF, if this setting is true, internal notes will be included in the PDF audit. If the setting is false, internal notes will not be included in the PDF audit.
core.audit.printView.showStudentId	Text	For Responsive Dashboard PDF output generated in the UI and through Transit, N means do not show the student ID, 0 (zero) to show the student ID and not mask any characters, and 1-10 to show the student ID and mask this number of characters, starting from the first character.
core.audit.printView.studentHeader.custom.items	Text	For Responsive Dashboard PDF output generated in the UI and through Transit, list the custom data that should appear in the student header of the worksheet. This is usually set to the same value as dash.studentHeader.custom.items.
		To display custom data that should appear only for users of a certain user class, you can use Controller to create a new setting with the user class as a suffix. For example, create new setting dash.worksheets.custom.items.adv with the list of the custom data items that should appear only for users in the ADV user class.
core.audit.printView.studentHeader.goals.items	Text	For Responsive Dashboard PDF output generated in the UI and

Key	Value Type	Description
		through Transit, list the goals that should appear in the student header of the worksheet. This is usually set to the same value as dash.studentHeader.goals.items.
core.audit.process.runOnScribeChanges.enable	Boolean	When viewing the most recent audit in the dashboard, the API, or Transit, run a new audit if any changes were made to the Scribe blocks in the student's audit after the last audit was generated. The default is false.
core.audit.programAsDegree	Boolean	If set to true, Program will be used as the overall Degree while generating an API What-If Audit and the Degree field will be ignored. This is only used by Banner schools and this field must be true when the UCX-CFG020 BANNER Use Program as Degree flag is Y and it must be false when the flag is N.
core.audit.useEtsType	Boolean	If set to true, the ets-type field on the rad-ets-mst for the transfer school associated with each transfer class will be sent to the auditor as an attribute with a key of DWETSTYPE and a value of the ets-type. This DWETSTYPE can then be used in a WITH qualifier when scribing.
core.audit.view.reduceAuditTree.enabled	Boolean	If set to true, performance when generating new audits may be improved by limiting the XML/JSON audit tree to only what is needed by the worksheet as defined by the RPT036 Show flags.

core.security

Updated: September 29, 2023

The core.security* settings control security configurations and behavior for the Degree Works Java software.

Key	Value Type	Description
core.security.authenticationType	Text	<p>Determines how the login credentials are collected. Defaults to SHP, which uses the internal Degree Works login screen for the application. Other valid values are CAS or SAML, which redirect to the appropriate URL configured in other settings. If set to SHP, then one or both of core.security.ldap.enable or core.security.shp.authentication.enabled must be set to true.</p>
core.security.cas.callbackUrl	Text	<p>The URL used by the CAS server to return to the application after authentication has succeeded. This is usually the root of the application. You will need a separate instance of this setting, with different spec values, for each application.</p> <p>For example: https://your.server.com:8443/your_Dashboard</p> <p>Make sure there is no / at the end of the URL.</p>
core.security.cas.idAttribute	Text	The attribute which maps the CAS user to the Degree Works user.
core.security.cas.loginUrl	Text	The URL of the login form for your CAS server. For example: https://cas.myschool..edu/cas/login.
core.security.cas.serverUrlPrefix	Text	The start of the CAS server URL. For example: https://cas.myschool..edu/cas.
core.security.contentSecurityPolicy.connectSrc	Text	<p>Supplement to the Content Security Policy connect-src directive. Do not include self. Changes to this setting require a restart of all web applications. Multiple values should be separated by a single space.</p>
core.security.contentSecurityPolicy.formAction	Text	<p>Supplement to the Content Security Policy form-action directive. Do not include self.</p>

Key	Value Type	Description
core.security.contentSecurityPolicy.frameAncestors	Text	Changes to this setting require a restart of all web applications. Multiple values should be separated by a single space.
core.security.contentSecurityPolicy.imgSrc	Text	Supplement to the Content Security Policy img-src directive. Do not include self . Changes to this setting require a restart of all web applications. Multiple values should be separated by a single space.
core.security.contentSecurityPolicy.scriptSrc	Text	Supplement to the Content Security Policy script-src directive. Do not include self . Changes to this setting require a restart of all web applications. Multiple values should be separated by a single space.
core.security.contentSecurityPolicy.styleSrc	Text	Supplement to the Content Security Policy style-src directive. Do not include self . Changes to this setting require a restart of all web applications. Multiple values should be separated by a single space.
core.security.enableIpAddressCheck	Boolean	This setting defines if it is required that every request comes from the same IP address as when the user logged in. Note, the new IPV6 format is not yet supported.
core.security.externalAccessManager.assertionIsCookie	Boolean	If true, expect to find assertion value in a cookie. If false, expect to find it in a header.
core.security.externalAccessManager.assertionName	Boolean	The name of the external access assertion token.
core.security.externalAccessManager.enable	Boolean	Enable external access manager. It is important to note that care must be takenbe careful to never leave this flag enabled (true)

Key	Value Type	Description
		unless an external access manager is in place, configured and operational. Otherwise, applications are vulnerable to unauthorized access where anyone could impersonate any user.
core.security.ldap.adminDn	Text	Defines the LDAP admin distinguished name (DN). This is used to locate the users dn in the LDAP server.
core.security.ldap.adminPassword	Text	Defines the LDAP admin password. It is stored in an encrypted format.
core.security.ldap.enable	Boolean	When true a user's credentials will be validated against LDAP when If this setting is set to true, LDAP authentication will be enabled. This will only happen, however, if the core.security.authenticationType setting is SHP. All settings prefixed with core.security.ldap should be set appropriately when this setting is true. If set to false when the authenticationType is set to SHP, then the core.security.shp.authentication.enabled must should be set to true or else you would have no data source enabled for authentication. Both SHP and LDAP can be enabled concurrently if desired. Default value is set to false.
core.security.ldap.serverUrl	Text	Defines the LDAP server URL. Example ldaps:// ldap.myschool.edu:636.
core.security.ldap.studentId.attribute	Text	Defines the LDAP attribute that is a SHP user's alternate ID.
core.security.ldap.studentId.suffix	Text	When not empty, this suffix is expected to be found appended to every users' ID attribute, as defined in core.security.ldap.studentId.attribute. It will be removed before the value is used to find the user's SHP record.

Key	Value Type	Description
core.security.ldap.userDnPattern	Text	This setting defines the LDAP user distinguished name (DN) pattern. It is relative to the base defined in the core.security.ldap.serverUrl setting. Use the token {0} to substitute the user's access ID into the pattern. For example, uid={0}. Either this setting or the core.security.ldap.userSearchFilter setting or both must be specified.
core.security.ldap.userSearchBase	Text	This setting defines the LDAP user search base. It is used to find the user's distinguished name. This restricts the scope of the core.security.ldap.userSearchFilter pattern during the search.
core.security.ldap.userSearchFilter	Text	This setting defines the LDAP user search filter. It is used to find the user's distinguished name. This follows the standard LDAP filter expression format as defined in RFC 2254. You should include the special token {0} to represent the user's Access ID as entered in the login screen. For example uid={0}. The scope of this search may be limited by using the core.security.ldap.userSearchBase setting. Either this setting or the core.security.ldap.userDnPattern setting or both must be specified.
core.security.logoutUrl	Text	This setting defines the url to redirect after a successful logout. Only for treqss specification, the value should be / and is not customizable.
core.security.newUser.roles	Text	This setting determines the roles required by the newly registered user to use some of the Degree Works applications.
core.security.newUser.userClass	Text	This setting determines the userClass of the newly registered user in some of the Degree Works applications. .

Key	Value Type	Description
core.security.passport.timeoutIncrementDefault	Number	The default passport timeout increment. This will be used if the core.security.passport.timeoutPrecedence is set to D or no other timeout increments are specified for a user. This setting can have a significant effect on your Java server (for example, Tomcat) memory requirements.
core.security.passport.timeoutMaximumDefault	Number	The default passport timeout maximum. This will be used if the core.security.passport.timeoutPrecedence is set to D or no other timeout maximums are specified for a user.
core.security.passport.timeoutPrecedence	Text	Determines the source of the passport timeout increment and maximum. Setting this to D will cause the timeout values to come from the shp settings core.security.passport.timeoutIncrementDefault and core.security.passport.timeoutMaximumDefault. Setting this to U will cause the timeout values to come from the Shp User record if it is non-zero. Setting this to G will cause the timeout values to come from the Shp Group records for the groups to which the user belongs. If the setting is either G or U, and the initial setting does not result in a non-zero timeout value, then it will try the other option.
core.security.passport.timeoutPreference	Text	Used when setting the passport timeout values from the user's groups. Because a user may belong to multiple groups, this setting specifies which group to use by indicating whether to use the group with the longest timeout maximum or the shortest. The valid values are L for longest timeout maximum, and S for shortest.
core.security.passport.validationInterval	Number	This setting defines the time interval, in milliseconds, after

Key	Value Type	Description
		which the passport in user's session is refreshed.
core.security.passwordCheck.clearText.enabled	Boolean	If this setting is set to true no password encoding methods will be used and passwords will be in clear text form. Currently the default value is set to true.
core.security.passwordCheck.sha1.enabled	Boolean	If this setting is set to true the SHA1 encoding method will be used as a password encoding method. Currently the default value is set to false.
core.security.passwordEncoding.enabled	Boolean	If this setting is set to true it enables password encoding and encrypts user passwords. Currently the default value is set to false, however, setting encoding to false is not recommended due to security reasons.
core.security.referenceUser.password	Text	This setting determines the default user's password. This is to be used for communication with Degree Works classic for applications which allow anonymous users. (This field is encrypted.) This value must be the same as the TREQER user password on the shp_user_mst.
core.security.referenceUser.username	Text	This setting determines the default user's username. This is to be used for communication with Degree Works classic for applications which allow anonymous users.
core.security.referrerAllowableUrls	Text	This setting helps stop cross site request forgeries by limiting the origins of http requests coming into our applications. It is a comma-delimited list of server names, and it should contain the names of all servers that host Degree Works applications. In addition, you must include any server that is referenced by or may redirect to Degree Works such as your CAS, RabbitMQ or portal page server. For example:

Key	Value Type	Description
		degreeworks.myschool.edu, adminapp.myschool.edu, cas.myschool.edu
		Do not use http:// or https:// prefixes.
		This field is required; it must not be empty.
core.security.rules.shpcfg	CLOB	The rules used to assign keys to a user when they login. These dynamic key assignments are based on user attributes, such as their role (e.g. Student, Advisor, etc.). This setting should now be maintained. The SHPCFG file on the classic server is now obsolete.
core.security.saml.assertionConsumerUrl	Text	Optional. If used, it is added to the last part of the AssertionConsumerService URL in the SAML assertion request. The first part is https://server.edu/context/saml/SSO/. It must match the Identity Provider configuration for this service provider. You need to configure this value of this setting for each application by using the spec value associated with that application. Modification of this setting does not require a restart of the applications.
core.security.saml.entityId	Text	The entityId attribute in the SAML assertion request sent to the Identity Provider. This must match the Identity Provider configuration. You must configure a different value for this setting for each application by using the spec value associated with that application. Modification of this setting does not require a restart of the applications.
core.security.saml.idAttribute	Text	The attribute which maps the SAML user to the Degree Works user.

Key	Value Type	Description
core.security.saml.keystore.defaultKey	Text	A default key name to retrieve from the keystore for signing SAML requests. Usually would be set equal to the same value as core.security.saml.keystore.keypair.signingKey. Modification of this setting does not require a restart of the applications.
core.security.saml.keystore.keypair.password	Text	The password for the key specified by core.security.saml.keystore.keypair.signingKey. It is stored in an encrypted format. Modification of this setting does not require a restart of the applications.
core.security.saml.keystore.keypair.signingKey	Text	The alias of key to retrieve from the keystore for signing SAML requests. Modification of this setting does not require a restart of the applications.
core.security.saml.keystore.location	Text	<p>The location of a java keystore which contains certificate for signing SAML requests. This points to a file on your server, and the value should start with file:</p> <p>Example: file:/u01/jdk1.7.0_11/bin/keystore.jks.</p> <p>Modifications of this setting do not require a restart of the applications.</p>
core.security.saml.keystore.password	Text	The password to access the keystore. It is stored in an encrypted format. Modification of this setting does not require a restart of the applications.
core.security.saml.metadata.identityProviderUrl	Text	The URL at the Identity Provider that provides its configuration metadata. Either this setting or core.security.saml.metadata.xml.identityProvider must be provided, with this setting taking precedence over the other. Modification of this setting does not require a restart of the applications.

Key	Value Type	Description
core.security.saml.metadata.xml.identityProviderXML	XML	An XML document which defines the identity provider metadata conforming to schema: urn:oasis:names:tc:SAML:2.0:metadata.
		This setting will be used only if the setting core.security.saml.metadata.identityProviderUrl is empty. Modification of this setting does not require a restart of the applications.
core.security.saml.metadata.xml.serviceProviderXML	Obsolete	Use core.security.saml.assertionConsumerUrl and core.security.saml.entityId to configure the service provider.
core.security.sessionTimeout.warningSeconds	Number	The number of seconds before Degree Works displays a warning and the user session times out. Default value = 180.
core.security.shp.authentication.enabled	Boolean	When true a user's credentials will be validated against the Shep user database. This will only happen, however, if the when core.security.authenticationType setting is SHP. If set to false when the authenticationType is set to SHP, the core.securityldap.enable should be set to true or you would have no data source enabled for authentication. Both SHP and LDAP can be enabled concurrently if desired. Default value is set to true.
core.security.shp.failLoginResetMinutes	Number	The number of minutes after the user has exceeded their maximum number of failed login attempts, as specified in the setting core.security.shp.maxLoginAttempts, that the user may again try to

Key	Value Type	Description
		login. If they try before the given number of minutes, even successful logins will fail. Setting this to zero will disable the check completely.
core.security.shp.maxLoginAttempts	Number	The maximum number of consecutive times a user can fail a login before further logins are ignored (that is, automatically failed). The logins will be failed automatically for a period defined in the setting core.security.shp.failLoginResetMinutes, after which the user may try again.
core.security.singleSignoutEnabled	Boolean	When true, logging out of Degree Works will also log users out of their single sign-on session. There are other configurations tied to this setting. Modification of this setting does not require a restart of the applications. For more information, see SAML single sign-on and sign-off .
core.security.ssold.assertionPattern	Text	A Java style regular expression matching the SAML claim (specified in core.security.saml.idAttribute). If the claim does not match this pattern, the authentication will be rejected. The pattern must include a named capturing group with the name id (for example, (?<id>.+)). This group will be used to extract the value used to retrieve the Degree Works user record by the Single Sign-on ID field.
core.security.stateless.secretkey	Text	The key used for encoding and decoding the stateless token. This will be used to encode the user details while creating the token and to decode the user details from the token during authentication. This key must be at least 32 bytes in length.
core.security.stateless.token.customer.audience	Text	An optional value which will be added to the stateless tokens created by Degree Works. It isn't

Key	Value Type	Description
		validated by applications but may be validated in the future, so its benefit is to add customer-specific identifying information to those tokens.
core.security.stateless.token.customer.issuer	Text	The aud (audience) claim identifies the recipients that the JWT is intended for. In the general case, the aud value is an array of case-sensitive strings, each containing a StringOrURI value.

core.site

Updated: September 29, 2023

The core.* settings are used in Transfer Finder to define site specific variables.

Key	Value Type	Description
core.site.displayName	Text	Defines the school name as it should appear on the transfer audit in Transfer Equivalency Self-Service and Transfer Finder. Default is Local School Name.
core.site.id	Text	The School ID as defined in the SchoolDirectory. It should not be the alternate school ID if the institution uses one. This is used

Key	Value Type	Description
		by etssync to map the rad_ets_user field on the ets_mst for the alternate school ID. For more information, see the School directory topic.
core.site.idType	Text	Defines the Local School Code Type, this is used to translate schools codes with a Global Central reference.

core.whatIf

Updated: March 25, 2022

The core.whatIf* settings define behavior for the What-If functionality.

Key	Value Type	Description
core.whatIf.changeCatalogYear	Boolean	This setting defines whether a user can change the Catalog Year field in the What If audit.
core.whatIf.concTiedToMajor	Boolean	If set to true, the Concentration under the primary area will be filtered by curriculum rules and the selected Major value. The Banner CurrRules extract (RAD35 in Transit) also uses this flag so if this setting is changed, this extract must be rerun.
		Used only when Curriculum Rules is enabled.
		See also the related setting core.whatIf.showAllConcsInPrimary .
core.whatIf.defaultCatalogYear	Boolean	This setting defines what default value displays for the Catalog Year. This will override student data. Leave it blank if you want the what-if to default to student's catalog year.
core.whatIf.degreeBeforeCollege	Boolean	This setting defines the order of the Degree and College fields. If true, Degree will display before the

Key	Value Type	Description
		College field. Used only when Curriculum Rules is enabled.
core.whatIf.enableCurriculumRules	Boolean	<p>Enable filtering of criteria by curriculum rules by setting this to true. If set to false, no filtering will take place: only values from UCX tables where "Show in SEP" is "Y" will be displayed.</p> <p>When this setting is true then setting treq.selfService.goalPresentation is ignored.</p>
		This field is only used by Transfer Equivalency Self-Service. The Responsive Dashboard uses the core.whatIf.filterMode setting to enable Curriculum Rules.
core.whatIf.filterMode	Text	<p>This setting defines what filter mode should be used in what-if audits in the Responsive Dashboard.</p> <p>N=No filtering</p> <p>R=Curriculum rules</p> <p>D=Degree drives major</p> <p>M=Major drives degree/level/college</p> <p>C=Major drives college</p>
core.whatIf.filterConcentrationByMajor	Boolean	<p>Filter concentrations by the chosen major based on the filter fields in UCX-STU563.</p> <p>For Responsive Dashboard, this is only valid when core.whatIf.filterMode is C, D or M. The core.whatIf.showConcentration setting is ignored; the concentration will always show.</p>
core.whatIf.majorRequired	Boolean	This setting defines whether the Major is required in the primary

Key	Value Type	Description
		area. This is only applicable if the Major field displays.
		For Responsive Dashboard, if core.whatIf.filterMode is D, C or M, this setting is ignored and the major will always display.
core.whatIf.mustUsePrimaryRule	Text	<p>This setting defines whether the College and Degree in the additional area will be used from Primary area. Valid values are:</p> <p>D: Additional area must use Degree from primary area</p> <p>C: Additional area must use Degree and College from primary area</p> <p>N: Degree and College in additional area can be different than primary area</p> <p>Used only when Curriculum Rules is enabled.</p>
core.whatIf.programAdditionalRequired	Boolean	<p>If the showProgramAdditional is true, the Program will display in the additional area. If configured to display in the additional area, the Program will display first if programAdditionalRequired = true, or after college/degree/major/concentration/minor if programAdditionalRequired = false.</p> <p>Used only when Curriculum Rules is enabled.</p>
core.whatIf.showAllConcsInPrimary	Boolean	<p>If set to true, and there are no concentrations attached to the selected curriculum rule, show all concentrations in the drop-down in the Areas of study area.</p> <p>Used only when Curriculum Rules is enabled.</p>

Key	Value Type	Description
core.whatIf.showAllConcsInSecondary	Boolean	If set to true, and there are no concentrations attached to the selected curriculum rule, show all concentrations in the dropdown in the Additional areas of study area. Used only when Curriculum Rules is enabled.
core.whatIf.showAllMinors	Boolean	If set to true, and there are no minors attached to the selected curriculum rule, show all minors in the dropdown in the Areas of study area and Additional areas of study area. Used only when Curriculum Rules is enabled.
core.whatIf.showCampus	Boolean	This setting defines whether the Campus will be displayed under the primary area after the Catalog Year. Used only when Curriculum Rules is enabled.
core.whatIf.showCollege	Boolean	This setting defines whether the College will display in the primary and additional areas. For Responsive Dashboard, if core.whatIf.filterMode is C then this setting is assumed to be true and college will be a required field also. If core.whatIf.filterMode is M and this setting is true then college will be a required field.
core.whatIf.showConcentration	Boolean	This setting defines whether the Concentration will display in the primary and additional areas. When core.whatIf.filterConcentrationByMajor is enabled this show setting is ignored.
core.whatIf.showLiberalLearning	Boolean	When this setting is set to true, the Liberal Learning field will display in the Areas of study section for

Key	Value Type	Description
		core.whatIf.filterMode is N, C, D, or M, and in the Additional areas of study section for core.whatIf.filterMode is N or R.
core.whatIf.showMajor	Boolean	This setting defines whether the Major will display on the what-if page. In Responsive Dashboard, this setting is assumed be true when core.whatIf.filterMode is D, C or M.
core.whatIf.showMinor	Boolean	This setting defines whether the Minor will display on the what-if page.
core.whatIf.showProgramAdditional	Boolean	This setting defines whether the Program will display in the additional area.
core.whatIf.showProgramPrimary	Boolean	This setting defines whether the Program will display in the primary area.
core.whatIf.showSchool	Boolean	This setting defines whether the Level (aka School) will display in the primary area.
core.whatIf.showSpecialization	Boolean	When this setting is set to true, the Specialization field will display in the Areas of study section for core.whatIf.filterMode is N, C, D, or M, and in the Additional areas of study section for core.whatIf.filterMode is N or R.

core.workflow

Updated: March 25, 2022

The core.workflow* settings define variable used in the integration between Degree Works and Banner Workflow.

Key	Value Type	Description
core.workflow.externalSourceName	Text	In Banner Workflow, the external source name. Default is DegreeWorks.

Key	Value Type	Description
core.workflow.password	Text	In Banner Workflow, the password for the core.workflow.username. This field is encrypted.
core.workflow.petition.enabled	Text	When enabled, Degree Works will attempt to launch a Workflow event when an exception petition is saved with PE pending status.
core.workflow.petition.eventName	Text	The name of the event to be initiated. DW_PETITION is the sample Banner Workflow event name that is delivered with Degree Works.
core.workflow.petition.modelName	Text	The name of the model to be initiated. DW_PETITION is the sample Banner Workflow model that is delivered with Degree Works.
core.workflow.productTypeName	Text	In Banner Workflow, the product type name. Default is DegreeWorks.
core.workflow.username	Text	In Banner Workflow, the web service username. Default is wfwebservices.
core.workflow.wsdl	Text	The WSDL URL for your Banner Workflow server.

dash

Updated: March 24, 2023

The dash.* settings control the Degree Works Java software interaction with the Responsive Dashboard.

Key	Value Type	Description
dash.audit.athletic.football.enabled	Boolean	Show the football status for those football athletes in the Athletic eligibility details section.
dash.audit.process.inprogress.defaultValue	Boolean	True=Include In-progress check box is checked by default. Additionally, if no include-inprogress value is included in the run-audit request this default value is used.

Key	Value Type	Description
dash.audit.process.inprogress.enabled	Boolean	Show the Include In-progress check box when processing a new audit.
dash.audit.process.preregistered.defaultValue	Boolean	True=Include Preregistered check box is checked by default. Additionally, if no include-preregistered value is included in the run-audit request this default value is used.
dash.audit.process.preregistered.enabled	Boolean	Show the Include Preregistered check box when processing a new audit.
dash.audit.repeat.codes	Text	When the Show Repeats flag in UCX-RPT036 is enabled the classes with one of these repeat codes will appear with a repeat indicator in the worksheet. The class status on each class is matched against the codes in this comma separated list.
dash.classHistory.auditSection.enabled	Boolean	In Class History, show the audit section to which a class applies if it does not apply to a requirement.
dash.classHistory.termSummary.enabled	Boolean	In Class History, after each term, show both a set of term and cumulative calculated values. Any item in the summary can be disabled by localizing the properties file. See the dash.classHistory.termSummary section of DashboardMessages.properties.
dash.courseInformation.enabled	Boolean	This setting defines whether to enable third-party course information functionality in the Responsive Dashboard. When true, course information will be retrieved from the source defined in dash.courseInformation.url, not Banner.
dash.courseInformation.url	Text	The url to use when description.dash.courseInformation.enabled is true.

Key	Value Type	Description
		This should contain the URL and request format specified by the third-party course information vendor.
dash.courseLink.campus.enabled	Boolean	This setting defines whether to display the campus in the course section information shown in CourseLink.
		When true, a column with the campus will display after the meeting times.
dash.courseLink.meetingTime.24hour	Boolean	True=the meeting time format is 24-hours, otherwise it shows with the 12-hour AM/PM format.
dash.exceptionManagement.exceptionsReport.allowSearchOnAll	Boolean	True=users are allowed to search on all exceptions in Exception Management.
dash.exceptions.details.option	Text	N=do not allow exceptions details to be entered. Y=allow exceptions details to be entered. R=require the user to enter exceptions details.
dash.exceptions.runAuditOnEnter	Boolean	True=a new audit is generated when the user goes to the exceptions page.
dash.exceptions.saveNewAudit	Boolean	True=the audit that is run after an exception is save will itself be saved. False=the audit is only shown to the user but not saved to the database.
dash.gpaCalculator.decimals	Number	The number of decimals to show for calculated values.
dash.gpaCalculator.round	Boolean	True=Round the calculated GPA. False=Truncate the GPA at the number of digits specified in the decimals setting.
dash.navigation.links.items	Text	Define a list of items to show in the LINKS menu at the top of the dashboard separated by a comma

Key	Value Type	Description
		and a space. For example, item1, item2, item3. You may add as many links as you link. See the dash.navigation.links properties in the DashboardMessages.properties resource for defining each of the link items.
dash.notes.internal.enabled	Boolean	True=enable the internal notes check box.
dash.notes.predefined.enabled	Boolean	True=enable the predefined notes drop-down.
dash.refresh.external.enabled	Boolean	True=enable the external refresh process.
		See the Student Data Refresh topic. If enabled, the following settings must also be completed: dash.refresh.external.password dash.refresh.external.url dash.refresh.external.username
dash.refresh.external.password	Text	The password part of the basic authentication sent as credentials to the external refresh API.
dash.refresh.external.url	Text	The URL of the external refresh API that is called by the Dashboard when a refresh is requested. It should include a special token {studentId} that will be replaced by the student's Degree Works ID. The optional token {userId} may be used. It will be replaced by the user's Degree Works access ID and may be used for logging or other purposes.
dash.refresh.external.username	Text	The user ID part of the basic authentication sent as credentials to the external refresh API.
dash.search.catalogYear.enabled	Boolean	True=enable the catalog year drop-down on the student search page.
dash.search.college.enabled	Boolean	True=enable the college drop-down on the student search page.

Key	Value Type	Description
dash.search.concentration.enabled	Boolean	True=enable the concentration drop-down on the student search page.
dash.search.custom.cst001.code	Text	Code of custom search item 1 in the advanced search window.
dash.search.custom.cst001.enabled	Boolean	Show custom search item 1 drop-down in the advanced search window.
dash.search.custom.cst002.enabled	Boolean	Show custom search item 2 drop-down in the advanced search window.
dash.search.custom.cst002.code	Text	Code of custom search item 2 in the advanced search window.
dash.search.custom.cst003.enabled	Boolean	Show custom search item 3 drop-down in the advanced search window.
dash.search.custom.cst003.code	Text	Code of custom search item 3 in the advanced search window.
dash.search.custom.cst004.enabled	Boolean	Show custom search item 4 drop-down in the advanced search window.
dash.search.custom.cst004.code	Text	Code of custom search item 4 in the advanced search window.
dash.search.custom.cst005.enabled	Boolean	Show custom search item 5 drop-down in the advanced search window.
dash.search.custom.cst005.code	Text	Code of custom search item 5 in the advanced search window.
dash.search.custom.cst006.enabled	Boolean	Show custom search item 6 drop-down in the advanced search window.
dash.search.custom.cst006.code	Text	Code of custom search item 6 in the advanced search window
dash.search.custom.cst007.enabled	Boolean	Show custom search item 7 drop-down in the advanced search window.
dash.search.custom.cst007.code	Text	Code of custom search item 7 in the advanced search window.
dash.search.custom.cst008.enabled	Boolean	Show custom search item 8 drop-down in the advanced search window.
dash.search.custom.cst008.code	Text	Code of custom search item 8 in the advanced search window.

Key	Value Type	Description
dash.search.custom.cst009.enabled	Boolean	Show custom search item 9 drop-down in the advanced search window.
dash.search.custom.cst009.code	Text	Code of custom search item 9 in the advanced search window.
dash.search.custom.cst010.enabled	Boolean	Show custom search item 10 drop-down in the advanced search window.
dash.search.custom.cst010.code	Text	Code of custom search item 10 in the advanced search window.
dash.search.degree.enabled	Boolean	True=enable the degree drop-down on the student search page.
dash.search.degreeSource.enabled	Boolean	True=enable the degree source drop-down on the student search page.
dash.search.level.enabled	Boolean	True=enable the student class level (aka classification) drop-down on the student search page.
dash.search.liberalLearning.enabled	Boolean	True=enable the liberal learning drop-down on the student search page.
dash.search.major.enabled	Boolean	True=enable the major drop-down on the student search page.
dash.search.maximum	Number	The maximum number of students to be returned in the student search results.
dash.search.minor.enabled	Boolean	True=enable the minor drop-down on the student search page.
dash.search.program.enabled	Boolean	True=enable the program drop-down on the student search page.
dash.search.school.enabled	Boolean	True=enable the school (aka level) drop-down on the student search page.
dash.search.specialization.enabled	Boolean	True=enable the specialization drop-down on the student search page.
dash.search.studentId.mask	Number	How many characters of the student ID should be masked on the main dashboard page. For example, given a 9-character student ID and this mask set to 5 the ID will appear as *****6789 – the first 5 characters are masked.

Key	Value Type	Description
dash.search.studentStatus.enabled	Boolean	True=enable the student status (aka student type) drop-down on the student search page.
dash.studentHeader.custom.enabled	Boolean	True=show the custom items specified in the custom.items setting in the header of the dashboard.
dash.studentHeader.custom.items	Text	<p>An ordered list of custom items to show in the student header separated by a comma. The space after the comma is optional. For example, acadstanding, writexam, mathtest.</p> <p>Each code must be lowercase and either be set up in UCX-SCR002 or UCXRPT046, or in a code on rad_report_dtl. If a value does not exist for a student, then it and its label will not appear in the header. The label for the value must be added to DashboardMessages.properties using the form dash.studentHeader.custom.xxxxx where xxxx is the lowercase value added to this items list.</p> <p>If you want the custom label to appear when the custom value does not exist for the student, add the appropriate property with the none suffix. For example, dash.studentHeader.custom.xxxxx.none =Unknown macro: {label} (no xxxx). If this setting is empty and custom.enabled is true, all report values (but not custom values) will show for the student. This is the default behavior.</p> <p>To display custom data that should appear only for users of a certain user class, you can use Controller to create a new setting with the user class as a suffix. For example, create new setting dash.worksheets.custom.items.adv with the list of the custom data</p>

Key	Value Type	Description
		<p>items that should appear only for users in the ADV user class.</p> <p>The order of the items in this list determines the order the items will appear in the student header.</p> <p>Also see <code>dash.worksheets.custom.items</code> for the planner audit.</p>
<code>dash.studentHeader.goals.items</code>	Text	<p>An ordered list of goal items to show in the student header, separated by a comma. The space after the comma is optional. The full list of goals is school, classification, major, minor, program, conc, college, spec, libl. Each code must be lowercase. To hide a goal from showing simply remove it from the list. If a goal does not exist for a student then it and its label will not appear in the header. If you want the goal label to appear when the goal does not exist then add the appropriate property with the “none” suffix. For example, <code>dash.home.context.MAJORFormat.none={label}</code> (no major). See the baseline properties file for a full list of the none goal properties. The order of the items in this list determines the order the goals will appear in the student header.</p> <p>Also see <code>dash.worksheets.goals.items</code> for the planner audit.</p>
<code>dash.worksheets.custom.enabled</code>	Boolean	True=show the custom items specified in the <code>custom.items</code> setting in the planner audit.
<code>dash.worksheets.custom.items</code>	Text	<p>An ordered list of custom items to show in the planner audit separated by a comma. The space after the comma is optional. For example, <code>acadstanding, writexam, mathexam</code>.</p>

Key	Value Type	Description
		<p>Each code must be lowercase and either be set up in UCX-SCR002 or UCXRPT046, or in a code on rad_report_dtl. If a value does not exist for a student, then it and its label will not appear in the header. The label for the value must be added to DashboardMessages.properties using the form dash.worksheets.custom.xxxxx where xxxx is the lowercase value added to this items list.</p> <p>If you want the custom label to appear when the custom value does not exist, add the appropriate property with the “none” suffix. For example, dash.worksheets.custom.xxxxx. none = Unknown macro: {label}(no xxxx). If this setting is empty and custom.enabled is true, all report values (but not custom values) will show for the student. This is the default behavior.</p>
		<p>To display custom data that should appear only for users of a certain user class, you can use Controller to create a new setting with the user class as a suffix. For example, create new setting dash.worksheets.custom.items.adv with the list of the custom data items that should appear only for users in the ADV user class.</p>
		<p>The order of the items in this list determines the order the items will appear in the planner audit.</p>
		<p>Also see dash.studentHeader.custom.items for the student header.</p>
dash.worksheets.goals.items	Text	An ordered list of goal items to show in the planner audit, separated by a comma. The space after the comma is optional. The full list of goals is school,

Key	Value Type	Description
		<p>classification, major, minor, program, conc, college, spec, libl. Each code must be lowercase. To hide a goal from showing simply remove it from the list. If a goal does not exist for a student then it and its label will not appear in the header. If you want the goal label to appear when the goal does not exist then add the appropriate property with the none suffix. For example, dash.worksheets.context.MAJORFormat.none={label} (no major). See the baseline properties file for a full list of the none goal properties. The order of the items in this list determines the order the goals will appear in the planner audit.</p> <p>Also see dash.studentHeader.goals.items for the student header.</p>

email

Updated: September 29, 2023

These email settings are used to control where emails should be sent when certain processes complete.

You can use a single email address, or you can string together a list of email addresses separated by semicolons (do not include spaces). For example:

user.name1@myschool.edu;user.name2@myschool.edu

All of these settings are optional. However, if you do populate any of the toEmail fields with more than one email address, the fromEmail field must be filled in.

Key	Value Type	Description
core.audit.batch.notification.toEmail	Text	Email address(es) where results of running batch audits should be sent.
core.audit.whatif.alternate.batch.notification.toEmail	Text	Email address(es) where results of running alternate what-if batch audits should be sent.

Key	Value Type	Description
core.audit.whatif.batch.notification.toEmail	Text	Email address(es) where results of running batch what-if audits should be sent.
core.notification.ccEmail	Text	When an email is sent, the address(es) specified here will be copied on the email.
core.notification.fromEmail	Text	When emails are sent, they will appear to be from this email address. If the recipient of the email replies, the email will be sent to this address. Note: If left blank and the associated toEmail setting is populated, the toEmail address will be used as the from address. To prevent errors, be sure to populate fromEmail if two or more addresses are entered in one of the toEmail fields.
integration.banner.sap.notification.toEmail	Text	Email address(es) where results of running Banner SAP processor should be sent.
integration.capture.notification.toEmail	Text	Email address(es) where results of running the Banner extract or bridge should be sent.
parser.batch.notification.toEmail	Text	Email address(es) where results of running DAP16 should be sent.
treq.export.articulation.notification.toEmail	Text	Email address(es) where results of running the Transfer Equivalency export should be sent.

integration.banner

Updated: September 29, 2023

These entries define various default data values used by Banner schools.

See also the additional Banner flags in [UCX-CFG020 BANNER](#).

Key	Value Type	Description
integration.banner.extract.<user class>.sql.daily	CLOB	<p>The default sql to use when running the daily/nightly Banner extract for the given user class using the bannerextract script or when running RAD33 Banner Non-Student Extract and Bridge in Transit. This sql should select your active pool of users for the user class from Banner to pull over into Degree Works. For example, when extracting deans, this setting would be integration.banner.extract.dean.sql.daily. For advisors, you can have both adv and advx settings depending on which user classes you are using.</p>
integration.banner.extract.applicant.sql.daily	CLOB	<p>The default sql to use when running the daily/nightly applicant Banner extract using the bannerextract script or when running RAD32 Banner Applicant Extract and Bridge in Transit. This sql should select the applicants from Banner to pull in to Degree Works.</p>
integration.banner.extract.student.sql.daily	CLOB	<p>The default sql to use when running the daily/nightly student Banner extract using the bannerextract script or when running RAD30 Banner Student Extract and Bridge in Transit. This sql should select your active pool of students from Banner to pull over into Degree Works.</p>
integration.banner.extract.cache.timeoutMinutes Number		<p>How often the Banner extract as part of web07 should refresh its cache of Banner records (STVTERM, SSBSECT, SHRGRDE, and so on). If you are doing a nightly webrestart, you can choose to disable this by setting it to 0. A setting of 1440 minutes means the cache will be cleaned one time a day, which essentially has no affect if you are doing a webrestart each day. When the cache is cleaned, the current student being extracted will take a</p>

Key	Value Type	Description
		little longer to be refreshed, so using a low timeout value of 10 minutes, for example, will impact performance of your web requests. This setting has no impact on the Banner batch extract because the cache is loaded on startup one time only.
integration.banner.extract.config	CLOB	Contains the SQL WHERE and FROM clauses used by the Banner extract used to select student and other data.
integration.banner.extract.curriculumRules.multipleRanges.enabled	Boolean	true to extract multiple ranges for curriculum rules that were disabled and enabled again in different catalog years. Do the same for the associated majors, minors, and concentrations. Also allow these items to be automatically ended if a new year starts but the item is not included in the new year, even if SORMCRL_DAU_IND is not set to N.
integration.banner.extract.equiv.crosslistedNewInd	Boolean	Set to true when you want CFG073 cross-listed records created for SCREQIV cross-listings, even if the new course on SCREQIV is not active in SCBCRSE or SCBCRKY.
integration.banner.extract.equiv.crosslistedRange	Boolean	Set to true when you want CFG073 cross-listed records created for SCREQIV cross-listings with a range of terms, where the END-TERM is not 999999. A pair of CFG073 records is created for the start term and the end term with an operator of RS (range start) and RE (range end).
integration.banner.extract.progress.interval.seconds	Number	The number of seconds between checking to see how many students have been processed so far. Used by the bannerextract script when extracting students/applicants only. This allows those viewing the extract log to determine approximately when the extract will finish.

Key	Value Type	Description
integration.banner.extract.staff.name.format	Text	<p>Define how to format staff names. This setting is a string of tokens that are replaced by the name components. For example, <last>, <first> <middle></p> <p>You can use <finitial> in place of <first> to get the first initial.</p> <p>You can use <minitial> in place of <middle> to get the middle initial, or you can choose to omit the middle name completely.</p>
integration.banner.extract.staff.name.pref.enabled	Boolean	When true, use the staff.name.pref.format instead of the staff.name.format setting if the person has a preferred name.
integration.banner.extract.staff.name.pref.format	Text	<p>See the description for the staff.name.format setting above. In addition, for this setting you can specify the person's preferred first name as <pref>. Example format:</p> <p><last>, <first> <middle> (<pref>)</p> <p>Here, parentheses are placed around the preferred name but that is purely a suggestion. The above format would give you the following name if the person had a preferred name of Bob:</p> <p>Smith, Robert William (Bob)</p> <p>This setting is only used if staff.name.pref.enabled is true and if the person has a preferred name and if it is different from the first name.</p>
integration.banner.extract.student.name.format	Text	<p>Define how to format student names. This setting is a string of tokens that are replaced by the name components. For example, <last>, <first> <middle></p> <p>You can use <minitial> in place of <middle> to get the student's</p>

Key	Value Type	Description
		middle initial, or you can choose to omit the middle name completely.
		For searching on students you must specify <last> as the first token followed by a comma.
integration.banner.extract.student.name.pref.enabled	Boolean	When true, use the student.name.pref.format setting instead of the student.name.format setting if the student has a preferred name.
integration.banner.extract.student.name.pref.format	Text	See the description for the student.name.format setting above. In addition, for this setting you can specify the student's preferred first name as <pref>. Example format:
		<last>, <first> <middle> (<pref>)
		Here, parentheses are placed around the preferred name but that is purely a suggestion. The above format would give you the following name if the student had a preferred name of Bob:
		Smith, Robert William (Bob)
		This setting is used only if student.name.pref.enabled is true and if the student has a preferred name and if it is different from the first name.
integration.banner.extract.student.useAdmitTerm	Boolean	When true, the extract will use the SORLCUR_TERM_CODE if it is not a future term and it is greater than the SORLCUR_TERM_CODE_ADMIT. Otherwise, the most recent SORLCUR_TERM_CODE_ADMIT is used as the active term bridged to Degree Works, even if it is a future term. When false, use the SORLCUR_TERM_CODE. Note: Classes are also used in the logic

Key	Value Type	Description
		to determine the active term.
		For more information, see R011PRIM - Primary Record .
integration.banner.registration.apiUrl	Text	The base URL where the Banner registration API is deployed and required to be up and running. It should be an absolute URL (including the /api/registration-register endpoint). For example: <code>https://server.myschool.edu:7443/production/banner/registration/api/registration-register</code>
		This ERP registration API is part of the Student APIs deployment on Banner.
integration.banner.registration.enable	Boolean	Enable Banner registration from Course Link in Responsive Dashboard.
integration.banner.registration.password	Text	Defines the password for the user to authenticate to the Banner registration API. (This field is encrypted.)
integration.banner.registration.username	Text	Defines the user name to authenticate to the Banner registration API.

integration.bridge.audits

Updated: September 29, 2023

These entries define the flags for running the bridge.

Key	Value Type	Description
integration.bridge.audits.buildCpaResults	Boolean	Build CPA data when running a bridge audit.
integration.bridge.audits.includeInProgress	Boolean	Include in-progress classes when running a bridge audit.
integration.bridge.audits.includePreregistered	Boolean	Include preregistered classes when running a bridge audit.

localization

Updated: March 25, 2022

The localization.* settings configure the general behavior of how localizations are handled.

Key	Value Type	Description
localization.css.enable	Boolean	Set to false to disable localized CSS in any application, so only baseline CSS will be used. Default is true.
localization.generatedPages.enable	Boolean	Set to false to disable localized SHP Scripts in Dashboard, so only baseline Scripts will be used. Default is true.
localization.images.cacheMilliseconds	Number	Number of milliseconds browser will cache localized images. New browser sessions or visitors can't cache until after the first request. So, you can bypass by clearing your browser cache or doing a forced cache refresh.
localization.images.enable	Boolean	Localization of images is enabled by default. If you want to ignore any localizations that may be present, either system wide or for a particular project Spec, set this to false.
localization.internationalization.cacheMilliseconds	Number	Localizations to internationalization properties will be cached for maximum this number of milliseconds. So, wait at least this long after making modifications to properties in Composer before refreshing browser to see those changes reflected.
localization.internationalization.enable	Boolean	Set to false to disable localized internationalization properties in any application, so only baseline properties will be used. Default is true.
localization.xsl.enable	Boolean	XSL localization is enabled by default. Create localizations in Composer. Set to false to disable localized XSL in any application so only baseline XSL files will be used.

scribe

Updated: March 25, 2022

The scribe.* settings configure the general behavior of the Scribe administrative module.

Key	Value Type	Description
scribe.query.threshold.requisite	Number	This setting defines the maximum number of REQUISITE blocks that will be returned when searching for requirement blocks in Scribe. If the number of matching blocks exceeds this value, the user will be prompted to refine their query to return fewer results.
scribe.save.enable.confirmation	Boolean	The setting causes the user to be prompted to confirm that they want to update an existing block. By default, this setting is set to true.
scribe.secondaryTag.showCollege	Boolean	This setting defines whether the College drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showConcentration	Boolean	This setting defines whether the Concentration drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showDegree	Boolean	This setting defines whether the Degree drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showLiberalLearning	Boolean	This setting defines whether the Liberal Learning drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showMajor1	Boolean	This setting defines whether the Major 1 drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showMajor2	Boolean	This setting defines whether the Major 2 drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showMinor	Boolean	This setting defines whether the Minor drop-down list will be

Key	Value Type	Description
		displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showProgram	Boolean	This setting defines whether the Program drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showSchool	Boolean	This setting defines whether the School drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showSpecialization	Boolean	This setting defines whether the Specialization drop-down list will be displayed in the Secondary Tags section of the search screen.
scribe.secondaryTag.showStudentId	Boolean	This setting defines whether the Student ID field will be displayed in the Secondary Tags section of the search screen.
scribe.template.newBlock	Text	The setting contains the block template in the clob field.
scribe.upshift.codes	Boolean	Defines whether Scribe upshifts the REQUISITE and OTHER block values. If true, all alpha characters a user types will be upshifted in the user interface and when saved to the database.

studentPlanner

Updated: September 30, 2022

The studentPlanner.* settings configure general behavior for both Plans and Template Management.

Key	Value Type	Description
studentPlanner.choiceReq.enableCourseAttribute	Boolean	This setting defines whether to enable the course attributes for courses in the choice requirement. If true, the course attribute field is displayed alongside the course field in the choice requirement.
studentPlanner.choiceReq.enableScribePointer	Boolean	This setting defines whether to enable the scribe pointer field for the choice requirement. If true, the Pointer field is displayed for the

Key	Value Type	Description
		choice requirement. This property is set to false by default.
studentPlanner.courseLink.enable	Boolean	This setting defines whether to enable Course Link functionality on plans and templates.
studentPlanner.courseReq.showCampus	Boolean	This setting defines whether the campus field is displayed for course and choice requirements.
studentPlanner.courseReq.showCritical	Boolean	This setting defines whether the critical check box is displayed for all requirements except placeholders.
studentPlanner.courseReq.showDelivery	Boolean	This setting defines whether the delivery field is displayed for course and choice requirements.
studentPlanner.courseReq.showGrade	Boolean	This setting defines whether the minimum grade field is displayed for course and choice requirements.
studentPlanner.courseReq.showHonors	Boolean	This setting defines whether the honors check box is displayed for course and choice requirements.

studentPlanner.planner

Updated: September 30, 2022

The studentPlanner.planner* settings configure the overall behavior of Plans.

Key	Value Type	Description
studentPlanner.planner.addOnlyCoursesOffered	Boolean	This setting defines whether a course can be added to a term in which it is not offered, based on CFG072. If true, the user will be given the option to add the course anyway. If false, the course cannot be added to the term.
studentPlanner.planner.allowChangesToApproved	Boolean	This setting allows users to modify plans that have already been approved. If true, users can modify approved plans and resubmit them for approval. If false, users must use the Save As option to save the plan with modifications, and that

Key	Value Type	Description
studentPlanner.planner.allowChangesToCurrentTerm	Boolean	<p>new plan needs to be approved again. Note: Only used if studentPlanner.planner.planApprovalMethod = W.</p>
studentPlanner.planner.allowChangesToCurrentTerm	Boolean	<p>This setting defines whether to allow changes to plan requirements in the student's current or active term. If true, a requirement in a current term can be modified or deleted.</p> <p>However, if studentPlanner.planner.tracking.enable is true, this setting will restrict only changes to ONTRACK requirements on a tracked plan. OFFTRACK and not tracked requirements in the student's current term can still be modified or deleted, regardless of this setting.</p>
studentPlanner.planner.allowChangesToPastTerms	Boolean	<p>This setting defines whether to allow changes to plan requirements in the student's past terms. If true, a requirement in a past term can be modified or deleted.</p> <p>However, if studentPlanner.planner.tracking.enable = true, this setting will restrict only changes to ONTRACK requirements on a tracked plan. OFFTRACK and not tracked requirements in the student's past terms can still be modified or deleted, regardless of this setting.</p>
studentPlanner.planner.allowMultipleActivePlans	Boolean	<p>This setting defines whether to allow multiple active plans for the same degree. If false, a student can have only one active plan per degree.</p>
studentPlanner.planner.audit.usePlannedCourses	Boolean	<p>This setting defines whether planned courses should be included when generating a planner audit. If true, both</p>

Key	Value Type	Description
		registered and planned courses for the student's active term are included in the audit. Otherwise, only the registered courses for the student's active term are included in the audit, provided the student has registered courses in the active term.
studentPlanner.planner.audit.usePreregistered	Boolean	This setting defines whether preregistered classes should be included when generating a planner audit. If a planned class is a duplicate of a preregistered class on the same term, it will not be included when this setting is true.
studentPlanner.planner.createBlock.classIndent	Number	Number of spaces to indent the Class/Credits/RuleIncomplete line when creating a Scribe block from a plan.
studentPlanner.planner.createBlock.dateFormat	Text	Format for dates when creating a Scribe block from a plan.
studentPlanner.planner.createBlock.hourFormat	Text	Format for time when creating a Scribe block from a plan.
studentPlanner.planner.createBlock.labelIndent	Number	Number of spaces to indent Label/ProxyAdvice line when creating a Scribe block from a plan.
studentPlanner.planner.createBlock.placeholderLabelPrefix	Text	Value that will precede the placeholder requirement literal when creating a Scribe block from a plan.
studentPlanner.planner.createBlock.planBlockTitleText	Text	Title of the Scribe block created from a plan. If not specified, the default value of Planner Block will be used.
studentPlanner.planner.createBlock.planBlockValueText	Text	Block value of the Scribe block created from a plan. If not specified, the default value of PLAN will be used.
studentPlanner.planner.createBlock.showCredits	Boolean	If true, the created Scribe block rules will use - x Hours after the class title in the label for each class requirement. If false, the rules will show only the class title in the label for each class requirement.

Key	Value Type	Description
studentPlanner.planner.createBlock.subsetLabelPrefix	Text	Value that will precede the term literal in the Scribe block created from a plan.
studentPlanner.planner.createBlock.subsetsForTerm	Boolean	This setting defines whether a subset of requirements will be created for each planner term or if all planner requirements will be written in one set in the Scribe block created from a plan.
studentPlanner.planner.createBlock.useClassesIn	Boolean	If true, the rules will use 1 Class in before each class requirement. If false, the rules will use x Credits in before each class requirement in the Scribe block created from a plan.
studentPlanner.planner.currentTerm.isDeterminedByUcx	Boolean	This setting defines whether the current term for a student is determined from a UCX table. If true, the current term for the student is determined from STU016. Otherwise, the student's active term is used as the current term.
studentPlanner.planner.filterCoursesByPlanSchool	Boolean	This setting defines whether or not to filter out the courses offered by the plans school (aka level). When this is true, then only the courses offered by the plans school (those that have a rad_crs_attr_dtl with rad_attr_code = DW-SCHOOL and rad_attr_value equal to the school on the plan) will be displayed on the Courses list. When this is false, all courses will be displayed depending on how studentPlanner.planner.filterCoursesNoLongerOffered is configured.
studentPlanner.planner.filterCoursesNoLongerOffered	Boolean	This setting defines whether to filter out the courses that are no longer offered. When this is true, if the students active term is greater than rad_course_mst.rad_cat_yr_stop, the course will not appear in the list. When this is false, all courses will be displayed depending on how

Key	Value Type	Description
studentPlanner.planner.filterCoursesByPlanSchool	Boolean	This setting defines whether studentPlanner.planner.filterCoursesByPlanSchool is configured.
studentPlanner.planner.notes.showComesFromTemplateIndicator	Boolean	This setting defines whether to display a visual indicator in the plan note list window to show that a note came from a template. If true, the indicator will be displayed when a note is copied to a plan from a template.
studentPlanner.planner.planApprovalMethod	Text	Enables and disables plan approval functionality. Valid options are: W for Banner Workflow, L for Locking, or N for no approval or locking.
studentPlanner.planner.showDisclaimer	Boolean	This setting defines whether to show a disclaimer on the printed plan. The disclaimer text is defined in planner.planDetail.disclaimer in PlannerMessages.properties.
studentPlanner.planner.showScope	Text	This setting will control whether the plan scope (type) is visible on the screen, and whether it is required. Possible values are: No - do not display scope, Optional - display scope but it is not required, and Required - display scope and require a valid value.
studentPlanner.planner.sidebar.coursesOpenedByDefault	Boolean	This setting defines whether to initially show the course list in the Plans sidebar and requirement drawer by default. If false, the still needed list will display initially.
studentPlanner.planner.sidebar.enableCourses	Boolean	This setting defines if the courses list displays in the Plans sidebar and requirement drawer.
studentPlanner.planner.sidebar.enableStillNeeded	Boolean	This setting defines if the still needed list displays in the Plans sidebar and requirement drawer.
studentPlanner.planner.sidebar.sidebarOpenedByDefault	Boolean	This setting defines whether to expand the Plans sidebar by default.
studentPlanner.planner.tracking.allowTakenTermDifferentThanPlanned	Boolean	This setting defines whether a planned course can be taken after the planned term, with the term and plan still considered ONTRACK. If true, even if a student takes a planned course in

Key	Value Type	Description
studentPlanner.planner.tracking.batch.numberOfStudentsToProcess	Integer	a term after the term defined as the planned term, the term and plan will still be considered ONTRACK.
studentPlanner.planner.tracking.blankGradeTransfersMeetMinimumGrade	Boolean	Determines how many students will be processed on each batch iteration during DAP58 execution.
studentPlanner.planner.tracking.currentGpaMeetsRequirement	Boolean	This setting defines whether to consider transfer classes passed with a blank grade as passed with the minimum grade met.
studentPlanner.planner.enable	Boolean	This setting allows GPA requirements in the student's current term to always be ONTRACK. Because grading may not be complete for all classes in the current term, the calculated GPA may be incorrect and cause the requirement to be OFFTRACK. If true, GPA requirements on the student's current term will always be ONTRACK. When the term becomes historical, the GPA will be correctly calculated and evaluated by tracking.
studentPlanner.planner.incompleteClassesMeetRequirement	Boolean	This setting defines whether incomplete classes should be evaluated as satisfying a requirement by tracking.
studentPlanner.planner.offTrackTermsAllowed	Integer	The number of OFFTRACK terms that make the overall plan OFFTRACK.
studentPlanner.planner.passFailClassMeetMinimumGrade	Boolean	This setting defines whether passed classes taken as pass/fail satisfy a minimum grade on a requirement by tracking.
studentPlanner.planner.showTrackingForCriticalRequirements	Boolean	When enabled, only requirements marked as critical will display a tracking status.
		Non-critical requirements may be evaluated by tracking if studentPlanner.planner.trackCriticalRequirements is false, but if studentPlanner.planner.showTrackingForCriticalRequirements is true, only critical requirements will be tracked.

Key	Value Type	Description
studentPlanner.planner.tracking.showTrackingText	Boolean	is false, the tracking status will not display in Plans.
studentPlanner.planner.tracking.showTrackingText	Boolean	When true, the plan, term, and requirement tracking statuses will display on a tracked plan.
studentPlanner.planner.tracking.trackCriticalRequirementsOnly	Boolean	This setting defines whether to consider non-critical requirements when calculating the term tracking status. When true, tracking will be evaluated only for requirements marked as critical.
		If studentPlanner.planner.tracking.trackCriticalRequirementsOnly is true and studentPlanner.planner.tracking.showTrackingForCriticalRequirementsOnly is false, the tracking statuses for non-critical requirements will not be included in the term tracking status.
studentPlanner.planner.tracking.web.processInOfficialMode	Boolean	When tracking is triggered by the web application, this setting determines whether tracking is processed in official mode. If false, tracking is processed in unofficial mode.
		Ellucian recommends that you do not change this setting.
studentPlanner.planner.tracking.web.processTrackingStatusCurrent	Boolean	When tracking is triggered by the web application and if processInOfficialMode = true, each plan processed by tracking will have its is_tracking_status_current field set to this value.
		Ellucian recommends that you do not change this setting.
studentPlanner.planner.tracking.web.updatePlanTrackingStatus	Boolean	When tracking is triggered by the web application, this setting determines whether the tracking status fields will be updated on each plan processed by tracking.

Key	Value Type	Description
studentPlanner.planner.tracking.web.updateTermBookingStatus	Boolean	Ellucian recommends that you do not change this setting.
studentPlanner.planner.updateCreditsIfChanged	Text	When tracking is triggered by the web application, this setting determines whether the tracking status fields will be updated on each term processed by tracking. Ellucian recommends that you do not change this setting.
studentPlanner.planner.updateCreditsIfChanged	Text	This setting defines whether to update credits on a plan if the credits value on the rad_course_mst changed after the requirement was added. If false, the user will not be able to save the plan if the credits have changed. If true, the user can save the plan and the credits on the plan requirement will be updated with the value on the rad_course_mst. If it is set to warn, the user will get a warning that the credits are different when saving the plan, but the credits on the plan requirement will not be updated. The value will be set to false by default.
studentPlanner.planner.workflow.plannerEventName	Text	Event name to be used for Banner Workflow for plan approval. The event name default is DW_APPROVE_SEP_PLAN.
studentPlanner.planner.workflow.plannerModelName	Text	Model name to be used for Banner Workflow for plan approval. The model name default is DW_APPROVE_SEP_PLAN.

studentPlanner.template

Updated: September 30, 2022

The studentPlanner.template* settings configure the overall behavior of Template Management.

Key	Value Type	Description
studentPlanner.template.columnsToDisplay.modifiedByDate	Boolean	This setting defines whether to display the Modified date in the template list of the flat view.
studentPlanner.template.columnsToDisplay.modifiedByWho	Boolean	This setting defines whether to display the modify What in the template list of the flat view.
studentPlanner.template.columnsToDisplay.modifiedWho	Boolean	This setting defines whether to display the modify Who in the template list of the flat view.
studentPlanner.template.columnsToDisplay.templateId	Boolean	This setting defines whether to display the template ID in the template list of the flat view.
studentPlanner.template.columnsToDisplay.termScheme	Boolean	This setting defines whether to display the Term scheme in the template list of the flat view.
studentPlanner.template.initialDepth	Number	This setting defines the number of tag levels that should initially expand in the tree view. It will be used only if studentPlanner.template.showTreeView is true.
studentPlanner.template.showFlatView	Boolean	This setting defines if the flat view of templates should be displayed. It should be true if studentPlanner.template.showFlatViewByDefault is true.
studentPlanner.template.showFlatViewByDefault	Boolean	This setting defines whether to initially display the flat view of templates by default. If true, studentPlanner.template.showFlatView should also be true. If false, studentPlanner.template.showTreeView should be true.
studentPlanner.template.showTreeView	Boolean	This setting defines if the tree view of templates should be displayed. It should be true if studentPlanner.template.showFlatViewByDefault is false.
studentPlanner.template.sidebar.enableCourses	Text	This setting defines if the courses list is visible in the Template Management sidebar and requirement drawer.
studentPlanner.template.sidebar.sidebarOpenedByDefault	Boolean	This setting defines whether to expand the Template Management sidebar by default.

theme

Updated: September 30, 2022

The theme* settings control the Responsive Dashboard color scheme and logo. Changes to these settings impact all users so make sure to grant access to users authorized to make these changes.

Key	Value Type	Description
theme.color.primary	String	Defines the main color of your Responsive Dashboard deployment.
theme.color.secondary	String	Defines the second color of your Responsive Dashboard deployment.
theme.color.tertiaryName	String	Selection of available colors defines the third color of your Responsive Dashboard deployment.
theme.url.logo	String	URL of your logo graphic. This must be accessible to Responsive Dashboard users so it may need to reside outside your network or firewall. The host server also must be added to the Content Security Policy (CSP) by adding it to the Shepherd setting core.security.contentSecurityPolicy.imgSrc.

transferFinder

Updated: March 25, 2022

The transferFinder* setting configures Transfer Finder standard defaults.

Key	Value Type	Description
transferFinder.acknowledgement.show	Boolean	Turns on an acknowledgement message that the user must agree to before accessing the tab. Defines if the notification message will be displayed.

Key	Value Type	Description
transferFinder.courseSearch.enableSchoolSelection	Boolean	Enable ability to select a partner school and retrieve mappings from selected school.
transferFinder.courseSearch.limitResultsToPartners	Boolean	Determines if course equivalency results in Transfer Finder are limited to partner institutions.
transferFinder.creditType	Text	This setting defines the school's type of credit.
transferFinder.snapshot.audit.process.includeTransferWork.codes	Text	A semicolon-delimited list of transfer codes that will be included in Classwork History and Transfer Snapshot.
transferFinder.snapshot.audit.process.includeTransferWork.defaultValue	Boolean	Set to true if you want the corresponding switch to default to switched on or if you want additional transfer work included when the switch is not displayed.
transferFinder.snapshot.audit.process.includeTransferWork.enabled	Boolean	Display a switch that allows users to indicate they want additional transfer work included in Classwork History and Transfer Snapshot.
transferFinder.snapshot.audit.process.inprogress.defaultValue	Boolean	True=Include In-progress check box is checked by default. Additionally, If transferFinder.snapshot.audit.process.preregistered.enabled is false, then this is the default value used to run an audit.
transferFinder.snapshot.audit.process.inprogress.enabled	Boolean	Show the Include In-progress check box when processing a new audit
transferFinder.snapshot.audit.process.preregistered.defaultValue	Boolean	True=Include Preregistered check box is checked by default. Additionally, If transferFinder.snapshot.audit.process.preregistered.enabled is false then this is the default value used to run an audit.
transferFinder.snapshot.audit.process.preregistered.enabled	Boolean	Show the Include Preregistered check box when processing a new audit

Key	Value Type	Description
transferFinder.snapshot.enableDisciplines	Boolean	Show Academic Disciplines in Transfer Snapshot search criteria side bar.
transferFinder.snapshot.enablePaths	Boolean	Show Transfer Paths in Transfer Snapshot search criteria side bar.

transferFinder.central

Updated: March 25, 2022

The transferFinder.central* settings configures Transfer Finder standard defaults for the central server.

Key	Value Type	Description
transferFinder.central.classSchedule.banner.password	Text	Password used to authenticate to Banner API at all partner sites. This field is encrypted.
transferFinder.central.classSchedule.banner.termEndOffset.numberOfDays	Text	Number of days to add from today's date when retrieving terms for class schedule from Banner API.
transferFinder.central.classSchedule.banner.termStartOffset.numberOfDays	Text	Number of days to subtract from today's date when retrieving terms for class schedule from Banner API.
transferFinder.central.classSchedule.banner.username	Text	Username used to authenticate to Banner API at all partner sites.
transferFinder.central.directory.password	Text	Defines the password required to access the directory central webservices for Transfer Finder. This field is encrypted.
transferFinder.central.directory.username	Text	Defines the user name required to the directory central webservices for Transfer Finder.
transferFinder.central.globalDirectory.uri	Text	Defines the value of the URI for the Transfer Finder that will allow the customer to respond with the desired information for the end user.
transferFinder.central.partnerService.password	Text	The password for the central server user to run partner services in Transfer Finder. (This field is encrypted).

Key	Value Type	Description
transferFinder.central.partnerService.username	Text	The username for the central server user to run partner services in Transfer Finder.
transferFinder.central.schoolDirectory.uri	Text	This setting defines the URL of a central webservice that will return School Directory information.
transferFinder.central.transferDirectory.uri	Text	This setting defines the URL of a central webservice that will return Transfer Directory information.
transferFinder.central.worksheet.category.all.complete.minimum	Text	Defines minimum number of categories that need to be complete.
transferFinder.central.worksheet.category.all.exclude.list	Text	Defines the exclude categories – those that should not be considered when doing the all check.
transferFinder.central.worksheet.category.block.type	Text	Defines which block type will be used for the transfer category check. This is usually set to OTHER.
transferFinder.central.worksheet.category.block.value	Text	Defines which block value will be used for the transfer category check.
transferFinder.central.worksheet.category.classes.minimum	Text	Defines minimum classes applied that need to be complete.
transferFinder.central.worksheet.category.credits.minimum	Text	Defines minimum credits applied that need to be complete.
transferFinder.central.worksheet.category.discrete.list	Text	Defines the categories for the discrete check – the check for specific categories. For example, SCIENCE, MATH.
transferFinder.central.worksheet.category.special.complete.minimum	Text	Defines minimum number of special categories that need to be complete – based on those categories in the special list.
transferFinder.central.worksheet.category.special.list	Text	Defines the special categories.

transferFinder.service

Updated: March 25, 2022

The transferFinder.service* settings configures Transfer Finder standard defaults for web services.

Key	Value Type	Description
transferFinder.service.central.password	Text	The password associated with the username in the transferFinder.service.central.username value. This entry is stored encrypted and cannot be viewed or modified except by users with the SHENCRPT key.
transferFinder.service.central.uri	Text	The URI of the Central Server degreeworks-services deployment.
transferFinder.service.central.username	Text	The username to gain access to the central server services, such as ucx, shepard settings, and school codes. Used together with the password stored in the transferFinder.service.central.password entry.

transit

Updated: March 24, 2023

The transit.* settings configure behavior for Transit.

Key	Value Type	Description
core.security.transit.batch.password	Text	This setting determines the Degree Works Transit Batch user's password. This is to be used for communication between the Transit Batch Executor and the Transit APIs. This field is encrypted.
core.security.transit.batch.username	Text	This setting determines the Degree Works Transit Batch username. This is to be used for communication between the Transit Batch Executor and the Transit APIs.
core.security.transit.password	Text	This setting determines the default user's password in the Degree Works Transit application. This is to be used for communication with Degree Works Transit APIs. This field is encrypted.
core.security.transit.username	Text	This setting determines the default user's username in the Degree Works Transit application. This is to

Key	Value Type	Description
		be used for communication with Degree Works Transit APIs.
transit.api.url	Text	The full URL of the TransitUI deployment. This is used to launch new jobs, monitor their status, and get job specifications. Should be an absolute URL starting with https:// + location of TransitUI deployment. For example: https://server.myschool.edu:7443/transit/production/
		If this is not set correctly then the Transit Batch Executor running on the classic server will not be able to process new Transit UI launch requests.
transit.cleanup.completed.maxDays	Number	This setting defines how many days old a Transit job with a status of DONE, FAILED, or TERMINATED must be before it will be deleted. A value of 0 means that these jobs will never be removed. The default value is 14 days.
transit.cleanup.incomplete.maxDays	Number	This setting defines how many days old a Transit job with a status of PENDING, RUNNING, or CLEANUP must be before it will be deleted. A value of 0 means that these jobs will never be removed. The default value is 14 days.
transit.dap16.workerCount	Number	Number of processes to use to run the batch parser. If 1,000 blocks were selected and the count is 10, then 100 blocks will be parsed in parallel in 10 separate processes. If the count is too small or too large, the batch may run more slowly.
transit.dap22.workerCount	Number	Number of processes to use to run the audits. If 1,000 students are in the batch selection and this count is 10, then 100 audits will run in parallel in 10 separate processes. If the count is too small or too large, the batch may run more slowly.

Key	Value Type	Description
transit.dap27.workerCount	Number	Number of processes to use to run the audits. If 1,000 students are in the batch selection and this count is 10, then 100 audits will run in parallel in 10 separate processes. If the count is too small or too large, the batch may run more slowly.
transit.dap28.workerCount	Number	Number of processes to use to run the audits. If 1,000 students are in the batch selection and this count is 10, then 100 audits will run in parallel in 10 separate processes. If the count is too small or too large, the batch may run more slowly.
transit.rad11.workerCount	Number	Number of processes to use to run the bridge. If 1,000 students are in the BIF file and this count is 10, then 100 students will be bridged in parallel in 10 separate processes. If the count is too small or too large, the batch may run more slowly.
transit.rad30.workerCount	Number	The number of RAD08 daemons to run to support dynamic bridge requests (non-Banner).

treq.applicant

Updated: March 25, 2022

The treq.applicant* setting configures applicant default for the articulation status.

Key	Value Type	Description
treq.applicant.default.treqStatus	Text	This setting defines the default value for the status on the dap_applicant_mst. Default value = TR.

treq.goal

Updated: March 25, 2022

The treq.goal* setting configures applicant default values related to student goals.

Key	Value Type	Description
treq.goal.catalogYear.defaultValue	Text	This setting defines the default value for the catalog year on the dap_applicant_mst.
treq.goal.degree.defaultValue	Text	This setting defines the default value for the degree on the dap_applicant_mst.
treq.goal.school.defaultValue	Text	This setting defines the default value for the school on the dap_applicant_mst.
treq.goal.scr004.codesToShow	Text	This setting defines the default elements that can be selected as Student goals in the “Goal type” dropdown in Transcript and Audit modules. The valid values are: COLLEGE, CONC (for Concentration), LIBL (for Liberal Learning), MAJOR, MINOR, PROGRAM, SPEC (for Specialization).

treq.selfservice

Updated: March 24, 2023

The treq.selfService* setting configures various behaviors of the Transfer Equivalency Self-Service module.

Key	Value Type	Description
treq.selfService.authentication.social.facebook.appId	Number	This setting denotes the registration id of the TreqUI application with Facebook. This is a numeric string, generated by Facebook when the application is registered. This setting is required to enable Facebook authentication.
treq.selfService.createAuditPDF	Boolean	This setting controls whether the Download PDF button displays in the Results page of Transfer Equivalency Self- Service.
treq.selfService.degreeMajorFilter	Boolean	This setting determines if the major combination is filtered by the degree selected in the Goals Disclosure page. If true, then the contents of the major drop-down

Key	Value Type	Description
		will be only values from STU023 with Show in Transfer Equivalency Self-Service flag equal to Y AND the previously selected degree in one of the Transfer Equivalency Self-Service Degree Filter * fields. This setting determines if the major drop-down is filtered by the degree selected in the Goals Disclosure page.
treq.selfService.enableAnonymous	Boolean	This setting determines if Transfer Equivalency Self-Service may be used without logging in. Note: Either this setting or the treq.selfService.persistence.enableShpAccount must be true. Both of the settings cannot be false but both can be true.
treq.selfService.goalPresentation	Text	A comma-delimited list of goal prompts which defines their order and visibility. Default is term, level, degree, major, campus, program, college, concentration, minor and that is the exhaustive list of valid goal prompt names. The first four are required but their order can be changed: term, level, degree and major. Spaces around the commas aren't required and will be trimmed. Invalid or misspelled goal prompt names will be ignored. This setting is ignored when core.whatIf.enableCurriculumRules is true.
treq.selfService.nextSteps.*	Text	See the Other options topic.
treq.selfService.persistence.enableShpAccount	Boolean	This setting determines if saving data, login and logout should be enabled. Note: Either this setting or the treq.selfService.enableAnonymous must be true. Both of the settings cannot

Key	Value Type	Description
treq.selfService.persistence.enableSocial.facebook	Boolean	be false but both can be true.

treq.treqadmin

Updated: September 30, 2022

The treq.treqadmin* settings configure behavior for Transfer Equivalency Admin.

Key	Value Type	Description
treq.treqadmin.applicant.show.articulationConditions	Boolean	This setting defines if the Student conditions section will be displayed under the Student Information section in Transcript module, that will be considered for articulation. If set to true, the Student conditions section will be displayed in UI and if set to false, it will not be displayed.
treq.treqadmin.audit.report	Text	Report code to generate articulation and audit in Transfer Equivalency Admin. You can also use this setting to set the default report code to be returned while accessing the /requirements and /articulation request endpoints in API Services.
treq.treqadmin.default.gradeAssignMode	Text	This setting defines how the grade will be assigned on a many-to-one or many-to-many mapping. L=use lowest grade from transfer classes, H=use highest grade from transfer classes, P=use many-passfail-grade when assigning the grade. Default=H. This value is used for the articulation process.
treq.treqadmin.default.manyPassGrade	Text	This setting defines the default passfail grade to be used when the

Key	Value Type	Description
shpSetting		treq.treqadmin.default.gradeAssignMode grade assignment flag is P. Default=PF. This value is used for the articulation process.
treq.treqadmin.testscores.default.gradeType	Text	This setting defines the default Grade-Type from UCX-STU356 for the classes mapped from test scores. Default = TS. This value is used for the articulation process.
treq.treqadmin.testscores.default.schoolId	Text	This setting defines the default School-ID for test scores saved in the dap_transfer_dtl. Default = TESTS. This value is used for the articulation process.

UCX tables

Updated: March 25, 2022

The Universal Code eXtension (UCX) Tables store data validation and configuration values used in Degree Works. These tables are managed through the Controller application.

UCX-AUD012 User Class Codes

Updated: September 29, 2023

UCX-AUD012 defines the user class codes valid for Degree Works. Each person accessing Degree Works must be part of a User Class for security purposes. Degree Works security uses the user class to control access to notes and exceptions. The user class also sets up defaults to be used when a note or exception is added through the Web.

UCX KEY	Len	Description
User Class	4	The user class code. Alphanumeric. Alpha must be upper-case. Usually descriptive of a group of users, for example, "ADV" for Advisors. Each Degree Works user has an ID# and each ID# is mapped to a User Class in the shp_user_mst.
Reserved	26	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Description of this user class.
Reserved	13	31	Reserved for future use

UCX VALUE	Len	Start	Description
Are Internal Notes Visible	1	44	Y/N. Y = user can see notes marked "internal". N = user cannot see notes marked "internal". This controls the internal notes in the Notes dialog and the notes shown in worksheets. This does not control internal notes in the planner.
Reserved	5	45	Reserved for future use.
Audit Type	2	50	Default audit type (UCX-AUD034). This code controls whether or not the audit is saved to the database.
Default Audit Report	5	52	Obsolete
Reserved	5	57	Reserved for future use.
Default Internal Note Type	2	62	Note type (UCX-SCR008) used if note is specified on the Web as internal: Internal=Y
Default Public Note Type	2	64	Note type (UCX-SCR008) used if note is not specified on the Web as internal: Internal=N
Reserved	4	66	Reserved for future use
Default Exception Status	2	70	Exception status (UCX-AUD015) used when status is not specified.
Show in RAD33	1	72	Display in the RAD33 user class drop-down in Transit. Y or blank = Show, N = Hide.
Reserved	97	73	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD013 User Class Permissions

Updated: March 25, 2022

UCX-AUD013 maps a user class code to read or write permissions on data entered by other users of Degree Works. These permissions determine whether the user can read or write notes entered or maintained by other users.

The Degree Works software checks the user class of the person who is logged on against the user class of the Note record. The user class of the note record is the user class of the last modifier of the record. If the record's user class is in the user's UCX-AUD013 User Class Array, then the record can be read by the logged-on user. If the record's user class is in the UCX-AUD013 User Class Array with "W" (write) access then the record can be modified or deleted. If the logged-on user is the initiator (dap_enter_id) of the note then the user has write access to the note regardless of user class.

UCX KEY	Len	Description
User Class	4	The user class code.
Reserved	7	Reserved for future use. Must be blanks.
Sequence Number	1	A number, sequentially assigned and starting with "1". Valid values are 1-4.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
User Class	4	1, 6, 11...	A four byte user class code. Must be valid in UCX-AUD012.
Read/Write Flag	1	5, 10, 15...	A one byte read/write flag (R=read, W=write).

The above two fields create an array of 20 five-byte fields representing the user class interaction. Typically, users have write access to "lower" user class levels and read access to "higher" user class levels. Each field consists of a four byte user class code and a one byte read/write flag (R=read, W=write).

Reserved	69	101	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD014 Exception Type Codes

Updated: September 29, 2023

UCX-AUD014 defines the Exception Type codes and is used to supply the literal for the exception types drop-down list box.

Warning! These codes are defined by Ellucian and institutions should not create their own. For this particular table, you can change the Description, but do not modify anything else unless instructed by Ellucian.

UCX KEY	Len	Description	
Exception Type	2	The type of exception. This code is defined by Ellucian and should not be changed.	
Reserved	28	Reserved for future use.	
<hr/>			
UCX VALUE	Len	Start	Description
Description	30	1	Description of this exception type.
Reserved	139	31	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD015 Exception Status Codes

Updated: March 25, 2022

UCX-AUD015 defines the exception status codes. This status code describes the state of the exception, usually either pending or approved.

UCX KEY	Len	Description	
Exception Status	2	The exception status code, usually representing the source or state of the exception.	
Reserved	28	Reserved for future use.	
<hr/>			
UCX VALUE	Len	Start	Description
Description	30	1	Description of this exception status.
Reserved	139	31	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD027 Major What-If Picklist

Updated: March 25, 2022

UCX-AUD027 defines the codes allowed in the first Major field in the What-If drop-down. It can be used to map the degree, school and college from the Major during a What-If audit or define the degrees for which each is valid.

UCX KEY	Len	Description	
Major Code	12	The major code from the student system. The key is the Major code (for example, HIST). However, it is possible for this code to be anything that uniquely identifies Major, Degree, School, College combination, so the What-If drop-down list box will guide the student to select valid combinations of data.	
Reserved	18	Reserved for future use.	
<hr/>			
UCX VALUE	Len	Start	
Description	50	1	Description of this major.
Is Concentration required	1	51	Y=a concentration is required for this major (For the Responsive Dashboard core.whatIf.filterMode must be R. For Transfer Equivalency Self-Service core.whatIf.enableCurriculumRules must be true).
Additional	1	52	Not used by the Responsive Dashboard.
Reserved	19	53	Reserved for future use
Degree	12	72	The UCX-STU307 degree code associated with this major. See the additional notes in the Restricting What-If Picklists sections below.
School	12	84	The UCX-STU350 school code associated with this major. See the additional notes in the Restricting What-If Picklists sections below.
Reserved	5	96	Reserved for future use.
College	6	101	The UCX-STU560 college code associated with this major. See the additional notes in the Restricting What-If Picklists sections below.
Degree Filter 1	12	107	Used in degree-drives-major mode to filter the major drop-down. See the additional notes in the Restricting What-If Picklists sections below.

UCX VALUE	Len	Start	Description
Degree Filter 2	12	119	Used in degree-drives-major mode to filter the major drop-down. See the additional notes in the Restricting What-If Picklists sections below.
Degree Filter 3	12	131	Used in degree-drives-major mode to filter the major drop-down. See the additional notes in the Restricting What-If Picklists sections below.
Degree Filter 4	12	143	Used in degree-drives-major mode to filter the major drop-down. See the additional notes in the Restricting What-If Picklists sections below.
Degree Filter 5	12	155	Used in degree-drives-major mode to filter the major drop-down. See the additional notes in the Restricting What-If Picklists sections below.
Reserved	3	167	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

What-if picklist restriction

Degree-drives-major mode is enabled when core.whatIf.filterMode is set to D. When the selected degree matches any one of the Degree Filter codes in AUD027 then the AUD027 major will appear in the Major drop-down.

Major-drives-degree mode is enabled when core.whatIf.filterMode is M. The Degree, Level, and College drop-down list boxes will be filtered based on the Degree, School and College fields in AUD027.

These AUD027 fields apply to both the Worksheets What-If and Plans What-If.

Additional degree filters

For degree-drives-major, if more than five Degree Filters are required, additional records can be created for a major, allowing additional filters to be added. You may append a suffix to the key like _0, _1, ... _9 for up to 50 additional degrees.

For example, if 7 Degree Filters were required for the Math major:

1. Using Controller, copy “MATH” record to “MATH_2” in UCX-AUD027.
2. Change description on MATH_2 record to say “Math major – CONTINUATION”.

However, make sure the descriptions for this major sort together. That is, make sure that the description for some other major does not come between these two descriptions alphabetically. For this reason you may need to add several spaces before putting “-CONTINUATION”. You can use “(cont)” or anything else to help ensure that the second record is a continuation of the first.

3. Add the additional degree filters to the MATH_2 record.

This applies to both the Worksheets What-If and Plans What-If.

For major-drives-degree, the Degree, School and College fields are used and not the Degree Filter fields.

Curriculum rules

When using Curriculum Rules filtering, the Conc Required flag is used to enforce that the user choose a concentration for the selected major.

Other than this Conc Required flag no settings on this record are used. Specifically, the Degree, School, College and five Degree Filter fields are not used by the Curriculum Rules functionality.

UCX-AUD029 Minor What-If Picklist

Updated: March 25, 2022

UCX-AUD029 defines the codes allowed in the Minor field in the What-If drop-down list boxes.

UCX KEY	Len	Description	
Minor	12	The minor code from the student system. The key is usually the Minor code (e.g. HIST).	
Reserved	18	Reserved for future use.	
<hr/>			
UCX VALUE	Len	Start	Description
Description	50	1	Description of this minor.
Filter	12	51	Used to filter the data on the what-if tab.
Reserved	107	63	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD031 Athletic Eligibility Types

Updated: March 25, 2022

UCX-AUD031 defines the different ATHLETE types that may be used to save blocks in Scribe.

Any value can be used here to help you manage your ATHLETE blocks. For example, you may create a type of ACADSTANDING for the block that checks for the student being in good academic standing. You may create another type of 40-60-80 for the NCAA 40/60/80 rule. You can create as many types, and thus blocks, as you like.

UCX KEY	Len	Description
Type	12	The athlete type to use in Scribe.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Description of this award.
Reserved	139	31	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD032 Audit Freeze Types

Updated: September 29, 2023

The freeze type identifies that an audit is frozen and the reason for freezing it.

The freeze type is stored in the dap_freeze_type field on the dap_audit_dtl. Each freeze type defines a set of user classes. Only users associated with these user classes are allowed to use the particular freeze type when freezing an audit.

This table is supplied by Ellucian during the initial install but is expected to be modified by each institution as needed.

UCX KEY	Len	Description
Freeze Type	6	The freeze type to be placed on the dap_audit_dtl's dap_freeze_type field when the audit is frozen.
Reserved	24	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Description of this freeze type.
user classes 1-10	4x10	31, 35, 39...	List of user class codes from UCX-AUD012 that are valid for this freeze type. Each code is 4 bytes long and there can be no more than 10 codes. Only those users belonging to one of these user classes can use this freeze type when freezing an audit.
Reserved	99	71	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD033 Financial Aid Awards

Updated: March 25, 2022

UCX-AUD033 defines the different AWARD types that may be used to save blocks in Scribe.

UCX KEY		Description	
Award		The award code.	
Reserved		Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	Description of this award.
Reserved	139	31	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD034 Audit Type

Updated: September 29, 2023

UCX-AUD034 defines the valid audit type codes.

Warning! These codes are defined by Ellucian and institutions should not create their own. For this particular table, you can change the Description and the Save Audit flag, but do not modify anything else unless instructed by Ellucian.

An audit type is stored in the database for each audit that is run. It indicates which audits should be saved in the database. Audit Type is also used as criteria for deleting audits.

UCX KEY	Len	Description	
Audit Type	2	The Audit Type code.	
Reserved	28	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	Description of this audit type.
Reserved	69	31	Reserved for future use
Save Audit	1	100	Y/N. Y = save to the database. N = show audit results but do not save the audit.
Reserved	69	101	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-AUD047 Repeat Policies

Updated: September 29, 2023

UCX-AUD047 defines the valid Repeat Policy codes for Degree Works. Each Repeat Policy code represents a different method for handling repeated classes in a degree audit.

Warning! These codes are defined by Ellucian and institutions should not create their own. Modify only if you are using repeat policy 2. See Repeat Policy 2 Additional Control Flag.

See the [Repeated classes](#) topic for more information.

UCX KEY	Len	Description	
Repeat Policy	1	Repeat Policy code.	
Reserved	29	Reserved for future use.	

UCX VALUE	Len	Start	Description
Description	50	1	Description of this repeat policy.
Description 2	50	51	Description continued.
Description 3	50	101	Description continued.
Repeat Policy 2 Additional Control Flag	1	151	For Repeat Policy 2 when the grades are the same this can be set to O to keep the Oldest class and set to N to keep the Newest class.
Reserved	69	101	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-BAN001 Schedule Type Codes

Updated: March 25, 2022

UCX-BAN001 is used by the Banner Bridge extract to determine if “zero-credit” lab classes should be skipped and not rolled to Degree Works.

Note: Banner sites only.

If this feature is desired, then the Schedule Type Codes that are used on the “zero-credit” lab classes and are to be “excluded” from Degree Works should be loaded into this table. Use the SFRSTCR_SCHD_CODE for current classes and SHRTCKN_SCHD_CODE for historic classes. For example, if SCHD_CODES “CR”, “N” and “R” are used on lab classes and those with “0” credits are NOT to be rolled to Degree Works, then these 3 codes should be added to UCX-BAN001 using Controller.

Note: BAN001 and BAN002 may both be used to filter out “zero-credit” lab classes if a mix of SCHD_CODES and SEQ_NUMB values is used at your site.

UCX KEY	Len	Description
Schedule Code	3	The SCHD_CODE values that are to be excluded from Degree Works for ZERO-CREDIT classes.
Reserved	27	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Description of SCHD_CODE for ZERO-CREDIT current (SFRSTCR) and historic (SHRTCKN) classes.
Reserved	139	31	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-BAN002 Section Codes

Updated: March 25, 2022

UCX-BAN002 is used by the Banner Bridge extract to determine if “zero-credit” lab classes should be skipped and not rolled to Degree Works.

Note: Banner sites only.

If this feature is desired, then the SFRSTCR_SEQ_NUMB (current classes) and SHRTCKN_SEQ_NUMB (historic classes) values that are used on the “zero-credit” lab classes that are to be “excluded” from Degree Works should be loaded into this table. For example, if SEQ_NUMB codes of “L1”, “L2” and “LB” are used on lab classes and those with “0” credits are NOT to be rolled to Degree Works, then they should be added to UCX-BAN002.

Note: UCX-BAN001 and UCX-BAN002 may both be used to filter out “zero-credit” lab classes if a mix of SCHD_CODES and SEQ_NUMB values is used at your site.

UCX KEY	Len	Description
Section Code	3	The SEQ_NUMB values that are to be excluded from Degree Works for ZERO-CREDIT current (SFRSTCR) and historic (SHRTCKN) classes.
Reserved	27	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Description of SEQ_NUMB Section code.
Reserved	139	31	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code

UCX VALUE	Len	Start	Description
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-BAN003 Gmod Codes

Updated: March 25, 2022

UCX-BAN003 is used by the Banner Bridge extract to determine if classes should be skipped and not rolled to Degree Works.

If this feature is desired, then the Gmod Codes (Grade Types) that are to be “excluded” from Degree Works should be loaded into this table.

The SFRSTCR_GMOD_CODE for current classes and SHRTCKG_GMOD_CODE for historic SHRGCKN classes will be looked up in this table and if a match is found the class record will be skipped. For example, if a class with a GMOD_CODE of “A” is NOT to be rolled to Degree Works, then a UCX Key of “A” should be added to UCX-BAN003, using Controller.

UCX KEY	Len	Description	
Gmod Code	3	The Banner GMOD_CODE values that are to be excluded from Degree Works.	
Reserved	27	Reserved for future use.	

UCX VALUE	Len	Start	Description
Description	30	1	Description of GMOD_CODE for current (SFRSTCR) and historic (SHRTCKN) classes that should NOT be extracted into Degree Works.
Reserved	139	31	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-BAN080 Banner Custom Data Definitions

Updated: September 29, 2023

If your institution has stored non-standard data in Banner that is required in Degree Works to accurately define the rules to complete a degree but is NOT being extracted through the Degree

Works Banner Student extract program, then UCX-BAN080 can be used to specify this data so that it can be extracted into the rad_custom_dtl, rad_report_dtl or rad_aid_dtl.

UCX-BAN080 contains the location (database TABLE and COLUMN) of the desired data in Banner along with the required WHERE clause to retrieve this non-standard data and an ORDER BY clause to sort the data in the desired sequence. This table must contain the valid SQL pieces to be used by the banstudent Banner Student extract program to retrieve the desired data from the Banner database: COLUMN, TABLE, WHERE, and ORDER BY clauses. The COLUMN and TABLE clauses are required while the WHERE and ORDER BY clauses are optional. Whether the WHERE clause is left blank, the first seven characters from the TABLE entry will be used to create the name of the PIDM column. For example, if the TABLE entry is SGBSTDN, the PIDM column becomes SGBSTDN_PIDM. Thus, any custom view you use in the TABLE record must be seven characters and the PIDM column must be the view name, an underscore, and PIDM. If the ORDER BY clause is left blank or the entry is not included in UCX-BAN080, the ORDER BY clause will automatically be loaded with the COLUMN being selected (for example, SGBSTDN_ORSN_CODE).

Note: Access to your Banner database is required for this custom extract process.

If any custom database tables are referenced in UCX-BAN080 that are NOT included in the list of Banner tables, then READ access must be provided. For more information, see [Database table access](#).

For example, if an institution wanted to determine if the Core Requirements have been satisfied for a student which is indicated by a flag of S in the SGBSTDN_ORSN_CODE column from the SGBSTDN table, a Keyword of CORESAT could be assigned. This CORESAT Keyword would be required in every UCX-BAN080 entry so that a proper SQL statement to extract this data could be constructed.

UCX Key	UCX Value
CORESAT:COLUMN	SGBSTDN_ORSN_CODE
CORESAT:TABLE	SGBSTDN
CORESAT:WHERE	SGBSTDN_ORSN_CODE = S
CORESAT:ORDERBY	SGBSTDN_ORSN_CODE

When the following CORESAT SQL statement is executed for a given student:

```
SELECT SGBSTDN_ORSN_CODE
FROM SGBSTDN
WHERE SGBSTDN_PIDM = 33333 AND (SGBSTDN_ORSN_CODE = 'S')
ORDER BY SGBSTDN_ORSN_CODE
```

and an SGBSTDN record is selected for the student, an S would be returned and loaded into the rad_custom_value on the rad_custom_dtl with the rad_custom_code of CORESAT. A rule could then be defined in Scribe to check for a CORESAT code of S on the rad_custom_dtl for a given student:

```
If (CORESAT = 'S') Then
    Rule-Complete
    Label "Core Requirements Satisfied"
```

Else

... other requirements would be defined here;

Three types of records in the RAD database may be created using dynamic retrievals defined in UCX-BAN080:

- The default location for this custom data is the rad_custom_dtl. This custom data can then be included in rules scribed for your audits as show above. A unique Keyword must be chosen for each custom piece of data extracted from Banner. In the example below, the Keyword for the entry below is STUDENTHOLD (before the colon).

UCX-BAN080	STUDENTHOLD:COLUMN	SPRHOLD_HLDD_CODE
------------	--------------------	-------------------

- If the custom data is to be displayed in the audit header it should loaded into the rad_report_dtl. In this case a special Keyword:REPORT record must be added to the UCX-BAN080 table.
- If the custom data is to be used in the Financial Aid Audit it should loaded into the rad_aid_dtl. In this case a special Keyword:AID record must be added to the UCX-BAN080 table.

UCX-BAN080	AIDAWARD:AID	AWARD
------------	--------------	-------

The UCX Value area above contains the special Block Type of AWARD which would be used in the Scribe rules for AID blocks. In this case, Financial Aid awards would be stored in the rad_aid_dtl with the rad_aid_code of AWARD.

- If the custom data is to be used in core.security.rules.shpcfg in an IF-statement to assign keys based on the value of this custom data item, the special Keyword:SHPCFG record must be added to the UCX-BAN080 table. The value field in the record should be left blank.

UCX-BAN080	SOMETHING:SHPCFG
------------	------------------

This will ensure a SHP_USER_ATTRIB record is created with the ATTRIBUTE_NAME set to SOMETHING and ATTRIBUTE_VALUE set to the value pulled from Banner. In the shpcfg you can then specify SOMETHING in an IF-statement to assign keys or groups.

Currently only ONE database COLUMN may be loaded for a given keyword. The data returned from the SQL call will be loaded into the rad_custom_value, rad_report_dtl or rad_aid_dtl. Although only one value can be specified in the COLUMN record you can concatenate multiple columns together to create a single value. For example, SORLCUR_CAMP_CODE || - || SORLCUR_CACT_CODE. This will produce a value with two codes separated by a hyphen. You may also use the database functions to format the value you are extract. For example, if you are extracting a GPA field you may choose to use the TO_CHAR function to format the field like this: TO_CHAR(SHRLGPA_GPA, 0.000). As long as a single value is being selected you may specify more than just a column name in the COLUMN record.

Warning! If more than 50 records (defaulted in /src/include/ban40.h) are returned from the SQL call for ONE piece of data the WARNING counter will be incremented on the LOG file and a WARNING message will be displayed in the logdebug file. Review your UCX_BAN080 SQL statements to make sure the SQL returns less than 50 records for each SQL call. If necessary, run the bannerextract with debugon to see the exact warning message in the \$ADMIN_HOME/logdebug/rad30.????.xml file including the UCX_BAN080 KEYWORD involved.

Multiple lines may be defined for the WHERE clause used for a given Keyword up to a maximum of 10 lines. However, in this case a unique identifier must be included at the end of the WHERE keyword. It can be any set of characters desired as long as the characters sort in the desired sequence. For example, CORESAT:WHERE_A, CORESAT:WHERE_B, CORESAT:WHERE_C,

and so on would be concatenated together in that order to make one complex WHERE statement containing up to 2000-bytes. Parenthesis are recommended.

For data loaded into the rad_custom_dtl and rad_report_dtl, you can optionally also load the school/level and degree both associated with the data being extracted. For example, if you are pulling the ADMITCODE from SORLCUR, you may also want to pull over the SORLCUR_LEVEL_CODE or SORLCUR_DEGC_CODE. This can be helpful if the student is working on multiple, simultaneous degrees. This way multiple admit codes are pulled in to Degree Works along with the school/level and degree to which each is associated. To pull over these extra fields, you need a LEVEL (or SCHOOL) and a DEGREE record added to BAN080 indicating the column from which the data should be pulled. For example:

BAN080ADMITCODE:LEVEL SORLCUR_LEVEL_CODE

BAN080ADMITCODE:DEGREE SORLCUR_DEGC_CODE

When the Responsive Dashboard and PDF display this data in the student header, it uses this data bridged into the school and degree fields on the custom/report records to determine which records to show for the selected degree. For example, if the user chooses to view the student's MBA audit, the admit code for this degree will be shown. Additionally, when running an audit, the SCR002 custom records used in if-statements will be filtered by the audit's school and degree.

Simple example

UCX VALUE	Len	Description
Keyword:SQL Clause	30	The Keyword must be first (UPPER CASE) followed by a colon (:) and then the SQL Clause. No BLANKS are allowed in this key. For example, if the Keyword is CORESAT for Core Satisfied then the key entries would be:

CORESAT : COLUMN

CORESAT : TABLE

CORESAT : WHERE

CORESAT : ORDERBY

Additional informational keywords might be used for descriptive purposes:

CORESAT:DESCRIPTION – this entry will be ignored by banstudent but multiple lines of description might be included for documentation purposes.

The keyword cannot be longer than 12 characters. If you create a keyword that is longer than 12 characters, it will be truncated.

UCX VALUE	Len	Start	Description
SQL Statement or a Description	169	1	Text of the particular SQL clause or a description. For example, to illustrate the UCX-BAN080 entries required for the CORESAT example listed above the

UCX VALUE	Len	Start	Description
-----------	-----	-------	-------------

following entries may be loaded:

UCX Key	UCX Value
CORESAT:COLUMN	SGBSTDN_ORSN_CODE
CORESAT:TABLE	SGBSTDN
CORESAT:WHERE	BSTDN_ORSN_CODE = 'S'
CORESAT:ORDERBY	SGBSTDN_ORSN_CODE

The SQL statement when concatenated by the Banner Bridge extract would be similar to the following:

```
select SGBSTDN_ORSN_CODE from SGBSTDN
  where SGBSTDN_PIDM = 822 and
    (SGBSTDN_ORSN_CODE = 'S')
  order by SGBSTDN_ORSN_CODE;
```

Note: One set of parenthesis is automatically put around the entire UCX-BAN080 WHERE clause as the PIDM will always be included first outside of the parenthesis displayed in this example. If a complex WHERE clause is desired then other parenthesis may be used as well to ensure the correct results are returned.

Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

After these pieces of the SQL clause have been defined properly above then the CORESAT keyword above can be used in a rule used for an audit.

More complex example - multiple WHERE lines

UCX VALUE	Len	Start	Description
SQL Statement or a Description	169	1	Text of the particular SQL clause or a description. For example, to illustrate the UCX-BAN080 entries required for a more complex example used to retrieve the CAMPUS code different columns and a multiple line WHERE clause are displayed

UCX VALUE	Len	Start	Description
-----------	-----	-------	-------------

below:

UCX Key	UCX Value
CAMPUS:COLUMN	SORLCUR_CAMP_CODE
CAMPUS:ORDERBY	SORLCUR_CAMP_CODE
CAMPUS:REPORT	STVCAMP
CAMPUS:TABLE	SORLCUR a
CAMPUS:WHERE_1	a.SORLCUR_CACT_CODE = 'ACTIVE'
CAMPUS:WHERE_2	AND a.SORLCUR_SEQNO = (SELECT
CAMPUS:WHERE_3	MAX(b.SORLCUR_SEQNO) FROM SORLCUR b
CAMPUS:WHERE_4	WHERE b.SORLCUR_PIDM = a.SORLCUR_PIDM
CAMPUS:WHERE_5	AND b.SORLCUR_PRIORITY_NO =
CAMPUS:WHERE_6	a.SORLCUR_PRIORITY_NO
	AND b.SORLCUR_LMOD_CODE = 'LEARNER')

The SQL statement when concatenated by the Banner Bridge extract would be similar to the following:

```

SELECT SORLCUR_CAMP_CODE
FROM SORLCUR a
WHERE SORLCUR_PIDM = 33333 AND
( a.SORLCUR_CACT_CODE = 'ACTIVE' AND
a.SORLCUR_SEQNO = (SELECT
MAX(b.SORLCUR_SEQNO) FROM SORLCUR b
WHERE b.SORLCUR_PIDM = a.SORLCUR_PIDM AND
b.SORLCUR_PRIORITY_NO =
a.SORLCUR_PRIORITY_NO AND
b.SORLCUR_LMOD_CODE = 'LEARNER') )
ORDER BY SORLCUR_CAMP_CODE

```

In the CAMPUS:REPORT line above the STVCAMP validation table is listed in the UCX Value so that the actual Campus Name will be returned rather than just the Campus Code. If just the Campus Code is desired, then replace STVCAMP with NONE.

Note: One set of parenthesis is automatically put around the entire UCX-BAN080 WHERE clause as the PIDM will always be included first outside of the parenthesis displayed in this example. In this more complex WHERE

UCX VALUE	Len	Start	Description
			clause an additional set of parenthesis is used for clarity.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

REPORT: rad_report_dtl loaded instead of the rad_custom_dtl

UCX VALUE	Len	Start	Description
SQL Statement or a Description	169	1	<p>Text of the particular SQL clause or a description. For example, to illustrate the UCX-BAN080 entries required when the custom data needs to be loaded into the rad_report_dtl for use in the audit header note the REPORT keyword loaded below. The ucx_value contains the Banner STV validation table that is used to lookup the Code (STVASTD_CODE - Academic Standing retrieved from SHRTTRM) to get the Description (STVASTD_DESC – Description for the rad_report_value). If, however, the actual data from the specified column is desired instead of the Description then leave the UCX Value BLANK or load it with NONE:</p>

UCX Key	UCX Value
ACADSTANDING:COLUMN	SHRTTRM_ASTD_CODE_END_OF_TERM
ACADSTANDING:ORDERBY	SHRTTRM_ASTD_CODE_END_OF_TERM
ACADSTANDING:REPORT	STVASTD
ACADSTANDING:TABLE	SHRTTRM a
ACADSTANDING:WHERE_1	a.SHRTTRM_TERM_CODE =
ACADSTANDING:WHERE_2	(SELECT MAX(b.SHRTTRM_TERM_CODE)
ACADSTANDING:WHERE_3	FROM SHRTTRM b
ACADSTANDING:WHERE_4	WHERE b.SHRTTRM_PIDM =
	SHRTTRM_PIDM)

The Academic Standing code (for example, GS for Good Standing) returned by the sql statement created from the statements above will be translated by reading the STVASTD table using the STVASTD_CODE to SELECT the

UCX VALUE	Len	Start	Description
			appropriate records and returning the Description in the STVASTD_DESC column. The STVASTD_DESC will be loaded into the rad_report_value while the keyword ACADSTANDING will be loaded into the rad_report_code on the rad_report_dtl. If instead GS is desired in the rad_report_value, leave the UCX Value BLANK or load it with NONE (remove the STVASTD reference).
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

REPORT: rad_aid_dtl loaded instead of the rad_custom_dtl

UCX VALUE	Len	Start	Description														
SQL Statement or a Description	169	1	Text of the particular SQL clause or a description. For example, the records below illustrate the UCX-BAN080 entries required when the custom data needs to be loaded into the rad_aid_dtl for use in Financial Aid audits. The :AID keyword is required after the keyword. The UCX_Value contains the special Scribe Block that is used in the Scribe rules to specify requirements that must be met for the specified Financial Aid data: <table> <thead> <tr> <th>UCX Key</th> <th>UCX Value</th> </tr> </thead> <tbody> <tr> <td>AIDAWARD:AID</td> <td>AWARD</td> </tr> <tr> <td>AIDAWARD:COLUMN</td> <td>RPRAWRD_FUND_CODE</td> </tr> <tr> <td>AIDAWARD:ORDERBY</td> <td>RPRAWRD_FUND_CODE</td> </tr> <tr> <td>AIDAWARD:TABLE</td> <td>RPRAWRD</td> </tr> <tr> <td>AIDAWARD:WHERE_1</td> <td>RPRAWRD_AIDY_CODE = '0708'</td> </tr> <tr> <td>AIDAWARD:WHERE_2</td> <td>AND RPRAWRD_ANST_CODE = 'ACPT'</td> </tr> </tbody> </table>	UCX Key	UCX Value	AIDAWARD:AID	AWARD	AIDAWARD:COLUMN	RPRAWRD_FUND_CODE	AIDAWARD:ORDERBY	RPRAWRD_FUND_CODE	AIDAWARD:TABLE	RPRAWRD	AIDAWARD:WHERE_1	RPRAWRD_AIDY_CODE = '0708'	AIDAWARD:WHERE_2	AND RPRAWRD_ANST_CODE = 'ACPT'
UCX Key	UCX Value																
AIDAWARD:AID	AWARD																
AIDAWARD:COLUMN	RPRAWRD_FUND_CODE																
AIDAWARD:ORDERBY	RPRAWRD_FUND_CODE																
AIDAWARD:TABLE	RPRAWRD																
AIDAWARD:WHERE_1	RPRAWRD_AIDY_CODE = '0708'																
AIDAWARD:WHERE_2	AND RPRAWRD_ANST_CODE = 'ACPT'																

In the example above, the special :AID is appended to the UCX-BAN080 keyword of AIDAWARD making the UCX Key of AIDAWARD:AID so that the banner extract knows to load the data into the rad_aid_dtl. Note that the prefix of AID in

UCX VALUE	Len	Start	Description
AIDAWARD:AID			AIDAWARD:AID is NOT required – it is just included so all of the AID UCX-BAN080 records would appear together.
			The UCX Value contains the actual rad_aid_code (AWARD in the above example). The award records from the RPRAWRD table would be read for a specified student PIDM for the Aid Year of 0708 (20072008) with an Award Status Code of ACPT for Accepted. The RPRAWRD_FUND_CODE column will contain the actual AWARD codes like PELL, SEOG, and so on for the student and would be written to the rad_aid_dtl.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 BANNER

Updated: March 24, 2023

The BANNER record of UCX-CFG020 contains several settings to tell Degree Works how to handle various issues regarding data bridged from the Banner student system.

Note: Banner sites only.

For more information, the additional Banner settings starting with integration.banner.extract in [integration.banner](#).

UCX KEY		Description	
BANNER		BANNER	
Reserved		Reserved for future use.	
UCX VALUE	Len	Start	Description
Is this a Banner Site	1	1	A Y/N flag. Set to Y if your site uses the Banner student software. This will allow the Banner database to be used for search and refresh purposes for Degree Works.

UCX VALUE	Len	Start	Description
Email Code	4	2	The email code used to find the desired email address in the GOREMAL Banner table for the rad_primary_mst in Degree Works.
Email Override	1	6	A Y/N flag. If set to Y and the email code specified above is not found the Active email address will be loaded into the rad_primary_mst.
Repeat Skip A	1	7	A Y/N flag. Set to Y to skip the repeated classes in Banner with an A in the shrtckn_repeat_course_ind and NOT load them into the rad_class_dtl.
Repeat Policy A	1	9	<p>A valid Repeat Policy must be defined (valid values are 0-7 and B) – do not leave blank even if the skip flag is set to Y. Refer to the UCX-AUD047 Repeat Policy documentation included later in this document for a definition of these values.</p> <p>Policy B is recommended – the Averaged classes will end up in the Insufficient section of the audit and they will affect the overall GPA but will not be counted in the overall credits towards a degree.</p>
			<p>Although there are three repeat policy fields here they should be all set to the same value. These also should match the values set for the Transfer Repeat Policy fields below.</p>
Repeat Skip E	1	9	A Y/N flag. Set to Y to skip the repeated classes in Banner with an E in the shrtckn_repeat_course_ind and NOT load them into the rad_class_dtl.
Repeat Policy E	1	10	<p>If the Repeat Skip E flag above is set to N, then a valid Repeat Policy must be defined (valid values are 0-7 and B). Refer to the UCX-AUD047 Repeat Policy documentation included later in this document for a definition of these values.</p> <p>Policy B is recommended – the Excluded classes will end up in the Insufficient section of the audit but they will not affect the overall GPA or credits.</p>

UCX VALUE	Len	Start	Description
Repeat Policy I	1	11	A valid Repeat Policy (0-7 or B) must be defined for repeated classes that have an I in the shrckn_repeat_course_ind. Refer to the UCX-AUD047 Repeat Policy documentation included later in this document for a definition of these values. Policy B is recommended – the Included classes will apply to rules as normal classes affecting the GPA and total credits.
GPA Type	1	12	The desired SHRLGPA record to be used to retrieve GPA information for the rad_term_dtl: I = Institutional GPA (no transfer classes), O = Overall GPA (includes transfer classes).
Add UCX Entries Only	1	13	Obsolete – moved to CFG020 RADBRIDGE. Still works with the Banner extract but has been generalized to work with any SIS bridging UCX data.
Use Term as Catalog Year	1	14	A Y/N flag. Set to Y if the Term Code is ALSO being used as the Catalog Year. If set to N, the STVTERM_ACYR_CODE will be used as the Catalog Year in Degree Works.
Search in Banner	1	15	Not used by Responsive Dashboard.
Advisor User Class	4	16	This default User Class is used by the banstaff advisor/staff load program, ban45.ec, when creating dap_user_mst records for Advisors. The standard values are: ADV which means advisors CAN make exceptions or ADVX which means advisors CANNOT make exceptions.
Staff User Class	4	20	This default User Class is used by the banstaff advisor/staff load program, ban45.ec, when creating dap_user_mst records for Staff members. The standard value is: REG which has the highest level of security available in Degree Works.
Non-Course Score	1	24	C=the SHRNCRS_NCST_CODE is loaded into the rad_non_score column. I=the corresponding STVNCST satisfied Indicator (Y/N) is loaded into the rad_non_score column based on a lookup using the SHRNCRS_NCST_CODE

UCX VALUE	Len	Start	Description
Default Grade	4	25	Used when encountering current classes with BLANK grades on Banner. BAN40 will load this Default Grade (for example: NA, NR or whatever is entered here) into the rim_final_grade on the rad_class_dtl if the Banner SFRSTCR_GRDE_CODE is BLANK.
Transfer Repeat Skip A	1	29	A Y/N flag. Set to Y to skip the repeated transfer classes in Banner with an A in the shrtrce_repeat_course and NOT load them into the rad_transfer_dtl.
Transfer Repeat Policy A	1	30	If the Repeat Skip A flag above is set to N, then a valid Repeat Policy may be defined (valid values are 0-7 and B) or be left BLANK. Refer to the UCX-AUD047 Repeat Policy documentation included later in this document for a definition of these values. Policy B is recommended – the Averaged classes will end up in the Insufficient section of the audit and they will affect the overall GPA but will not be counted in the overall credits towards a degree. Although there are three repeat policy fields here they should be all set to the same value. These also should match the values set for the Repeat Policy fields above.
Transfer Repeat Skip E	1	31	A Y/N flag. Set to Y to skip the repeated transfer classes in Banner with an E in the shrtrce_repeat_course and NOT load them into the rad_transfer_dtl.
Transfer Repeat Policy E	1	32	If the Repeat Skip E flag above is set to N, then a valid Repeat Policy may be defined (valid values are 0-7 and B) or be left BLANK. Refer to the UCX-AUD047 Repeat Policy documentation included later in this document for a definition of these values. Policy B is recommended – the Excluded classes will end up in the Insufficient section of the audit but they will not affect the overall GPA or credits.
Transfer Repeat Policy I	1	33	A valid Repeat Policy (0-7 or B) may be defined for repeated transfer classes that have an I in the shrtrce_repeat_course or be left BLANK. Refer to the UCX-AUD047

UCX VALUE	Len	Start	Description
			Repeat Policy documentation included later in this document for a definition of these values. Policy B is recommended – the Included classes will apply to rules as normal classes affecting the GPA and total credits.
Advisor Method	1	34	<p>This flag controls how advisors are retrieved from the SGRADVR table – the SPRIDEN_ID for the Advisor records selected for the Advisor Methods below will be written to the rad_goaldata_dtl rad_goal_value with a rad_goal_code of ADVISOR. Each Advisor ID will be written to a separate rad_goaldata_dtl record.</p> <p>A - the four sets of Advisor Codes below the Follow Gradable Indictor flag will be searched for a match in up to 3 SGRADVR_ADVR_CODES in each set. This means that a maximum of four different ADVISOR records may be created for a given student using this method. If a match is found the associated SPRIDEN_ID for the SGRADVR advisor will be loaded into the rad_goaldata_dtl.rad_goal_value with a rad_goal_code of ADVISOR. Duplicate ADVISOR records will NOT be created if the SGRADVR_ADVR_CODE is found in more than one of the 12 available UCX_CFG020 ADVR_CODES listed below.</p> <p>Only the ADVISOR from the first matching SGRADVR_ADVR_CODE will be written to the rad_goaldata_dtl. In addition, the SGRADVR_ADVR_CODE is written to the rad_goaldata_dtl.rad_attach_code and the matching Advisor Code set below (1-4) is written to the rad_goaldata_dtl.rad_attach_value.</p> <p>C - the advisors will be filtered by matching the SGRADVR_ADVR_CODE to the Advisor Major and Advisor Minor listed below if the student has a valid rad_goaldata_dtl record for a MAJOR or a MINOR.</p> <p>S – the SPRIDEN_IDs for all of the selected advisors will be written to a separate rad_goaldata_dtl rad_goal_value</p>

UCX VALUE	Len	Start	Description
			<p>with a rad_goal_code of ADVISOR. Duplicate ADVISOR records will NOT be created if multiple records with the same SGRADVR_ADVR_PIDM are found for a given student.</p>
Advisor Major	4	35	<p>If the Advisor Retrieval flag is set to C above, the SGRADVR records retrieved for a given student will be searched for values matching this Advisor Major. If the student has a valid MAJOR rad_goaldata_dtl record and a Primary Advisor has been found for the student (SGRADVR_PRIM_IND = Y), the associated SPRIDEN_ID for the SGRADVR advisor will be loaded into the rad_goaldata_dtl.rad_goal_value with a rad_goal_code of ADVISOR.</p> <p>The record with the SGRADVR_PRIM_IND = Y will always be loaded with a rad_goal_seq of 0001. All other SGRADVR records selected for the student with the SGRADVR_ADVR_CODE equal to the Advisor Major value will also be loaded. The Advisor SPRIDEN_ID for the matching SGRADVR records will be written to the rad_goaldata_dtl.rad_goal_value with a rad_goal_code of ADVISOR.</p> <p>When extracting students' advisors, BAN40 looks first for a primary advisor (SGRADVR_PRIM_IND = Y). If it does not find one, it stops and does nothing else with advisors. If it does find one, it continues and extracts additional (non-primary) advisors.</p>
Advisor Minor	4	39	<p>If the Advisor Retrieval flag is set to 'C' above then the SGRADVR records retrieved for a given student will be searched for values matching this Advisor Minor. If the student has a valid MINOR rad_goaldata_dtl record then all SGRADVR records selected for the student with the SGRADVR_ADVR_CODE equal to the Advisor Minor will be loaded. The Advisor SPRIDEN_ID for the matching SGRADVR records will be written to the</p>

UCX VALUE	Len	Start	Description
			rad_goaldata_dtl rad_goal_value with a rad_goal_code of ADVISOR.
Reserved	1	43	Reserved for future use.
Cross List in SCREQIV	1	44	A Y/N flag. Set to Y if cross listed courses are to be filtered out by the ban43 equivalency load program. This means dap_eqv_crs_mst records will NOT be created for cross listed courses. If cross listed courses are not stored in the SCREQIV table at your site then set this flag to N. Setting this flag to N will preserve any cross-listed courses you may have manually added to UCX_CFG073.
			Please also see the integration.banner.extract.equiv.crosslistedRange setting.
Repeatable Option	1	45	This option flag contains several values to assist in the appropriate handling of repeatable courses that can be taken several times for credit (e.g., Music Lessons, PE activity classes, etc.). It is important for Degree Works to be able to differentiate these repeatable classes from classes that have been repeated for a better grade. There are currently seven different Repeat Policies that may be used to control the handling of classes repeated for a better grade in Degree Works (e.g., 1 = Keep most recent repeat, 2 = Keep class with highest grade, etc.). These Repeat Policies are defined in the UCX-AUD047 section. Each SHRTCKN_REPEAT.Course_IND (A-Averaged, E-Excluded and I-Included) must have one of these seven Repeat Policies loaded into the following fields defined above in this UCX-CFG020BANNER record: Repeat Policy A, Repeat Policy E and Repeat Policy I. The Banner Extract uses these values to load the appropriate UCX-CFG020BANNER Repeat Policy into the rad_repeat_plcy on the rad_class_dtl.

UCX VALUE	Len	Start	Description
			<p>In addition, Banner has several rules used to control the registration of repeatable classes outlined in Chapter 12 of the Banner Registration documentation. Because both repeatable classes and classes that may be repeated for a better grade may have an I value in the SHRTCKN_REPEAT_IND, these registration rules are being used in the Banner Extract to correctly identify these different types of classes when extracted into Degree Works so that rad_class_dtls are created correctly and subsequent degree audits are generated correctly.</p> <p>These rules are explained in more detail in the Class repeats/multiple occurrences topic. There are 5 different options to assist Degree Works in extracting the I (Included) class records appropriately based on values found in the Banner SCBCRSE record associated with each SHRTCKN class.</p>

Valid values:

N – DO NOT check the
SCBCRSE_REPEAT_LIMIT or
SCBCRSE_MAX_RPT_UNITS

L – Check the SCBCRSE_REPEAT_LIMIT
Only (Default if Repeatable Option BLANK)

U – Check the
SCBCRSE_MAX_RPT_UNITS Only

B – Check Both the
SCBCRSE_REPEAT_LIMIT and
SCBCRSE_MAX_RPT_UNITS

I – Include the class credits in the
SCBCRSE_REPEAT_LIMIT and
SCBCRSE_MAX_RPT_UNITS before
checking for repeatable classes using the
SCBCRSE_CREDIT_HR_LOW

Current Course 1	46	A flag used to indicate how the Banner Equivalency extract (ban43) determines if an OLD SCREQIV Course Key (Subject + Course Number) looked up on the SCBCRSE table with the maximum (MAX)
------------------	----	--

UCX VALUE	Len	Start	Description
			Effective Term is included in the 'Current Course Catalog'. This knowledge is required so that the 'circular' or 'reversal' entries contained in the Banner SCREQIV table (required for Banner registration purposes) can be filtered out and NOT extracted into Degree Works.
			The valid values are:
A			A – Look up the SCBCRSE_CSTA_CODE for the OLD Course Key on the STVCSTA table. If the STVCSTA_ACTIVE_IND = A, the course is considered Current and if the OLD Course Key is NOT REUSED (found on CFG074) then the course will be considered a circular or reversal course and will NOT be extracted into Degree Works.
C			C- The SCBCRSE_CSTA_CODE itself is checked. If it is equal to a C then the course is considered Current and if the OLD Course Key is NOT REUSED (found on CFG074) then the course will be considered a circular or reversal course and will NOT be extracted into Degree Works. The associated STVCSTA_ACTIVE_IND for the C STVCSTA_CODE can still be set to A for Banner purposes. This setting allows Degree Works to identify 'Current' Courses in Banner (as opposed to just looking at the 'Active' Indicator) as courses appear to be 'Active' in Banner when they are really not part of the 'Current Course Catalog' which is what Degree Works really needs to know.
K			K – The SCBCRKY record is looked up for the OLD course. If the SCBCRKY_TERM_CODE_END is equal to '999999' the course is still Current/Active then the course will be considered a circular or reversal course and will NOT be extracted into Degree Works.
Follow Gradable Indicator	1	47	A Y/N flag. If set to Y and the SSBSECT_GRADABLE_IND = N for a current SFRSTCR class, then the class will be skipped and NOT extracted into Degree Works by the Banner student extract.

UCX VALUE	Len	Start	Description
Advisor #1 Codes	3 *4	48	<p>If the Advisor Method above is set to A this array of three 4-byte Advisor Codes (SGRADVR_ADVR_CODE) will be searched.</p> <p>If a match is found the associated Advisor SPRIDEN_ID code will be loaded into a rad_goal_value on the rad_goaldata_dtl with a rad_goal_code of ADVISOR by the Banner student extract (ban40). In addition, the SGRADVR_ADVR_CODE will be loaded into the rad_goaldata_dtl.rad_attach_code and a 1 will be loaded into the rad_goaldata_dtl.rad_attach_value.</p> <p>If NO matches are found, no ADVISOR rad_goaldata_dtl record will be generated for this set of Advisor Codes.</p>
Advisor #2 Codes	3 * 4	60	<p>If the Advisor Method above is set to A this array of three 4-byte Advisor Codes (SGRADVR_ADVR_CODE) will be searched.</p> <p>If a match is found the associated Advisor SPRIDEN_ID code will be loaded into a rad_goal_value on the rad_goaldata_dtl with a rad_goal_code of ADVISOR by the Banner student extract (ban40). In addition, the SGRADVR_ADVR_CODE will be loaded into the rad_goaldata_dtl.rad_attach_code and a 2 will be loaded into the rad_goaldata_dtl.rad_attach_value.</p> <p>If NO matches are found, no ADVISOR rad_goaldata_dtl record will be generated for this set of Advisor Codes.</p>
Advisor #3 Codes	3 * 4	72	<p>If the Advisor Method above is set to A this array of three 4-byte Advisor Codes (SGRADVR_ADVR_CODE) will be searched.</p> <p>If a match is found the associated Advisor SPRIDEN_ID code will be loaded into a rad_goal_value on the rad_goaldata_dtl with a rad_goal_code of ADVISOR by the Banner student extract (ban40). In addition,</p>

UCX VALUE	Len	Start	Description
			<p>the SGRADVR_Advr_CODE will be loaded into the rad_goaldata_dtl.rad_attach_code and a 3 will be loaded into the rad_goaldata_dtl.rad_attach_value.</p> <p>If NO matches are found, no ADVISOR rad_goaldata_dtl record will be generated for this set of Advisor Codes.</p>
Advisor #4 Codes	3 * 4	84	<p>If the Advisor Method above is set to A this array of three 4-byte Advisor Codes (SGRADVR_Advr_CODE) will be searched.</p> <p>If a match is found the associated Advisor SPRIDEN_ID code will be loaded into a rad_goal_value on the rad_goaldata_dtl with a rad_goal_code of ADVISOR by the Banner student extract (ban40). In addition, the SGRADVR_Advr_CODE will be loaded into the rad_goaldata_dtl.rad_attach_code and a 4 will be loaded into the rad_goaldata_dtl.rad_attach_value.</p> <p>If NO matches are found, no ADVISOR rad_goaldata_dtl record will be generated for this set of Advisor Codes.</p>
Check for Dual Degree	1	96	<p>A Y/N flag.</p> <p>If set to 'Y' then a Dual Degree will be checked for on the SGBSTDN record.</p> <p>If non-blank values are found in the SGBSTDN_LEVL_CODE_DUAL, SGBSTDN_DEGC_CODE_DUAL and SGBSTDN_MAJR_CODE_DUAL fields then the Dual Degree will be processed along with the other degree information found in either the SORLCUR/SORLFOS records (used if they exist) or the SGBSTDN record (if no SORLCUR/SORLFOS records exist for a given student).</p> <p>If the Dual Degree LEVL/DEGC codes are unique then a separate rad_goaldata_dtl will be built for the MAJOR with the rad_goal_value loaded with the Dual MAJOR and the rad_goal_code loaded with</p>

UCX VALUE	Len	Start	Description
			ADVISOR. Otherwise the Dual Degree data will be skipped and NOT written to the rad_goaldata_dtl in the Degree Works database.
Always Process SHRATTC	1	97	A Y/N flag. Historic class attributes are stored in two tables: SHRATTR and SHRATTC. This flag controls whether attributes from SHRATTC are extracted by ban40 if attributes from both tables exist. SHRATTR class attributes are processed first. If any SHRATTR attributes are found for a given class they are written to the rad_attr_dtl. Then this Always Process SHRATTC flag is checked:
			Set to 'N' if the SHRATTC attributes are to be skipped for a given class if attributes from SHRATTR already exist for that class. This is the default value if this flag is left BLANK.
			Set to 'Y' if the SHRATTC class attributes should always be processed and written to the rad_attr_dtl if found for a given class, even if SHRATTR attributes exist for that class. In this case class attributes from both the SHRATTR/SRATTC Banner tables would be written to the rad_attr_dtl.
Create GPA RAD_REPORT_DTL	1	98	A Y/N flag. Set to Y if the records in the SHRLGPA GPA/HOURS_EARNED are to be formatted and written to the rad_report_dtl by the Banner Bridge extract for each student. The rad_report_codes used are: GPAXx where xx is loaded based on the SHRLGPA_GPA_TYPE_IND; I= IN for Institutional GPA, O=OV for Overall GPA and T=TR for Transfer GPA. For hours/ credits earned the rad_report_codes used are: GPACREDITSxx where the xx is loaded the same as the GPA values. Valid rad_report_codes: GPAIN, GPAOV, GPATR,

UCX VALUE	Len	Start	Description
			GPACREDITSIN, GPACREDITSOV and GPACREDITSTR.
			These GPA/Credits values may then be displayed on the audit headers using these keywords.
Use Program as Degree	1	99	A Y/N flag.
			Set to Y to load the Banner Program (SORLCUR_PROGRAM if using Concurrent Curriculum, SGBSTDN_PROGRAM1, SGBSTDN_PROGRAM2 if not using Concurrent Curriculum) as the Degree Works degree (rad_goal_dtl.rad_degree_code, rad_goaldata_dtl.rad_degree_code). Default is N. Also see Process Applicants below.
Skip Inactive Records in SCBCRKY	1	100	A Y/N flag. Used by the EQUIV data extract (ban43). The default is Y or blank.
			Set to Y or leave blank if Inactive records found in the SCBCRKY table for a given SCREQIV New Course Key (SCBCRKY_TERM_CODE_END NOT equal to '999999') are to be skipped and NOT processed as possible course equivalents.
			Set to N if Inactive records in the SCBCRKY table for a given SCREQIV New Course Key (SCBCRKY_TERM_CODE_END NOT equal to '999999') are NOT to be skipped and should be evaluated as possible course equivalents.
Process Applicants	1	101	A Y/N flag.
			Set to Y if applicant processing by the ban40 extract is to be performed.
			Set to N if NO applicant data is to be looked up or potentially extracted into Degree Works when running in STUDENT

UCX VALUE	Len	Start	Description
			mode. This flag is ignored when running in APPLICANT mode.
			When this flag is set to Y and the Program as Degree flag is set to Y and you might end up with degree records for each LEARNER record and for each ADMISSIONS record. For this reason you may want to turn this flag off when Program as Degree is set to Y.
Process Both Goals	1	102	A Y/N flag. Set to Y if goal (degree) data from Banner student data and applicant data is to be processed and imported into Degree Works.
			Set to N if the 'ADMISSIONS' SORLCUR/ SORLFOS records are NOT to be looked up or imported into Degree Works if at least one set of 'LEARNER' SORLCUR/ SORLFOS records is found.
Load SARADAP Goals	1	103	A Y/N flag. Set to Y if applicant data from the SARADAP table is to be imported into Degree Works if no SORLCUR 'ADMISSIONS' record is found.
			Set to N if NO SARADAP goal data is to be loaded into Degree Works.
Cross List Term	1	104	A flag that is used to control what SCREQIV term is used to load the UCX-CFG073 Cross-listed Term by the EQUIV extract (ban43). The valid values are: B = Load the UCX-CFG073 Cross-listed Term with BLANKS. Also, loads the UCX-CFG073 With Operator with BLANKS. L = Load the UCX-CFG073 Cross-listed Term with the lower of the two terms: SCREQIV_EFF_TERM or the SCREQIV_START_TERM if the start term does NOT match 000000 and does NOT

UCX VALUE	Len	Start	Description
			match the lowest STVTERM. Also, loads the UCX-CFG073 With Operator with >=.
			S = Load the UCX-CFG073 Cross-listed Term with the SCREQIV_START_TERM if the start term does NOT match 000000 and does NOT match the lowest STVTERM. Also, loads the UCX-CFG073 With Operator with >=. This setting is also the DEFAULT if this flag is left blank.
Transfer/ Program Attributes	1	105	A Y/N flag.
			Set to Y to generate a rad_attr_dtl record for Transfer Classes which have been associated to a Program in SHRDGMR. Such Transfer Classes are identified as having SHRTRCD_APPLIED_IND = 'Y'. The rad_attr_value will be populated by the associated SHRDGMR_PROGRAM value.
Reserved	1	106	Reserved for future use.
Allow writing to SHRTRIT and SHRTRAM	1	107	A Y/N flag. Set to Y to enable Transfer Equivalency to write required SHRTRIT and SHRTRAM records in Banner before rolling articulated transfer classes to SHRTTRK. If blank or N then the user must complete the SHATRNS screen in Banner before rolling classes from Transfer Equivalency.
In-progress repeat of historic class check	1	108	A Y/N flag. Set to Y to tell the Banner student extract (RAD30) to check for an equivalent Course Key match using the dap_eqv_crs_mst when trying to identify In-Progress classes that are repeats of historic/transfer classes that have different Course Keys due to changes in Course Keys over time. The Banner equivalent extract (RAD38) must have been run to load the dap_eqv_crs_mst with equivalencies from the SCREQIV Banner table or CFG070 must be manually updated using Controller and the dapucx2eqv script run to load the dap_eqv_crs_mst from the CFG070 table

UCX VALUE	Len	Start	Description
			after manual changes have been completed.
			If set to N the student extract will continue to only check for exact Course Key matches when trying to identify current classes that are In-Progress repeats of historic/transfer classes.
			Note: If the class has been extracted with Apply Historic Repeats set to No but the client runs the audit and deselects the In-progress flag, the system will revive and apply the historic class instead of sending it to insufficient.
Count Excluded Class in Major/Minor GPA	1	109	A Y/N flag. Only valid when the Repeat Policy E or the Transfer Repeat Policy E field is set to B and when the Skip E flags are set to N.
			Set to Y to have the excluded classes count in the GPA for the major/minor blocks where they would have applied. The UCX-CFG020 DAP14 Insufficient Major GPA or Insufficient Minor GPA or Insufficient OTHER GPA flags must be set to Y of course.
Count Averaged Class in Major/Minor GPA	1	110	A Y/N flag. Only valid when the Repeat Policy A or the Transfer Repeat Policy A field is set to B and when the Skip A flags are set to N.
			Set to Y to have the averaged classes count in the GPA for the major/minor blocks where they would have applied. The UCX-CFG020 DAP14 Insufficient Major GPA or Insufficient Minor GPA or Insufficient OTHER GPA flags must be set to Y of course.
Create RAD_ATTR_DTL	1	111	A Y/N flag.

UCX VALUE	Len	Start	Description
for classes in SHRDMGR			Set to Y to generate a rad_attr_dtl record for Current/History Classes which have been associated to a Program in SHRDGMR.
			For Current Classes (SFRSTCR) the SGBSTDN record with the max effective term <= SFRSTCR_TERM_CODE is found. The rad_attr_value will be populated by the SGBSTDN_Program_1.
			For History Classes (SHRTCKN) the SHRTCKN_SEQ_NO is used to lookup SHRTCKD using the SHRTCKD_TCKN_SEQ_NO. The SHRTCKN_TERM_CODE must also match the SHRTCKD_TERM_CODE.
			If the SHRTCKD_APPLIED_IND is not null the SHRTCKD_DMGR_SEQ_NO is used to lookup the SHRDGMR using the SHRDGMR_SEQ_NO. The rad_attr_value will be populated by the associated SHRDGMR_PROGRAM value.
Include In- Progress Withdrawn Classes	1	112	A Y/N flag. Set to Y to bridge to Degree Works any in-progress class (SFRSTCR) which has been set to Withdraw status. Rather than appear as InProgress = Y in Degree Works, classes with RSTS_CODE values where STVRSTS_WITHDRAW_IND = Y will appear as InProgress = N and Withdraw = Y. Note that clients who want to implement this feature must modify the SFRSTCR query in integration.banner.extract.config to allow SFRSTCR records with RSTS_CODE values where STVRSTS_WITHDRAW_IND = Y to be selected for import into Degree Works.
Apply Historic Repeat	1	113	A Y/N flag. Only obeyed when Repeat Policy=B.

UCX VALUE	Len	Start	Description
			Set to Y to apply the historic part of the repeat set and put in-progress portion of the repeat into insufficient.
			Set to N to apply the in-progress class and place the historic class into insufficient.
Load Current Grade from SFRSTCR	1	114	A Y/N flag. The purpose of this flag is to load the current grade from SFRSTCR into the rad_final_grade if one exists for current classes that have not been rolled to SHRTCKN during the grading period. Refer to the Load a current grade special edits topic for details. The Grading Period Inc and Grading Period Length values defined below are also used. Note: SFRSTCR2 SQL is required in integration.banner.extract.config. It does NOT filter out records based on NULL SFRSTCR_GRDE_DATE. Other customizations may be made to this query to select the appropriate SFRSTCR records for your student population. SFRSTCR records will be filtered by ban40 as it processes the special edits outlined in the area below this table.
			If set to N, load Default current grade listed above (Default Current Grade) into the rad_final_grade for the current SFRSTCR class record.
			If set to Y perform the special edits defined at the end of this document. See the Load a current grade special edits topic for details.
			Warning! If this flag is set to Y more records will be read from the Banner database tables SFRSTCR and SHRTCKN. Be aware that the

UCX VALUE	Len	Start	Description
			bannerextract may run longer as ban40 will now have to read more SFRSTCR records (not filtered by a NULL sfrstcr_grde_date because of the special processing rules required for this enhancement). During the grading period defined by the Grading Period Increment and Grading Period Length settings, bannerextract may take longer to complete as it will check each graded SFRSTCR record against SHRTCKN to prevent duplicates from being created.
Days after term end to load current grade	2	115	<p>The number of days (numeric and zero-filled) after the STVTERM_END_DATE that is considered the end of the grading period (for example, 00 – 99). Set to 00 if the end of the Grading Period is the same as the STVTERM_END_DATE. If this increment is > 0 it will be added to the STVTERM_END_DATE to calculate a new Grading Period End Date which will be used to determine the time-period where the special edits below will be performed to properly load the current grade.</p> <p>For example, if the STVTERM_END_DATE is 17-DEC-10 but the Grading Period actually ends 22-DEC-10 then the Grading Period Increment would be 05 (22 – 17).</p> <p>Warning! MUST be loaded with a valid numeric value 00 – 99 if the Load Current Grade if Exists flag above = Y.</p>
Days in term grading period	2	117	The number of days (numeric and zero-filled) in the STVTERM term's grading period (for example, 00 - 99). A Grading Period Length of 00 indicates that the grading period extends from the STVTERM_START_DATE through the Grading Period End Date

UCX VALUE	Len	Start	Description
			(STVTERM_END_DATE + Grading Period Increment above).
			This value contains the number of days for a given term that is set aside as a grading period where a graded class may exist in SFRSTCR but may or may not have been rolled to SHRTCKN. It is used in conjunction with the Grading Period Increment value above.
			For example, if the Grading Period End is 22-DEC-10, but faculty can start turning in grades on 15-DEC-10 the Grading Period Length would be 7 (Dec 22 – Dec 15).
			Warning! MUST be loaded with a valid numeric value 00 – 99 if the Load Current Grade if Exists flag above = Y.
Reserved	1	119	Reserved for future use.
Load Grade Issued by Transfer School	1	120	A Y/N flag. Set to Y to load the grade issued by the transfer school (SHRTRCR_TRANS_GRADE) rather than the grade issued by your institution (SHRTRCE_GRDE_CODE) into Degree Works.
			Note that RAD_FINAL_GR_NUM will be populated by the lookup of the SHRTRCE_GRDE_CODE in SHRGRDE and will not reflect the SHRTRCR_TRANS_GRADE.
Override SHRTRCR Grade	1	121	A Y/N flag. Set to Y if you want to override the RAD_FINAL_GRADE value. Note that the preceding flag Load SHRTRCR Grade must also be set to Y to use this functionality. In order to override the grade, create an entry in UCX-STU385 with a composite key of Level/Grade Type/Grade.

UCX VALUE	Len	Start	Description
Load Midterm Grade if it exists	1	122	Y=If midterm grade exists extract it instead of using the default grade.

Compare Applicant Degrees	1	123	A Y/N flag.
---------------------------	---	-----	-------------

Used only if NO valid SORLCUR/SORLFOS records are found for a given ID code and the three Applicant flags above are set to Y: Process Applicants, Process Both Goals and Load SARADAP Goals.

If this Applicant Compare Degrees flag is set to BLANK or Y the SARADAP Applicant compare logic will look for a match with the SGBSTDN goal records using the School (Level) and Degree Code (this is how the pre_DW4.1.0 Banner extract currently works).

If set to N the comparison will only be made using the School code.

For example:

SGBSTDN School = UG, Degree = BA,
Term = 201110

SARADAP School = UG, Degree = BS,
Term = 201110

If this flag = N the SGBSTDN record with the BA degree would be extracted. The student LEARNER record takes precedence over an ADMISSIONS record when the Terms are equal or the SGBSTDN Term is greater than the SARADAP Term. In this case the SARADAP BS degree would be skipped and not imported into Degree Works. However, if the Terms are NOT equal and the SARADAP Term (for example, 201210) is higher than the SGBSTDN Term (for example, 201110) the SARADAP BS degree would be extracted and the SGBSTDN BA degree would be skipped.

If this flag = Y or BLANK both the SGBSTDN record with the BA degree and the SARADAP record with the BS degree would be extracted and imported into Degree Works. The Term, regardless of the

UCX VALUE	Len	Start	Description
			value on each record, would not be involved in this case because the School/Degree combinations do NOT match.
			Note: If the Terms are different, the matching record with the highest (most recent) term would be extracted and created as a rad_goal_dtl record in Degree Works.
Unlimited Repeats when REPEAT_LIMIT is null	1	124	A Y/N flag. If set to Y and the SCBCRSE_REPEAT_LIMIT is NULL then the Banner Course extract, ban41.ec, will load the RAD.Course_Mst.RAD_REPEAT_MAX with 99 (infinitely repeatable). If set to N or BLANK and the SCBCRSE_REPEAT_LIMIT is NULL then the RAD.Course_Mst.RAD_REPEAT_MAX will be loaded with 00 (NOT repeatable). If the SCBCRSE_REPEAT_LIMIT is NOT NULL then whatever value is in the SCBCRSE_REPEAT_LIMIT will be moved to the RAD_REPEAT_MAX.
Load SAP data for Primary Degree Only	1	125	A Y/N flag. If set to N only SAP data for the Primary SORLCUR record (lowest SORLCUR_PRIORITY_NO) selected for a student will be written to the Banner SAP tables (SHRSAPP and SHRSARJ). If set to Y SAP data for all SORLCUR Degree records selected for a student will be written to the Banner SAP tables (SHRSAPP and SHRSARJ). However, due to restrictions on Banner only one degree is expected per request. It is recommended that this setting be set to N. The SORLCUR3 query in integration.banner.extract.config is used for

UCX VALUE	Len	Start	Description
			SAP to control the Degree/Program records desired.
Mapping Term Option	1	126	T= SHBTATC term maps to transfer term/cat-yr (dap_tr_catyr_beg/end on the dap_mapping_dtl) M=SHBTATC term maps to the matriculation term/cat-yr (dap_cat_yr_start/stop on the dap_mapping_dtl)
Map terms to stu016 academic year	1	127	For the equivalency extract, set this flag to Y to map the SCREQIV terms to the academic years in UCX-STU016. Set the flag to N to use STVTERM to map the term to an academic year. You might need to use STU016 instead of STVTERM if the financial aid years on STVTERM do not match your academic years.
Reserved	42	128	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Load a current grade special edits

Updated: March 25, 2022

Extension of the Load Current Grade from SFRSTCR flag.

1. The goal of these special edits is to load the correct Current Grade from the Banner SFRSTCR record into the rad_class_dtl if one exists. Before these special edits being BAN40 moves the SFRSTCR_GRDE_CODE into the rad_final_grade regardless of whether it contains NULL or a valid grade in case the logic exits before the end of these special edits. If the rad_final_grade is NULL or BLANK after exiting these edits the Default Current Grade (listed above) will be loaded.
2. If the Grading Period Inc is numeric ("00" – "99") lookup the STVTERM using SFRSTCR_TERM_CODE. If the STVTERM_START_DATE is NOT in the future load it into the Grading Period Start Date and calculate the Grading Period End Date by adding the Grading Period Increment to the STVTERM_END_DATE:

$$\text{STVTERM_END_DATE} + \text{Grading Period Increment} = \text{Grading Period End Date}$$

For example, if the STVTERM_END_DATE = '17-DEC-10' and the GradingPeriodInc below is '05' the calculated Grading Period End Date would be '22-DEC-10'. This date will be used in future edits.

3. Check the Grading Period Length. If set to "00" that means the Grading Period extends from the STVTERM_START_DATE through the Grading Period End Date.

Now BAN40 checks Today's Date to see if it is within the Grading Period Start/End dates:

- If SFRSTCR_GRDE_DATE is NULL or BLANK perform additional edits:
 - a. If Today's Date > Grading Period End Date
 - a. The SFRSTCR class will be SKIPPED because the class is from an old term.
 - b. Go to Step #7 below – exit these edits and read/process the next SFRSTCR class.
 - b. If Today's Date < Grading Period Start Date
 - a. The SFRSTCR class is in the future and will be marked as "In-Progress" (rad_inprog_flag = 'Y').
 - b. The SFRSTCR class is in the future and will be marked as "In-Progress" (rad_inprog_flag = 'Y').
 - c. If Today's Date is >= Grading Period Start Date and Today's Date <= Grading Period End Date
 - a. The SFRSTCR class is current and will be marked as "In-Progress" (rad_inprog_flag = 'Y') and
 - b. the SFRSTCR_GRDE_CODE will be loaded into the rad_final_grade (in case it is loaded),
 - c. go to Step #6 below – exit these edits and continue processing this class.
 - If Grading Period Length = "00"
 - a. If Today's Date is >= Grading Period Start Date and Today's Date <= Grading Period End Date, then go to Step #5 below.
 - b. Else
 - a. The SFRSTCR class is NOT within the Grading Period and will be SKIPPED,
 - b. go to Step #7 below – exit these edits and read/process the next SFRSTCR class.
4. If the Grading Period Length is from "01" – "99", then BAN40 calculates the Number of Elapsed Days –this represents the number of days from Today's Date until the grading period ends. Then BAN40 compares the Number of Elapsed Days to the Grading Period Length. If the Number of Elapsed Days is LESS THAN or EQUAL TO the Grading Period Length and is NOT NEGATIVE, then SHRTCKN is read. If not, then the SFRSTCR class will be SKIPPED because it has been graded, but it is not yet time to grade the current class or because Today's Date is greater than the Grading Period End Date which means the class should already be in SHRTCKN.

Subtract Today's Date from the Grading Period End Date to calculate the Number Of Elapsed Days:

Grading Period End Date – Today's Date = Number Of Elapsed Days

For example, if the Grading Period Length = "07" then:

Grading Period	Today's	Number of	
End Date	- Date	= ElapsedDays	
22-DEC-10	- 01-SEP-10	= 113	-> Skip SFRSTCR (GREATER THAN 99)
22-DEC-10	- 14-DEC-10	= 8	-> Skip SFRSTCR (8 > 7)
22-DEC-10	- 15-DEC-10	= 7	-> Check SHRTCKN (7 = 7)
22-DEC-10	- 22-DEC-10	= 0	-> Check SHRTCKN (<= 7 but NOT NEGATIVE)
22-DEC-10	- 23-DEC-10	= -1	-> Skip SFRSTCR (NEGATIVE)

If the NumElapsedDays is NEGATIVE (e.g., -1) or GREATER THAN 99 (e.g., >= 100) then the SFRSTCR class will be SKIPPED and NOT imported into Degree Works. Go to Step #7 below – exit these edits and read/process the next SFRSTCR class.

If the Number Of Elapsed Days <= Grading Period Length and NOT NEGATIVE then go to Step #5 below.

5. Read SHRTCKN to see if a matching class is found for the SFRSTCR class, matching on Term and CRN.

If a match is found in SHRTCKN the SFRSTCR class is SKIPPED as it is a DUPLICATE.

If a match is NOT found in SHRTCKN then the SFRSTCR_GRDE_CODE will be used as it will have already been loaded into the rad_final_grade).

Conditions where the SFRSTCR class is SKIPPED and NOT imported into Degree Works:

- a. If the Number Of Elapsed Days > Grading Period Length
- b. The Number Of Elapsed Days is NEGATIVE
- c. Today's Date is LESS THAN the calculated "Grading Period Start Date"
- d. Today's Date is GREATER THAN the calculated "Grading Period End Date"

It should have a valid SFRSTCR_GRDE_DATE and is not within the grading period window so the class should already exist in SHRTCKN.

6. If the rad_final_grade is still blank the UCX_CFG020 BANNER Default Current Grade listed above will be loaded into the rad_final_grade for the current SFRSTCR class BIF record.
7. Exit these special edits.

UCX-CFG020 BLOCKPRI

Updated: March 25, 2022

When an audit is run, this record is used to control the priority of the blocks used in the audit.

The blocks are placed on the audit tree in this order. However, the order of the blocks is unimportant to the auditor when applying courses and the blocks may be reordered when the worksheet is presented to the user.

UCX KEY	Len	Description	
BLOCKPRI	8	BLOCKPRI	
Reserved	22	Reserved for future use.	
UCX VALUE	Len	Start	Description
ID Priority	2	1	Priority of the ID requirements block if a block of type ID is part of the audit. Default = 01. Valid values are 01 - 10. Each number can be used only one time in this record.
Degree Priority	2	3	Priority of the DEGREE requirements block if a block of type DEGREE is part of the audit. Default = 02. Valid values are 01 - 10. Each number can be used only one time in this record.
Major Priority	2	5	Priority of the MAJOR requirements block if a block of type MAJOR is part of the audit. Default = 03. Valid values are 01 - 10. Each number can be used only one time in this record.
Concentration Priority	2	7	Priority of the CONC requirements block if a block of type CONC is part of the audit. Default = 04. Valid values are 01 - 10. Each number can be used only one time in this record.
Minor Priority	2	9	Priority of the MINOR requirements block if a block of type MINOR is part of the audit. Default = 05. Valid values are 01 - 10. Each number can be used only one time in this record.
Specialization Priority	2	11	Priority of the SPEC requirements block if a block of type SPEC is part of the audit. Default = 06. Valid values are 01 - 10. Each number can be used only one time in this record.
Liberal Learning Priority	2	13	Priority of the LIBL requirements block if a block of type LIBL is part of the audit. Default = 07. Valid values are 01 - 10. Each number can be used only one time in this record.
College Priority	2	15	Priority of the COLLEGE requirements block if a block of type COLLEGE is part of the audit. Default = 08. Valid values are 01 - 10. Each number can be used only one time in this record.
School Priority	2	17	Priority of the SCHOOL requirements block if a block of type SCHOOL is part of the

UCX VALUE	Len	Start	Description
			audit. Default = 09. Valid values are 01 - 10. Each number can be used only one time in this record.
Program Priority	2	19	Priority of the PROGRAM requirements block if a block of type PROGRAM is part of the audit. Default = 10. Valid values are 01 - 10. Each number can be used only one time in this record.
Reserved	30	21	Reserved for future use.
Starting Block Type	12	51	Default = "DEGREE". Valid values are: DEGREE, ID, MAJOR, MINOR, CONC, SPEC, LIBL, COLLEGE, SCHOOL, PROGRAM, OTHER
Starting Block Value	12	63	Default = spaces. Must be filled in if START-BLOCK is "OTHER".
One Per Tag	1	75	Y/N flag. For each blocktype/value on a block, should a student be audited against all blocks that match OR only the most specific? Y = Use only one requirement block per primary Database tag, the BEST match. N = Use all matching requirement blocks per primary Database tag. For example, WEB28.
Other Tag	1	76	Y/N flag. For each OTHER blocktype/value on a block, should a student be audited against all OTHER blocks that match OR only the most specific? Y = Use only one requirement block per primary OTHER Database tag, the BEST match. N = Use all matching requirement blocks per primary OTHER Database tag. For example, WEB28.
Reserved	93	77	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.

UCX VALUE	Len	Start	Description
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

The first block is the one designated in UCX-CFG020 as the starting block. The remaining blocks are sorted by BLOCKTYPE using the UCX-CFG020 priority rank as the sort order. The priority rank assigned to each of the ten blocktypes in UCX-CFG020 must be a unique number from one to ten; no duplicates are allowed. Blocks of type OTHER are placed at the end of the audit tree unless the OTHER block is the starting block.

The starting blocktype and value fields tell the software which block should be used as the starting (initial) block. This is usually the degree block but could be a university requirements block that is universal to all degrees. The starting block must exist for the audit to proceed. The default starting blocktype is DEGREE with value = " ". The starting value could contain a wildcard, for example, starting blocktype = OTHER and starting value = UR@.

Typically, the priority values are left alone and not changed by schools. The default values installed are usually never changed.

ONE-PER-TAG example for Anthropology Student with catalog year 20052006:

Two blocks have been defined in Scribe for DEGREE=BA with the following tags:

BLOCK #1: Blocktype	DEGREE
Value	BA
Start Catalog Year	20022003
Stop Catalog Year	99999999
BLOCK #2: Blocktype	DEGREE
Value	BA
Start Catalog Year	20022003
Stop Catalog Year	99999999
Major	ANTH

If ONE-PER-TAG is Y, then the student is audited with BLOCK #2 – because of the extra secondary tag for major. If ONE-PER-TAG is N, then the student is audited with BOTH blocks.

UCX-CFG020 COURSELINK

Updated: March 25, 2022

The COURSELINK record of UCX-CFG020 controls the Section information of the pop-up that appears when a course from the worksheet advice is clicked.

UCX KEY	Len	Description
COURSELINK	10	COURSELINK

UCX KEY	Len	Description	
Reserved	20	Reserved for future use.	
UCX VALUE	Len	Start	Description
Sections start term	8	1	When reading from Banner this term is used to select course sections greater than or equal to this term. Normally this field is left blank – but is useful for testing.
Sections Term days old	2	9	When reading from Banner sections are found with a start date that is within the last 14 days or has a start date that is in the future. This value changes that 14 days value to some other number. When left blank 14 days is used.
Reserved	159	11	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 DAP13

Updated: March 25, 2022

The DAP13 record of UCX-CFG020 controls the operation of the Parser Engine. The GETREQ, SAVEREQ, and SAVEADVICE subroutines also use the Upshift-Flag setting.

UCX KEY	Len	Description	
DAP13	5	DAP13	
Reserved	25	Reserved for future use.	
UCX VALUE	Len	Start	Description
Maximum Discipline Length	2	1	Should always be 12. Maximum length of a discipline code.
Maximum Parser Errors	3	3	Maximum number of Parser errors allowed. Any value from 000 through 999, the default is 025. The Parser uses this value to short-circuit the parsing process after the maximum number of errors has been found.

UCX VALUE	Len	Start	Description
Validate Courses	1	6	<p>Y/N.</p> <p>Y = validate course in student system. Y = validate course in the rad_course_mst. All courses that do not include wildcards or course number ranges must be valid in the student system. Courses must be set up in the student system before entering Degree Works requirements.</p> <p>N = do not validate courses in the rad_course_mst from the student system. Degree Works requirements can be entered before courses are loaded from the student system.</p> <p>The default is "Y".</p>
Maximum Numeric Grade or GPA	6	7	<p>Maximum numeric grade or GPA that the student system will assign or calculate. It is in the form 999v999, with the decimal point implied by the "v". The default is 004000 (equivalent to 004.000). The Parser validates the values associated with MINGPA and MINGRADE against this maximum.</p>
Upshift Codes	1	13	<p>Y/N.</p> <p>Y = Upshift codes before validation or search. Set this flag to Y if all codes are upper-case for Degree, School, Major, Minor, Concentration, Liberal Learning, Specialization, Program, Discipline, and Transfer.</p> <p>N = do not upshift codes.</p> <p>Default = "Y".</p>
Show Errors	1	14	<p>Controls the reporting of Parser errors. A = Parser reports all errors for a token. F = Parser reports only the first error for a token.</p>
Require Labels	1	15	This flag should always be set to Y.
Maximum Course Number Length	2	16	Should always be 12. Length of course number in the course-key.

UCX VALUE	Len	Start	Description
Validate Disciplines	1	18	Y/N. Y = validate discipline in UCX-STU352. Default=Y
Allow Duplicate Labels	1	19	Y/N. Y=Duplicate labels cause problems with exceptions. N is recommended.
Wildcard Character Match	1	20	0=Wildcard matches 0 or more characters, 1=Wildcard matches 1 or more characters, etc.
Process cross-listed courses	1	21	N, Y, W, R, A. Process cross listed courses from UCX-CFG073 and alter Scribed rules. See the Process Cross-Listings topic for additional information.
Process Equivalent Courses	1	22	Y=Process equivalence courses from dap_eqv_crs_mst (UCX-CFG070) and alter Scribed rules. See the Process Equivalences topic for additional information.
Reserved	147	23	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Process Cross-Listings

Updated: March 25, 2022

When a block is parsed and the Process cross-listed courses flag is enabled, the parser looks up each course in the cross-listings table.

If a match is found then the parser adds the course on the requirement with the new course within Hide. The parser does not change the scribe text; the parser only changes the parsed file saved in the daptrees directory that is used by the auditor. For example, if the rule is scribed like this:

```
1 Class in MATH 123
Label MKNYBL "Intro to Logic";
```

And if there is a cross-listing for MATH 123 to PHIL 145 in the cross-listing table then the rule is changed to this:

```
1 Class in MATH 123 {Hide PHIL 145}
Label MKNYBL "Intro to Logic";
```

You cannot see this change in Scribe; you can see this change in the Diagnostics Report.

Other than the N to disable this feature there are four other flags available to enable this feature:

- Y - Process specific courses only
- A - Process specific courses, wildcards and ranges
- W - Process specific courses and wildcards
- R - Process specific courses and ranges

When the flag is A or W and you have scribed this: 1 Class in ACCT 2@ and you have these cross-listings set up:

ACCT214 =>	BUSN	234
ACCT219 =>	BUSN	226
ACCT237 =>	BUSN	241

Then you will get this result:

```
1 Class in ACCT 2@ {Hide BUSN 234, 226, 241}
```

However, if you have wildcards in both the discipline and course number no cross-listings will be used. Here are two examples which will not result in any cross-listings being inserted:

```
1 Class in @ @ (With Attribute=ABCD) # scribing a single @
is equivalent to using two wildcards
3 Credits in MA@ 2@
```

When the flag is A or R and you have scribed this: 1 Class in ACCT 200:299 and you have these cross-listings setup:

ACCT214 =>	BUSN	234
ACCT219 =>	BUSN	226
ACCT237 =>	BUSN	241

Then you will get this result:

```
1 Class in ACCT 200:299 {Hide BUSN 234, 226, 241}
```

Setting the flag to just Y means that wildcards and ranges are not processed.

Process Equivalences

Updated: March 25, 2022

When a block is parsed and the Process equivalent courses flag is enabled, the parser looks up each course in the equivalence table.

If a match is found then the parser replaces the course on the requirement with the new course. The parser does not change the scribe text; the parser only changes the parsed file saved in the daptrees directory that is used by the auditor. For example, if the rule is scribed like this:

```
1 Class in ACCT 123
Label ASD0I2 "Business Accounting";
```

And if there is an equivalence for ACCT 123 to BUSN 145 in the equivalence table then the rule is changed to this:

```
1 Class in BUSN 145
Label ASD0I2 "Business Accounting";
```

On the parsed file we also keep a note of what the former course was and this then shows in the Diagnostics Report as a "formerly" notation showing the course that was scribed.

When the requirement is scribed with a wildcard or range no lookup on the equivalence table is made. For example, if this is scribed in the same block:

```
1 Class in ACCT 1@*
Label ASD234 "Accounting Requirement";
```

Then no replacement is made – the rule remains as-is.

When making this replacement, the catalog year of when the student took the course is not known – because there is no student involved in the parsing process. Because of this, the first catalog year field on the UCX-CFG070 record is ignored because it represents when the class was taken. But note, when an audit is run and the classes the student did take are looked up on the equivalence table the term on each class is used to find the correct equivalence record. For example, if this same block was being parsed and these records existed in the equivalence table:

```
2008 ACCT 123 BUSN 139
2015 ACCT 123 BUSN 145
```

The parser would find the first ACCT 123 record and it would be replaced with BUSN 139 and not BUSN 145. The first record is used because the catalog year is ignored. You may not want to use this CFG020 setting if you have past equivalencies in place for the same course; you may want to make the scribe changes manually.

UCX-CFG020 DAP14

Updated: September 29, 2023

The DAP14 record of UCX-CFG020 controls certain aspects of the Auditor Engine. This record is also used by many Degree Works subroutines.

UCX KEY	Len	Description	
DAP14	5	DAP14	
Reserved	25	Reserved for future use.	
UCX VALUE	Len	Start	Description
Passfail	1	1	Y/N.
Satisfies			Y = Courses with Passfail can satisfy a requirement that has a MinGrade qualifier.
Mingrade			N = Passfail course cannot be applied to a requirement that has a MinGrade qualifier.
			Default = N.
Reserved	1	2	Reserved for future use.
Transfer	1	3	Y/N.
Satisfies			Y = Transfer course with a blank letter grade or 0.0 numeric grade can apply to a requirement that has a MinGrade qualifier.
MinGrade			N = Transfer course with a blank letter grade or 0.0 numeric grade cannot apply to a requirement that has a MinGrade qualifier.
Qualifier			
			Default = N.
Reserved	1	4	Reserved for future use.
Audit History Depth	2	5	Audit history depth. The maximum number of audits to keep per student per School/Degree combination. 99 or 00 = keep all audits. 2 digit number 00 - 99.
			When set to 03, for example, 3 academic audits, 3 financial aid audits, and 3 athletic eligibility audits will be kept. This setting is used for academic, financial aid, and athletic audits but the counts are kept track of separately. This is not used for the what-if audits – see the What-if History Count flag below.
Over-the-limit Courses	1	7	Y/N/G.

UCX VALUE	Len	Start	Description
Will Over-the-Limit Courses Count in Major GPA	1	8	<p>Y = Courses in the Over-the-Limit section of the audit count in overall GPA and total credits toward the degree.</p> <p>N = Courses in the Over-the-Limit section of the audit do not count toward the degree. They print on the audit output but are otherwise unused.</p> <p>G = Courses in the Over-the-Limit section count towards the overall GPA and major GPA calculations (excluding insufficient classes moved here).</p> <p>Default = N.</p>
Will Insufficient Courses Count in Major GPA	1	8	<p>Y/N.</p> <p>Default = N.</p> <p>Y = Courses in the Insufficient section of the audit count in the major GPA if they could have applied. In addition, failed classes that could have applied to blocks that are being pulled into the major block are also included in the GPA. For example, if the major block is pulling in a CONC or OTHER block – failed classes applying to those blocks are also included in the GPA.</p> <p>However, if you have the UCX-CFG020 BANNER Repeat Policy set to B and you do not have the Excluded Repeats Count or the Averaged Repeats Count flags set to Y then these repeats will be forced directly to the insufficient section and will not apply to the Major GPA.</p>
Will Transfer Courses count in GPA	1	9	<p>Y/N.</p> <p>Y = Transfer courses count in block GPA and overall GPA.</p> <p>N = Transfer courses do not count in GPA calculation.</p> <p>Default = N.</p>
Will Incomplete Courses Apply	1	10	Y/N.

UCX VALUE	Len	Start	Description
			Y = Courses with grade of incomplete are applied by the Auditor to requirements blocks.
			N = Courses with grade of incomplete are not applied by the Auditor to requirements blocks but are put into the insufficient section of the audit.
			Default = N.
Will In-Progress courses apply	1	11	Y/N. Y = Courses that are in-progress are applied by the Auditor to requirements blocks.
			N = Courses that are in-progress are not applied by the Auditor to requirements blocks but they are put into the in-progress section of the audit tree.
			Default = Y.
Filter Classes by School	1	12	Y/N. Y = Filter classes based on the school on the rad_goal_dtl record.
			Default = N.
Approximate average credits per class	7	13	Average number of credits per class, approximately. A number with three significant digits, a decimal, and 3 digits after the decimal, for example, 004.000. The Auditor uses this number when calculating the percent complete for each rule that is not yet complete. Default = 003.000.
Reserved	5	20	Reserved for future use.
Check the CFG020 TIEBREAK First	1	25	Y/N. Y = For starting block qualifiers, the Auditor will evaluate the TIEBREAK configurations from UCX-CFG020 TIEBREAK before checking the number of fits for each course.

UCX VALUE	Len	Start	Description
			<p>N = For starting block qualifiers, the Auditor will evaluate the number of fits for each course before checking the UCX-CFG020 TIEBREAK config.</p> <p>Default = N.</p> <p>See the Check CFG020 TIEBREAK First topic for additional information.</p>
Round the GPA (Y) or Truncate GPA (N)	1	26	<p>Y/N.</p> <p>Default = Y.</p> <p>Y or Blank = Round GPA (2.1495 --> 2.150).</p> <p>N = Truncate GPA (2.1495 --> 2.149).</p> <p>This flag is used when the audit is transformed to an XML document and when the GPA appearing in ProxyAdvice and Display text is formatted.</p> <p>The Banner extract also uses this flag when extracting the SHRLGPA_GPA into the rad_cum_gpa field on the term record.</p>
Range In-Progress	1	27	<p>Y/N.</p> <p>Y or Blank = Remove in-progress classes from course rules expressed as ranges of classes and or credits, so the rule can be 100% complete.</p> <p>N = Keep in-progress classes on course rules expressed as ranges, even though the rule will be 98% complete.</p> <p>Default = Y.</p> <p>The Range In-Progress flag controls how the Auditor Engine handles in-progress classes applied to a rule with ranges of classes and credits. For example, MATH 100 is complete and MATH 200 is in progress. Both classes apply to the rule 3:6 CREDITS IN MATH @. If Range In-Progress is set to N, the rule is 98% complete even though 6 credits applied because one of the classes is still in</p>

UCX VALUE	Len	Start	Description
			progress. If Range In-Progress is set to Y, the Auditor will remove MATH 200, satisfying the rule with only the completed class, thereby making the rule 100% complete.
NOCOUNT to the Over-the-limit	1	28	Y/N. Y = Place any classes applied to NOCOUNT requirements in the over-the-limit section. Default = N.
Show Advice on Range Rules	1	29	Y/N. Y = Show advice on range rules if block qualifiers have not been satisfied. Default = N.
			The Range Advice flag controls whether advice should be given on course rules with ranges when block header qualifiers have not been satisfied. There may be course rules that say 2:6 CLASSES IN ENGL 2@ and 1:4 CLASSES IN LIT 2@ and a block qualifier may exist that says MINCLASSES 3 IN ENGL 2@, LIT 2@. By setting the Range Advice flag to Y the auditor creates advice for the rules as long as their maximums have not been reached and as long as the header qualifiers are still unsatisfied.
Reserved	29	30	Reserved for future use.
MAX Qualifier Advice	1	59	Y: If MaxCredits or MaxClasses qualifier is at its max change the rule advice within this scope to not include any classes listed under this qualifier. N: do not alter any rule advice.
Follow STU307 Remedial Mode	1	60	Y: If UCX-STU307 DgwRemedialMode is set to B, all remedial classes will be forced into over-the-limit. If UCX-STU307 DgwRemedialMode is set to A (or blank), remedial classes will be handled like normal classes.

UCX VALUE	Len	Start	Description
			N: Special Remedial mode not enabled.
			The Remedial Mode controls whether classes for certain degrees should be forced into over-the-limit. Remedial classes are defined as: any classes with a level under 100. Any classes that have letters in the level are not considered remedial. Examples of Remedial: 099, 95, 009, 45, 5. Examples of Not Remedial: 123, A99, 99A, 5AB, 300. This flag works in conjunction with the UCX-STU307 RemedialMode setting for each degree. If that flag is set to B, then remedial classes for that degree are forced into over-the-limit. If the flag is set to A, then remedial classes for that degree are treated as normal classes.
Deprecated - should be set to N	1	61	Y/N.
Will Insufficient courses count in Minor GPA	1	62	Y=Perform main audit logic again to maybe get better results. This will make the audit slower, and may actually result in less optimal results. This feature has been DEPRECATED and it should be set to N.
			Y=Insufficient minor courses count in minor GPA if they could have applied. In addition, failed classes that could have applied to blocks that are being pulled into the minor block are also included in the GPA. For example, if the minor block is pulling in a CONC or OTHER block – failed classes applying to those blocks are also included in the GPA.
Display GPA number of decimals	1	63	1,2, or 3. When displaying GPA, use 1, 2, or 3 decimal places. This works in combination

UCX VALUE	Len	Start	Description
			with the GPA Round Flag. This is primarily for the GPA values displayed in PROXY-ADVICE and DISPLAY statements. Other GPAs in the audit can be localized using XSL.
Check old course name	1	64	Y=when an equivalence is in place and the requirement contains a wildcard or range we should check the old course name in addition to new course name to see if a class meets a requirement.
In-progress vs Completed	1	65	Y=Remove in-progress courses from a rule in favor of keeping the completed course. For more information, see the Class evaluation on a rule topic. Warning! Setting this flag to Y alters the best-fit algorithm in such a way that you may not get ideal results.
Insufficient courses count in OTHER blocks GPA	1	66	Y/N. Y=Insufficient courses count in OTHER blocks GPA if they could have applied if they were not failed. In addition, failed classes that could have applied to blocks that are being pulled into the OTHER block are also included in the GPA. For example, if the OTHER block is pulling in another OTHER block – failed classes applying to that block are also included in the GPA. However, if you have the UCX-CFG020 BANNER Repeat Policy set to B and you do not have the Excluded Repeats Count or the Averaged Repeats Count flags set to Y then these repeats will be forced directly to the insufficient section and will not apply to the OTHER GPA.
Fallthru courses counts in the Overall GPA	1	67	Y/N. Y=classes in the fall-through section count in the overall GPA and credits/classes calculation. However, this flag does not affect how fall-through classes are counted against Max/Min qualifiers in the starting block – these classes are always counted

UCX VALUE	Len	Start	Description
			against these qualifiers regardless of this flag.
			N=fall-through classes are not counted in the overall GPA and credits/classes applied calculation.
Out of Sequence Classes count in the Overall GPA	1	68	Out of Sequence classes count in the Overall GPA
Out of Sequence count in the Residency Calcs	1	69	Out of Sequence classes count in the Residency Calculations
Maximum What-If audits per student	2	70	What-if Audit history depth. The maximum number of what-if audits to keep per student. 99 = keep all what-if audits. 2 digit number 00 - 99. When set to 03, for example, 3 what-if audits will be kept. However, additional frozen what-if audits may exist for each student.
			Set this flag to 00 or blanks to tell Degree Works to never save what-if audits. Only when what-if audits are saved (count >= 1) can they be frozen. Planner What-if audits are never saved.
			This setting is not used by the What-if API.
Apply exception to a similar block	1	72	If the block on which the exception was originally applied is not found in the audit then the auditor will attempt to apply the exception to another block of the same type/value. This is useful for when a student changes catalog years and a different MAJOR=CHEM block, for example, is pulled into the audit. The student still has a MAJOR=CHEM block in the audit but for a different catalog year and thus with a different RA number. The auditor will use the label-tag stored on the exception to locate a rule in this new block with the same label-tag; the node-id will not be used.

UCX VALUE	Len	Start	Description
			Set this flag to Y to always do this when the original block is not found.

Set this flag to A to only do this when the label-tag contains at least one alphabetic character (A-Z). Label-tag values like CHEMINTR0 are more trustworthy than a label-tag of 7, for example, because it is very easy for two rules to have a label-tag 7 but yet not be the same rule in the two blocks.

When this flag is Y or A the Similar major field on UCX-STU023 is also used. This is helpful when the major changes from ART to ARTH, for example. See the STU023 documentation for more details.

(N is the default.)

Calculate Elective Credits Allowed	1	73	<p>Y=Calculate a separate credits-applied value based on elective credits allowed. Using the CheckElectiveCreditsAllowed qualifier trumps this flag and produces different results so use one or the other – not both. See the CreditsAppliedTowardsDegree section of the Athletic Eligibility documentation to see how these credits are calculated.</p> <p>This flag tells the auditor to calculate this new credits-applied value in an identical way to how it is done for athletic audits. When this flag is enabled, the credits progress bar on worksheets will make use of this value instead of the other value that is calculated. Using this flag the auditor determines which blocks are required versus optional and makes appropriate credits calculations. This extra logic is not performed using the qualifier.</p> <p>When the Fallthru Counts in Overall flag is set to N these two credit calculations will be identical. When the flag is Y the overflow fall-through credits are subtracted from the primary credits total to give this additional credits value that is used in the progress bar.</p>
------------------------------------	---	----	--

UCX VALUE	Len	Start	Description
For more information, see the Elective Credits Allowed (ECA) Calculations in topics.			
Save GPA History	1	74	When using Tracking in SEP and you are using Overall or Major GPA requirements, you should set this flag to Y to have the auditor save overall and major GPA data, per term, to the dap_gpa_history table.
Alphabetic chars allowed in course range	1	75	Y=course numbers with alphabetic characters are allowed to fit in course ranges. For example, BIOL 123L will fit in the requirement BIOL 100:199. When this flag is Y the software simply strips out the letters from the course number and does a comparison using the remaining value. For example, BIOL 123L is treated as BIOL 123 for comparison purposes when dealing with a course number range. Course numbers with special characters (asterisk, underscore, etc.) will never apply to course ranges; the auditor traps for such course numbers and skips the comparison.
MinGrade insufficient classes count in GPA	1	76	Y=Classes not applied because of a MinGrade qualifier will count in block GPA and also any MinGPA class-list qualifiers. This is useful for D grades that are thrown out because of a MinGrade 2.0 qualifier; when this flag is set to Y the D grades can count in the GPA even though they won't count against any requirements or the block credits.
Obey equivalences in ranges and wildcards	1	77	<p>Y or blank=Equivalences are obeyed in courses with ranges and wildcards. That is if the course scribed in a rule or qualifier is MATH 2@ or MATH 200:299 and MATH 189 was renamed MATH 205 then the course will apply to both requirements because we use the new name of the course, MATH 205, when comparing it to the wildcard or range.</p> <p>When this flag is N this course will not apply because we then use the old name of the course, MATH 189, when comparing and it does not fit the range or the wildcard. When this flag is set to N you should also set the Check Old Coursename flag to N.</p>

UCX VALUE	Len	Start	Description
Failed classes count GPA	1	78	Y=failed classes do count in the overall GPA calculation. This also applies to all MinGPA qualifiers you have in your degree block – even those with a list of classes.
Recalculate the fit-rank frequently	1	79	Y=Recalculate the fit-rank each time a requirement is processed- recalculate it for the classes that were removed and for those that were kept. N is the old behavior wherein the auditor recalculated the fit-rank less frequently sometimes giving incorrect results. Changing the flag to Y does cause the auditor to do a bit more work but the performance impact should be very minor. You will likely see more accurate results with the flag set to Y but it is possible you will not – it all depends on how your blocks are scribed. It is recommended that you set the flag to Y and test and if you see problems you might find that at your school a setting of N works best.
Reserved	90	80	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Check CFG020 TIEBREAK First

Updated: March 25, 2022

For example, the starting block includes the following qualifier: MAXCLASS 1 IN PHYS @.

The student took PHYS 115 in Fall 1995 and PHYS 221 in Fall 1996. To satisfy the MAXCLASS qualifier, the Auditor must remove one of these two classes to Over-The-Limit. PHYS 115 was applied to rule: 1 CLASS IN PHYS @;. PHYS 221 was applied to rule: 1 CLASS IN PHYS 200, 221;. The UCX-CFG020 TIEBREAK is set to keep the course with the highest term (most recent course). PHYS 221 fits in two places (it fits both rules) but PHYS 115 fits in only one place. If the DAP14 Check the CFG020 TIEBREAK first flag is "N" then PHYS 221 is removed and PHYS 115 is kept because the number of fits for PHYS 221 is higher than the number of fits for PHYS 115. If the DAP14 Check the CFG020 TIEBREAK first flag is "Y" then PHYS 115 is removed and PHYS 221 is kept because PHYS 221 has term Fall 1996, which is more recent than Fall 1995 for PHYS 115.

The Check the CFG020 TIEBREAK first parameter is used only in evaluation of the starting block qualifiers. For the rest of the audit, the match level (exact match vs. range or wildcard) and the

number of fits are evaluated before the TIEBREAK config to get the best spread of classes across the rules, avoiding Fallthrough whenever possible.

UCX-CFG020 DAP15ADVICE

Updated: March 25, 2022

The DAP15ADVICE record of UCX-CFG020 supplies the text the auditor displays when rule advice has limitations involving MINPERDISC, MINSPREAD, or SAMEDISC.

UCX KEY	Len	Description	
DAP15ADVICE	11	DAP15ADVICE	
Reserved	19	Reserved for future use.	
UCX VALUE	Len	Start	Description
Advice Text	70	1	Free-text string that prints on audit output when rule advice has limitations by discipline.
Reserved	99	71	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 DAP15LABEL

Updated: March 25, 2022

The DAP15LABEL record of UCX-CFG020 supplies the text the auditor displays when a requirement label is missing.

UCX KEY	Len	Description	
DAP15LABEL	10	DAP15LABEL	
Reserved	20	Reserved for future use.	
UCX VALUE	Len	Start	Description
Default Label	50	1	Free-text string that prints on audit output if requirement label is missing.
Reserved	119	51	Reserved for future use.

UCX VALUE	Len	Start	Description
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 DAP15LIST

Updated: March 25, 2022

The DAP15LIST record of UCX-CFG020 supplies the text the auditor displays when a course list is not qualified with a number of classes or credits.

UCX KEY	Len	Description	
DAP15LIST	9	DAP15LIST	
Reserved	21	Reserved for future use.	
UCX KEY	Len	Start	Description
Course rule text	50	1	Free-text string that prints on audit output if a course list is not qualified by classes or credits.
Reserved	119	51	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 EXCEPTIONS

Updated: March 25, 2022

The EXCEPTIONS record of UCX-CFG020 controls how exceptions are brought into a degree audit.

UCX KEY	Len	Description
EXCEPTIONS	10	EXCEPTIONS

UCX KEY	Len	Description	
Reserved	20	Reserved for future use.	
UCX VALUE	Len	Start	Description
Exceptions Tied to Degree	1	1	Y/N. Default = N. Y = Exceptions are tied to degree. Degree Works will skip the exception if the degree being audited does not match the degree on the exception. N or blank = Degree is ignored when dealing with exceptions
Exceptions Tied to School	1	2	Y/N. Default = N. Y = Exceptions are tied to school. Degree Works will skip the exception if the school being audited does not match the school on the exception. N or blank = School is ignored when dealing with exceptions.
Reserved	167	3	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 RADBRIDGE

Updated: March 25, 2022

The RADBRIDGE record of UCX-CFG020 controls the behavior of the Bridge (RAD41) from the student system to the Degree Works Repository for Academic Data.

UCX KEY	Len	Description	
RADBRIDGE	10	RADBRIDGE	
Reserved	20	Reserved for future use.	
UCX VALUE	Len	Start	Description
Reserved	13	1	Reserved for future use.
DELETEID will delete from DAP	1	14	Y/N. Y = DELETEID transaction will delete records from the DAP and SEP tables. This includes notes, exceptions, transfer equivalency data, CPA data, plans, and audits. This flag is used only when bannerextract deleteids is run from the

UCX VALUE	Len	Start	Description
			command line OR when BIF records are sent with an action=D. Note: Even the frozen audits will be deleted.
SHP-USER-MST Modify Flag	1	15	F = Be FLEXIBLE with the shp_user_mst Modify flag. Just load what is bridged and do NOT pay attention to the shp_user_mst Modify flag. If the shp_user_mst exists then modify the data using the bridged data. If the shp_user_mst does NOT exist then add the record using the bridged data. I = Be INFLEXIBLE. Follow the Modify flag no matter what. If an "M" is bridged and the shp_user_mst does not exist then nothing will happen. A shp_user_mst record will NOT be added. If an "A" is bridged and a shp_user_mst already exists, then nothing will happen. The existing shp_user_mst will NOT be modified.
Add UCX Entries Only	1	16	Note: Refer to the chart below for details.
User Attributes School	1	17	A Y/N flag used to determine whether or not data should only be "Added" to the UCX and NOT "Deleted". Set to "Y" to ONLY ADD new records to the "UCX" table from the bridge file. Set to "N" if the existing table being processed should first be DELETED from the UCX and then loaded completely from the bridged data.
User Attributes Degree	1	18	Y=copy the degree from the rad_goal_dtl to the shp_user_attrib for use in SHPCFG.
User Attributes Catalog Year	1	19	Y=copy the catalog year from the rad_goal_dtl to the shp_user_attrib for use in SHPCFG.
User Attributes Student Level	1	20	Y=copy the student level from the rad_goalData_dtl to the shp_user_attrib for use in SHPCFG.
User Attributes Degree Source	1	21	Y=copy the degree source from the rad_goal_dtl to the shp_user_attrib for use in SHPCFG.

UCX VALUE	Len	Start	Description
User Attributes Active Term	1	22	Y=copy the active term from the rad_student_mst to the shp_user_attrib for use in SHPCFG.
User Attributes Major	1	23	Y=copy the major from the rad_goalData_dtl to the shp_user_attrib for use in SHPCFG.
User Attributes Program	1	24	Y=copy the program from the rad_goalData_dtl to the shp_user_attrib for use in SHPCFG.
User Attributes College	1	25	Y=copy the college from the rad_goalData_dtl to the shp_user_attrib for use in SHPCFG.
Reserved	144	26	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Flexible / Inflexible (F/I)	Modify Flag (A/M)	SHP Record Exists (Y/N)	Result
F	A	Y	Shepherd (SHP) record is modified with data in bridge R171SHPU record
F	A	N	Shepherd record is added with data in bridge R171SHPU record
F	M	Y	Shepherd record is modified with data in bridge R171SHPU record
F	M	N	Shepherd record is added with data in bridge R171SHPU record
I	A	Y	Error/Warning: Attempt to add a record that already exists. No Shepherd record is modified, nor is it added.
I	A	N	Shepherd record is added with data in bridge R171SHPU record
I	M	Y	Shepherd record is modified with data in bridge R171SHPU record
I	M	N	Error/Warning: Attempt to modify a record that does not exist. No Shepherd record is modified, nor is it added.

UCX-CFG020 REFRESH

Updated: March 25, 2022

The REFRESH record of UCX-CFG020 contains global configuration settings for Degree Works that are shared by many programs.

Note: Banner sites only.

UCX KEY	Len	Description	
REFRESH	7	REFRESH	
Reserved	23	Reserved for future use.	
UCX VALUE	Len	Start	Description
Dynamic Refresh	1	1	Y/N. Y = Dynamic data refresh from the student system is allowed. This capability is currently only available for Banner sites.
Refresh Timeout	4	2	Time in minutes before a new refresh is allowed. The number of minutes in a day is 1440. For example, if the Refresh Timeout is set to 120 minutes (0120) and the difference between the system's current time and the rad_primary_mst Bridge Date/Time is LESS THAN or EQUAL TO 120 minutes, then a Refresh will NOT be performed. If the time differential is 120.01 minutes then a Refresh may be performed depending on the settings below. "9999" = Never refresh. "0000" = Always refresh.
Run Audit Refresh	1	6	Y/N. Y = Refresh the data from the student system before running an Audit.
What-If Refresh	1	7	Y/N. Y = Refresh the data from the student system before performing a What-If audit.
Look Ahead Refresh	1	8	Y/N. Y = Refresh the data from the student system before performing a Look Ahead audit.
Planner Audit Refresh	1	9	Y/N. Y = Refresh the data from the student system before performing a Planner Audit
Exception Audit Refresh	1	10	Y/N. Y = Refresh the data from the student system before performing an Exception Audit.
View Audit Refresh	1	11	Y/N. Y = Refresh the data from the student system. If there are data changes, then a

UCX VALUE	Len	Start	Description
			new audit will be run instead of just doing a view audit.
Requisite Audit Refresh	1	12	Y/N. Y = Refresh the data from Banner before checking the performing the requisite check. This is also used by the Planner when a plan is being saved and prerequisites are being checked.
Reserved	157	13	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 RESULTS

Updated: March 25, 2022

UCX-CFG020RESULTS defines which dap_result_dtl (CPA) result-types are created when building CPA data.

See [Advanced Reporting](#) for additional information.

UCX KEY	Len	Description
RESULTS	7	RESULTS
Reserved	23	Reserved for future use.

UCX VALUE	Len	Start	Description
Build Stuinfo	1	1	Build CATALOGYEAR and GOALDATA records
Build Classes Applied	1	2	Build list of classes applied to rules
Build Remarks	1	3	Build remarks for header and for rules
Build Fall-through section	1	4	Build fallback (electives) section
Build Over-The-Limit section	1	5	Build over-the-limit (not used) section
Build Insufficient section	1	6	Build insufficient (failed) section

UCX VALUE	Len	Start	Description
Build Audit Errors section	1	7	Build the audit errors section
Build Block Header CrCl Needed	1	8	Build the block header's credits/classes needed
CrCl Applied	1	9	Build the block header's credits/classes applied
Advice	1	10	Build the block header advice
Requirements	1	11	Build the block header qualifier requirements
Build Rule ProxyAdvice	1	12	Build rule proxy-advice
Block Rule Label	1	13	Build the block rule label and percent complete
Advice	1	14	Build the block rule advice
Blocktype Rule Label	1	15	Build the blocktype rule label and percent complete
Advice	1	16	Build the blocktype rule advice
Rule-Complete Label	1	17	Build the rule-complete label
Group Rule Label	1	18	Build the group rule label and percent complete
Advice	1	19	Build the group rule advice
Subset Rule label	1	20	Build the subset rule label and percent complete
Advice	1	21	Build the subset rule advice
Noncourse Rule Label	1	22	Build the noncourse rule label and percent complete
Advice	1	23	Build the noncourse rule advice
Course Rule Label	1	24	Build the course rule label, percent complete, and credits/classes applied
Advice	1	25	Build the course rule advice
CrCl Required	1	26	Build the course rule classes/credits required
Reserved	143	27	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.

UCX VALUE	Len	Start	Description
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 TIEBREAK

Updated: March 25, 2022

The Auditor Engine uses this record to break ties between two courses whose best fit is the same requirement but only one of the courses is needed for the requirement.

The factors used in this decision are assigned a position in this record. For each factor, the user assigns a priority and a Keep-Flag. The priority tells the order in which the Auditor will evaluate the factors until the tie is broken. If both courses have the same value for a factor then check the next factor in priority order. If none of the factors break the tie then the Auditor arbitrarily keeps the first course on the rule and removes the other course. The Keep-Flag indicates which of the courses to apply to the rule, for example, how to break the tie.

The factors and possible keep-flag values are:

Grade (numeric)	:	H/L	H=keep the highest grade.
Term	:	H/L	H=keep the highest (most recent).
DAP Credits	:	H/L	H=keep the highest credits.
Course-Number	:	H/L	H=keep the highest course number.
Pass-Fail	:	Y/N	Y=keep the Pass-Fail course.
Transfer	:	Y/N	Y=keep the Transfer course.
Do-Over	:	Y/N	Y=keep the Do-Over course.
In-Progress	:	Y/N	Y=keep the In-Progress course.
Incomplete	:	Y/N	Y=keep the Incomplete course.

Each of the above factors is assigned a unique priority from one to nine. A priority number cannot be duplicated on multiple factors.

For example, if MATH 100 and MATH 105 both fit the same requirement but only one course is needed, then the Auditor uses the TIEBREAK record to guide the selection of which course to apply to the requirement.

If TIEBREAK indicates grade has priority “01” and Keep-Flag “H”, then the course with the highest grade is applied to the requirement. If both courses have the same grade, then the Auditor examines the factor with priority “02”. If term is “02” and its Keep-Flag is “H” then the course with the highest term (most recent) is kept on the requirement. Evaluation continues through all the factors, in priority order, until the tie is broken. If the TIEBREAK factors fail to break the tie, then the Auditor will keep the first course in its internal list.

UCX KEY	Len	Description
TIEBREAK	8	TIEBREAK
Reserved	22	Reserved for future use.

UCX VALUE	Len	Start	Description
Grade Rank	2	1	Priority of the Grade factor, "01" – "09". Default = "01".
Grade Keep	1	3	Keep-Flag of the Grade factor. H = keep the course with the highest grade. L = keep the course with the lowest grade. Default = "H".
Term Rank	2	4	Priority of the Term factor, "01" – "09". Default = "02".
Term Keep	1	6	Keep-Flag of the Term factor. H = keep the course with the most recent term. L = keep the course with the oldest term. Default = "H".
Credits Rank	2	7	Priority of the Credits factor, "01" – "09". Default = "03".
Credits Keep	1	9	Keep-Flag of the Credits factor. H = keep the course with the highest credits. L = keep the course with the lowest credits. Default = "H".
Course Number Rank	2	10	Priority of the Course Number factor, "01" – "09". Default = "04".
Course Number Keep	1	12	Keep-Flag of the Course Number factor. H = keep the course with the highest course number. L = keep the course with the lowest course number. Default = "H".
Passfail Rank	2	13	Priority of the Passfail factor, "01" – "09". Default = "05".
Passfail Keep	1	15	Keep-Flag of the Passfail factor. Y = keep the Passfail course. N = do not keep the Passfail course. Default = "N".
Transfer Rank	2	16	Priority of the Transfer factor, "01" – "09". Default = "06".
Transfer Keep	1	18	Keep-Flag of the Transfer factor. Y = keep the Transfer course. N = do not keep the Transfer course. Default = "N".
Repeat Rank	2	19	Priority of the Repeat factor, "01" – "09". Default = "07".
Repeat Keep	1	21	Keep-Flag of the Repeat factor. Y = keep the repeated (do-over) course. N = do not keep the repeated (do-over) course. Default = "N".
Incomplete Rank	2	22	Priority of the Incomplete factor, "01" – "09". Default = "08".
Incomplete Keep	1	24	Keep-Flag of the Incomplete factor. Y = keep the Incomplete course. N = do not keep the Incomplete course. Default = "N".

UCX VALUE	Len	Start	Description
In-progress Rank	2	25	Priority of the In-Progress factor, "01" – "09". Default = "09".
In-progress Keep	1	27	Keep-Flag of the In-Progress factor. Y = keep the In-Progress course. N = do not keep the In-Progress course. Default = "N".
Reserved	142	28	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.
Grade Rank	2	1	Priority of the Grade factor, "01" – "09". Default = "01".
Grade Keep	1	3	Keep-Flag of the Grade factor. H = keep the course with the highest grade. L = keep the course with the lowest grade. Default = "H".
Term Rank	2	4	Priority of the Term factor, "01" – "09". Default = "02".
Term Keep	1	6	Keep-Flag of the Term factor. H = keep the course with the most recent term. L = keep the course with the oldest term. Default = "H".
Credits Rank	2	7	Priority of the Credits factor, "01" – "09". Default = "03".
Credits Keep	1	9	Keep-Flag of the Credits factor. H = keep the course with the highest credits. L = keep the course with the lowest credits. Default = "H".
Course Number Rank	2	10	Priority of the Course Number factor, "01" – "09". Default = "04".
Course Number Keep	1	12	Keep-Flag of the Course Number factor. H = keep the course with the highest course number. L = keep the course with the lowest course number. Default = "H".
Passfail Rank	2	13	Priority of the Passfail factor, "01" – "09". Default = "05".
Passfail Keep	1	15	Keep-Flag of the Passfail factor. Y = keep the Passfail course. N = do not keep the Passfail course. Default = "N".
Transfer Rank	2	16	Priority of the Transfer factor, "01" – "09". Default = "06".

UCX VALUE	Len	Start	Description
Transfer Keep	1	18	Keep-Flag of the Transfer factor. Y = keep the Transfer course. N = do not keep the Transfer course. Default = "N".
Repeat Rank	2	19	Priority of the Repeat factor, "01" – "09". Default = "07".
Repeat Keep	1	21	Keep-Flag of the Repeat factor. Y = keep the repeated (do-over) course. N = do not keep the repeated (do-over) course. Default = "N".
Incomplete Rank	2	22	Priority of the Incomplete factor, "01" – "09". Default = "08".
Incomplete Keep	1	24	Keep-Flag of the Incomplete factor. Y = keep the Incomplete course. N = do not keep the Incomplete course. Default = "N".
In-progress Rank	2	25	Priority of the In-Progress factor, "01" – "09". Default = "09".
In-progress Keep	1	27	Keep-Flag of the In-Progress factor. Y = keep the In-Progress course. N = do not keep the In-Progress course. Default = "N".
Reserved	142	28	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 TRANSFER

Updated: March 25, 2022

The TRANSFER record controls how the software processes transfer courses from the student system. It indicates if the transfer record contains grading and class status information, and optionally allows transfer courses to be evaluated as repeats.

UCX KEY	Len	Description
TRANSFER	8	TRANSFER
Reserved	22	Reserved for future use.

UCX VALUE	Len	Start	Description
Reserved	14	1	Reserved for future use.

UCX VALUE	Len	Start	Description
Use TreQ	1	15	Y/N/B. Default = N. Y = Degree Works will use the DAP transfer data entered through Transfer Equivalency. N = Degree Works will use the transfer data sent from the student system (RAD). B = Degree Works will look for the transfer class data in the RAD system first. If NO RAD transfer data is found then it will look for DAP transfer data entered through Transfer Equivalency.
Reserved	154	16	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 TREQ

Updated: March 25, 2022

Transfer Equivalency and Degree Works subroutines use this record to control how transfer records are created in the DAP database.

Other settings tell Degree Works how a transfer school is identified on the system. This record should be examined and changed by each institution.

UCX KEY	Len	Description
TREQ	4	TREQ
Reserved	26	Reserved for future use.

UCX VALUE	Len	Start	Description
School Key	1	1	Default = E. Which field on the rad_ets_mst is the SCHOOL-ID: E = rad_ets_code D = rad_ets_doe

UCX VALUE	Len	Start	Description
			F = rad_ets_fice U = rad_ets_user
School Elem	4	2	The UCX-SYS999 element that tells Degree Works where to find the School-ID on the rad_ets_mst. For example: 0086 = rad_ets_code 2590 = rad_ets_doe 2591 = rad_ets_fice 2592 = rad_ets_user 0088 = rad_ets_tax_id
Default Calendar	2	6	Default calendar code from UCX-STU346 (quarter, semester, credit hours, etc.) This value is used by Transfer Equivalency Self-Service and Transfer Equivalency Admin if the school selected does not have a default calendar selected.
Default Credit Type	2	8	Default credit-type from UCX-STU355 for transfer classes stored in transfer-dtl. This value is used by Transfer Equivalency Admin.
Reserved	3	10	Reserved for future use.
Default Grade Type	2	13	Default Grade Type from UCX-STU356 for transfer classes stored in the rad_transfer_dtl (A-F, Pass-fail, etc). This value is used by Transfer Equivalency Admin for the articulation process.
Default Transfer Grade	4	15	If the Transfer Grade Required flag listed below is N and the user leaves the grade blank this grade should be used as the default transfer grade.
Default School	12	19	Default School code from UCX-STU350.
Reserved	12	31	Reserved for future use.
Grade Assign Mode	1	43	Default=H. How should the grade be assigned on a many-to-one or many-to-many mapping: L=use lowest grade from transfer classes; H=use highest grade from transfer classes;

UCX VALUE	Len	Start	Description
			P=use many-passfail-grade when assigning the grade.
			This value is used by Transfer Equivalency Admin and Transfer Equivalency Self-Service for the articulation process.
Many Pass Grade Type	2	44	Default=PF. The default passfail grade type to use when the MANY grade assignment flag is P.
Many Pass Grade	4	46	Default=PA. The default passfail grade to be used when the MANY grade assignment flag is P. This value is used by Transfer Equivalency Admin and Transfer Equivalency Self-Service for the articulation process.
Reserved	3	50	Reserved for future use.
Tests ID	10	53	School-ID for test scores saved in the dap_transfer_dtl. Default = "TESTS". This value is used by Transfer Equivalency Admin and Transfer Equivalency Self-Service for the articulation process.
Reserved	2	63	Reserved for future use.
Test Credit Type	2	65	The credit type to use on classes mapped from test scores.
Test Grade Type	2	67	Default Grade-Type from UCX-STU356 for classes mapped from test scores.
Test Grade	4	69	The grade to use on classes mapped from test scores.
Test Indicator	2	73	Will be set to "NM" – no match if a TEST-DTL does NOT match a dap_transfer_dtl record. Will be set to "RO" if the TEST-DTL is rolled. Will be set to "FR" if the records rolled from the dap_transfer_dtl are frozen.
Test Zero-fill	1	75	Default = N. If set to Y, then the TEST-SCORE will be right-justified and zero-filled if rolled.
Transfer Grade Required	1	76	Default=N. Should Transfer Equivalency require a transfer grade when the transfer transcript is being entered? Y/N
Upshift Transfer Discipline	1	77	Default=N. Upshift Transfer Discipline flag. Y/N.
Upshift Home Discipline	1	78	Default=N. Upshift Discipline (home school). Y/N.

UCX VALUE	Len	Start	Description
Use Section	1	79	When defining mappings in Transfer Equivalency should Section appear as a field for the home school class? Y/N
Show Transfer Grade Points	1	80	Y/N. Default = Y. Y = Transfer Equivalency will show Transfer Grade Points. N = Transfer Equivalency will not show Transfer Grade Points.
Course Mst Data	1	81	Y/N. Default = Y. Y = Transfer Equivalency will default the following fields on the dap_transfer_dtl from the rad_course_mst: Acad-Votech, Class-Type, Division, and Department. N = Transfer Equivalency will default these fields to blanks.
Default Active Term	8	82	When loading transcript data through RADBRIDGE, the term on the RAD-APPLICNT-DTL and RAD-STUDENT-MST are defaulted to this value if term is blank on these records. This occurs only when adding a new RAD-APPLICNT-DTL or RAD-STUDENT-MST. Must be valid in UCX-STU016.
School Type	1	90	Load Dap-Transfer-Dtl User-Def1 with ETS Type field. Should only be set to Y if Scribed blocks use WITH operator on SchoolType value.
Reserved	10	91	Reserved for future use.
Default Degree	12	101	When loading transcript data through RADBRIDGE and adding a new RAD-APPLICNT-DTL, the degree code on the RAD-APPLICNT-DTL is defaulted to this value if degree is blank. When loading a student for the first time into Transfer Equivalency that does not have a RAD-APPLICNT-DTL we also used this degree as the default when creating the dap-applicnt-mst. Must be valid in UCX-STU307.
Reserved	57	113	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.

UCX VALUE	Len	Start	Description
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG020 WEB

Updated: March 25, 2022

The Degree Works web interface uses this record to control different aspects of the software, such as the GPA Calculator, Exception Management, displaying the audit, and others.

UCX KEY	Len	Description
WEB	3	WEB
Reserved	27	Reserved for future use.
UCX VALUE	Len	Start
GPA Calculator classes per term	2	1
		Not used by Responsive Dashboard.
GPA Calculator grade method	1	3
		Not used by Responsive Dashboard.
Show notes student checkbox	1	4
		Not used by Responsive Dashboard.
		For the Responsive Dashboard see dash.notes.internal.enabled
Show in-progress checkbox	1	5
		Not used by Responsive Dashboard.
		For the Responsive Dashboard see dash.audit.process.inprogress.enabled
In-progress default	1	6
		Not used by Responsive Dashboard.
		For the Responsive Dashboard see dash.audit.process.inprogress.defaultValue
Show pre-registered checkbox	1	7
		Not used by Responsive Dashboard.
		For the Responsive Dashboard see dash.audit.process.preregistered.enabled

UCX VALUE	Len	Start	Description
Pre-registered default	1	8	Not used by Responsive Dashboard. For the Responsive Dashboard see dash.audit.process.preregistered.defaultValue
Audit title style	1	9	Not used by Responsive Dashboard.
Run new audit if no audit exists	1	10	If a user clicks "View Audit" on the web, but no audit exists, "Y" = run new audit and return to screen; "N" = show error message that no audit exists.
View last audit in Exceptions	1	11	Not used by Responsive Dashboard. For the Responsive Dashboard see dash.exceptions.runAuditOnEnter
Save Exceptions audit	1	12	Not used by Responsive Dashboard. For the Responsive Dashboard see dash.exceptions.saveNewAudit
Allow searching on all exceptions	1	13	Not used by Responsive Dashboard.
Use CFG071 predefined notes	1	14	Not used by Responsive Dashboard. For the Responsive Dashboard see dash.notes.predefined.enabled
Mask student ID	1	15	Not used by Responsive Dashboard. For the Responsive Dashboard see dash.search.studentId.mask
Header frame height	3	16	Not used by Responsive Dashboard.
GPA Calculator decimals	1	19	Not used by Responsive Dashboard.

UCX VALUE	Len	Start	Description
			For the Responsive Dashboard see dash.gpaCalculator.decimals
GPA Calculator round	1	20	Not used by Responsive Dashboard.
			For the Responsive Dashboard see dash.gpaCalculator.round
Date format	3	21	Not used by Responsive Dashboard.
Reserved	2	24	Reserved for future use.
Allow audit freezing	1	26	Not used by Responsive Dashboard. In the Responsive Dashboard users with the key can freeze audits.
Allow audit description	1	27	Not used by Responsive Dashboard. In the Responsive Dashboard users with the key can enter descriptions.
Add a note from any tab	1	28	Not used by Responsive Dashboard.
Allow input of exception details	1	29	Not used by Responsive Dashboard. For the Responsive Dashboard see dash.exceptions.details.option
Notes tied to degree	1	30	Y=Notes are tied to the student's degree.
Notes tied to school	1	31	Y=Notes are tied to the student's school/level.
Reserved	138	32	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Audit title style settings

Style	Description and example
A	Ellucian Degree Works audit
B	Degree Works: [Report Type] audit Example: Degree Works Detailed Advice audit
C	[School Name:] [Report Type] audit Example: Generic College: Detailed Advice audit
D	[Report Type] audit for [Student Name] Example: Detailed Advice audit for Thomas Long
E	[Report Type] audit for [Student Name] – [Student ID] Example: Detailed Advice audit for Thomas Long – 123456789
F	[School Name:] [Report Type] audit for [Student Name] Example: Generic College: Detailed Advice audit for Thomas Long
G	[School Name:] [Report Type] audit for [Student Name] [Student ID] Example: Generic College: Detailed Advice audit for Thomas Long 123456789

UCX-CFG020 WEBPARAMS

UCX KEY	Len	Description
WEBPARAMS	9	WEBPARAMS
Reserved	21	Reserved for future use.

UCX VALUE	Len	Start	Description
Create Logs	1	11	A Y / N flag controlling the creation of the shp_log_dtl entries for every request to Degree Works. If this byte is "Y" then the logs will be created; otherwise they will not be created. See below for more information on the data stored on the shp_log_dtl.
Encrypt Password	1	12	A Y / N flag indicating if access codes are being encrypted in the shp_user_mst. This flag should be set before any shp_user_mst records are created and should not be changed unless all user-mst records are to be recreated.
			A "Y" indicates that the shp_user_mst is storing the access code in an encrypted fashion. Any authentication that takes places must encrypt the user's access code before comparing it to what is in the database. The encryption method is one-way - there is no way to decrypt the code stored in the database.
Term used for filtering in student search	8	65	The term to be used for performing a student search. This is the oldest current, valid term. Students with an active term on the rad_student_mst older than this term will be filtered out from the search results. This is useful for excluding students who have graduated or who have dropped out. Set this field to blanks if you do not want term filtering to be performed. This term is ignored when performing an ID search so that you may specifically call up students who are no longer active.
Changed Password	1	93	A Y/N flag. If set to 'N' then the shp_access_code will NOT be updated by the shpsave routine which is called by the extract programs (RAD11JOB for non-Banner sites and RAD30JOB for Banner sites). If this flag is 'Y' or BLANK then the shp_access_code WILL be updated if data has changed for the individual (student, advisor or staff member) and one of the extract programs is run on that individual.

UCX-CFG068 Course Sequencing

Updated: March 25, 2022

UCX-CFG068 contains the Course Keys (Discipline + Course Numbers) that define a course sequence.

The key into this table is the end-point, or last course in the sequence, with the value of this record specifying the other courses in the sequence.

For example, Spanish 101, 102 and 103 is a language sequence that must be taken in that order. While prerequisite checking within registration will ensure that the student has completed 101 before taking 102 and has completed 102 before taking 103 there is usually nothing in the registration process that forbids a student from testing into Spanish 103 and then later registering for and taking Spanish 101 or 102 for easy credit. It is this latter scenario that UCX-CFG068 defines and what the auditor forbids based on the sequences defined.

The Course discipline 1 and Course number 1 would be filled in with SPAN 101. The Course discipline 2 and Course number 2 would be filled in with SPAN 102 with the key to this record being "SPAN103".

When an audit is run and a UCX-CFG068 record is found for SPAN 103 the auditor checks to see if the student took (or is taking) any of the seven classes on this record in a term after SPAN 103 was taken. Any class found that was taken after SPAN 103 will be placed in the over-the-limit section and will not apply towards degree requirements.

UCX KEY	Len	Description
Course Key (no spaces)	24	The Course Key with spaces removed between the Discipline and Course (e.g., "STAT123", "SPAN103", "MATH129").
Reserved	6	Reserved for future use.

UCX VALUE	Len	Start	Description
Course discipline 1	12	1	Discipline of 1st course that cannot be taken after the course in the key.
Course number 1	12	13	Number of 1st course that cannot be taken after the course in the key.
Course discipline 2	12	25	Discipline of 2nd course that cannot be taken after the course in the key.
Course number 2	12	37	Number of the 2nd course that cannot be taken after the course in the key.
Course discipline 3	12	49	<similar to the above>
Course number 3	12	61	<similar to the above>
Course discipline 4	12	73	<similar to the above>

UCX VALUE	Len	Start	Description
Course number 4	12	85	<similar to the above>
Course discipline 5	12	97	<similar to the above>
Course number 5	12	109	<similar to the above>
Course discipline 6	12	121	<similar to the above>
Course number 6	12	133	<similar to the above>
Course discipline 7	12	145	<similar to the above>
Course number 7	12	157	<similar to the above>
Reserved	1	169	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Turn on/off course sequencing check

There is no UCX-CFG020 flag to turn on/off Course Sequencing. If records exist here in UCX-CFG068 they will be used. If there are no records in UCX-CFG068 the auditor will not attempt to lookup courses here to check for course sequencing. To turn off sequencing you need to empty out this table.

Failed classes

If SPAN 103 has a failing grade no lookup on UCX-CFG068 is performed. It is expected that the student will take SPAN 101 or SPAN 102 after failing SPAN 103.

If any of the courses listed on the record, such as SPAN 101 and 102, that should not have been taken after SPAN 103 have a failing grade they will not be placed into over-the-limit. These failed classes will instead be placed in insufficient and will apply to the major or minor block if they could have applied. Essentially, the course sequencing rule is ignored if either the master or subordinate class is failed.

GPS and residency

Any classes that are placed in the over-the-limit section because of a course sequencing check will still count in the overall GPA calculation and also in any residency (LastRes) check. These classes will not count in the overall degree credits/classes or against any other qualifier in the degree block.

Renamed classes

If the student took SPAN 103 but it was then renamed to SPAN 103 the UCX-CFG068 record needs to have the SPAN 103 name listed. Degree Works knows SPN 103 has been renamed to SPAN 103 because of the equivalence records stored in Degree Works. The same is true for any of the courses listed on the record, like SPAN 102 for example. If the student took SPN 102 and it was renamed to SPAN 102 and an equivalence record exists the new name will be used for comparison. Therefore, the UCX-CFG068 records need to contain the new names of your courses if the associated equivalence records exist in Degree Works.

Exceptions

If SPAN 101 is moved to over-the-limit because SPAN 103 was taken previously you cannot add an exception to allow SPAN 101 to apply to a requirement. If you think you may need to allow SPAN 101 to apply for certain students then using UCX-CFG068 will prevent you from doing that using an exception.

UCX-CFG070 Course Equivalence Records

Updated: March 25, 2022

UCX-CFG070 is used by Degree Works to equate course keys which have been changed or obsoleted to current or new course keys. It can be maintained manually in Controller.

Records entered in UCX-CFG070 will not be available for use in Degree Works until the records are loaded into the dap_eqv_crs_mst table by running dapucx2eqv under ADMIN in Transit. It is important to consider the implications for your old scribe blocks when making global changes using the UCX-CFG070 table.

RAD clients can update the dap_eqv_crs_mst table directly by using the RAD11 bridge.

UCX KEY	Len	Description	
Record number	30	Unique identifier. Can be discipline/number (MATH101) or can be any unique number	
UCX VALUE	Len	Start	Description
Catalog Year class was taken	12	1	Catalog year the class was taken (Must be valid in UCX-STU035 or can be a wildcard [@])
Old Course Discipline	12	13	Old course discipline.
Number	12	25	Old course number. Can be wildcard (@) if discipline changed but number did not.
Student's Catalog Year	12	37	Student's catalog year (Must be valid in UCX-STU035 or can be a wildcard [@])
New Course Discipline	12	49	New course discipline code

UCX VALUE	Len	Start	Description
Number	12	61	New course number. Can be wildcard (@) if discipline changed but number did not.
Note	50	73	50 byte free text note.
Reserved	47	123	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG071 Note Text

Updated: March 25, 2022

UCX-CFG071 contains predefined note text which is available for use in the Notes feature.. This works in conjunction with the UCX-CFG020 WEB “Show Notes Picklist” flag. It must be set to “Y” for this picklist to show.

To add a new record, copy an existing record and increment the key by one. The picklist will sort by the key.

UCX KEY	Len	Description
Sequential Number	3	Sequential number that determines the order of the predefined notes in the picklist. It sorts ascending.
Reserved	27	Reserved for future use.

UCX VALUE	Len	Start	Description
Note text	70	1	Text of the note
cont'd	70	71	More note text.
Reserved	29	141	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG072 Planned Courses

Updated: September 30, 2022

UCX-CFG072 defines the term(s) a course is to be offered during future terms. This is used in the Student Educational Planner. You can also specify terms that courses will not be offered.

This functionality is available in the Plans tab in Responsive Dashboard, but is not available in the Template Management functionality in Responsive Dashboard.

UCX KEY	Len	Description	
Course Key	24	Course Key (without a space, for example, ART101, MATH211, ENGR344, PE125)	
Reserved	6	Reserved for future use.	
UCX VALUE	Len	Start	Description
Term 1	8	1	Term Code. See below for allowable characters.
Term 2	8	9	Term Code. See below for allowable characters.
Term 3	8	17	Term Code. See below for allowable characters.
Term 4	8	25	Term Code. See below for allowable characters.
Term 5	8	33	Term Code. See below for allowable characters.
Term 6	8	41	Term Code. See below for allowable characters.
Term 7	8	49	Term Code. See below for allowable characters.
Term 8	8	57	Term Code. See below for allowable characters.
Term 9	8	65	Term Code. See below for allowable characters.
Term 10	8	73	Term Code. See below for allowable characters.
Term 11	8	81	Term Code. See below for allowable characters.
Term 12	8	89	Term Code. See below for allowable characters.
Term 13	8	97	Term Code. See below for allowable characters.
Term 14	8	105	Term Code. See below for allowable characters.

UCX VALUE	Len	Start	Description
Term 15	8	113	Term Code. See below for allowable characters.
Term 16	8	121	Term Code. See below for allowable characters.
Message to User	40	129	A user-friendly message that will display to users if they attempt add a course requirement to a term in which it is not offered.
Reserved	1	169	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Term code examples

@	Any term
@03	Any term ending in 03
2008@	Any term starting with 2008
200803	This specific term
!@03	Any term except those ending in 03
!200803	Any term except this specific term

UCX-CFG073 Cross Listed Courses

Updated: March 25, 2022

UCX-CFG073 contains the Course Keys (Discipline + Course Numbers) that are considered cross-listed courses. These courses are loaded into UCX-CFG073 in two different ways.

Banner sites

Cross-listed courses may be extracted into Banner using the RAD38 -Banner Equivalencies Extract. If the UCX-CFG020BANNER “Cross List in SCREQIV” flag is ‘Y’ the Equivalency extract (BAN43) performs the following edits to determine if the SCREQIV OLD Course is Cross-listed:

- The SCREQIV_END_TERM = ‘999999’ or if integration.banner.extract.equiv.crosslistedRange is true
- The OLD SCREQIV_SUBJ_CODE_EQIV and SCREQIV_CRSE_NUMB_EQIV are used to look up the associated SCBCRKY record. The SCBCRKY_TERM_CODE_END must = ‘999999’

- The NEW SCREQIV_SUBJ_CODE and SCREQIV_CRSE_NUMB are used to look up the associated SCBCRKY record. The SCBCRKY_TERM_CODE_END must = '999999'

Courses that satisfy all three requirements are considered cross-listed. If the SCREQIV record is cross-listed the new SCREQIV_SUBJ_CODE and SCREQIV_CRSE_NUM (subject and course number in the key block of SCADET) are loaded in the UCX-KEY while the SCREQIV_SUBJ_CODE_EQIV and SCREQIV_CRSE_NUMB_EQIV are loaded into the UCX-VALUE and written to this UCX-CFG073 table for use later by the parser and auditor.

Non-Banner (RAD) sites

Cross-listed courses may be manually loaded into this table using Controller, if any such courses exist.

After making changes to UCX-CFG073 you must reparse your blocks to have them take effect. Each cross-listed record is processed by the parser and the rules are altered on the syntax tree – you will not see these changes in Scribe. The Degree Works auditor will use the information in UCX-CFG073 when trying to satisfy requirements for a given student.

UCX KEY	Len	Description
Course Key (no spaces)	27	The cross-listed Course Key with ALL spaces removed between the Discipline and Course Number followed by ONE SPACE and a 2-digit sequence number that must start with "01" (e.g., "STAT123 01", "STAT123 02", "STAT123 03", "PE101 01", "ANTHRO304 01").
Reserved	3	Reserved for future use.

UCX VALUE	Len	Start	Description
Cross Listed Course Discipline	12	1	Cross-listed Discipline (SubjCode for Banner sites).
Number	12	13	Cross-listed Course Number (CrseNumb for Banner sites).
WITH DWTerm Operator	2	25	Optional; >, <, <=, >=, = or <> (usually set to >=). This can also be RS (range start) and RE (range end).
Cross-listed Term	8	27	Optional; specify the term for this cross-listing (length is 6 for Banner sites).
Reserved	135	35	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG074 Reused Courses

Updated: March 25, 2022

UCX-CFG074 contains Course Keys that have been reused over the years.

This table is used for two different purposes:

- Banner Equivalent Extract (ban43): If a “Reversal” is found where a NEW Course maps back to an OLD course normally this SCREQIV record would be “skipped” because this is NOT an equivalent as far as Degree Works is concerned. However, if a “Reversal” is found the Course Key is looked up in UCX-CFG074. If a match is found then the “Reversal” is NOT skipped. Instead processing continues and equivalent records for the Start/End catalog year range are created.
- The parser when altering the scribed courses to match what you have in equivalent records.

UCX KEY	Len	Description
Course Key	24	The Discipline + Course Number with ALL spaces removed (for example, “ART 101 “ ® “ART101”).
Reserved	6	Reserved for future use.

UCX VALUE	Len	Start	Description
Note	50	1	Description
Reserved	119	51	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG075 Repeatables

Updated: March 25, 2022

UCX-CFG075 contains Course Keys defining how a class can be repeated for credit.

This table is used when repeat policy 7 is in use. Classes listed here are allowed to be repeated for credit up to the maximum number of classes or credits specified. For more information, see the [Repeated classes](#) topic.

Classes that are deemed to be over the repeat limit will appear in the over-the-limit (Not Counted) section of the worksheet but are considered insufficient and will be treated as such for the normal GPA calculations performed by the auditor.

UCX KEY	Len	Description
Course Key	24	The Discipline + Course Number with ALL spaces removed (for example, "ART 101" ® "ART101"). Wildcards may be used so that you don't have to list every course. For example, if all 100-level ART classes can be repeated as many times as desired you can create an "ART@{@}" record with a Limit of "99". Each @ character matches against one character in the course key.
Sequence number	6	Optional – needed if more than one record is needed to define different term ranges. Add a space after the course key before putting a sequence number (for example, "ART101 1" and "ART101 2").

UCX VALUE	Len	Start	Description
Begin Term	8	1	Start of term range.
End Term	8	9	End of term range.
Limit	2	17	00, 99, blanks = unlimited. 01-98 define limit of classes or credits/hours. 01 means the classes can be taken one time and never repeated. 02 means the class can be taken one time and repeated one time. And so on...
Limit Type	1	19	C=Class limit; H=Hour/Credit limit
Reserved	119	20	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CFG078 Split Courses

Updated: March 25, 2022

UCX-CFG078 contains the Course Keys (Discipline + Course Numbers) that were split into two or more courses. Because split courses cannot be handled in the equivalence table this table is used to tell Degree Works how a course was split.

For example, MATH 101 was a 5 credit Algebra class but in 2017 it was split into MATH 112 (2 credits) and MATH 113 (3 credits). The key into this table would then be "MATH101". The **From Catalog Year** and **To Catalog Year** would be filled in so that students within this range will have their class split; student's whose catalog year is not within this range will not have their class split.

Note, when the student took the course is irrelevant – it is the student's catalog year that matters as it determines what requirements are scribed.

The **Course discipline 1** and **Course number 1** would be filled in with MATH 112. The **Course discipline 2** and **Course number 2** would be filled in with MATH 113.

When a UCX-CFG078 record is found and the student took the class within the specified term range the class will not be looked up in the equivalence table; this table is used in place of the equivalence table for the courses listed.

The credits given to each course listed come from the course-mst. When the sum of these credits is more than the credits earned by the student the credits are reduced to give the student exactly what they earned. Conversely, if the student earned more than the sum of the credits for the courses the first course listed is given more credits to properly indicate the credits earned by the student.

Given the MATH 101 example above, the auditor should apply MATH 101 to the MATH 112 rule and also to the MATH 113 rule. The worksheet will show "MATH 101" as the class that was taken but with the appropriate number of split credits.

UCX KEY	Len	Description
Course Key (no spaces)	24	The split Course Key with ALL spaces removed between the Discipline and Course (e.g., "STAT123", "PE101", "ANTHRO304").
Reserved	6	Reserved for future use.

UCX VALUE	Len	Start	Description
From Catalog Year	12	1	Starting student's catalog year when this class could be taken to be considered for the split.
To Catalog Year	12	13	Ending student's catalog year when this class could be taken to be considered for the split.
Course discipline 1	12	25	Discipline of 1st course that was a result of the split.
Course number 1	12	37	Number of 1st course that was a result of the split.
Course discipline 2	12	49	Discipline of 2nd course that was a result of the split.
Course number 2	12	61	Number of 2nd course that was a result of the split.
Course discipline 3	12	73	Discipline of 3rd course that was a result of the split.
Course number 3	12	85	Number of 3rd course that was a result of the split.
Course discipline 4	12	97	Discipline of 4th course that was a result of the split.

UCX VALUE	Len	Start	Description
Course number 4	12	109	Number of 4th course that was a result of the split.
Course discipline 5	12	121	Discipline of 5th course that was a result of the split.
Course number 5	12	133	Number of 5th course that was a result of the split.
Course discipline 6	12	145	Discipline of 6th course that was a result of the split.
Course number 6	12	157	Number of 6th course that was a result of the split.
Reserved	1	169	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-CST001 – CST010 Custom Tables

Updated: March 25, 2022

UCX-CST001 through UCX-CST010 are reserved for special customizations made for schools.

These tables can also be used to define custom data items for the Responsive Dashboard Advanced Search.

UCX KEY	Len	Description
Key	30	Key for this UCX table

UCX VALUE	Len	Start	Description
Description	50	1	Description of entry
Reserved	119	51	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-RPT036 Audit Report Formats

Updated: September 29, 2023

UCX-RPT036 defines a report format in Degree Works. These entries are used to configure various report characteristics.

Report formats prepared by Ellucian follow a naming convention of xxx##, where xxx is RPT, SEP, TRF, TRQ, or WEB; and ## is a two-digit number.

These flags are a convenient way to toggle between different report styles and output.

UCX KEY	Len	Description	
Audit Report	5	The 5-character Report code from UCX-RPT036.	
Reserved	25	Reserved for future use.	
UCX VALUE	Len	Start	Description
Title	30	1	Report Title. This value displays in the report picklist. For Transit, if you change this title you need to change the corresponding report titles in TransitJobMessages.properties in Controller.
XSL Stylesheet	20	31	Obsolete
Show Block Remarks	1	51	Show block remarks
Show Block Qualifiers	1	52	Show block qualifiers
Show Block Exceptions	1	53	Show block qualifier exceptions
Show Block Include List	1	54	Show block include lists in the starting block regardless of the Include Blocks flag below.
Show Block Advice	1	55	Show block advice.
Show Rule Remarks	1	56	Show rule remarks.
Show Rule Qualifiers	1	57	Show rule qualifiers.
Show Rule Exceptions	1	58	Show rule exceptions.
Show Rule Advice	1	59	Show rule advice.

UCX VALUE	Len	Start	Description
Show Requirement Text	1	60	Show requirement text.
Show Courses Applied	1	61	Show courses applied to rules.
Show Fallthrough (Electives)	1	62	Show fall-through (electives) section. Y = show a single fall-through section. N = do not show the fall-through section.
			E = Elective Credits Allowed (ECA) - show one section for the fall-through classes that are counted towards the overall degree credits, and show another section for those that were not counted in the overall credits. The UCX-CFG020 DAP14 Calculate Elective Credits Allowed flag must be enabled.
Insufficient (Failed)	1	63	Show insufficient (failed) section.
Over the Limit (Not Counted)	1	64	Show over-the-limit (not counted) section.
In-Progress	1	65	Show in-progress section.
Notes	1	66	Show notes section.
Exceptions	1	67	Show exceptions section.
Errors	1	68	Obsolete
Show Legend	1	69	Show legend.
Show Disclaimer	1	70	Show disclaimer.
Show Progress Bar	1	71	Obsolete
And/Or Advice	1	72	Obsolete
Show Prerequisite Indicator	1	73	Show/hide an indicator on the courses in the web advice to show that prerequisite conditions exist.
Show Course Link	1	74	Y = allow the user to click on course advice to link to get more information.
Show Course Keys Only	1	75	Y = show course key only (no title, and so on).

UCX VALUE	Len	Start	Description
			N = show course key, title, credits, grade, and term.
Show Student Header	1	76	Obsolete
Show Student Alerts	1	77	Obsolete
Show in Picklist	1	78	Make this report available in the web picklist.
Hide inner-group labels	1	79	The labels for the rules within the groups are hidden.
Hide Subset label	1	80	The subset label is hidden.
Reserved	1	81	Reserved for future use
Show Split Credits Section	1	82	Classes that were split over blocks show in this section.
Include blocks	1	83	Each block that uses block/blocktype/includeblocks will have a list of the blocks that were included. Also see Show Block Include List flag above.
Show Label	1	84	Show the label and completeness icon for each rule. Normally this is set to Y, but on the Registration Checklist, some schools set this to N.
Show SS GPA in header	1	85	Show the Student System GPA in the student header. If set to N, the Degree Works calculated GPA will display.
Course Link Title Order	2	86	The order (1-99) where Course Title information displays in Course Link. Do not duplicate order values between these Course Link entries. 0 means do not display.
Course Link Description Order	2	88	The order (1-99) where Course Description information displays in Course Link. Do not duplicate order values between these Course Link entries. 0 means do not display.
Course Link Co Prereq Order	2	90	The order (1-99) where Course Co-Prerequisite information displays in Course Link. Do not duplicate order values between these Course Link entries. 0 means do not display.
Course Link Attributes Order	2	92	The order (1-99) where Course Attribute information displays in Course Link. Do not duplicate order values between these

UCX VALUE	Len	Start	Description
			Course Link entries. 0 means do not display.
Course Link Sections Order	2	94	The order (1-99) where Course Section information displays in Course Link. Do not duplicate order values between these Course Link entries. 0 means do not display.
Reserved	2	96	Reserved for future use.
Course Link Transfer Order	2	98	The order (1-99) where Course Transfer information displays in Course Link. Do not duplicate order values between these Course Link entries. 0 means do not display.
Reserved	2	100	Reserved for future use.
FOP XSL Stylesheet	20	102	Obsolete
Show Progress Bar Requirements	1	122	Show the requirements progress bar.
Show Progress Bar Credits	1	123	Show the credits progress bar.
Show Petitions	1	124	In the notes section of the worksheet, also show petitions.
Show Overall GPA	1	125	Show the overall GPA in the Degree Progress section. The student system GPA will display; the Degree Works calculated GPA will not display.
Show Repeated	1	126	Show an indicator if a class was repeated. A class is considered repeated if its class status matches one of the values in the dash.audit.repeat.codes setting.
Degree Credits Option	1	127	T = show total credits applied to the degree in the starting block's header. E = show total credits applied to the degree in the starting block's header. This is the total credits minus the overflow fall-through credits. This flag has an effect only if the elective credits calculation is being performed, that is, if CFG020 DAP14 Calculate Elective Credits Allowed is Y or the

UCX VALUE	Len	Start	Description
			CheckElectiveCreditsAllowed qualifier is used.
			For more information, see the UCX-CFG020 DAP14 or Elective Credits Allowed (ECA) Calculations in topics.
Show Noncourses in Fall-through	1	128	Show unused noncourses in the fall-through section.
Show classes in fall-through header	1	129	Show classes in fall-through section header. Also applies to fall-through ECA sections.
Show credits in fall-through header	1	130	Show credits in fall-through section header. Also applies to fall-through ECA sections.
Show classes in insufficient header	1	131	Show classes in insufficient section header.
Show credits in insufficient header	1	132	Show credits in insufficient section header.
Show classes in over-the-limit header	1	133	Show classes in over-the-limit section header.
Show credits in over-the-limit header	1	134	Show credits in over-the-limit section header.
Show classes in in-progress header	1	135	Show classes in in-progress section header.
Show credits in in-progress header	1	136	Show credits in in-progress section header.
Show classes in split-credits header	1	137	Show classes in split-credits section header.
Show credits in split-credits header	1	138	Show credits in split-credits section header.
Reserved	31	139	Reserved for future use.

UCX VALUE	Len	Start	Description
Status	1	170	Y = Active N = Inactive W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-RPT046 Custom Report Data

Updated: March 25, 2022

UCX-RPT046 contains one record for each report data item from the student system to be used in a degree audit report.

See UCX-SCR002 for custom data items.

Institutions wishing to display data on the student header that is not provided with the standard Degree Works software may create Custom Report Data items by adding them to UCX-RPT046. Degree Works uses this table to ascertain which pieces of data from the student system should be sent to the auditor in addition to the standard academic and biographic data.

For example, the student's email address is not a piece of data that is typically part of a student's degree audit information, but it can be configured to display in the student header. To construct the requirement, EMAIL must be added as a Custom Report Data item to UCX-RPT046, with a Data Element = 2522.

For each entry in UCX-RPT046, the Description and Data Element must be entered in the value-area. The rest of the value-area is optional and is typically used only when retrieving data from a detail table (where more than one record can exist per student, for example, degree details or school details). These fields indicate which records from the table to use and which to skip. Only records that pass the edits are used as a source of Custom Report Data.

The Data Element and Edit Element 1, 2, and 3 fields must contain a valid UCX-SYS999 value.

Note: You do not need to add an entry in UCX-RPT046 for those items in the rad_report_dtl as they are handled automatically. Only add entries for values located in other tables.

UCX KEY	Len	Description	
Custom Report Data	12	The name of the Custom Report Data element to be used by Degree Works. For example: EMAIL, ADVISORNAME, RELIGION.	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	Free-text description.
Data Element	4	31	Element number from UCX-SYS999 indicating the report data source.
Edit Element 1	4	38	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the Report Element data should be extracted from this detail. This element number must be from the same table as the Report Data Element.
Edit Type 1	2	42	The type of check to do on the data in Edit Element 1. EV or blanks = use the value in Edit Value 1.
Edit Value 1	12	44	The specific data value against which the data in Edit Element 1 should be checked. The record is skipped if it does not match.
Edit Element 2	4	56	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the Report Element data should be extracted from this detail. This element number must be from the same table as the Report Data Element.
Edit Type 2	2	60	The type of check to do on the data in Edit Element 2. EV or blanks = use the value in Edit Value 2.
Edit Value 2	12	62	The specific data value against which the data in Edit Element 2 should be checked. The record is skipped if it does not match.
Edit Element 3	4	74	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the Report Element data should be extracted from this detail. This element number must be from the same table as the Report Data Element.
Edit Type 3	2	78	The type of check to do on the data in Edit Element 3. EV or blanks = use the value in Edit Value 3.
Edit Value 3	12	80	The specific data value against which the data in Edit Element 3 should be checked. The record is skipped if it does not match.

UCX VALUE	Len	Start	Description
Reserved	78	92	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-RPT050 Course Link Title Settings

Updated: March 24, 2023

UCX-RPT050 defines the settings for the Course Link Title feature of Course Link.

For more information, see [UCX-RPT036 Audit Report Formats](#).

UCX KEY	Len	Description	
Audit Report	5 or more	The 5-character Report code from UCX-RPT036.	
Reserved	25	Reserved for future use.	
UCX VALUE	Len	Start	Description
Course Title and Credits Source	1	1	Source of the Title and Credits information. B = Banner
Version	1	2	B = Brief - credits are hidden. S = Standard V = Verbose Standard and Verbose are equivalent.
Reserved	167	3	Reserved for future use.
Status	1	170	Y = Active N = Inactive W = issue Warning message for override.
Reserved	30	171	Reserved for future use.

UCX VALUE	Len	Start	Description
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-RPT052 Course Link Attribute Settings

Updated: March 24, 2023

UCX-RPT052 defines the settings for the Course Link Attribute feature of Course Link.

For more information, see [UCX-RPT036 Audit Report Formats](#).

UCX KEY	Len	Description
Audit Report	5 or more	The 5-character Report code from UCX-RPT036.
Reserved	25	Reserved for future use.

UCX VALUE	Len	Start	Description
Course Attributes Source	1	1	Source of the Course Description information. B = Banner
Version	1	2	B = Brief S = Standard V = Verbose - attribute code and description are shown; otherwise, only the attribute code is shown, but the description appears as hover text over the code.
Reserved	167	3	Reserved for future use.
Status	1	170	Y = Active N = Inactive W = issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.

UCX VALUE	Len	Start	Description
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-RPT054 Course Link Sections Settings

Updated: March 24, 2023

UCX-RPT054 defines the settings for the Course Link Sections feature of Course Link.

See also [UCX-RPT036 Audit Report Formats](#). This determines the ordering of all Course Link features.

UCX KEY	Len	Description
Audit Report	5 or more	The 5-character Report code from UCX-RPT036.
Reserved	25	Reserved for future use.

UCX VALUE	Len	Start	Description
Course Sections Source	1	1	Source of the Course Section information. B = Banner
Version	1	2	B = Brief S = Standard V = Verbose

In Verbose mode, the course title will show for each section. This is helpful when the course title can be different depending on the section.

Reserved	167	3	Reserved for future use.
Status	1	170	Y = Active N = Inactive
			W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-RPT056 Course Link Transfer Settings

Updated: March 24, 2023

UCX-RPT056 defines the settings for the Course Link Transfer feature of Course Link.

This feature is also referenced as Course Finder.

For more information, see [UCX-RPT036 Audit Report Formats](#).

UCX KEY	Len	Description	
Audit Report	5 or more	The 5-character Report code from UCX-RPT036.	
Reserved	25	Reserved for future use.	
UCX VALUE	Len	Start	Description
Transfer Mappings Source	1	1	Source of the Transfer Mapping information. D = Degree Works
Version	1	2	B = Brief - shows in a table with column headers. S = Standard - shows on one line. V = Verbose - shows with the school name, city, and state. The course titles appear and the credit and grade restrictions appear.
Show Favorites Only	1	3	Y = show mappings for favorite/feeder schools only. N = show mappings for all schools. Favorite schools are set up in UCX-TRQ060.
Link to School URL	1	4	Y = include link to school's URL (see UCX-TRQ060), or do a search on the school name if no URL is found.
Reserved	167	3	Reserved for future use.
Status	1	170	Y = Active

UCX VALUE	Len	Start	Description
N = Inactive			
W = Issue Warning message for override			
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SCR001 Reserved Words

Updated: September 29, 2023

UCX-SCR001 contains one record for each word of the Degree Works language. It is used to create drop-down list boxes for the Scribe program.

Warning! These codes are defined by Ellucian and institutions should not create their own. For this particular table, you can change the Description and the Output flag, but do not modify anything else unless instructed by Ellucian.

UCX KEY	Len	Description	
Reserved Word	12	Degree Works Reserved Word	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	Free-text description.
Note 1	59	31	Documentation note (helpful to define if nomenclature differences).
Code Length	2	90	This is the length of the Code in the Scribe drop-down list box of valid values for a blocktype, for example, "06" for COLLEGE.
Literal Length	2	92	This is the length of the Description in the Scribe drop-down list box of valid values for a blocktype, for example, "30" for COLLEGE.
Output	1	94	Y/N. This flag controls whether or not this rule or block qualifier is included on Degree Works audit reports. Y = include on output. N = omit from output.
UCX Reference	3	95	The UCX table against which this reserved word is validated. If the Cross-Reference Table is blank then this reserved word is

UCX VALUE	Len	Start	Description
			not validated against any UCX table. If it is filled in, then the UCX table must be valid in UCX-SYS001. For example, the DEGREE reserved word has its values validated in UCX-STU307, but the MINGPA reserved word does not validate its values against any UCX table.
Block	1	98	Y = this reserved word is valid in a BLOCK rule, for example, 1 BLOCK (MAJOR=PSY) and appears on the Scribe drop-down list box for the BLOCK rule. N = this reserved word is not valid in a BLOCK rule and is not part of the Scribe drop-down list box for the BLOCK rule.
Blocktype	1	99	Y = this reserved word is valid in a BLOCKTYPE rule, for example, 1 BLOCKTYPE (MAJOR) and appears on the Scribe drop-down list box for the BLOCKTYPE rule. N = this reserved word is not valid in a BLOCKTYPE rule and is not part of the Scribe drop-down list box for the BLOCKTYPE rule.
Enable	1	100	<p>This flag is only used if the Blocktype Flag is "Y". It indicates whether or not the institution uses this kind of data.</p> <p>For example, if this flag is set to "Y" for the MINOR reserved word then the institution will write requirements for minors.</p> <p>If set to "N" for MINOR then the institution does not allow students to choose a minor and no requirements for minors will be written in Degree Works. The drop-down list boxes in Scribe for BLOCK and BLOCKTYPE do not include reserved words with the Enable Flag set to "N".</p>
Note 2	40	101	Documentation note (helpful to define web usage).
Reserved	29	141	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.

UCX VALUE	Len	Start	Description
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SCR002 Custom Data

Updated: March 25, 2022

UCX-SCR002 contains one record for each data item from the student system to be used in an IF requirement.

If an institution wants to construct requirements that will necessitate evaluation of data not included in the "standard" then these custom data items must be added to UCX-SCR002 before the requirement can be written.

For example, religion is not a piece of data that is typically part of a degree requirement. However, a school may have a requirement that varies based on the student's religion. In order to construct the requirement, religion must be added as a custom data item to UCX-SCR002. Custom data items can be used in an IF rule in a requirements block. Degree Works uses this table to ascertain which pieces of data from the RAD tables should be sent to the Auditor Engine in addition to the standard academic data.

For each entry in UCX-SCR002, the Description, IF-Element, and UCX Table must be entered in the value-area. The rest of the value-area is optional and is typically used only when retrieving data from a detail table (where more than one record can exist per student, for example, transfer courses or test scores). These fields indicate which records from the table to use and which to skip. Only records that pass the edits are used as a source of custom data. For example, if you want a test score from the many stored in the database, but you only want the one for test code "GRE" then use test score as the IF-Element, test code as the Edit Element, and "GRE" as the Edit Value.

DO NOT use any Scribe Reserved Word as the beginning of your key into UCX-SCR002. "COLLEGE" or "COLLEGE2" as custom data items will NOT work. The Parser Engine sees "COLLEGE" and then gives an error on "2".

The UCX-SYS999 entry for Edit Element 1, 2, 3 and If-Element MUST contain the correct starting byte position of the data item (UCX-SYS999-Mask) and the correct data length.

Most custom data items are bridged to the rad_custom_dtl. The rad_custom_code = 'R322' and the rad_custom_value = 'R323'. For example, with a "STATUS" bridged to the rad_custom_dtl the record displayed below should suffice.

Therefore, set the Data Element to R323 and set the Edit Element 1 to R322.

Here are these and other helpful values that are most commonly used in SCR002:

Table	Data Element	Edit Element 1
rad_CUSTOM_dtl	R323 – rad_custom_value	R322 – rad_custom_code
rad_TEST_dtl	1292 - rad_test_score	1291 – rad_test_code

Table	Data Element	Edit Element 1
rad_NONCRSE_dtl	R303 – rad_non_score	R302 – rad_non_course
rad_PREVINST_dtl	4311 – rad_prev_degree	n/a

See UCX-SYS999 for the element numbers, for other tables and fields.

For Banner sites, Student Attributes are placed into the rad_custom_dtl in Degree Works with a custom-code of “ATTRIBUTE” and a custom-value of the attribute code, such as “HONR” for example.

UCX-SCR002 must contain an ATTRIBUTE entry telling Degree Works to retrieve all rad_custom_dtl ATTRIBUTE records and send them to the auditor.

When running a Financial Aid audit all of the rad-aid-dtl codes and values are pulled into the audit automatically – you do not need to define any UCX-SCR002 records for the rad-aid-dtl items you want to use in your Scribing.

UCX KEY	Len	Description
Custom Data Item	12	The name of the custom data to be used by Degree Works, for example, RELIGION or ROTC.
Reserved 1	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Free-text description.
Data Element	4	31	Element number from UCX-SYS999 indicating the data source to be used in a Scribe IF statement.
Reserved	3	35	Reserved for future use.
Edit Element 1	4	38	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the If-Element data should be extracted from this detail. This element number must be from the same table as the If-Element.
Edit Type 1	2	42	The type of check to do on the data in Edit Element 1. EV or blanks means use the value in Edit Value 1.
Edit Value 1	12	44	The specific data value against which the data in Edit Element 1 should be checked. The record is skipped if it does not match.
Edit Element 2	4	56	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the If-Element data should be extracted from this detail. This element number must be from the same table as the If-Element.

UCX VALUE	Len	Start	Description
Edit Type 2	2	60	The type of check to do on the data in Edit Element 2. EV or blanks means use the value in Edit Value 2.
Edit Value 2	12	62	The specific data value against which the data in Edit Element 2 should be checked. The record is skipped if it does not match.
Edit Element 3	4	74	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the If-Element data should be extracted from this detail. This element number must be from the same table as the If-Element.
Edit Type 3	2	78	The type of check to do on the data in Edit Element 3. EV or blanks means use the value in Edit Value 3.
Edit Value 3	12	80	The specific data value against which the data in Edit Element 3 should be checked. The record is skipped if it does not match.
Reserved	78	92	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SCR003 Noncourse Codes

Updated: March 25, 2022

UCX-SCR003 contains one record for each data item from the student information system that Degree Works should evaluate against NONCOURSE rules.

Typically, noncourse data includes required activities such as thesis, chapel, recital, or proficiency exams. The software uses this table to ascertain which pieces of data bridged from the student system should be sent to the Auditor Engine as noncourse data.

For each entry in UCX-SCR003, the Description and Data Element must be entered in the value-area. The rest of the value-area is optional and is typically used only when retrieving data from a detail table (where more than one record can exist per student, for example, transfer courses or test scores). These fields indicate which records from the table to use.

Note: Place 0000 in the Data Element field for those items in the rad_noncrse_dtl; only entries for values located in other tables require a valid element number. The

software always reads the noncourse records stored in the rad_noncrse_dtl regardless of what is in UCX-SCR003. The only reason for a record here for those entries in the rad_noncrse_dtl is to prevent a parsing error when your Noncourse rule is parsed when saving in Scribe.

DO NOT use any Scribe Reserved Word as the beginning of your key into UCX-SCR003. "COLLEGE" or "COLLEGE2" as custom data items will NOT work. The Parser Engine sees "COLLEGE" and then gives an error on "2".

The UCX-SYS999 entry for Edit Element 1, 2, 3 and If-Element MUST contain the correct starting byte position of the data item (UCX-SYS999-Mask) and the correct data length.

If your noncourse value is stored in the rad_custom_dtl then set the Data Element to R323 and set the Edit Element 1 to R322.

If your noncourse value is stored in the rad_test_dtl, set the Data Element to 1292 and set the Edit Element 1 to 1291.

UCX KEY	Len	Description	
Noncourse Code	12	The name of the noncourse data item to be used by Degree Works, for example, THESIS or ENGTEST.	
Reserved 1	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	Free-text description.
Data Element	4	31	Element number from UCX-SYS999 indicating the data source for a Non Course entry. Use 0000 if the noncourse is to be pulled from the rad_noncrse_dtl. Use R323 if the noncourse is to be pulled from the rad_custom_dtl. Use 1292 if the noncourse is to be pulled from the rad_test_dtl.
UCX Table	3	35	UCX table number in which this element is validated. If the element is not validated in a UCX table (i.e. the data is a date, number, or free-text) then leave UCX Table blank.
Edit Element 1	4	38	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the noncourse should be extracted from this detail. This element number must be from the same table as the Data Element.

UCX VALUE	Len	Start	Description
			Leave this field blank if the noncourse is stored on the rad_noncrse_dtl.
			Use R323 if the noncourse is to be pulled from the rad_custom_dtl.
			Use 1291 if the noncourse is to be pulled from the rad_test_dtl.
Edit Type 1	2	42	The type of check to do on the data in Edit Element 1. EV or blanks = use the value in Edit Value 1.
Edit Value 1	12	44	The specific data value against which the data in Edit Element 1 should be checked. The record is skipped if it does not match.
Edit Element 2	4	56	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the Noncourse-Element should be extracted from this detail. This element number must be from the same table as the Noncourse-Element.
Edit Type 2	2	60	The type of check to do on the data in Edit Element 2. EV or blanks = use the value in Edit Value 2.
Edit Value 2	12	62	The specific data value against which the data in Edit Element 2 should be checked. The record is skipped if it does not match.
Edit Element 3	4	74	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the Noncourse-Element should be extracted from this detail. This element number must be from the same table as the Noncourse-Element.
Edit Type 3	2	78	The type of check to do on the data in Edit Element 3. EV or blanks = use the value in Edit Value 3.
Edit Value 3	12	80	The specific data value against which the data in Edit Element 3 should be checked. The record is skipped if it does not match.
Reserved	78	92	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.

UCX VALUE	Len	Start	Description
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SCR004 Requirement Block Type Codes

Updated: September 29, 2023

UCX-SCR004 contains one record for each Degree Works reserved word that is a valid block type. A block type is the type of requirement block, the primary database tag. This table is used to create the drop-down list box of valid block types.

Warning! This table is constructed by Ellucian and institutions should not create their own. For this particular table, you can change the Description and flags, but do not modify anything else unless instructed by Ellucian.

UCX KEY	Len	Description
Block type	12	The Degree Works reserved word from UCX-SCR001 that is valid as a requirement block type.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Free-text description.
Show Credits Required	1	31	Y/N. In worksheet block headers in the Responsive Dashboard, show the Credits Required for blocks of this type. See the HeaderTag note below.
Show Credits Applied	1	32	Y/N. In worksheet block headers in the Responsive Dashboard, show the Credits Applied for blocks of this type. See the HeaderTag note below.
Show Catalog Year	1	33	Y/N. In worksheet block headers in the Responsive Dashboard, show the Catalog Year for blocks of this type. See the HeaderTag note below.
Show GPA	1	34	Y/N. In worksheet block headers in the Responsive Dashboard, show the GPA for blocks of this type. See the HeaderTag note below.
Reserved	135	35	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code

UCX VALUE	Len	Start	Description
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

You can use HeaderTags to override the block header Show flags above. For example, if Show GPA is N for the MINOR block type 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. Please see the [HeaderTag](#) topic for more information.

UCX-SCR007 Block Parsed Status

Updated: September 29, 2023

UCX-SCR007 contains one record for each parse status code. A parse status code indicates whether a requirements block has been parsed by the Parser without errors.

Warning! This table is constructed by Ellucian and institutions should not create their own. For this particular table, you can change the Description, but do not modify anything else unless instructed by Ellucian.

UCX KEY	Len	Description
Parse Status Code	2	The parse status code is one of the following: "NO" = did not parse successfully. Parsed with errors. "OK" = did parse successfully. Parsed without errors.
Reserved	28	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Free-text description of the parse status code.
Reserved	139	31	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SCR008 Note Type Codes

Updated: September 29, 2023

UCX-SCR008 defines the Note Type codes. The note type describes the category of note or the note source.

Note type codes can be referenced in the **Default Public Note Type** and **Default Internal Note Type** fields in AUD012 User Class Codes to define the default note type for a user class. The note type ends up in the **DAP_NOTE_TYPE** field of DAP_NOTE_DTL.

If the Internal Note flag is Y, the note is visible only to users with access to internal only notes.

UCX KEY	Len	Description	
Note Type	2	The type of note.	
Reserved	28	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	Free-text description.
Reserved	69	31	Reserved for future use.
Internal Note	1	100	Y/N. Y = Note is to display on output only if the user class has access to internal-only notes. N = Note is not restricted to internal users. Default = Y
Reserved	69	101	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SCR044 WITH Custom Class Data

Updated: March 25, 2022

UCX-SCR044 defines the custom class data to be used after the WITH reserved word. Each data item must exist in the rad_class_dtl.

Degree Works predefines certain standard WITH elements that do not need to be defined in UCX-SCR044. The predefined WITH items are: DWTITLE, DWTERM, DWLOCATION, DWGRADE, DWGRADETYP, DWCREDITS, DWRESIDENT, DWTRANSFER, DWCREDITTYPE, DWPASSFAIL.

Attributes bridged to the rad_attr_dtl need a record here, but the Element field should be ATTR (Offset and Length can be 00).

Degree Works looks up UCX-SCR044 WITH Element in UCX-SYS999 to get the data length and data mask of the data to be passed to the Auditor Engine. The data is extracted from the

rad_class_dtl and placed in the WITH buffer for the class. The location of the data in the WITH buffer is defined by UCX-SCR044 Offset. UCX-SCR044 Length is used by the Auditor Engine when evaluating the WITH class specifier and by the auditor when printing class information on an audit report.

UCX KEY	Len	Description	
Custom Data Item	12	The name of the custom class data to be used with "WITH".	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	
Description	30	1	Free-text description.
Element	4	31	Element number from UCX-SYS999 indicating the WITH data source. The software looks up the data length and data mask in UCX-SYS999, extracts the data and saves it in the WITH buffer for the class.
UCX Table	3	35	UCX table number in which this element is validated. If the element is not validated in a UCX table (i.e. the data is a date, number, or free-text) then leave UCX Table blank.
Offset	2	38	The starting position of the data in the WITH buffer for each class record. Start the first item at offset "01". The offset for the next item is then this offset plus this length. With an offset of 01 and a length of 04 the next offset will be 05.
Length	2	40	The length in bytes of the data in the WITH buffer, used by the auditor when evaluating WITH. Usually the UCX-SCR044 Length is the same as UCX-SYS999 Data Length.
Reserved	9	42	Reserved for future use.
Advice Literal	30	51	The free-text description to be printed as part of the audit advice.
Reserved	89	81	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SCR045 DECIDE Options

Updated: March 25, 2022

UCX-SCR045 defines the DECIDE codes to be used after the DECIDE reserved word in block qualifiers and course rules.

The DECIDE rules may vary by school and college. Therefore, the key allows optional school and college.

The Auditor Engine, DAP14, subroutine looks up the UCX-SCR045 DECIDE code to determine how it should decide which classes to remove and which to keep when a qualifier's or course rule's maximum has been exceeded.

By leaving a flag blank for one of the items in UCX-SCR045 the auditor does not consider this characteristic when deciding which classes to remove.

If the GRADE-KEEP flag is set to "H", for example, then the auditor keeps the class that has a higher grade.

If the flag is "L" then the auditor keeps the class with a lower grade.

If the flag is left blank then the auditor does not consider grades when making the decision.

The priority tells the auditor the order in which the characteristics should be checked. If any characteristic does not help the auditor distinguish one class from another then the next characteristic is considered.

UCX KEY	Len	Description
DECIDE Code	12	The code to be used with the "DECIDE" keyword on qualifiers and rules.
DECIDE School	12	The school code from UCX-STU350. Leave blank if DECIDE rules are the same for all schools and colleges.
DECIDE College	6	The college code from UCX-STU560. Leave blank if DECIDE rules are the same for all colleges.

UCX VALUE	Len	Start	Description
Grade Rank	2	1	Priority of the Grade factor, "1" - "9". Default = "1".
Grade Keep	1	3	Keep-Flag of the Grade factor. H = keep the course with the highest grade. L = keep the course with the lowest grade. Default = blank – do not look at the grade.
Term Rank	2	4	Priority of the Term factor, "1" - "9". Default = "2".
Term Keep	1	6	Keep-Flag of the Term factor. H = keep the course with the most recent term. L = keep

UCX VALUE	Len	Start	Description
			the course with the oldest term. Default = blank – do not look at the term.
Credits Rank	2	7	Priority of the Credits factor, "1" - "9". Default = "3".
Credits Keep	1	9	Keep-Flag of the Credits factor. H = keep the course with the highest credits. L = keep the course with the lowest credits. Default = blank – do not look at the credits.
Course Number Rank	2	10	Priority of the Course Number factor, "1" - "9". Default = "4".
Course Number Keep	1	12	Keep-Flag of the Course Number factor. H = keep the course with the highest course number. L = keep the course with the lowest course number. Default = blank – do not look at the course number.
Passfail Rank	2	13	Priority of the Passfail factor, "1" - "9". Default = "5".
Passfail Keep	1	15	Keep-Flag of the Passfail factor. Y = keep the Passfail course. N = do not keep the Passfail course. Default = blank – do not look at the passfail status.
Transfer Rank	2	16	Priority of the Transfer factor, "1" - "9". Default = "6".
Transfer Keep	1	18	Keep-Flag of the Transfer factor. Y = keep the Transfer course. N = do not keep the Transfer course. Default = blank – do not look at the transfer flag.
Repeat Rank	2	19	Priority of the Repeat factor, "1" - "9". Default = "7".
Repeat Keep	1	21	Keep-Flag of the Repeat factor. Y = keep the repeated (do-over) course. N = do not keep the repeated (do-over) course. Default = blank – do not look at the repeat flag.
Incomplete Rank	2	22	Priority of the Incomplete factor, "1" - "9". Default = "8".
Incomplete Keep	1	24	Keep-Flag of the Incomplete factor. Y = keep the Incomplete course. N = do not keep the Incomplete course. Default = blank – do not look at the incomplete flag.
In-Progress Rank	2	25	Priority of the In-Progress factor, "1" - "9". Default = "9".
In-Progress Keep	1	27	Keep-Flag of the In-Progress factor. Y = keep the In-Progress course. N = do not keep the In-Progress course. Default = blank – do not look at the in-progress flag.

UCX VALUE	Len	Start	Description
Reserved	142	28	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SEP001 Template Tags

Updated: March 25, 2022

This table is used to define the tags that are saved on templates. Some tags can be required so that the tag must be placed on all templates and some tags may not even need to match against student curriculum data perhaps to be used for organizational purposes only.

UCX KEY	Len	Description
Tag Type	12	A tag type code – such as MAJOR, DEGREE, CONC, etc.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	Describes this template tag.
Validation Table	6	51	The UCX table containing the values for this type.
Match Student Goal Data	1	57	Y=this type should be found in students' curriculum data; N=this type will not be found in student's data and is used for template organization only.
Hierarchy Order	2	58	Used in the template tree view
Required	1	60	Y=when saving templates this tag must be assigned; N=tag is optional
Reserved	109	61	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.

UCX VALUE	Len	Start	Description
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

If you don't want one of the UCX-SEP001 tags to appear anywhere in SEP then simply delete the entry. For example, if you don't want Liberal Learning to appear in the Required or Optional list of template tags you can simply delete the LIBL record from this table.

The reserved tag types used by Degree Works are listed below. Only these types can be matched against student goal data and thus the Match Student Goal Data flag should be set to Y for these entries and no others.

Tag Type	Description
CATYEAR	Catalog/Academic year
COLLEGE	College
CONC	Concentration
DEGREE	Degree
LIBL	Liberal Learning
MAJOR	Major
MINOR	Minor
PROGRAM	Program
SCHOOL	Level/School
SPEC	Specialization

You can, however, create your own template tags. An example of such a tag is CO-OP. This can be used to help organize your templates but would not be used by the software when creating plans from templates in batch. The Match Student Goal Data flag would be set to N for this type of record. The Validation Table setting should be filled in pointing to one of the 10 special UCX tables set aside for custom template tags: UCX-SEP060 through UCX-SEP069. In these tables you would then list all of the valid values for the tag. If the template tag was CO-OP in UCX-SEP060 you would list all valid CO-OP values and place "SEP060" in the Validation Table field.

UCX-SEP002 Template Term Schemes

Updated: March 25, 2022

This table is used to define the types of terms and the number of terms that are to be built as part of the template.

Additionally, this is used when creating a plan for a template to define the specific terms that are to be placed on the plan. For example, if your template, and thus your plans created from the template, is to have eight terms then you need eight records defined in your term scheme with

sequence number from “001” through “008”. Each of the records in your scheme may then have FALL and SPRING for the two terms of each academic year.

UCX KEY	Len	Description	
Term Scheme ID	27	An identifier for the type of scheme being defined.	
Sequence	3	A 3-digit number, padded with zeroes, which represents the order of the literal on the template.	
UCX VALUE	Len	Start	
Literal	50	1	The literal that will be displayed in the template. For example, “Semester 3” or “3rd Spring Term” or “Year 1 – Spring”
Description	50	51	Any description you like – but you may simply want to use the same value as the Literal.
Term Type	10	101	The type of term: SUMMER, SPRING, WINTER, FALL. This type is used along with the starting term to find the correct term from UCX-STU016 when building plans from templates. This value should always be upper-case.
Reserved	59	111	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Example term scheme for a 4-year semester school

Term Scheme ID	Sequence	Literal	Description	Term Type
COMPLETE_4YEAR	001	Fall – Year 1	1st fall	FALL
COMPLETE_4YEAR	002	Spring – Year 1	1st spring	SPRING
COMPLETE_4YEAR	003	Fall – Year 2	2nd fall	FALL
COMPLETE_4YEAR	004	Spring – Year 2	2nd spring	SPRING
COMPLETE_4YEAR	005	Fall – Year 3	3rd fall	FALL
COMPLETE_4YEAR	006	Spring – Year 3	3rd spring	SPRING

Term Scheme ID	Sequence	Literal	Description	Term Type
COMPLETE_4YEAR	007	Fall – Year 4	4th fall	FALL
COMPLETE_4YEAR	008	Spring – Year 4	4th spring	SPRING

UCX-SEP003 Requirement Delivery Codes

Updated: March 25, 2022

This table is used to define valid values for the delivery requirement attribute.

UCX KEY	Len	Description
Delivery Code	10	An identifier for the method of delivery for a course on a template or plan requirement. For example, ONLINE, SELFPACED, ONCAMPUS, etc.
Reserved	20	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	The description of this delivery type
Show in SEP Picklist	1	51	Y=show this delivery method in SEP
Reserved	118	52	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SEP004 Requirement Types

Updated: September 29, 2023

This table is used to define valid values for the requirement types.

Warning! This table is constructed by Ellucian and institutions should not create their own. For this particular table, you can change the Description and the Show in SEP Picklist flag, but do not modify anything else unless instructed by Ellucian.

UCX KEY	Len	Description	
Requirement type	30	A template or plan requirement type.	
UCX VALUE	Len	Start	Description
Description	50	1	The description of this requirement type
Show in SEP Picklist	1	51	Y=show this delivery method in SEP
Reserved	118	52	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SEP005 Placeholder Requirement Types

Updated: March 25, 2022

This table is used to define valid placeholder requirement types.

UCX KEY	Len	Description	
Requirement Type	12	A template or plan placeholder requirement type.	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	50	1	The description of this placeholder requirement type
Show in SEP Picklist	1	51	Y=show this delivery method in SEP
Reserved	118	52	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SEP006 Test Codes

Updated: March 25, 2022

This table is used to define valid planner test codes.

UCX KEY	Len	Description
Test Code	10	A template or plan test code
Reserved	20	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	The description of this test code.
Score length	2	51	Length of the score for this test code
Score type	1	53	A=alphabetic (A-Z); N=nemonic
Minimum Valid Score	4	54	The lowest possible score for this test
Maximum Valid Score	4	58	The highest possible score for this test
Show in SEP Picklist	1	62	Y=show this test code in SEP
Reserved	107	63	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SEP007 Test Score Sort Order

Updated: March 25, 2022

This table is used to define the sort order for valid planner test codes. Entries are needed in this table for tests using non-numeric test scores such as K# or %P. The planner needs to know how one test score compares to all others.

UCX KEY	Len	Description
Test Code	10	A template or plan test code from UCX-SEP006
Reserved	2	Reserved for future use.
Test Score	6	The test score for this test code
Reserved	12	Reserved for future use.

UCX VALUE	Len	Start	Description
Sort order	3	1	A value used to compare against other values for the different test scores for this test code. The lowest test score should have the lowest sort order value, with the highest test score containing the highest sort order value. For example, if there are 5 entries for a test code with "A" being the highest test score value, the sort order for "A" should be 5 (and not 1).
Reserved	166	4	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SEP008 GPA Requirement Types

Updated: March 25, 2022

This table is used to define valid GPA requirement types.

UCX KEY	Len	Description	
GPA Type	10	A template or planner GPA requirement type.	
Reserved	20	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	50	1	The description of this GPA requirement type.
Show in SEP Picklist	1	51	Y=show this GPA requirement type in SEP
Reserved	118	52	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code

UCX VALUE	Len	Start	Description
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

The GPA codes supported by Degree Works are listed below. These values should not be deleted but you can set the Show In SEP Picklist flag to "N" for any entries you do not want to appear in the planner and you can alter the description as needed. You should not create your own GPA codes.

GPA Code	Description
CLASS	Class list GPA
MAJOR	Major GPA
OVERALLSIS	Student System Overall GPA
OVERALLDGW	Degree Works Calculated Overall GPA

UCX-SEP009 Choice Scribe Pointers

Updated: March 25, 2022

This table is used to define valid Scribe pointers for the Choice requirement of the Student Educational Planner (SEP).

When enabled for display, Scribe pointers are listed in the 'Pointer' drop-down list of the Choice requirement.

If the user does not make a selection on the Choice requirement, and if the Scribe pointer value is not blank, then the pointer is passed to the auditor when a planner audit is run and the Choice requirement will be applied to the associated rule.

UCX KEY	Len	Description
Scribe Pointer	12	The name of the Scribe pointer that will be displayed in the drop-down list. For example, "COREMATH".
Filler	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	The description of this Scribe Pointer. For example, "Core Math Requirement".
Show in SEP Picklist	1	51	Y=show this GPA requirement type in SEP
Reserved	118	52	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.

UCX VALUE	Len	Start	Description
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SEP011 Plan Scope Values

Updated: March 25, 2022

This table is used to define valid Plan Scope Values for the Student Educational Planner.

UCX KEY	Len	Description
Plan Scope Code	6	The Plan Scope key
Filler	24	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	The Plan Scope description that will be displayed on the plan. For example, "Abbreviated Plan".
API Code	6	51	The code used in the Student System for this Plan Scope code.
Validation Flag	1	57	The Validation Flag indicates what type of validation should be done on this Plan Scope. Valid options are A-Abbreviated, C-Comprehensive and N-None.
Reserved	8	58	Reserved For future use.
Show in SEP Picklist	1	66	Y=show this GPA requirement type in SEP Picklist
Reserved	103	67	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SEP060 - UCX-SEP069 User-Defined Template Tags

Updated: March 24, 2023

A template administrator can define valid values for user-defined tags on templates. These 10 UCX tables are used to validate those values. In UCX-SEP001 you can add additional template tags beyond those automatically supported by Degree Works.

Also review the UCX-SEP001 documentation.

Example

If you added CO-OP to UCX-SEP001 and set the Validation Table field to SEP060 then in UCX-SEP060 you would place all of your CO-OP codes and descriptions.

UCX KEY	Len	Description	
User-Defined Tag Code	12	The code for the user-defined template tag	
Filler	18	Reserved for future use.	
<hr/>			
UCX VALUE	Len	Start	Description
Description	30	1	The description of this template tag
Show in SEP Picklist	1	31	Y = show this template tag value in SEP
Reserved	138	32	Reserved For future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SHP078 SHP Keys

Updated: March 25, 2022

UCX-SHP078 contains the keys used for access to services within Degree Works.

Warning! Do not modify.

These keys are supplied by Ellucian corresponding to specific services delivered within the product.

UCX KEY	Len	Description
SHP Key	8	Key corresponding to a Degree Works service
Reserved	22	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	100	1	Description of key
Reserved	69	101	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SHP080 Locale Codes

Updated: March 25, 2022

UCX-SHP080 contains locale codes used by application resources. This table is used for validation in Controller.

A set of example entries are delivered with Degree Works, but these can be deleted and additional entries added as needed.

UCX KEY	Len	Description
Locale Code	10	IETF language tag to be used for the localization of application properties.
Reserved	20	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	Description of locale code
Reserved	119	51	Reserved for future use
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU016 DegreeWorks Term Descriptions

Updated: March 24, 2023

UCX-STU016 maps the student system term to the Degree Works catalog year. The term literal may be printed on the audit report.

UCX KEY	Len	Description	
Term	12	The term code from the student system.	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Short Description	12	1	Appears in audit worksheets when no long description exists, but always appears in plans.
Long Description	30	13	A longer version of the term description, which will display by default in the audit worksheets. If blank, the short description is used instead.
Reserved	8	43	Reserved for future use.
Catalog Year	12	51	Degree Works catalog year. Must be valid on UCX-STU035.
Reserved	1	63	Reserved for future use.
Term Type	10	64	SPRING, FALL, etc. This value should always be upper-case and must match the Term Type values used in SEP002 if the Student Planner is used.
Show in Transfer Equivalency Self-Service	1	74	<p>Y/Q/B/N.</p> <p>Y = Include in Transfer Equivalency Self-Service transfer classes screen picklists,</p> <p>Q= Show on Transfer Equivalency Self-Service questions screen.</p> <p>B= Show on Both screens.</p> <p>N = Exclude from Transfer Equivalency Self-Service picklists.</p>
Financial Aid Year	4	75	The Financial Aid year in which this term lies. Used for Financial Aid Audits.
Show in SEP Picklist	1	79	Y=Show in Student Educational Planner picklists.
Term Start Date	8	80	The term start date. Used to determine the current term in the next generation Student Educational Planner and for Years Ago conditions on Transfer Equivalency mappings. This is also used for Transfer Equivalency Self-Service audits when DWAge is used to restrict older classes.

UCX VALUE	Len	Start	Description
			For these types of audits the active term is not defined for the student so we use the date ranges here to determine the current term. The date is in the YYYYMMDD format.
Term End Date	8	88	The term end date. Used to determine the current term in the next generation Student Educational Planner and for Years Ago conditions on Transfer Equivalency mappings. This is also used for Transfer Equivalency Self-Service audits when DWAge is used to restrict older classes. For these types of audits the active term is not defined for the student so we use the date ranges here to determine the current term. The date is in the YYYYMMDD format.
Transfer Finder Period	12	96	Corresponding Term Global Mapping Code from TRF100. Used by the Transfer Finder application only.
Reserved	62	108	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU023 Student Major Codes

Updated: March 24, 2023

UCX-STU023 defines the student Major codes, the codes that are stored in the RAD student system major fields.

UCX KEY	Len	Description
Student Major	12	The major code from the student system, without college, position, or school.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	Free-text description.

UCX VALUE	Len	Start	Description
Show in Transfer Equivalency Self-Service Picklist	1	51	Y/N. Y = Include in Transfer Equivalency Self-Service picklists. N = Exclude from Transfer Equivalency Self-Service picklists.
Filter 1	12	52	If the treq.selfService.degreeMajorFilter setting is Y and one of these degrees is selected on the Transfer Equivalency Self-Service page, this major will appear in the drop down list box. Only majors valid for the degree chosen will appear. This is same for "Filter 1 to Filter 5".
Filter 2	12	64	
Filter 3	12	76	
Filter 4	12	88	
Filter 5	12	100	
Show in SEP Picklist	1	112	Y=Show in Student Educational Planner picklists
Similar major	12	113	See notes below.
Show in Advanced Search Picklist	1	125	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	45	125	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	44	126	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Similar major

For exceptions this major is considered similar to the major in the key. If an exception was applied to the major in the key but that major no longer appears in the student's audit then the auditor will attempt to apply the exception to this similar major if this similar major does exist in the audit. This is helpful for cases wherein a student is first assigned to a pre-major and then is later moved to the declared major. For example, the student is assigned the pre-nursing major before being accepted into the nursing program and being assigned the nursing major. The two blocks are similar or perhaps identical and any exceptions applying to the first major should be applied to the second

major. This can also be used for students switching between similar majors like art and art history. When the label tags are identical between majors the auditor can apply exceptions from the first major into the second. In the case of art and art history, you may want to have ARTH in the similar major field on the ART record and, conversely, place ART into the similar major field on the ARTH record. This will allow students going from either major to the other major have their exceptions correctly carried over.

Keep in mind that the UCX-CFG020 DAP14 **Apply exception to a similar block** setting must be Y or A for this similar major functionality to work.

UCX-STU024 Student Minor Codes

Updated: March 25, 2022

UCX-STU024 defines the student Minor codes, the codes that are stored in the RAD student system minor fields.

UCX KEY	Len	Description	
Student Minor	12	The minor code from the student system, without college, position, or school.	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	50	1	Free-text description.
Show in Transfer Equivalency Self-Service	1	51	Y/N. Y = Include in Transfer Equivalency Self-Service picklists. N = Exclude from Transfer Equivalency Self-Service picklists.
Show in SEP Picklist	1	52	Y=Show in Student Educational Planner picklists
Show in Advanced Search Picklist	1	53	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	117	53	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	116	54	Reserved for future use.
Revision	6	201	Revision code

UCX VALUE	Len	Start	Description
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU035 Degree Works Catalog Year

Updated: March 25, 2022

UCX-STU035 defines the valid Degree Works catalog year codes.

Catalog year is the name of an edition of a course catalog, typically a range of years. Catalog years must be created so that they sort in ascending order, that is, the last entry in UCX-STU035 should be the latest catalog year.

UCX KEY	Len	Description
Catalog Year	12	The Catalog Year.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Free-text description.
Reserved	1	31	Reserved for future use.
Show in Web What-If Picklist	1	32	Y/N. Y = Include in What-If picklist. N = Exclude from What-If picklist. A blank is treated as a Y.
Show in Transfer Equivalency Self-Service Picklist	1	33	Y/N. Y = Include in Transfer Equivalency Self-Service picklists. N = Exclude from Transfer Equivalency Self-Service picklists.
Show in SEP Picklist	1	34	Y=Show in Student Educational Planner picklists
Show in Advanced Search Picklist	1	35	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	134	36	Reserved for future use.

UCX VALUE	Len	Start	Description
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU050 Course Attributes

Updated: March 25, 2022

This table stores course attributes. Currently, these attributes are used only in the planner's attribute drop-down list when adding a course requirement.

For Banner, this table is populated from the STVATTR

UCX KEY	Len	Description
Attribute	12	Attribute Code. Banner: stvattr_code
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	The normal length description of the attribute. Banner: stvattr_desc
Reserved	139	31	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU305 Student Level Codes

Updated: March 25, 2022

This table is used to validate student level codes such as FR, SO, JR, SR, and so on.

In Banner this is called "classification" and this table is akin to STVCLAS.

UCX KEY	Len	Description	
Student Level	12	Student Level Code.	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	The normal length description of the Student Level Code.
Show in Advanced Search Picklist	1	31	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	138	32	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU306 Student Status Codes

Updated: March 25, 2022

This table is used to validate the Student Status Code.

Examples of student status include continuing student, holdover student, new or first time student, readmitted student, transfer student, withdrawn student, or graduated student.

UCX KEY	Len	Description	
Student Status	6	Student Status Code.	
Reserved	24	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	50	1	The normal length description of the Student Status.

UCX VALUE	Len	Start	Description
Show in Advanced Search Picklist	1	51	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	118	52	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU307 Degree Codes

Updated: March 25, 2022

This table is used to validate the Degree Code. It contains a list of the degrees granted at your institution.

UCX KEY	Len	Description
Degree Code	12	Degree Code
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	The normal length description of the Degree Code.
Reserved	40	51	Reserved for future use.
Short Description	10	91	Short description of the degree code.
Reserved	36	101	Reserved for future use.
Remedial Mode	1	137	When UCXCFG020 DAP14 "Degree Drives Remedial Mode" flag set to "Y", this flag is enabled. A: Treat Remedial classes as normal classes. B: Force Remedial classes into over-the-limit.

UCX VALUE	Len	Start	Description
Show in Transfer Equivalency Self-Service Picklist	1	138	Y/N. Y = Include in Transfer Equivalency Self-Service picklists. N = Exclude from Transfer Equivalency Self-Service picklists.
Show in What-if Picklist	1	139	Y/N. Y = Include in What-if picklists. N = Exclude from What-if picklists.
			Note, when curriculum rules are being used to filter the What-If display in Transfer Equivalency Self-Service it is this flag that is used to filter the drop-down list and not the flag above.
Show in SEP Picklist	1	140	Y=Show in Student Educational Planner picklists
Show in Advanced Search Picklist	1	141	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	28	142	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Remedial classes

The Remedial Mode controls whether classes for certain degrees should be forced into over-the-limit. Remedial classes are defined as: any classes with a level under 100. Any classes that have letters in the level are not considered remedial. Examples of Remedial: 099, 95, 009, 45, 5. Examples of Not Remedial: 123, A99, 99A, 5AB, 300. This flag works in conjunction with the UCX-CFG020 DAP14 RemedialMode setting. If that flag is set to "Y," then remedial class functionality is enabled. If the flag is set to "N," then remedial class functionality is disabled.

Class level examples

Level	Status
99	Remedial
099	Remedial

Level	Status
5	Remedial
005	Remedial
A99	Not Remedial
99A	Not Remedial
100	Not remedial
100A	Not remedial
A100	Not remedial
ABC	Not remedial
A9C	Not remedial

UCX-STU314 Test Codes

Updated: March 25, 2022

This table is used by Transfer Equivalency to maintain and validate all test types and scores.

Existing codes have been constructed to conform to national standard coding from the testing agencies and should not be altered. Additional Test Codes may be added into this table.

UCX KEY	Len	Description
Test Code	12	The Test Code.
Reserved	18	Reserved for future use.
<hr/>		
UCX VALUE	Len	Description
Description	30	1 The normal length description of the test.
Reserved	40	31 Reserved for future use.
Treq Minimum Score	4	71 A 4-digit minimum score that may be used if the test score reflects credit-by-exam. For example, Minimum score for "AP Computer Science" is "0003". This field may be left blank.
Credit-by-Exam	1	75 A Y/N flag indicating whether or not the test code represents a credit-by-exam code. Used by Transfer Equivalency in transfer articulation.
Reserved	25	76 Reserved for future use.
Reserved	69	101 Reserved for future use.
Status	1	170 Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171 Reserved for future use.

UCX VALUE	Len	Start	Description
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU316 Program Codes

Updated: March 25, 2022

This table is used to validate the Program Code.

This code may be used to indicate that the student is involved in a special program (for example, Honors program, Continuing Ed program).

UCX KEY	Len	Description
Program Code	12	The Program Code.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	The normal length description of the test.
Show in What-if Picklist	1	51	Y/N. Y = Include in What-if picklists. N = Exclude from What-if picklists.

Note, when curriculum rules are being used to filter the What-If display in Transfer Equivalency Self-Service it is this flag that is used to filter the drop-down list and not the flag below.

Show in SEP Picklist	1	52	Y=Show in Student Educational Planner picklists.
Show in Transfer Equivalency Self-Service Picklist	1	53	Y/N. Y = Include in Transfer Equivalency Self-Service picklists. N = Exclude from Transfer Equivalency Self-Service picklists.
Show in Advanced Search Picklist	1	54	Y/N. Y = Include in Advanced Search dropdown.

UCX VALUE	Len	Start	Description
			N = Exclude from Advanced Search dropdown.
Reserved	115	55	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU323 Specialization Codes

Updated: March 25, 2022

This table is used to validate the Specialization Code.

UCX KEY		Description	
Specialization		The Specialization Code	
Reserved		Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	The normal length description of the Specialization Code.
Show in What-if	1	31	Y/N. Y = Include in What-if picklists. N = Exclude from What-if picklists.
Show in SEP Picklist	1	32	Y=Show in Student Educational Planner picklists
Filter 1	12	33	Used to filter the data on the special what-if page.
Filter 2	12	45	Used to filter the data on the special what-if page.
Filter 3	12	57	Used to filter the data on the special what-if page.
Filter 4	12	69	Used to filter the data on the special what-if page.

UCX VALUE	Len	Start	Description
Show in Advanced Search Picklist	1	81	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	88	82	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU324 Liberal Learning Codes

Updated: March 25, 2022

This table is used to validate the Liberal Learning Code.

UCX KEY	Len	Description	
Liberal. Learning Code	12	The Liberal Learning Code.	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	The normal length description of the Liberal Learning Code.
Show in What-if Picklist	1	31	Y/N. Y = Include in What-if picklists. N = Exclude from What-if picklists.
Show in SEP Picklist	1	32	Y=Show in Student Educational Planner picklists
Show in Advanced Search Picklist	1	33	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	136	34	Reserved for future use.

UCX VALUE	Len	Start	Description
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU346 Transfer Calendar

Updated: March 25, 2022

This table is used to define the various academic calendars used by transfer schools.

It defines the methodologies required to translate transfer credits into institutional credits. For example, if "Quarter" credits are worth 75 percent (0750) of the actual number of transfer credits, then a 4.000 credit transfer class would only be worth 3.000 credits at the given institution (if a "Quarter" credit method is specified for the transfer class). The conversion percentage is specified in the table along with the credit method literal.

UCX KEY	Len	Start	Description
Calendar	2		The Calendar code indicating the academic calendar in use at a transfer school.
Reserved	28		Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	The description of the academic calendar.
Convert Number	4	31	A number multiplied against the transfer credits to get equivalent resident credits. Formatted 9v999.
Show in Transfer Equivalency Self-Service picklist	1	35	Y=This Academic Calendar should be shown in Transfer Equivalency Self-Service, N= This Academic Calendar should not be shown in Transfer Equivalency Self-Service.
Reserved	134	36	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code

UCX VALUE	Len	Start	Description
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU350 School Code

Updated: March 25, 2022

This table is used to validate the School Code.

A common usage of this field is in separating undergraduates from graduates. The degree audits use school to separate classes to apply to an audit, making school a useful field for students who return for subsequent degrees.

In Banner this is referred to as "level" and this table is akin to the STVLEVL table.

UCX KEY	Len	Description
School Code	12	The school code such as UG, GR, etc
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Reserved	3	1	Reserved for future use.
Description	30	4	The description of the school code.
Show in Transfer Equivalency Self-Service Picklist	1	34	Y/N. Y = Include in Transfer Equivalency Self-Service picklists. N = Exclude from Transfer Equivalency Self-Service picklists.
Full-time Credits	2	35	For Athletic Audits - what is considered full-time for this school?
Show in What-if Picklist	1	37	Y/N. Y = Include in What-if picklists. N = Exclude from What-if picklists.
			Note, when curriculum rules are being used to filter the What-If display in Transfer Equivalency Self-Service it is this flag that is used to filter the drop-down list and not the flag above.

UCX VALUE	Len	Start	Description
Show in SEP Picklist	1	38	Y = Show in Student Educational Planner picklists
Show in Advanced Search Picklist	1	39	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	120	40	Reserved for future use.
Status	1	170	Y=Active, N=Inactive, W=Issue Warning message for override.
Client Reserve	30	171	Reserved for Client use.
Revision	6	201	Revision Code.
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU352 Discipline Codes

Updated: March 25, 2022

This table is used to validate the Discipline Code. The discipline may be the same as a department of study.

UCX KEY	Len	Description
Discipline Code	12	The Discipline Code. The length of the longest discipline code is set in UCX-CFG020 DAP13 DiscLen.

Additional Rules:

- If the Discipline Code contains imbedded spaces in the R800UCXT BIF record the spaces will be replaced with the underscore character (_) automatically by the extract programs. For example, "AU B" would become "AU_B", "A C" would become "A__C", "A BC" would become "A_BC" while "ART " would remain "ART " as the space is trailing.
- If new Discipline codes are being input manually into UCX-STU352 using Controller or some other tool then the same rule must be followed: replace imbedded spaces with underscores. Otherwise

UCX KEY	Len	Description	
Degree Works will not process the discipline codes properly.			
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Discipline Description	30	1	The normal length description of the Discipline Code.
Reserved	12	31	Reserved for future use.
Discipline Status	1	43	BANNER ONLY I=Inactive. A = Active. Blank = Active.
If the discipline status for a given discipline on a current, historic or transfer class is 'I' the class will be SKIPPED and NOT bridged to Degree Works by the Banner Student Extract.			
Reserved	126	44	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU355 Credit Type Codes

Updated: March 25, 2022

This table is used to define the types of credit that can be accumulated by a student.

For instance, this code can be used to differentiate between academic credit, transfer credit, credit by exam, and other types of credit.

UCX KEY	Len	Description	
Credit Type	2	The Credit Type Code.	
Reserved	28	Reserved for future use.	

UCX VALUE	Len	Start	Description
Description	30	1	The normal length description of the Credit Type Code.
Reserved	1	31	Reserved for future use.
DAP Process	1	32	This Y/N flag indicating whether this Credit Type should be used by the Degree Audit Engine (DAP14) to count toward meeting academic requirements. "Y"=Credit can count toward meeting requirements. "N"=Credit cannot count toward meeting requirements.
Credit By Exam	1	33	This Y/N flag indicates whether this Credit Type reflects credit taken by exam (CLEP, AP, etc.).
Reserved	136	34	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU356 Grade Type Codes

Updated: March 25, 2022

UCX-STU356 contains the valid codes for grade type.

Each class has a grade type that groups grades into broad categories and indicates which type of grade will be given for this class.

This table is also used by Transfer Equivalency as the source of a picklist.

UCX KEY	Len	Description
School	12	The course School Code (UCX-STU350).
Grade Type	6	The course Grade Type Code.
Reserved	12	Reserved for future use.

UCX VALUE	Len	Start	Description
Range Code	1	1	Not Implemented
			(A flag (F, R, or N) used to indicate whether grades associated with this grade type are numeric or letter grades.
			F = grade is a code and is verified from UCXDW385.
			R = grade is number within range as defined in UCX-STU385.)
Reserved	1	2	Reserved for future use.
Description	30	3	The normal length description of the Grade Type Code.
Reserved	137	33	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU385 Grade Table

Updated: March 25, 2022

This table is used to validate the actual grades that are entered into a student's record.

Each entry contains a number of flags that assist programs in processing the grade information. It is necessary to couple the Grade with the School (UCX-STU350) and Grade Type (UCX-STU356) to validate the grade.

UCX KEY	Len	Description
School Code	12	The School Code (From UCX-STU350).
Grade Type	6	The Grade Type (From UCX-STU356).
Grade	6	The Final Grade.
Reserved	6	Reserved for future use.

UCX VALUE	Len	Start	Description
Numeric Grade	7	1	The numeric representation of the grade in the format 9v999. Also grade points assigned to grade. For example, "0004000"

UCX VALUE	Len	Start	Description
			for an A grade, "0003300" for a B+ grade, "0002000" for a C grade, etc. This field is not used by the extract/bridge process. This field is used by the GPA Calculator if the Use in GPA Calculators flag below is enabled. It is also used by Transfer Equivalency.
Reserved	4	8	Reserved for future use.
Graded Attempted	1	12	A flag which indicates whether the credit will be added to the student's total graded credits attempted (Y/N). Used by Transfer Equivalency.
			Also used by the Advice Calculator. Set to N to exclude grade from Advice Calculator. Note that the grade will still be used in the Term Calculator if "Use in GPA Calculators" flag is Y.
Reserved	71	13	Reserved for future use.
Incomplete (Transfer Equivalency Only)	1	84	A flag which indicates whether or not the grade is an incomplete grade (Y/N). Default is set to "N".
Reserved	2	85	Reserved for future use.
Use in GPA Calculators (Web)	1	87	Y/N flag. Indicates if the Grade should be used in the GPA Calculator. Y = use, N or blank = do not use.
Reserved	13	88	Reserved for future use.
Overrides Allowed	1	101	Y/N flag. Used by the Banner student extract.
			For current classes found in SFRSTCR, historical classes found in SHRTCKN and transfer classes found in SHRTRCE. If "Y" and any of the 8 "Override" grade values listed below are NOT blank, the associated standard current/historical/transfer grade value will be overridden on the rad_class_dtl (current/historical classes) or the rad_transfer_dtl (transfer classes).
Override Audit value	1	102	If the "Overrides Allowed" flag above is "Y" and this Override Audit value is NOT BLANK it will be loaded into the rad_audit_flag.

UCX VALUE	Len	Start	Description
			Banner: On the rad_class_dtl (SFRSTCR/SHRTCKN) or rad_transfer_dtl (SHTRCE).
Override Insufficient value	1	103	If the “Overrides Allowed” flag above is “Y” and this Override Insufficient value is NOT BLANK it will be loaded into the rad_insuff_flag.
			Banner: On the rad_class_dtl (SFRSTCR/SHRTCKN) or rad_transfer_dtl (SHTRCE).
Override In-Progress value	1	104	Y/N flag.
			Banner: Extracts current classes found in SFRSTCR and historical classes found in SHRTCKN. The default In-Progress code for a historical class is ‘N’. However, if this flag is set to ‘Y’ for the School/GradeType/Grade combination then the rad_inprog_flag on the rad_class_dtl will be set to ‘Y’. This flag ONLY needs to be set for those special historical classes that for some reason must be considered In-Progress by Degree Works.
Override Withdraw value	1	105	If the “Overrides Allowed” flag above is “Y” and this Override Withdraw value is NOT BLANK it will be loaded into the rad_withdw_flag.
			Banner: On the rad_class_dtl (SFRSTCR/SHRTCKN) or rad_transfer_dtl (SHTRCE).
Override Incomplete value	1	106	If the “Overrides Allowed” flag above is “Y” and this Override Incomplete value is NOT BLANK it will be loaded into the rad_incomp_flag.
			Banner: On the rad_class_dtl (SFRSTCR/SHRTCKN) or rad_transfer_dtl (SHTRCE).
Override Pass Flag value	1	107	If the “Overrides Allowed” flag above is “Y” and this Override Pass Flag value is NOT BLANK it will be loaded into the rad_pass_flag.

UCX VALUE	Len	Start	Description
			Banner: On the rad_class_dtl (SFRSTCR/ SHRTCKN) or rad_transfer_dtl (SHRTRCE).
Override Pass Fail value	1	108	If the “Overrides Allowed” flag above is “Y” and this Override Pass Fail flag value is NOT BLANK it will be loaded into the rad_pass_fail flag.
			Banner: On the rad_class_dtl (SFRSTCR/ SHRTCKN) or rad_transfer_dtl (SHRTRCE).
Override Final Grade Number value	4	109	If the “Overrides Allowed” flag above is “Y” and this Override Final Grade Number is NOT BLANK it will be loaded into the rad_final_gr_num value. For a grade of 2 use “2” for a grade of 2.5 use “2.5” with the decimal.
			Banner: On the rad_class_dtl (SFRSTCR/ SHRTCKN) or rad_transfer_dtl (SHRTRCE).
Transfer Repeat	1	116	BANNER ONLY. Y/N flag. Used ONLY by the Banner Bridge extract to load a repeat policy for a transfer class found in the Banner SHRTRCE table that has been repeated if a match on the UCX-STU385 key combination (school = SHRTRCE_LEVL_CODE, grade type = SHRTRCE_GMOD_CODE and grade = SHRTRCE_GRDE_CODE) is found in UCX-STU385 for a given transfer class
Transfer Repeat Skip	1	117	BANNER ONLY. Y/N flag. If the Transfer Repeat flag above is “Y” and this Transfer Repeat Skip is a “Y” the transfer class will NOT be loaded into the rad_transfer_dtl.
Transfer Repeat Policy	1	118	BANNER ONLY. If the Transfer Repeat flag above is “Y”, the Transfer Repeat Skip above is “N” and this Transfer Repeat Policy is NOT BLANK it will be loaded into the rad_transfer_dtl rad_repeat_plcy. The discipline (SHRTRCE_SUBJ_CODE) and course number (SHRTRCE_CRSE_NUM) will be loaded into the rad_repeat_ptr.
Show in SEP Picklist	1	119	Y=Show in Student Educational Planner picklists

UCX VALUE	Len	Start	Description
Override Transfer Grade	6	120	Used by the Banner student extracts.
			Banner: If the “Overrides Allowed” flag above is “Y” and CFG020-BANNER “Load SHRTRCR Grade” and “Override SHRTRCR Grade” flags are both set to “Y” then you may override the rad_final_grade in rad_transfer_dtl with the value in this field. Note that the STU385 key must be composed as follows:
			School – SHRTRCE_LEVEL padded to 12 bytes
			Grade Type – SHRTRCE_GMOD_CODE padded to 2 bytes
			Grade – SHRTRCR_TRANS_GRADE.
Transfer Finder Grade	6	126	Contains the Transfer Grade Mapping value from STU398. This value is only used by the Transfer Finder application.
Reserved	38	132	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU398 Transfer Grade Mapping

Updated: March 25, 2022

This table is used to map grades from transfer institutions to grades valid at your institution.

A valid school and grade type must also be input on the Transfer Equivalency transfer records that are to be used in grade processing as these fields are used along with the final grade to look up the appropriate UCX-STU385 record. The Articulate Flag must be set to “Y” for transfer classes with this grade to be articulated in Transfer Equivalency.

UCX KEY	Len	Description
Transfer Grade	6	The grade from the transfer institution.
Reserved	24	Reserved for future use.

UCX VALUE	Len	Start	Description
Resident Grade	6	1	The grade used in your student system.
Articulate	1	7	Y=this grade is valid for articulation; N=transfer classes with this grade cannot be articulated
Show in Transfer Equivalency Self-Service	1	8	Y/N. Y = Include in Transfer Equivalency Self-Service picklists. N = Exclude from Transfer Equivalency Self-Service picklists.
Numeric Grade	7	9	Letter grade in the format 9999v999. A grade of 3.5 is recorded as "0003500", for example. This field is currently only used for sorting the grade in dropdowns in Transfer Finder.
Reserved	154	16	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU560 College Codes

Updated: March 25, 2022

This table is used to validate the college code. The college code indicates a particular college within a school, such as Arts and Letters, Science, Engineering, Business, and so on.

UCX KEY		Description	
Transfer Grade		The grade from the transfer institution.	
Reserved		Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	30	1	The normal length description of the College Code.

UCX VALUE	Len	Start	Description
Show in What-if Picklist	1	31	Y/N. Y = Include in What-if picklists. N = Exclude from What-if picklists.
			Note, when curriculum rules are being used to filter the What-If display in Transfer Equivalency Self-Service it is this flag that is used to filter the drop-down list and not the flag below.
Show in SEP Picklist	1	32	Y=Show in Student Educational Planner picklists.
Show in Transfer Equivalency Self-Service Picklist	1	33	Y/N. Y = Include in Transfer Equivalency Self-Service picklists. N = Exclude from Transfer Equivalency Self-Service picklists.
Show in Advanced Search Picklist	1	34	Y/N. Y = Include in Advanced Search dropdown. N = Exclude from Advanced Search dropdown.
Reserved	135	35	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-STU563 Concentration Codes

Updated: March 25, 2022

This table is used to validate Concentration Codes. A concentration is usually a focused subject area of academic study.

UCX KEY	Len	Description	
Concentration	12	Concentration Code	
Reserved	18	Reserved for future use.	
UCX VALUE	Len	Start	Description
Description	50	1	The normal length description of the Concentration Code.
Show in What-if Picklist	1	51	<p>Y/N.</p> <p>Y = Include in What-if picklists.</p> <p>N = Exclude from What-if picklists.</p> <p>Note, when curriculum rules are being used to filter the What-If display in Transfer Equivalency Self-Service it is this flag that is used to filter the drop-down list and not the flag below.</p>
Show in SEP Picklist	1	52	Y=Show in Student Educational Planner picklists
Show in Transfer Equivalency Self-Service Picklist	1	53	<p>Y/N.</p> <p>Y = Include in Transfer Equivalency Self-Service picklists.</p> <p>N = Exclude from Transfer Equivalency Self-Service picklists.</p>
Major Filter 1	12	54	These four filter fields contain the majors for which this concentration is valid. Used if core.whatif.filterConcentrationByMajor is enabled.
Major Filter 2	12	66	2 nd major
Major Filter 3	12	78	3 rd major
Major Filter 4	12	90	4 th major
Show in Advanced Search Picklist	1	102	<p>Y/N. Y = Include in Advanced Search dropdown.</p> <p>N = Exclude from Advanced Search dropdown.</p>
Reserved	67	103	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.

UCX VALUE	Len	Start	Description
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Add additional major filters

For major-drives-conc (core.whatIf.filterConcentrationByMajor setting), if more than four Major Filters are required, additional records can be created for a concentration, allowing additional filters to be added. The additional records would have an _0, _1, _2 ... _9 suffix appended to the key for up to 40 additional majors. The suffix is an underscore followed by a single digit.

For example, if seven Major Filters were required for the African History concentration:

1. Using Controller, copy “HSAF” record to “HSAF_1” in UCX-STU563.
2. Change description on HSAF_1 record to say “African History concentration – CONTINUATION”.

However, make sure the descriptions for this concentration sort together. That is, make sure that the description for some other concentration does not come between these two descriptions alphabetically. For this reason, you may need to add several spaces before putting “- CONTINUATION”. You can use “(cont)” or anything else to help ensure that the second record is a continuation of the first.

3. Add the three additional major filters to the HSAF_1 record.

This applies only to the regular What-If page and SEP4’s What-If page in the Responsive Dashboard.

UCX-STU576 Campus or Location Codes

Updated: March 25, 2022

This table is used to validate the Location Code. The Location Code indicates the location or remote campus of the class.

Location is also known as Campus.

UCX KEY	Len	Description
Location Code	2	The Location Code; limited to 2 bytes.
Reserved	28	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	The normal length description of the Location Code.
Show in SEP Picklist	1	31	Y=Show in Student Educational Planner picklists.

UCX VALUE	Len	Start	Description
Show in Transfer Equivalency Self-Service Picklist	1	32	Y=Show in Transfer Equivalency Self-Service
Show in What-if Picklist	1	33	Y/N. Y = Include in What-if picklists. N = Exclude from What-if picklists.
			Note, when curriculum rules are being used to filter the What-If display in Transfer Equivalency Self-Service it is this flag that is used to filter the drop-down list and not the flag above.
Reserved	137	34	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SYS001 Table of Contents

Updated: March 25, 2022

This table lists all the UCX tables and serves as the table of contents for the UCX.

UCX KEY		Description	
UCX Table		UCX table number.	
Reserved		Reserved for future use.	
UCX VALUE	Len	Start	Description
Reserved	4	1	Reserved for future use.
Description	80	5	Title of UCX table.
Reserved	7	85	Reserved for future use.
Cache Stamp	4	92	It is a 4 digit numeric used internally by ucx81. It should not be changed by the institution.

UCX VALUE	Len	Start	Description
Large Table	1	96	A Y/N flag indicating that the size of the UCX table is Large.
Shift Type	1	97	A flag that controls the case of the user's key entry. D = Downshift, U = Upshift, or Blank.
Reserved	3	98	Reserved for future use.
Different Layouts	1	101	A Y/N flag that indicates whether or not the table has different record layouts for different key structures. A "Y" indicates that the table has more than one record structure depending on the key. For example, UCX-CFG020 has a different record layout for each key.
Reserved	68	102	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SYS098 Error or Status Messages

Updated: March 25, 2022

Error Numbers are defined in each Degree Works program. The first two digits of the error number indicate the program that created the error message (for example, "4002" is from dap40).

Warning! Do not modify.

UCX KEY	Len	Description
Error Number	4	All UCX-SYS098 Error Numbers must be unique.
Reserved	26	Reserved for future use.

UCX VALUE	Len	Start	Description
Module	5	1	Name of the program/subroutine issuing the error.
What Happened?	80	6	A description of "What Happened".
Action Message	80	86	A message to the user indicating what action should be taken to fix this problem.

UCX VALUE	Len	Start	Description
Reserved	4	166	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SYS100 PDF Audit Page Dimensions

Updated: March 25, 2022

This table defines the page dimensions that are available in the drop-down lists in Responsive Dashboard and Transit when generating PDF audits. Ellucian provides several sample entries, but you can delete them if your site does not need them. Additionally, you can create new options as needed.

UCX KEY		Description	
Page Dimension Key		The page dimension code. Can be up to 30 characters with no spaces. Underscores are allowed. For example, "LETTER_PORTAIT".	
UCX VALUE	Len	Start	Description
Description	30	1	The page dimension description that will display in the user interfaces.
Measurement units	2	31	The unit of measurement for the page width and height. The options are: IN = inches MM = millimeters
Page width	10	33	The width of the PDF audit page dimension. Should be a number but can include a decimal. Format is 9999999V999. For example, "8.5".
Page height	10	43	The height of the PDF audit page dimension. Should be a number but can include a decimal. Format is 9999999V999. For example, "8.5".
Reserved	116	53	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.

UCX VALUE	Len	Start	Description
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SYS933 Data Names

Updated: March 25, 2022

UCX-SYS933 contains one record for each data item from the RAD/DAP database to be used in a Degree Works script or SHPCFG rule.

It provides a convenient way to use easily recognizable and mnemonic codes to access a particular data element. If an institution wants to construct scripts of SHPCFG rules that necessitate evaluation of data from the database then these data names must be added to UCX_SYS933 before those scripts and rules can be written.

For example, special treatment of athletic coaches may require that the SHPCFG grant certain keys to all coaches for access to specialized services. In order to construct the SHPCFG rule, "coach" must be added as a data name to UCX-SYS933. The shpparse tool uses this table to ascertain which pieces of data from Degree Works should be used in evaluation of the SHPCFG file and Degree Works scripts.

For each entry in UCX-SYS933, the Literal and Data Element must be entered in the UCX-VALUE. The rest of the UCX-VALUE is optional and is typically used only when retrieving data from a detail table (where more than one record can exist per student, for example, custom records or test scores). These fields indicate which records from the table to use and which to skip. Only records that pass the edits are used as a source of data.

For example, if you want a test score from the many stored in the database, but you only want the one for test code "GRE" then use test score (1292) as the Data Element, test code (1291) as the Edit Element, and "GRE" as the Edit Data.

UCX KEY	Len	Description
Data Name	12	A code that gives a name to a data element from UCX-SYS999.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	Free-text description.
Element	04	31	Element number from UCX-SYS999 indicating the data source.
UCX Table	03	35	UCX table number in which this element is validated. If the element is not validated in a UCX table (i.e. the data is a date,

UCX VALUE	Len	Start	Description
			number, or free-text) then leave UCX Table blank.
Edit Element 1	04	38	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the data should be extracted from this detail. This element number must be from the same dataset as the Data Element.
Edit Type 1	02	42	The type of check to do on the data in Edit Element 1. EV or blanks = use the value in Edit Value 1.
Edit Data 1	12	44	The specific data value against which the data in Edit Element 1 should be checked. The record is skipped if it does not match.
Edit Element 2	04	56	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the Data Element data should be extracted from this detail. This element number must be from the same dataset as the Data Element.
Edit Type 2	02	60	The type of check to do on the data in Edit Element 2. EV or blanks = use the value in Edit Value 2.
Edit Data 2	12	62	The specific data value against which the data in Edit Element 2 should be checked. The record is skipped if it does not match.
Edit Element 3	04	74	Element number from UCX-SYS999 indicating the data field to evaluate when determining if the Data Element data should be extracted from this detail. This element number must be from the same dataset as the Data Element.
Edit Type 3	02	78	The type of check to do on the data in Edit Element 3. EV or blanks = use the value in Edit Value 3.
Edit Data 3	12	80	The specific data value against which the data in Edit Element 3 should be checked. The record is skipped if it does not match.
Reserved	78	92	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.

UCX VALUE	Len	Start	Description
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SYS935 Web Server Function Buffers

Updated: September 29, 2023

UCX-SYS935 contains the layout for buffers used within Degree Works.

Warning! Do not modify.

It is used to format the data coming from the web page into a layout specific to the service requested. It is also used to take a formatted buffer and create a name-value pair string. The WEB84 subroutine reads this table and formats a buffer for the caller.

This table describes the buffers used by Degree Works programs. Altering this table in any way may prevent Degree Works services from working correctly. Only Ellucian may alter this table.

Most name-value pairs contain only one occurrence of each name. If a particular name for a buffer is not found in a name-value pair string then the bytes in the calling program's buffer are left unaltered, that is, they are not space filled. If two names exist in a string then WEB84 finds the first name and ignores the second. The Multiple Records flag may, however, indicate that WEB84 should parse for multiple occurrences of a particular name. The calling program should prepare an array of values to be returned in this case.

The Shift Flag is only used when values are pulled from the name-value pair string to create the formatted buffer. Values can be upshifted, downshifted, or left as-is.

The Crypt flag can force values that are pulled from the name-value pair string to be decrypted - the assumption is that the value is already in the encrypted form. The Crypt flag can also force values to be encrypted when a name-value pair string is created from a formatted buffer.

UCX KEY	Len	Description
Buffer	08	Name of buffer to be formatted.
Sequence	03	001-999
Reserved	19	Reserved for future use.

UCX VALUE	Len	Start	Description
Data Name	12	1	Dataname used in HTML query string
Offset	04	13	Offset in buffer sent to WEB84.
Length	04	17	Maximum length of data in buffer.
Type	04	21	SUPR or blank. "SUPR" = takes all query string items that are "Ennnn" and formats them as SUPRGET criteria. If Multiple

UCX VALUE	Len	Start	Description
			Occurrences field is Y then this field contains maximum occurrences.
Description	30	25	Item description.
Shift	01	55	U means upshift the value and D means downshift the value. N means leave as is.
Multiple Records	01	56	Y = Scan for multiple occurrences of this name; maximum occurrences found in Maximum field.
			R = follow the Record layout
Crypt	01	57	Y = Encrypt value when creating name-value string and decrypt value when pulling from name-value string.
Critical	01	58	Y = Critical item.
Element	04	59	Element number.
Protect	01	63	Y = protect, N = do not protect.
Mark Missing	01	64	Y = Buffer gets marked with ~ for each missing character.; N=missing characters are not marked
Reserved	105	65	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SYS998 System Standard Error Messages

Updated: September 29, 2023

For every error number that can be generated in the software, this table defines the error message associated with that error number.

When an error is detected by one of the data entry programs, an error number is generated and the first message that is defined here with that error number is written to the display screen. Thus, only the message that has a Sequence Number of 01 in its UCX-Key can actually be written to the screen. However, additional information regarding an error can be stored in this UCX Table under Sequence Numbers 02-99, for the benefit of users who look up the error in the UCX table.

Warning! Do not modify.

UCX KEY	Len	Description	
Error #	4	0001-9999	
Sequence #	2	01-99	
Reserved	24	Reserved for future use.	
UCX VALUE	Len	Start	Description
Reserved 1	1	1	Blank.
Type	5	2	"Error" or "Warn".
Reserved 2	1	7	Blank.
Number	4	8	Same as Error # in key.
Hyphen	3	12	" - "
Message	53	15	Free text message.
Reference	10	68	"Ref UCX####" referral to UCX table for codes.
Bell	1	78	Ctrl-G.
Reserved	91	79	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-SYS999 Data Dictionary

Updated: March 25, 2022

This table documents every data element in the software, including data element number (E#), data base and data set, data length, data type, initial value, if any, and UCX validation table, if any.

Warning! Do not modify.

Most of the elements listed in UCX-SYS999 are items in a database. However, some elements are "dummy" elements; that is they do not represent one particular database item. Dummy elements are used to represent such things as pieces of a database item or composites of database items.

UCX KEY	Len	Description
Element #	4	An E# in the range of 0001 to 9999 where ranges are based on a Degree Works module. Some element numbers may also start with alpha character (with 3 digits following).

UCX KEY	Len	Description	
Reserved	26	Reserved for future use.	
UCX VALUE	Len	Start	Description
Base Name	16	1	database (table) name (must end with a ;).
Dataset Name	16	17	data set (column) name (must end with a ;).
Data Element	17	33	Data item name (must end with a ;).
Description	20	50	Free-text description.
Initial Value	12	70	Default value.
Data Type	1	82	"X" = alphanumeric, "9" = numeric.
Data Length	3	83	Number of bytes: 001-999.
UCX Code	6	86	The number of the UCX Table which validates this element (UCXSYS001), if any. This field must contain the UCX table number if you expect to use the WINGSPAN UCX drop-down list box for this element.
UCX Mask	2	92	Starting position of literal in the UCX Table: 01-99. This value must be accurate to ensure that the WINGSPAN UCX drop-down list box displays a meaningful literal for the codes associated with this element.
UCX Length	2	94	Length of literal in the UCX Table: 01-99. This value must be accurate to ensure that the WINGSPAN UCX drop-down list box displays a meaningful literal for the codes associated with this element.
Search Item	1	96	"K" = IMAGE search key, Blank = not a key.
Dataset Type	1	97	"M" = master data set, "D" = detail data set.
Optimize	1	98	A Y/N flag.
Mask	4	99	The starting location of a data element within its table.
Reserved	1	103	Reserved for future use.
Edit Type	4	104	Describes how the element should be edited. It is used in limited situations by select programs. Refer to the individual program to determine the appropriate value for this field.
Error Number	4	108	The Error number from UCX-SYS998 that indicates an non-valid entry for this element.
Reserved	58	112	Reserved for future use.

UCX VALUE	Len	Start	Description
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-TRF100 Term Global Mapping Codes

Updated: March 25, 2022

This table stores the common term codes used by the system and is used to map the global codes used system-wide to the local school's codes.

It is typically updated automatically from the system's central code definition tables, and the local school needs to update the Local Term for each entry with the corresponding code from their STU016. Additionally, the local Term Code must be mapped to the corresponding global term in STU016.

UCX KEY	Len	Description
Term Code	12	The Code used in manual entry to indicate when the class was taken
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	The normal length literal associated with the term.
Reserved	127	31	Reserved for future use.
Local Term	12	158	The term code from UCX STU016.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-TRQ060 TreQ Favorite Schools

Updated: March 25, 2022

This table is used by Transfer Equivalency's Mapping function. The list of favorite transfer schools gives the user an easy way to get at the schools for which mappings are edited most often.

UCX KEY	Len	Description	
School ID	10	School ID from ETS-MST	
Reserved	20	Reserved for future use.	
UCX VALUE	Len	Start	Description
Transfer School Name	30	1	Transfer school name
Calendar Code	2	31	Calendar code from UCX-STU346. For example, if "S" is used in UCX-STU346 to refer to Semester Hours and this school's calendar is on a semester basis then the Calendar Code would be "S".
School URL	50	33	Transfer school's URL to their web site (for example, http://somecollege.edu). This is used when showing mappings in Course Link only if the UCX-RPT056 Link flag is set to Y. If this School URL field is blank, Course Link will use Google to try to find the school's web site.
Reserved	87	83	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-TRQ061 Transfer Mapping Conditions

Updated: March 25, 2022

This table is used in Transfer Equivalency and Transfer Finder when performing an articulation on a transfer student.

The condition names point to a data item in the dap_applicant_mst or dap_appdata_dtl. These condition names are specified in the Transfer Condition section on the Manage Mapping page of Transfer Equivalency Admin.

UCX KEY	Len	Description
Condition Name	12	Condition name that is placed on the dap_applicant_mst or dap_appdata_dtl. The

UCX KEY	Len	Description
		condition is used when a student's transfer classes are matched to mappings for the transfer school.
Reserved	18	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	50	1	Description of Condition Name.
Reserved	119	51	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

DEGREE, SCHOOL and APPCODE* are on the dap_applicant_dtl while the others come from the dap_appdata_dtl.

In Transfer Finder audits, mappings using As Long As conditions will be resolved only for the following TRQ061 codes: Degree, School, Major, Minor and Concentration. Any other TRQ061 code in ALA conditions will automatically skip the mapping.

The standard Mapping Condition codes supported by Degree Works are listed below. These codes are tied to specific applicant goal data the student has. These codes should not be deleted, but additional codes may be created. The articulation engine will look for a dap_appdata_dtl with that goal code for the student.

Key	Description
DEGREE	Student's intended Degree
SCHOOL	Student's intended School
COLLEGE	Student's intended College
MAJOR	Student's intended Major
MINOR	Student's intended Minor
PROGRAM	Student's intended Program
CONC	Student's intended Concentration
LIBL	Student's intended Liberal Learning
SPEC	Student's intended Specialization
APPCOND	Any Applicant Condition
APPCOND1	Applicant Condition 1
APPCOND2	Applicant Condition 2
APPCOND3	Applicant Condition 3
APPCOND4	Applicant Condition 4

Key	Description
APPCOND5	Applicant Condition 5
APPCOND6	Applicant Condition 6
APPCOND7	Applicant Condition 7
APPCOND8	Applicant Condition 8
APPCOND9	Applicant Condition 9
APPCOND10	Applicant Condition 10

UCX-TRQ062 Applicant TreQ Status

Updated: September 29, 2023

This table is used by Transfer Equivalency to track an applicant's progress through the transfer articulation process.

Warning! This table is constructed by Ellucian and institutions should not create their own. For this particular table, you can change the Description, but do not modify anything else unless instructed by Ellucian.

UCX KEY	Len	Description
TreQ Status	2	The Treq-Status on the dap_applicant_mst. Possible values are: AR - Articulated – Resolved NR - Not Resolved RO - Rolled to Student System TR - Transcript Entry

Reserved	28	Reserved for future use.
----------	----	--------------------------

UCX VALUE	Len	Start	Description
Description	30	1	The description that appears in Transfer Equivalency based on the status code.
Reserved	139	31	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.

UCX VALUE	Len	Start	Description
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-TRQ063 Transfer Articulation Status

Updated: September 29, 2023

This table is used by Transfer Equivalency to track the status of an applicant's transfer class. The status indicates the status of the transfer class after an articulation occurs and after the class has or has not been rolled to the student system.

Warning! This table is constructed by Ellucian and institutions should not create their own. For this particular table, you can change the Description, but do not modify anything else unless instructed by Ellucian.

UCX KEY	Len	Description	
Articulation Status	2	Status of transfer class in the articulation process.	
Reserved	28	Reserved for future use.	

UCX VALUE	Len	Start	Description
Description	30	1	A description of the articulation status.
Reserved	139	31	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

UCX-TRQ065 Transfer Articulation Codes

Updated: March 25, 2022

This table is used by Transfer Equivalency to track how the transfer course was articulated or how mappings should be articulated. For example, 1TO1 is one-to-one mapping which will articulate one transfer course to one native course.

Warning! This table is constructed by Ellucian and institutions should not create their own. For this particular table, you can change the Description, but do not modify anything else unless instructed by Ellucian.

UCX KEY	Len	Description
Articulation Code	4	Code indicating how the transfer course was articulated by Transfer Equivalency.
Reserved	26	Reserved for future use.

UCX VALUE	Len	Start	Description
Description	30	1	A description of the articulation code.
Reserved	139	31	Reserved for future use.
Status	1	170	Y = Active, N = Inactive, W = Issue Warning message for override.
Reserved	30	171	Reserved for future use.
Revision	6	201	Revision code
Who	8	207	User who last updated this record.
TimeStamp	6	215	CHRONOS stamp of last update.

Transfer Equivalency uses the following articulation codes:

1TO1	One-to-one
1TOM	One-to-many
2LOW	Grade/score not acceptable
ACYR	Applicant catalog year invalidates mapping
CCYR	Class catalog year invalidates mapping
COND	As-long-as condition not met
CRMN	Credit minimum not met
CRMX	Credit maximum not met
GRMN	Grade minimum not met
GRMX	Grade maximum not met
MTO1	Many-to-one
MTOM	Many-to-many
NOMP	No mapping found
NONE	No articulation
SRMN	Minimum test score not met
SRMX	Maximum test score not met
YAGO	Class outside of year-ago restriction

Technical Considerations

Updated: March 25, 2022

Review technical reference information as you work with Degree Works.

Parser Engine (DAP13)

Updated: March 25, 2022

The Parser Engine translates the requirements language into a format to be used for audit.

A Degree Works language consisting of certain keywords and syntactical rules is used to define institutional requirements into the computer. The institutional requirements are entered using Scribe, resulting in a series of requirement blocks. These blocks are then read by the Parser Engine. The Parser Engine validates the requirement blocks, assuring they are lexically and syntactically correct so that they may be properly interpreted by the Auditor Engine.

The Parser Engine is a program called DAPPARSE (DAP13) with the following characteristics:

- Resides on the host computer
- Accepts files of requirements (created with Scribe)
- Parses requirements, that is, translates the rules into syntax and remarks files
- Catches errors by validating disciplines and courses
- Returns error files if any errors are encountered during the parse
- Stores the syntax and remarks files on the host computer
- The parsed text is saved in the DAP database on the host computer

Scribe is the input mechanism for entering degree requirements. You can either enter the requirements yourself or contract with Ellucian to enter them for you. In either case, you must gather together the course catalogs and other documents that define your institution's degree requirements.

Requirement blocks

Degree Works stores requirements in requirement blocks. 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 (e.g., community service). The blocks will not necessarily be hierarchical, and links among blocks may be established but are not required. The order of the blocks and the linkages between the blocks are determined by the user. 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, or by client-defined data on the student record to indicate special activities or programs (e.g., ROTC). Blocks may be global as well (e.g., general education), so a link to student data may not be required.

Users are able to define the relationships among requirement blocks. Some will be sequential and hierarchical; some will be linked one to another (e.g., a concentration that cannot exist without an associated major); some will be conditional on student data that will determine blocks and point to other blocks (e.g., ROTC); and some will be isolated and required of all students regardless of their characteristics (e.g., a requirement that all students perform some kind of community service).

The requirement blocks may be defined for student attributes which may include school, college, department, major, minor, concentration, location, catalog year, and other client-defined characteristics. Custom sets of requirements by student may also be defined.

Unlimited courses are allowed within requirement blocks and unlimited nesting of requirements is allowed within a block.

The Scribe language for defining requirements lets users specify requirements by number of classes, by number of credits, or by a combination of both.

The definition of courses that can be taken to fulfill a requirement rule allows for the robust use of the wildcard (@) and also for use of a range operator (:) for course numbers. The wildcard (@) indicates one or more occurrences of any character.

Example: MATH@ = all classes in the discipline MATH

Example: MATH1@ = all classes in the discipline MATH where the first digit of the course number is a "1"

Example: MATH100:199 indicates all courses from MATH100 through MATH199

The definition of that portion of the course key that indicates the course discipline may require the use of a delimiter. If no course delimiter is specified then Degree Works assumes discipline is an alpha code.

Example: ART1200 = ART

Example: ART1 200 = ART1

Example: ARTA200 = ARTA

Example: ART A200 = ART

Degree Works allows for non-course requirements. These requirements may include theses, performances, attendance at required events, work experiences and should be allowed individually, in groups by student characteristics, or by requirement block.

There is an unlimited text capability for annotating the requirement blocks. This text can be used, in whole or part, to provide readable reports where requirements are described. The text may include comments internal to the requirement block or may be text for use on user reports.

Degree Works provides a mechanism for pointing multiple catalogs to the same requirement block to avoid duplicating blocks when requirements do not change. This is accommodated by identifying the catalog year as a range of begin/end years rather than as a single year.

Auditor Engine (DAP14)

Updated: March 25, 2022

The Auditor Engine reads the student's academic data, assembles the requirement blocks, performs the audit, and stores the audit results in the DAP database.

The Auditor Engine reconciles student academic data from the student information system with the requirement blocks that have been built by users and then validated by the Parser Engine. The Auditor Engine actually evaluates the student course data against the appropriate requirement blocks, determining which academic requirements have been satisfied and which await completion. The audit results are stored in a database for reporting and review by the institutional staff.

The Auditor Engine is a program called DAPAUDIT (DAP14) with the following characteristics:

- Resides on the host computer
- Accepts student data from the host (degree data, class data)
- Accepts "what-if" student data from the end-user
- Selects requirement blocks to be audited based on the degree data
- Processes the audit using the audit algorithm and site-defined configuration settings
- Stores the audit results in the Degree Works database on the host computer

The following student data is passed to Degree Works from the student database. These elements constitute the minimum data needed by the Auditor Engine but additional "custom" data may also be passed to the Auditor for evaluation or subsequent inclusion on audit reports.

From the student course record:

```
ID  
school  
college  
course discipline  
course number  
credits  
grade (letter and number)  
grade points (grade number * credits)  
term  
course type (resident, transfer)  
grade type (regular, pass/fail)  
credits type (e.g., academic, clep, ap)  
location (site of class)  
transfer xref (local course equivalence)  
class status (added, dropped, withdrawn, repeated)
```

From the student academic record:

```
ID  
school  
degree(s)  
    catalog year  
program(s)  
    catalog year  
college(s)  
    catalog year  
major(s)  
    catalog year  
minor(s)  
    catalog year  
concentration(s)  
    catalog year  
specialization(s)  
    catalog year  
liberal learning(s)  
    catalog year  
non-course requirement data  
    (e.g., exams, performance, language, comments)  
client-defined data (e.g., ROTC, religion)
```

From the course catalog record:

```
course discipline  
course number  
catalog year  
course equivalents
```

Student data drives the Auditor Engine. There are standard characteristics that would be expected from any student database including degree, major(s), concentration(s), minor(s), catalog year(s). There may be client-specific characteristics that also drive the Auditor Engine, including (but not limited to) ROTC and religion.

Requirement blocks are assembled for processing by searching for blocks that match student attributes, looking first for the degree block. Blocks are matched first by ID. If no blocks matched by ID, other student attributes are combined to find blocks with matching attributes. Configuration settings control the processing of the requirements.

The Auditor Engine matches all resident courses, transfer courses, and non-course activities to the requirement blocks that have been assembled. Resident courses can be evaluated in the context of the selected course catalog, taking into account changes in course identifiers (e.g., recycled courses/AKA's by catalog year). Courses may be excluded from analysis by the Auditor Engine based on credit type or grade type, (e.g., academic bankruptcy classes or classes that do not carry academic credits).

Degree Works can handle transfer courses that do not map directly to a resident course but do map to a range of courses or to any course in a discipline. Direct equivalencies are not required, for example, transfer courses that are different in credit value, courses that are a series of two at one institution and three at another, or situations when course transfer will be allowed but no direct equivalent exists.

Exception handling in Degree Works lets the user lock-in evaluation decisions so that any subsequent running of an audit will not undo them. These decisions might include substitutions of specific courses for specific requirements, waivers of classes, exemptions from certain credits or requirements. Implicit in this is the option to "unlock" such decisions.

Degree Works supports an option to "split" credits from a single course and apply the remaining credits to other requirements. This is done either through requirement definition in Scribe or through a special kind of exception.

It is possible within Degree Works for the user to indicate that a requirement is complete using Exceptions even when the Auditor Engine cannot complete it through.

Degree Works allows variations in the processing of repeated or retaken courses. Courses that may be repeated for credit up to a maximum number of credits or courses must be counted appropriately, and those courses may or may not be limited to one per term. Courses that may not be repeated for credit but are repeated to improve a student's GPA can be identified and the GPA can be calculated correctly. The rules for applying credits from repeated classes and the rules for calculating GPA for repeated classes are site-defined within a set of repeat policies provided by Degree Works.

For purposes of evaluation, the Auditor Engine assumes all courses are applied to requirements in an exclusive manner, i.e., one class and the associated credits apply to one requirement. The nonexclusive application of classes can then be specified with the appropriate limits. The range of users' needs on this issue is great. It includes everything from "you can use anything wherever it fits" to "you can't use anything twice". The middle ground is "no more than 10 credits that applied toward the major requirements can also be applied toward minor requirements."

For the processing of student records against requirement data, the Auditor Engine uses a "best fit" algorithm. To accomplish this, the Auditor Engine may have to perform a number of passes through the course information. For example, on the first pass, all courses that fit a requirement are placed. Classes that can be applied to multiple rules then needed to be weeded out, keeping the "best fit". The fit is made against requirement blocks based on a client-defined order.

To identify those students nearing requirements completion, the Auditor Engine calculates the overall percent complete and percent complete for each requirements block (e.g. Major, Minor)

The Notes capability in Degree Works provides unlimited text capabilities for recording notations on the student's record. This includes, but is not limited to, special circumstances, advisor notes, reasons for exceptions, who made decisions and when.

Grade Point Average is calculated overall and by requirement block. The cumulative GPA calculated by Degree Works may be different from that calculated by the student system due to Degree Works ability to process repeats and courses that exceed limits. Users have the ability to exclude specific courses from block GPA calculations through the NOTGPA reserved word.

GPA calculations

Updated: March 25, 2022

All courses applied to each block are used to calculate the BLOCK GPA.

For the Degree Block (or the "starting" block), the GPA is calculated using all the courses applied to the audit (all sections). The Over-the-Limit section can be included or excluded from this calculation (see CFG020 DAP14 parameter) – but typically the setting is N so that these classes do not count.

The Major and Minor Block GPAs will be calculated using all courses that are or could be applied to this block. Courses that end up in "insufficient" or "fail" sections of the audit can be included or excluded from the calculation (see CFG020 DAP14 parameters for Major and Minor). Classes that could have applied to the Major or Minor block but ended up in the GENED block, for example, will not count in the Major or Minor's GPA calculation – unless the class is being shared between blocks of course.

Transfer courses and GPA calculations

The CFG020 DAP14 parameter has values to control the application of transfer courses to the MINGPA and MINGRADE Scribe Reserved Words and their use in GPA calculations.

Redemption algorithm

Updated: March 25, 2022

After the auditor completes its primary pass through the blocks and rules, it enters one of the last phases of the degree audit process called the Redemption Algorithm.

If the class was removed from a rule by mistake, the algorithm works to make the correction. Either the class was removed from a rule and ended up in the fall-through (electives) section, or it was removed from a rule that is allowed to be shared with another rule which the class is also applying. A class is usually removed from a rule because too many credits/classes are applied to a rule and the auditor needs to reduce the number of classes that are applying. Later on, however, one or more of the classes that were kept on the rule may then have to be removed for other reasons (max qualifiers, class should be applied to another block, etc). It is this issue that the Redemption Algorithm attempts to correct.

Fall-through redemption

Updated: March 25, 2022

When evaluating fall-through classes, the auditor checks all the rules where each class was originally applied at the start of the audit process.

For each class, the auditor first examines fits that were on group rules. When a group rule is found that is not complete, the auditor reapplies this class and all other fall-through classes that also fit on this group rule. The auditor then reevaluates the group rules to check if another group(s) option should be kept instead of the one that was previously chosen.

After the group logic has been performed and if the group rule on which the class was reapplied was not chosen as the group rule to keep, the auditor continues inspecting the other fits for the class. The auditor starts with the best fit for the class and works its way down to the least-best fit –

skipping all fits on group rules. After an incomplete rule has been found, the class is reapplied and no other rules are examined.

Each time a class is placed back on a rule, the Max qualifiers on that rule are reevaluated.

Nonexclusive redemption

Updated: March 25, 2022

After the fall-through classes have been evaluated, the auditor attempts to find classes that are not doing enough sharing based on the Nonexclusive (also referred to as ShareWith) qualifiers found throughout the blocks.

These classes already fit in one or more rules but are possibly allowed to fit in one or more additional rules. For example, the Major rules can share the GenEd block but HIST 1916 was removed from the GenEd for some reason. This step in the redemption algorithm attempts to discover that this sharing is allowed and also attempts to reapply the class to the GenEd block.

For this piece of the algorithm, the auditor steps through each class that is currently applying to at least one rule. The list of the original fits from the start of the audit process is then inspected. This list includes links between fits that indicate if sharing is allowed. The auditor uses the links to check if it can reapply one or more of the fits for the class. When a class is reapplied to a rule, the Max qualifiers on that rule are reevaluated.

After both the Fall-through and Nonexclusive parts of the Redemption algorithm are performed, the auditor performs another check on the header qualifiers in all blocks. In addition, each rule's percent-complete is recalculated.

Too many classes on a rule

Updated: March 25, 2022

At the start of the audit each class is placed on all requirements where the class fits.

The auditor then attempts to figure out the best place to keep the class, removing it from the other requirements. In addition, when the auditor has found that too many classes are on a requirement it steps through a set of decision points to determine which classes to keep and which to remove. Each class is compared to each of the other classes on the rule with the auditor making a decision to keep or remove based on the decision points.

For example, let's say the rule has classes A, B, C, D and E but the rule only needs two of the classes. The auditor first compares A and B. If it turns out that B is the better class, based on the decision points, then A is marked as the one to remove. The auditor then compares A with C. If A is now considered to be the better class then C is marked as the one to remove. This continues on down the line until the last class is reached. At this point the auditor has determined which class is the one to remove. If it was determined that class D was the worst out of the list, for example, then D is removed and the cycle starts again comparing the remaining list of A, B, C and E. Because only two classes are needed on the rule the auditor stops when there are two classes remaining. These two remaining classes are the best classes to keep on the rule.

Below are the decision points the auditor uses when comparing two classes on a rule. The auditor steps through each point finding that one of the classes is better than the other or finding that they are equal for the given point. If they are equal only then does the auditor move to the next decision point.

This is the list of decision points, in order. Based on each comparison, one class is kept and the other is marked for removal. The one that is marked for removal is then compared to the next class on the rule and so on. When a decision is made a code is recorded by the auditor and this code shows up in the Diagnostics Report. This information helps you the user figure out why the auditor made the decisions it did.

- Kept class in the including list (SQ)
- Kept class needed by MinAreas (SQ)
- Kept class needed by MinPerDisc (SQ)
- Kept class needed by MinSpread (SQ)
- Kept class that matches rule exactly (1 Class and 3 Credits in) (SQ)
- Kept class with Apply Here exception (li)
- Kept class that was completed over in-progress class (see UCX-CFG020 DAP14 flag) (inpr)
- Kept class that originally only fit this rule and no other (e1f)
- Kept class that currently fits this rule and no other (1f)
- Kept class that is reapplied here from fall-through through Redemption (fara)
- Kept class based on Decide option - if specified (d45)
- Kept class that has more header MIN qualifiers that may have need this class (HMinQ)
- Kept class based on the fit rank (See Fit Rank section) (firk)
- Kept class that is less likely to be removed because of a header qualifier (dect)
- Kept class with fewer exclusive fits on other rules (exfi)
- Kept class with a higher match level (lvl)
- Kept class based on UCX-CFG020 TIEBREAK (tie)
- Kept class based on coin flip

Match level

Updated: March 25, 2022

The match level is one of the decision points the auditor examines to help determine the best location for the class and also which are the best classes to keep on a rule that has too many classes.

On each requirement, the match level is calculated based on how the requirement is scribed.

- Exact match courses get a match level of 6; example: MATH 123
- Courses with a range get a match level of 5; example: MATH 100:199
- Courses with a wildcard number get a match level of 4: example: MATH 1@

However, these changes are then made to the match level:

- If an Apply Here exception is in place – level is set to 99
- If the course is in an Including list – level is set to 16; example: Including MATH 123
- If the course is in a plus-list list – level is set to 16; example: 2 Classes in MATH 123 + 124
- If the course is the only course listed - level is set to 16; example: 1 Class in MATH 123
- For each HighPriority on the block increase the level by 5
- For each HighPriority on the rule increase the level by 5
- For each LowPriority on the block decrease the level by 5
- For each LowPriority on the rule decrease the level by 5
- If the rule's credit range starts with zero - level is set to 2; example: 0:9 Credits in...
- If the rule has the LowestPriority qualifier - level is set to -9876
- If the rule has a Force Complete exception - level is set to -99

The match level by itself does not determine which classes are removed or kept on rules that have too many credits and it does not determine the requirement on which a class is placed when the class fits on multiple rules, but is one of the decision points used.

Fit rank

Updated: March 25, 2022

The fit rank is used by the auditor to determine where a class should be placed when it fits on multiple requirements.

The auditor uses the decision points below to either increase or decrease the fit rank for a class on a particular requirement. The fits for a class are compared with each other to determine which is the best. A fit with a rank of 1 means it is the best fit; a rank of 2 means it is the 2nd best fit, etc. However, it is possible for two fits to have the same rank - meaning the two fits are equal when it comes to the decision points; neither fit is better than the other. A fit with a higher rank (lower number) will be kept before one with a lower rank (bigger number).

The auditor steps through these decision points in this order. Only when neither fit is determined to better or worse does the auditor proceed to the next decision point.

- Fit is on a rule with a Force Complete exception - worse fit because we can remove the class and the rule will still be complete
- Fit is on a StandAloneBlock - worse fit but only because we know all StandAloneBlock fits will remain so some other fit should be considered a better fit
- Fit has an Apply Here exception - better fit because the class must be applied here no matter what
- Fit is on a plus-list or including-list rule (and not in a group or a wildcard/range w/ multiple fits) - better fit because the rule explicitly says the student must take this class
- Fit is on a rule with another class that only fits here - worse fit because most likely this class will be removed from this rule in favor of the other class
- Fit's rule has a higher priority (per HighPriority, LowPriority, LowestPriority) - better fit because of the priorities specified
- Fit is only class on this rule - and does not have LowPriority - better fit because this rule needs this class; without it, the rule will be 0% complete
- Fit has a higher match level - better fit compared to the levels of other fits (which may be on a range or a wildcard requirement etc)
- Fit is on a group rule that may not be needed - worse fit because by definition a group rule is a list of options and this may not be the best option to take
- Fit is needed on the rule; removal would make rule incomplete - better fit because without this class the rule will not be 100% complete

Group processing

Updated: March 25, 2022

When classes are matching multiple rules within a group the auditor must determine which of the options to keep and which to discard.

The auditor uses the set of decision points below to make this decision. One important thing to keep in mind here is that the auditor does this group processing early on in the audit and thus it is making decisions before sharing has been resolved and before excess classes have been removed from requirements.

Here is the list of decision points to determine which groups to keep and remove:

- Keep the rule if it has an exception
- Keep the rule that has complete qualifiers (removing those rules with incomplete qualifiers)
- Keep the rule if it is more complete than another rule
- Keep the rule that has at least one completed class if multiple rules are incomplete because of in-progress classes

- Keep the rule that has a higher priority
- Keep the rule that is needed to satisfy the group's MinPerDisc qualifier
- Keep the rule whose classes have a higher average fit rank
- Keep the rule with fewer min/max qualifiers (since it a less complicated rule)
- Keep the rule whose classes have fewer fits on other rules (averaged over the number of classes on the rule)

Remove classes when too many fit on a rule

Updated: March 25, 2022

When too many credits/classes are applied to a rule, the auditor needs to figure out which classes to remove from the rule.

Example

3 credits in MATH 101, ENGL 101, PSYC 101, HIST 101

Label INTRO "One introduction-level class";

We will assume:

- These are all 3-credit classes
- The student has taken all 4 of these classes
- PSYC 101 is found out to be the best fit when comparing these four classes for this student

Again, we only need 3 credits. We take these 12 credits and ask, "which is the worst fit"? We start by comparing two classes to one another. The order in which these are applied is not important. For clarity's sake we will assume they are compared in the order Scribed.

MATH 101 is first compared with ENGL 101

We use the "Remove excess classes" decision-process (this is near the bottom of every Diagnostics Report) to compare these two classes. Inevitably, one is considered a "worse fit" than the other. Let's say ENGL 101 is the worse fit. MATH 101 is kept (for now), and ENGL 101 is then compared to PSYC 101. ENGL 101 is still worse than PSYC 101, then class C is kept (for now) and ENGL 101 is compared to HIST 101. Let's say HIST 101 is worse than ENGL 101. Now there are no more classes to compare.

So:

- MATH 101 is better than ENGL 101
- PSYC 101 is better than ENGL 101
- ENGL 101 is better than HIST 101

This means that HIST 101 is the worst. We remove it from the rule. We have not compared HIST 101 to PSYC 101, nor to MATH 101, but because of the transitive property, we can rely on the fact that MATH 101 is better than ENGL 101, and ENGL 101 is better than HIST 101. Therefore MATH 101 must be better than HIST 101.

Getting back to the example, now we have three classes still on the rule because we just removed only HIST 101. That's still too many credits. So we do it again. This time around we determine that:

- MATH 101 is better than ENGL 101
- PSYC 101 is better than ENGL 101

This means that ENGL 101 is the worst. We remove it from the rule.

That leaves us with MATH 101 and PSYC 101, and that's still too many credits. So we do it again. This time, we determine (again, based on the "Remove excess classes" decision-process) that:

- PSYC 101 is a better fit than MATH 101.
- PSYC 101 is better than MATH 101

So – MATH 101 is removed. Notice that this is the first time that MATH 101 was compared to PSYC 101, but because of the work that was done before this we can with confidence conclude that:

Based on the classes that originally fit on this rule, when comparing these classes – PSYC 101 is the best fit.

Note that decisions later in the audit process could impact this rule. Perhaps PSYC 101 fits on a different rule in this block or elsewhere in the audit. At that point (later in the audit process) a decision would have to be made: should PSYC 101 stay here on this "One introduction-level class" rule – or should it apply to a different fit? If it turns out that PYSC 101 should be placed on some other rule, the redemption step of the audit process will come into play reapplying one of those other 101 classes to this rule – but only if they themselves are not applying to some other rule.

Class evaluation on a rule

Updated: September 29, 2023

When too many credits/classes are applied to a rule, the auditor needs to figure out which classes to remove from the rule.

The auditor goes through the following steps to determine which classes to keep and which to remove:

1. Kept class in the including list.

Given a rule such as the following, the auditor will keep the included class because it is required.

5 Classes in MATH ↗ Including MATH 201

-
- 2. Kept class with Apply Here exception.

If a class has an Apply Here exception, it will be kept on a rule before all others.

- 3. Kept class needed by MinAreas.
- 4. Kept class needed by MinPerDisc.
- 5. Kept class needed by MinSpread.

Given a rule such as the following:

2 Classes in MATH @, CHEM @, BIOL @, PHYS @ MinSpread 2

Assuming MATH 101, CHEM 101, and CHEM 102 are on this rule, the auditor will keep the MATH 101 class on the rule because it is needed to satisfy the MinSpread qualifier. The same approach is used for MinAreas and MinPerDisc as well.

- 6. Kept class that matches the rule exactly (1 Class and 3 Credits in).

Given a rule such as the following:

1 Class and 3 Credits in ACCT 2@

Assuming ACCT 201 for 1 credit, ACCT 202 for 2 credits, and ACCT 203 for 3 credits are applied, the auditor will keep ACCT 203 because it matches the requirement of 1 class and 3 credits exactly.

- 7. Kept completed class and discarded in-progress class.

When the CFG020 DAP14 **In-progress vs Completed** flag is set to Y, a completed class is given preference over an in-progress class at this step in the algorithm. This usually works because in-progress classes that end up in fall-through can be reapplied to this rule by the redemption algorithm, if the rule still needs additional classes/credits. If the configuration flag is set to N, this step is skipped. Because setting this flag to Y alters the auditor's best-fit algorithm, it is recommended that this flag be left as N so that this step is skipped. When this flag is set to Y, you are telling the auditor to make a different choice than it would normally make and thus it is unable to apply classes in the most efficient manner.

- 8. Kept class that originally fit only this rule and no other.

If a class can fit only this rule and no other rule, it is kept over another class that has or had multiple possible fits.

- 9. Kept class that currently fits this rule and no other.

If a class did fit on multiple rules at some point, but now fits only this rule, it will be kept over a class that is now still placed on multiple rules.

- 10. Kept class that is reapplied here from fall-through through Redemption.

If a class was removed from all rules and placed in fall-through, but was then placed back onto this rule through redemption, it will be kept.

- 11. Kept class based on Decide option, if specified.

Given a rule such as the following:

2 Classes (Decide = BESTGRADE) in MATH @, CHEM @

The auditor will obey the DECIDE operator when deciding which class to keep if none of the above steps has helped make a decision.

- 12. Kept class based on the fit rank. (See the Fit Rank section in the Diagnostics Report.)

When a class has multiple fits, its fits are ranked from 1-x, where 1 is the best place to keep the class, and x being the least best place to keep the class. So when comparing two classes that fit this rule, the fit rank of each on this rule is considered. If one of them has a fit rank of 2

-
- on this rule and the other has a fit rank of 3 on this rule, the class with a 2 is kept because for that class, this is the better place to be kept.
13. Kept class that is less likely to be removed because of a header qualifier.
If we have a header qualifier such as the following:
MaxClasses 3 in ART 108, 109, 2@
And we have a rule such as the following:
1 Class in ART 108, 123, 145
The auditor sees that ART 108 is part of a MAX qualifier and so has a greater chance of being removed from this rule. The other class (that is not part of any qualifier) is kept on this rule instead.
14. Kept class with fewer exclusive fits on other rules.
Given a rule such as the following:
1 Class in BUS 106, 108;
BUS 106 may fit on two other rules, both of which are not shared with this rule.
BUS 108 may fit on two other rules, but one of them is in a block shared with this block.
This means that BUS 106 has two other exclusive (not shared) fits, while BUS 108 has only one other exclusive fit (with the other one being shared). Thus, BUS 108 will be kept on this rule because there is a greater chance that BUS 108 will end up on this rule because it will either go here or on that other exclusive fit, while there is only a 1-in-3 chance that BUS 106 will end up on this rule.
15. Kept class with a higher match level.
Given a rule such as the following:
1 Class in ANTHRO 266, 277 3@;
Given the choice between keeping ANTRHO 266 or ANTRHO 304 on this rule, the auditor will keep ANTRHO 269 because it is specifically listed. However, if given the choice between ANTHRO 266 and ANTHRO 277, both are specifically listed, but if one of them is in-progress, its match level is actually slightly lower. In this case, the completed class will be kept instead of the in-progress class. In other words, in-progress classes always have a match level lower than they would have if they were completed.
16. Kept class based on CFG020 TIEBREAK.
If all other steps cannot help the auditor make a decision about which class to keep, it uses the CFG020 TIEBREAK setting to figure out which class to keep. Usually this step decides which to keep because one is in-progress and one is completed, or one has a higher grade, or one has a higher course number, and so on. You can control whether the in-progress vs completed comparison is the first TIEBREAK check the auditor performs, but you need to alter the settings using Controller to configure it according to your requirement.
17. Kept class based on coin flip.
In very rare situations, for example, when classes are repeated many times (such as PE and ART classes), the TIEBREAK checks do not help the auditor make a decision. At this point, the two classes in question are basically identical so the auditor makes a decision based on a randomly generated choice.

In-progress vs completed

When each step is evaluated, if both classes have equal values for that particular step, the auditor continues to the next step to see if it can find a reason to keep one class over another class. The

auditor looks at many properties for the classes in question, and based on a CFG020 DAP14 configuration flag, it also checks to see if one of the classes is in-progress.

At step 7, which is used or ignored based on a CFG020 DAP14 flag, the algorithm directly examines whether a class is in-progress when deciding which rule to keep.

Percent complete calculation

Updated: September 29, 2023

A percent complete calculation is done for each rule, each block, and for the overall audit. It is this calculation that is used to mark a rule, a block, or the overall audit as completed.

Rule completeness

Updated: March 25, 2022

Rule completeness calculations can vary for different types or rules.

Course rule

At the course rule level, the calculation can take on some complex characteristics. Basically, the calculation determines the number of classes or credits that are required for the rule and then calculates the number of classes or credits applied to the rule. The percent complete is calculated from those two factors. If a course is applied to a rule but has not yet been graded (that is, it is "in progress") and the rule would be completed with that course, the percent complete is reduced to 98%. In the Degree Works reports we show these in-progress rules with a single squiggle signifying that the rule is close to being completed. If the student ends up failing the class it will be placed in Insufficient and the rule will end up with an empty box. There is no guarantee that the in-progress class will actually end up on the rule after it has been completed because other classes the student registers for may cause classes to be shifted around.

A rule qualifier that is not met will make the rule become 99% complete – given the required credits/classes were taken. A MinSpread or MinPerDisc qualifier that has not been met will cause the rule to be marked as 99% complete and will appear with a box with a double-squiggle on the Degree Works worksheet.

Noncourse rule

The percent complete is calculated based on the total number of noncourses required and the number of noncourses completed.

Subset rule

All the rules within the subset are counted in the block and overall percent complete calculations; the subset rule itself is ignored. The rules within the subset are used to calculate the completeness of the subset itself of course. If any subset qualifiers are not satisfied (and all the rules are complete) the percent complete is reduced to 99%. If all rules are complete but one or more contains an in-progress the subset will be considered 98% also – the subset will inherit this property of its child.

Group rule

The group that is the "most complete" is used as the basis of the percent complete calculation.

For example, a group rule states that 1 group is needed from a list of four groups.

```
1 GROUP in  
  (8 CREDITS IN BIOL 100:199) or  
  (8 CREDITS IN CHEM 100:199) or  
  (8 CREDITS IN PHYS 100:199) or  
  (9 CREDITS IN MATH 250 + CHEM 200 + CHEM 220)
```

If 6 credits have been applied to the last group and 4 credits to the first group, the last group will be used to do the percent complete calculation.

If the first two rules have 4 credits applied the auditor will simply chose one given everything else equal between the classes in question. The class on the rule not chosen is freed up to be used on another rule or will be placed in fall-through.

Block and blocktype rule

The percent complete is based on the details of the rule in the referenced block. If the referenced block is not found in the audit, the percent complete is zero.

If/Then/Else Rule:

When the IF condition is FALSE and there is no ELSE rule, the IF statement is not included in the block percent complete.

When the IF condition is true, the calculation is based on the rule type in the THEN portion of the IF rule.

These rules are ignored when calculating the parent blocks and the overall completeness except in the case when the rule is zero percent complete.

Block completeness

Updated: March 25, 2022

Each block's percent complete is based on the rules and the block qualifiers.

The rules are counted at the highest level for this calculation. If a Subset Rule is included in the block, it is counted as a single rule (its completeness has already been calculated at the rule level). If all rules within a block are complete but there are block qualifiers that are not satisfied, the completeness is reduced to 99%.

If the block is optional then it is 100% complete. If it is not optional then add up the number of rules at the first level and their total percent complete. Divide the total percent complete by the number of rules counted to get the block's completeness. If all rules have been completed but there is a block header qualifier that is not satisfied, the percent complete is reduced to 99%.

Overall audit completeness

Updated: March 25, 2022

The overall audit percent complete is calculated from the rules within each block being used by the Auditor. (If a rule has not been used, for example, within an IF statement, it is not included in the calculation.)

For each block that does not have an OPTIONAL qualifier do the following: Add up the number of rules at the first level - this means do not count each rule in a subset or group. Add up the percent complete of each of the rules counted. Divide the total percent complete by the number of rules counted to get the overall completeness. If all rules have been completed but there is a block header qualifier that is not satisfied, the percent complete is reduced to 99%.

Output options

Updated: March 25, 2022

Instead of showing the percent complete for each block, Degree Works maps each percent value to a graphic.

Percent complete	Meaning	Visual display
0 - 97	Not complete	Empty box
98	In-progress incomplete	Single squiggle box
99	Qualifier incomplete	Double squiggle box
100	Complete	Checked box

Output Engine (DAP15)

Updated: March 25, 2022

The Output Engine reads the audit results from the database and formats the results into an XML or JSON document.

The Output Engine interprets the audit results and produces printed and online audit reports. Online viewing of audit results is accomplished using the Responsive Dashboard, which may be made available to Registrar, faculty, or students. It is through the Responsive Dashboard that exceptions and substitutions are entered in addition to advisor notes. Security to control "who accesses what" is enforced by the Responsive Dashboard.

The Output Engine consists mainly of one subroutine, DAPEXTRACT (DAP15), with the following characteristics:

- Resides on the classic server
- Extracts the requested audit from the Degree Works database
- Formats the audit results as an XML or JSON tree

The presentation layer takes in the XML or JSON tree and formats it using XSL or some other tool. In the Responsive Dashboard the react-js code creates the worksheets from the JSON audit. In batch mode FOP is used to convert the XML trees into PDF.

Custom data items

Updated: March 25, 2022

Custom data items are pieces of data that are not part of the standard data items passed from the student system to Degree Works. Custom data items are also data available for use in an IF expression in Scribe.

Custom data items are defined in SCR002. They can be used to display this data in the student data card or used in Scribe in an IF expression where a requirement varies based on student data. For more information, see the [UCX-SCR002 Custom Data](#) and [Custom student data display](#) topics.

For example, if Catholic students must take RLGN 300 and non-Catholic students must take RLGN 305, the requirement could be written as follows:

```
If (RELIGION = CA) Then  
    1 Class in RLGN 300 Label "Catechism"  
Else  
    1 Class in RLGN 305 Label "Religion for the masses";
```

The above requirement will parse with an error indicating RELIGION is not valid. To add RELIGION or another piece of data as a custom data item, follow these steps:

1. Determine where the custom data item is stored in the student system. For example, RELIGION could be stored on a user-def field on the RAD-PRIMARY-MST or in the RAD-CUSTOM-DTL, which is the most logical place to store custom data.
2. If you want to use the data item in an IF expression in Scribe, it must also be added to SCR002.
3. Check SCR002 to see that the data item (for example, RELIGION) is not part of the custom data items. If it is, use the code from SCR002 in the IF expression.
4. If the data item is not in SCR002, add the data item to SCR002. Choose a name for the data item that is from one to twelve characters long (for example, RELIGION). This name is the Degree Works name of the custom data item. It is usually upper-case. Find out the element number of the data item by looking in SYS999. For example, the element number for the Custom-Value is R323. Use Controller to add the new code to SCR002. Create a new record with a code of RELIGION or whatever name you want to use in Scribe. In the **Description** field, enter something meaningful (for example, **Religious affiliation**). The **Data Element** field should contain your value from SYS999, in this case **R323**. Because this value is coming from a Dtl record, you need to specify which record to read. This is done by filling out the Edit Element information. Here we want the record with **RELIGION** in the **Custom-Code** field, which is element R322. The **Type** field should be filled with **EV** and the value should be **RELIGION**.

NonCourse data items

Updated: March 25, 2022

NonCourse data items are pieces of data from the student system that are non-standard requirements, not a traditional course but something required for graduation.

They are defined in SCR003. Examples of NonCourse data items are music recital, art gallery show, chapel attendance, and placement exams. In addition to using NonCourse data items in Scribe, they can be added as requirements on a student's plan. For more information, see the [UCX-SCR003 Noncourse Codes](#) and [Non-course](#) topics.

```
1 NonCourse (RECITAL)
  ProxyAdvice "At least one recital is required"
  Label RECITAL "Recital";

1 NonCourse (MATHEXAM > 12)
  ProxyAdvice "A math exam with a score better than 12 is needed"
  Label MATHEXAM "Math exam";
```

The above requirements will parse with errors if the NonCourse code in parentheses is not valid.

To add MATHEXAM or another piece of data as a NonCourse data item, follow the steps outlined below.

Step 1

Determine where the noncourse data is stored in the student system. For example, MATHEXAM might be stored in the RAD-CUSTOM-DTL, but often noncourses are stored in the RAD-NONCRSE-DTL. You can check the Student Data Report to verify where the data is being housed for your students.

Step 2

If the data item is not in SCR003 then add the data item to SCR003. Choose a name for the data item that is not longer than twelve characters, for example, MATHEXAM. This name is the Degree Works name of the noncourse data item. It should be upper-case. If the data is not on the RAD-NONCRSE-DTL, find out the element number of the data item by looking in SYS999. Use element number 0000 if the value is on the RAD-NONCRSE-DTL.

Use Controller to add the new code to SCR003. Add a new record and enter the Degree Works name for the data item, for example, MATHEXAM, as the key. In the Description field enter a description of the data item (for example, "Math Placement Exam"), followed by the element number in the Data Element field. The Edit Element, Type and Value fields are used if the value you are getting is from a "detail" record.

Advisee filtering

Updated: September 30, 2022

Non-student users such as advisors, department heads, and deans can be configured to have access only to their advisees or students in their department, school/level, or college.

This ability is particularly useful for users that have no assigned advisees or for allowing advisors access to all students in their department in addition to their advisees.

The following Shepherd keys are used for advisee filtering:

- ADFILTER
- SDSTUANY
- SDSTUMY
- SDFINDID
- SDFIND

Basic advisee filtering

Advisors are linked to their advisees by the ADVISOR rad_goaldata_dtl record. To configure advisors to have only their assigned advisees load in Responsive Dashboard, assign the SDSTUMY Shepherd key. When they log in to Responsive Dashboard, they will see a Select Student drop-down that contains all students with a rad_goaldata_dtl record, where rad_goal_code=ADVISOR and rad_goal_value is equal to the advisor's ID.

The SDSTUMY Shepherd key can be assigned explicitly user by user through Controller, or implicitly by user class in the core.security.rules.shpcfg Shepherd setting. It is also in the baseline SRNADV and SRNADVX Shepherd groups.

Department advisee filtering

Department heads can be assigned to a major or majors with their department, which will link them to all students with that major. To configure users to have only students in their department load in Responsive Dashboard, use Controller to add a department filter on the users' access records and select up to 10 majors for this filter type. Additionally, assign the ADFILTER Shepherd key. When they log in to Responsive Dashboard, they will see a Select Student drop-down that contains all students with a rad_goaldata_dtl record, where rad_goal_code=MAJOR and rad_goal_value is equal to the specified filters on the advisor's shp_user_mst.

The ADFILTER Shepherd key can be assigned explicitly user by user through Controller, or implicitly by user class in the core.security.rules.shpcfg Shepherd setting.

See the [User Access](#) topic for additional information on adding a department filter for a user.

School/Level advisee filtering

Deans or other users can be assigned to a particular school/level or schools/levels, which will link them to all students in that school/level. An example might be users who should be restricted to viewing students in the law school only, with no access to undergraduate students. To configure users to have only students in their school/level load in Responsive Dashboard, use Controller to add a school filter on the users' access records and select up to 10 schools for this filter type. Additionally, assign users the ADFILTER Shepherd key. When they log in to Responsive

Dashboard, they will see a Select Student drop-down that contains all students with a rad_goal_dtl record, where rad_school is equal to the specified filters on the advisor's shp_user_mst.

The ADFILTER Shepherd key can be assigned explicitly user by user through Controller, or implicitly by user class in the core.security.rules.shpcfg Shepherd setting.

See the [User Access](#) topic for additional information on adding a school filter for a user.

College advisee filtering

Deans or other users can be assigned to a particular college or colleges, which will link them to all students in that college. An example might be users who should be restricted to viewing students in the College of Arts and Sciences only, with no access to students in the College of Education. To configure users to have only students in their college load in Responsive Dashboard, use Controller to add a department filter on the users' access records and select up to 10 colleges for this filter type. Additionally, assign users the ADFILTER Shepherd key. When they log in to Responsive Dashboard, they will see a Select Student drop-down that contains all students with a rad_goaldata_dtl record, where rad_goal_code=COLLEGE and rad_goal_value is equal to the specified filters on the advisor's shp_user_mst.

The ADFILTER Shepherd key can be assigned explicitly user by user through Controller, or implicitly by user class in the core.security.rules.shpcfg Shepherd setting.

See the [User Access](#) topic for additional information on adding a college filter for a user.

Advisee searching

To allow advisors and other users access to search by student ID, they can be assigned the SDFINDID Shepherd key. In addition to having their assigned advisees load in the Select Student drop-down, they will be able to search by ID in the Student ID field. Note that by default, they will be allowed to search only for their assigned advisees. If they enter an ID for a student that is not an assigned advisee, they will not be able to view that student. However, if users have been assigned the SDSTUANY Shepherd key, they will be able to search for any student, not just their assigned advisees.

To allow advisors and other users access to the Advanced search functionality, they can be assigned the SDFIND Shepherd key. In addition to having their assigned advisees load in the Select Student drop-down, they will be able to use Advanced search to further filter their list of assigned advisees. Note that by default, they will see only their assigned advisees in their search results. However, if users have been assigned the SDSTUANY Shepherd key, they will see all students in their search results, not just their assigned advisees.

SDFINDID and SDFIND are in the baseline SRNADV, SRNADVL, and SRNREG Shepherd groups. SDSTUANY is in the baseline SRNREG Shepherd group.

Examples

Scenario	Assigned Shepherd keys and filters
Registrars who have access to and can search for any student	SDSTUANY, SDFINDID, SDFIND

Scenario	Assigned Shepherd keys and filters
Advisors who have access to their advisees only with no searching capabilities	SDSTUMY
Advisors who have access to their advisees and can search for their advisees only	SDSTUMY, SDFINDID, SDFIND
Advisors who have access to their advisees and can search for any student	SDSTUMY, SDFINDID, SDFIND, SDSTUANY
Political Science department heads who have access to students in that department only	ADFILTER with a department advisee filter for Political Science
Deans in the College of Arts and Sciences who have access to all students in that college with searching capabilities	ADFILTER with a college advisee filter for College of Arts and Sciences, SDFINDID, SDFIND
Deans in the graduate school who have access to all students in that school with searching capabilities	ADFILTER with a school advisee filter for the graduate school, SDFINDID, SDFIND
Business department heads who have access to students in that department but can also access any student's record	ADFILTER with a department advisee filter for Business, SDFINDID, SDFIND, SDSTUANY
Advisors who are also department heads with access to their advisees and students in the Chemistry department	SDSTUMY, ADFILTER with a department advisee filter for Chemistry
Advisors who are also department heads with access to their advisees and students in the History department in addition to all students	SDSTUMY, ADFILTER with a department advisee filter for History, SDFINDID, SDFIND, SDSTUANY

Degree Works bridge

Updated: March 25, 2022

The Degree Works Bridge is a mechanism for loading the Degree Works data structure with relevant data from the student database.

It has a well defined API-format that may be used by a university that is extracting data in a static fashion on a regular basis, or in a dynamic fashion on-demand. Degree Works also has a “native” integrated extract for selected student systems.

Degree Works static bridge

Updated: March 25, 2022

Student data is typically moved from the University’s student database by using the batch process RAD11.

This process can be scheduled to run on a regular periodic basis or can be launched on a manual basis. For more information about this process, please refer to the [Degree Works bridge specifications](#) topic.

Degree Works dynamic refresh

Updated: March 25, 2022

Although the recommended “best practice” is that the RAD11 static bridge be used to load student data in batch mode on a regular periodic basis, the RAD08 dynamic bridge may be used to load student data throughout the course of the day as events trigger the need.

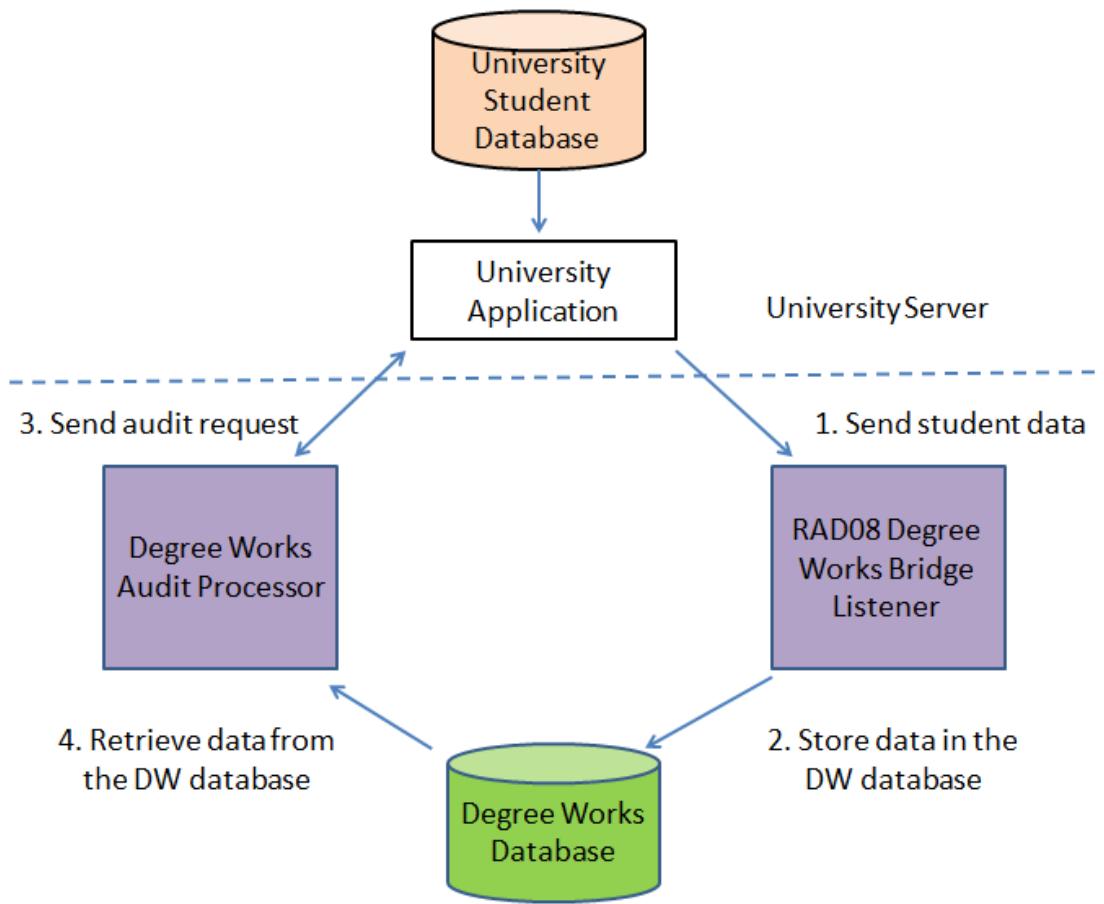
RAD08 is a daemon process that listens for incoming bridge requests and saves the incoming data to the Degree Works database. When a request is sent to RAD08 and processed, subsequent run-audit requests will use the new data for the given student.

The University must write an application which issues the request for Refresh and sends it to RAD08 running on the Degree Works classic server.

The number of RAD08 processes that listen for and process bridge requests is controlled by the classic.daemons.rad08.count setting. This setting controls the number of children the parent RAD08 will create to listen for requests. Because the parent also listens for and processes requests, the total number of RAD08 processes will be the number of children plus one.

You may decide to URL encode the request before sending it RAD08. If URL encoding is being used, the WEB84_URL_DECODE flag in rad08job must be set. Setting this flag tells the WEB84 subroutine that RAD08 calls to decode the request when breaking out the name-value pairs contained within.

Banner sites using the "native" Degree Works Integrated Interface should not use this RAD08 process. For more information, see the [Batch Extract flow diagram](#) topic.



Dynamic Refresh process flow

1. The University's application sends request to Degree Works to store student data in the Degree Works database.
2. Degree Works stores the student data in the Degree Works database.
3. The University's application sends a Web run audit request to Degree Works.
4. Degree Works reads the student data from the Degree Works database, processes the audit and returns the audit report to the user.

Refresh Request

The refresh request is in name-value pair format and contains the following information:

Name	Description	Length
ACTION	"REFRESH" for student data refresh	07
STUID	ID of the student for which the refresh is needed	10

Name	Description	Length
RECCOUNT	Count of student records returned as REC name-value pairs	04

After the header information, student data records must be sent in the refresh request.

Name	Description	Length
REC	Repeated up to 256 times. Each REC name-value pair contains a record in Bridge-Interface-Format including a HEADER = 28 bytes, DATA = 972 bytes for a maximum total of 1000 bytes.	1000

Example Refresh request (not all student records are shown):

```

ACTION="REFRESH"&STUID="123456"&RECCOUNT=0003&
REC="123456" R010PRIM A 123456 Johnson, Joyce"&
...
REC="123456" R020STUD A 123456 000000 20001"&
...
REC="123456" R050BIOG A 123456 542661234 19801231"&#
<$ENDMSG$>
```

An ampersand “&” separates each name-value pair, with an ampersand and pound-sign “#” signaling the end of the data. The end of the entire response is signaled by “<\$ENDMSG\$>”. Each value may or may not be enclosed by quotation marks but quotation marks are required if the value may contain an ampersand. All data for a student must be bridged; do not bridge just the changed or new records.

To increase the efficiency of the data transfer over the network, trailing spaces from each REC name-value pair should be removed. If only the first 48 bytes contain actual data then the trailing spaces should be removed before placing the record in the REC name-value pair. Records must already be sorted by the bridge-indicator field (R010PRIM, R020STUD, etc) before sending them across the network.

Refresh Response

The RAD08 process returns a message containing status information concerning the processing of the refresh request. The name-value pairs are listed in the tables below. Note that “Length” is the maximum length in bytes.

The status information in the Refresh Response includes:

Name	Description	Length
STATUS	OK or FAIL. Return FAIL if error encountered.	04

Name	Description	Length
ERROR	Error number specifying error encountered; e.g., 4321. Omit if no error encountered.	04
ERRMSG1	Error message string 1 describing error. Omit if no error.	80
ERRMSG2	Error message string 2 describing error. Omit if no error.	80
DATACHANGED	YES if the Degree Works bridge has detected changes as of the last time the student was bridged – either through RAD08 or through the batch process. NO if the bridge did not detect data changes.	3

Example Refresh Response:

```
STATUS=OK&ERROR= ""&ERRMSG1= ""&ERRMSG2= ""&DATACHANGED=YES&ACTION=REFRESH&...<br/>STUID=123456&#*<$FINISHED$>
```

The error number returned along with the error message strings will be returned. The error messages may say "Student ID number not found in student system" or "Database error occurred when writing student classes." An error is returned only if the data cannot be saved, is missing, or is not valid.

Refresh Thank You

After the University receives the status response a thank-you reply should be sent to the Degree Works refresh listener to indicate that the response has been read and it is ok to close the socket connection.

The thank-you message should be a simple "<\$THANKYOU\$>" value. When the Degree Works listener receives this message the current socket connection will be closed.

Equivalent course tracking

Updated: March 25, 2022

Equivalent courses are courses that changed discipline or number at some point in time.

These equivalents are important to Degree Works when a student is being evaluated against a catalog year that uses a different course-key than the course-key on the student's class. If Degree Works does not know about the equivalency, the rule will never be satisfied because the course-

key on the student's record does not match the course key Scribed in the rule. The solution to this problem is to set up the equivalence in the dap_eqv_crs_mst.

For example, John took ENGL 101 in the fall of 1990, took a year off and then returned to school as a member of the class of 1996. ENGL 101 changed to ENGL 115 in the fall of 1995. The course catalog requirements include ENGL 115 as a requirement, not ENGL 101 (which is now a different course). John's ENGL 101 course from 1990 should fulfill the ENGL 115 requirement of 1996.

In order for the Auditor Engine to automatically evaluate ENGL 101 as ENGL 115, an equivalent course record must be created. When created, the equivalent courses serve as a map of course number and discipline changes. The equivalent course record is not specific to a student. The equivalent course record is stored in the dap_eqv_crs_mst.

Note:

- This assumes your institution does not reuse course keys. A simple example of reusing course keys: MATH 103 was "Advanced Algebra II" during catalog year 19992000, then was changed to MATH 106. Then in 20002001 MATH 103 was reused to be a completely different course such as "Pre-Calculus I". If your institution has reused course keys, do not attempt to use the dap_eqv_crs_mst as described here; please contact Ellucian for assistance in setting this up.
- The structure of the dap_eqv_crs_mst is:
class_dtl information ("When the course was taken")

TERM-CATALOG-YR	X12	Degree Works catalog year (STU035) mapped from the term the class was taken. Mapped from term to catalog year through STU016. The character "@" can be used as a wildcard. Using the wildcard here tells the auditor that it does not matter when the student took the class.
OLD-COURSE-DISCIPLINE	X12	Discipline from the course-key of the class taken (STU352).
OLD-COURSE-NUMBER	X12	Number from the course-key of the class taken. The character "@" can be used as a wildcard (primarily used if there is a discipline change, for example, ENGL has changed to ENG).

Course Catalog Information ("When the equivalency exists")

TARGET-CATALOG-YR	X12	Degree Works catalog year (STU035) mapped from the student's catalog year. The
-------------------	-----	--

character “@” can be used as a wildcard. Using the wildcard here tells the auditor to apply this equivalency to all catalogs.

NEW-COURSE-DISCIPLINE	X12	Discipline from the course-key in the target catalog year – what the discipline is in requirements written for the target catalog year.
NEW-COURSE-NUMBER	X12	Course number from the course-key in the target catalog year – what the course number is in the requirement block written for the target catalog year. The character “@” can be used as a wildcard (primarily used if there is a discipline change, for example, ENGL has changed to ENG).

Set up equivalent course tracking

Setting up equivalent course tracking is done through the bridge extract and Controller.

To use dap_eqv_crs_mst, you must:

1. Update all prior catalogs with the change.
 - Discipline code change - find all instances of the discipline in the prior catalogs and change it to the new discipline code.
 - Specific class change - find all instances of the class in the prior catalogs and change it to the new class.
2. Add the equivalency to the dap_eqv_crs_mst.
 - a. Bridge the records using the R190DEQV layout (Refer to Degree Works Bridge to Student Record Systems Technical Specifications). You can bridge the entire contents of the dap_eqv_crs_mst or bridge only new records as needed.
 - b. Use Controller to maintain your equivalences in CFG070. You should then run the dapucx2eqv script. Be careful about using both the bridge method and CFG070 to maintain equivalences.

Discipline code change example

The school has decided to change the course discipline code from ENGL to ENG. The school will either manually change the requirement blocks referencing ENGL to say ENG, or the school will set the CFG020 DAP13 Process Equivalences flag to Y and run DAP16 to reparse all of the

blocks. With this flag set to Y, the underlying block requirements will now say ENG instead of ENGL.

For example, “1 Class in ENGL 101” will be changed to “1 Class in ENG 101”. This will happen through the laborious manual process or by running DAP16 with that flag enabled. Though the changes will not be reflected in Scribe when the block is viewed.

However, the auditor will now not apply any ENGL 101 classes to this rule. Setting up dap_eqv_crs_mst records is needed to tell Degree Works to apply ENGL classes against ENG rules.

1. Update prior catalogs with the new course number.

In Scribe, locate any instance of ENGL and change it to ENG for every catalog – or rely on DAP16 with the CFG020 DAP13 Process Equivalences flag set.

2. Add an entry to CFG070 in Controller.

Degree Works will apply all ENGL courses against ENG requirements.

- Catalog year class was taken = @
- Old Course Discipline = ENGL
- Old Course Number = @
- Student's Catalog Year = @
- New Course Discipline = ENG
- New Course Number = @

Course number change example

The school has decided to change the course number for the Math Statistical Analysis class. In the past, it had been listed as MATH 176. From now (catalog year 20042005) on, this class will be listed as MATH 200. The current catalog now contains rules such as “1 CLASS in MATH 200”. Without a dap_eqv_crs_mst, Degree Works would not apply MATH 176 to the MATH 200 rule. With the dap_eqv_crs_mst, you are telling Degree Works that the classes are really the same and thus MATH 176 can apply to the MATH 200 rule.

1. Update prior catalogs with the new course number.

In Scribe, locate any instance of MATH 176 and change it to MATH 200 for every catalog – or rely on DAP16 with the CFG020 DAP13 Process Equivalences flag set.

2. Add an entry to CFG070 in Controller.

Degree Works will take every instance of MATH 176 and treat it as MATH 200.

- Catalog year class was taken = @
- Old Course Discipline = MATH
- Old Course Number = 176
- Student's Catalog Year = @

- New Course Discipline = MATH
- New Course Number = 200

CFG070 Equivalence course records

Updated: March 25, 2022

The CFG070 Equivalence Course Records table contains the equivalence records for mapping historic course keys to current course keys.

This table provides an easy to use interface for maintaining the equivalence course records for use in Degree Works. Each record consists of a 30 byte key followed by a number of 12 byte fields for defining the course equivalence. The key can be any alphanumeric value but must be unique for each record. Wildcards can be used in the catalog year and course number fields. This is useful for changing course keys across multiple catalog years or for changing discipline codes globally. In the screen shot below, MLFRATH 100 taken in the 19901991 catalog year became FREN 1000 for students with a catalog year of 20012002.

The Note field is a 50 byte free text field which can be used for internal documentation. This information is not used in audits and is not displayed on any of the audit reports.

You can easily create new equivalence course records using the Controller data entry screen. New CFG070 records created in Controller will not be available for use in Degree Works until they are loaded into the dap_eqv_crs_mst table in the database. To load new records into the database, you should manually load the records by running dapucx2eqv under ADMIN in Transit. You can also use the Bulk Operations function in Controller to load these records from a flat file.

Process equivalences into scribed courses

Updated: March 25, 2022

When course numbers change (HIS 206 was renamed HIST 212) changes need to be correctly applied to Degree Works.

We have to make sure students taking the old course or the new course get credit for the particular requirement – and we need to make sure that students get the correct advice – which is to take the new course number. Scribes may go through historic blocks and make alter the requirements to list the new course number instead of the old one – but this can be a daunting task if there are a lot of changes.

Degree Works can alter the requirements listing the old course key and change it to the new course key based on the equivalence records that are in place.

To enable this feature you must do the following:

1. Set the CFG020 DAP13 Process Equivalences flag to Y.
2. Run DAP16 in Transit to reparse all of your blocks

Each time the equivalence table changes you should rerun DAP16 to pick up the changes and apply them to your rules.

With these dap_eqv_crs_mst records in place in CFG070:

Catalog year taken	Old course discipline	Old number	Student's catalog year	New course discipline	New number
2002	DANI	1	@	ANTH	100
2003	DANI	1	@	ANTH	100
2004	DANI	1	@	ANTH	100

With this block in place:

```
BEGIN
;
1 Class in DANI 1
    Label "My rule 1";
END.
```

The Diagnostics Report shows the Requirement with the equivalence information. The Requirement line shows what the rule looks like after the conversion has taken place and also the original course that was scribed. For example:

1 Classes in ANTH 100[Formerly DANI 1]

Note:

- Although the Diagnostic Audit is showing the equivalence information in the Requirement line, our standard Registrars Report worksheet will not.
- Requirements containing ranges or wildcards such as the following will not be processed following these rules. These requirements will have to be taken care of manually at this time.
5 Credits in MATH 1@
5 Credits in MATH 100:199

Simple scenario 1

Course A was offered from 2002 to 2005, but then in 2006 course A was renamed B.

These CFG070 records were built:

Catalog year taken	Old course discipline	Old number	Student's catalog year	New course discipline	New number
2002	A	001	@	B	001
2003	A	001	@	B	001
2004	A	001	@	B	001
2005	A	001	@	B	001

The 2002 scribed block looks like this:

```
1 Class in A001, X001, Y001
```

Problem:

Under normal operations we see that the student has taken A in 2002 and we rename it to B before trying to apply it to rules. When doing this against this block B does not fit because the rule still says A. For some schools it is a lot of work to go through and fix all of their rules to list the current course name.

Solution:

When saving the blocks the parser will make this conversion.

```
1 Class in A001, B001, X001, Y001
```

This change will be made to the saved syntax tree that is then pulled into the auditor.

When auditing a student who took this course when it was named A, the normal processing will take effect and A will be renamed to B and thus will apply to this rule. Students taking the course after 2005 will take it as B and it will apply normally also.

Complex scenario 2

A changed to B but then B was changed to C

These CFG070 records were built:

Catalog year taken	Old course discipline	Old number	Student's catalog year	New course discipline	New number
2002	A	001	@	B	001
2003	A	001	@	B	001
2004	B	001	@	C	001
2005	B	001	@	C	001

Yes, A was renamed to B but then B was renamed to C.

The 2002, 2003, 2004 and 2005 blocks should all list the rule using C:

```
1 Class in A001 C001, X001, Y001  
1 Class in B001 C001, X001, Y001
```

This should be handled as required.

Complex scenario 3

A (Intro to Art) was renamed to B but then A was reused for some other course (Sculpturing). When this happens, the logic is complicated. We will be looking up the course in the CFG074 table to find out if the course was reused. If the course was reused the parser will skip the processing of this course - these reused courses will have to be rescribed manually.

Process cross-listings into scribed courses

Updated: September 29, 2023

There are two options when dealing with scribing of cross-listed courses in Degree Works.

- Manually list all of the cross-listed pieces together in rules using Scribe:
1 Class in MATH 101, PHIL 101, STAT 101;
- Manually list just one piece of a cross-listed set using Scribe rules. Use CFG073 plus the parser to automatically insert the rest of the cross-listed set inside curly braces {Hide...}:
1 Class in MATH 101 {Hide PHIL 101, STAT 101};

To enable the second option, follow these steps:

1. Set the CFG020 DAP13 **Process Cross-Listings** flag to Y.
2. Set up CFG073. For the example above, two records would need to be added:
MATH101 would be added as the UCX Key to CFG073 with the UCX Value of PHIL101.
MATH101 would be added as the UCX Key to CFG073 with the UCX Value of STAT101.
For non-Banner clients, CFG073 must be loaded manually using Controller or inserted from a file containing records formatted correctly for CFG073 using the Bulk option of Controller.
For Banner clients, CFG073 CAN be automatically loaded by the Banner Equivalencies Extract (RAD38) process, if desired, or it could be manually loaded. To automatically load CFG073:
 - a. Set the CFG020 BANNER Cross List in SCREQIV to Y.
 - b. Use Transit to launch RAD38 – the Banner Equivalencies Extract.
CFG073 will be generated automatically if any cross listed course records are found in the SCREQIV table.
3. Use Transit to launch DAP16 to reparse all blocks.

Each time the CFG073 cross-listings table changes, DAP16 should be run to pick up the changes and apply them to your rules.

With these cross-listing in place in CFG073:

MATH 101	cross-listed with	PHIL	101
MATH 101	cross-listed with	STAT	101
MATH 102	cross-listed with	PHIL	102

With this block in place:

```
BEGIN
  MaxCredits 9 in MATH 101, 102
;
  5 Credits in MATH 101 , 102
    Label "My rule 1";
END.
```

The Diagnostics Report displays the cross-listed courses in three different places:

- The Header Qualifiers line shows the MaxCredits qualifier with the cross-listed information.
- The advice (Still Needed) line shows the cross-listed information.
- The Requirement line shows what the real rule looks like, which is different from the Scribed block above.

Note:

- Although the Diagnostics Report is showing the cross-listed information in the advice, the standard Student View worksheet will not. If the same type of changes are to take place whenever PHIL 101 is scribed, CFG073 records must be created pointing PHIL 101 to MATH 101 and PHIL 101 to STAT 101. The same goes for STAT 101. In all, six records would be needed to cover these three courses in the cross-listing set.
- The parser will not be creating header qualifiers to make sure the student will only get credit for the course one time in case they are allowed to register for it under the two different names. A header qualifier like this will not be created. It should be handled by the registration system.

MaxClasses 1 in MATH 102, PHIL 102

- These qualifiers may be added manually, if required. Requirements containing ranges or wildcards such as the following will not be processed following these rules. These requirements will have to be taken care of manually at this time.

5 Credits in MATH 1@

5 Credits in MATH 100:199

Freeze audits

Updated: September 29, 2023

Audits can be frozen so they aren't deleted when the audit history count is exceeded.

The CFG020 DAP14 Audit History Count setting controls how many audits to keep per student. When a new audit is processed, Degree Works will delete the oldest audit to ensure this history count is obeyed. However, there are certain times when you want to keep particular audits that do not get deleted you can freeze those audits.

Enabling audit freezing

Users can choose to enter a description on audits but not freeze them, or vice versa. However, normally your users may want to only enter a description when freezing audits –it is recommended that you establish a best practice for your institution.

Only those users who have been given the AUDFREEZ key will be allowed to freeze audits. Additionally, only users with the AUDDESCR key will be able to enter a description on audits. By default the REG, ATHL, and AID user classes are given these keys. You can modify the

`core.security.rules.shpcf` Shepherd setting to add or remove these keys on user classes as needed.

Enabling audit freezing on What-If audits

For what-if audits the keys are WIFFREEZ and WIFDESCR to allow the freezing of audits and to allow users to enter a description. Neither of these keys are assigned to any user classes by default. Additionally, saving of what-if audits must be enabled by setting the CFG020 DAP14 What-if History Count value to a number greater than or equal to 01. When this setting is blank or 00 no what-if audits are saved and thus none can be frozen.

Setting up freeze types

When an audit is frozen, the user specifies a freeze type. The freeze type not only specifies that the audit is frozen but also gives an indication as to why it was frozen. The AUD032 table contains a list of the valid freeze types that your institution is using. This table is initially installed with a set of example freeze types, but you can delete, modify, and add to this list as needed. Each freeze type is associated with up to 10 user classes. Only users in those user classes will be allowed to freeze audits with the given freeze type.

Entering a freeze type and description

When viewing the most recent audit on the Worksheets tab, users with the ability to freeze audits or enter descriptions are given the option to do so. The select box of freeze types is restricted to those set up in AUD032. However, if the current freeze type on an audit is not associated with the current user's user class, the freeze type will still appear (selected) in the select box. This allows the user to see the current freeze type and optionally change it to another freeze type.

The user can enter a description without choosing a freeze type, and can also enter a freeze type without entering a description – both fields are optional. The user may un-freeze an audit by choosing the “(not frozen)” option. These un-frozen audits will be deleted from the system when new audits are generated.

On the History tab you can see who froze the audit and when it was frozen. You can also modify the description and freeze type as needed. The description and freeze type also appear in the header of the worksheet itself, so that users who do not have the ability to freeze audits or enter a description can also see this information.

On the History tab you can also Delete any audit – whether it is frozen or not.

Freezing audits in a batch

Transit users may freeze audits and enter a description on audits when running DAP22 batch audits. The list of freeze types that appear in Transit are those associated with the REG user class in AUD032. If you want all freeze types to appear in Transit, then ensure that all freeze types have REG listed as one of the user classes in Controller.

Deleting old audits

Frozen audits do not count against the history count value. The Audit History Count may be set to “03”, for example, but a student may have several frozen audits and three non-frozen audits. There is no limit on the number of frozen audits that can be kept for each student. For this reason, it is important to limit the number of frozen audits that persist in your database for long periods of time.

Audits take up substantial space in your database and freezing many audits that never get deleted will cause you to run out of space eventually. If you do decide to freeze audits, you can consider reducing the history count value to 01 or 02 to help reduce the number of audits saved.

You can have audits frozen with a freeze type of DEGAWD (Degree Awarded), for example, that you do not want to delete but you may have frozen others that you only wanted to keep for a year or two. You can use the dapdelaudits script to delete audits older than a specific date and optionally specify a freeze type. If no freeze type is entered, only those non-frozen audits will be deleted.

```
$ dapdelaudits 20091231
Audits older than 20091231 will be deleted.
By default only non-frozen audits will be deleted - but you may choose
to delete specific audits with a particular freeze type.
Enter the freeze type (or just hit enter for non-frozen audits)? > TRMFAL
Deleting audits older than 20091231 with a freeze type of TRMFAL
Continue with delete? (y/N) >
```

You can supply the freeze type as the second parameter:

```
$ dapdelaudits 20091231 TRMFAL
Deleting audits older than 20091231 with a freeze type of TRMFAL
Continue with delete? (y/N) >
```

You can also supply a confirmation "Y" value as the third parameter to avoid the confirmation message. This is useful when calling dapdelaudits from a script:

```
$ dapdelaudits 20091231 TRMFAL Y
```

You may also use AUD02 in Transit to delete audits in much the same fashion as using the script.

Multi-entity processing in Degree Works

Updated: September 29, 2023

Degree Works can be deployed across many campuses or entities while specifying particular types of data that ought to be shared among those entities.

Some institutions need to deploy Degree Works across multiple degree-granting campus entities within their institution. Other institutions serve constituents at only one campus and have no need to share certain information beyond a single entity. Degree Works is configurable so that institutions may elect to use only one instance of Degree Works at a single entity or campus, or deploy Degree Works services across many campuses or entities while specifying particular types of data that ought to be shared among those entities. Configuration to specific institutional needs allows sites to manage both data and system administration in a manner best suited to their particular needs. Only students who have a degree record for a particular college can be accessed and processed within the Degree Works web interface.

Recognizing that terminology in Higher Education can be ambiguous, understand that in the context of multi-entity processing, we are using "campus" to mean a separate, degree-granting institution which is either "all by itself" (Entity=1) or part of a district or system of sister institutions (Entity=many) which has a central administrative arm and installation of Degree Works. We recognize that "campus" can also mean "location", where the campus is a satellite venue for the larger degree-granting institution. In this discussion of multi-entity processing do not think of a "campus" or "campus entity" as simply a locale or the case study will be confusing.

There are different reasons why an institution would opt for multi-entity processing among multiple campuses. For multi-entity institutions, sharing data enables campuses to use commonly accessed information, thereby reducing the disk space that database tables require. Sharing Oracle databases among entities conserves memory needs on the system and reduces the level of redundant system administration needed for managing the application. Degree Works has the ability to support multiple campuses sharing a variety of academic and biographic data, in addition to the sharing of academic requirements among campuses. Degree Works may be configured to share certain tables among some or all campuses, and also allows certain tables to be configured uniquely for individual campus entities.

Institutions supporting only one campus or entity need not configure anything special for their campus installation or updates; they may use default values delivered with the product.

Institutions supporting multiple campuses or entities have several options when activating multi-entity processing. These options include:

1. Individual Oracle database schemas for each campus entity with unique data for each campus (that is, nothing shared)
2. Individual Oracle database schemas for each campus entity with some shared data used by some or all campus entities, along with some unshared campus-specific data
3. A commonly shared Oracle database schemas with common access to all data by each campus entity

The Multiple-campus case study focuses on scenario #2 in which an institution deploys individual databases with some shared data and some unique data. The case study is intended to familiarize institutions with some of the configuration issues to consider before adopting multi-entity processing for multiple campuses. For more information, see the [Multiple-campus case study](#) topic.

Multiple-campus case study

Updated: March 24, 2023

In this case study, the Eloyce Community College District is comprised of four campus entities: North College, South College, East College, and West College.

Students can attend any of the four campuses sequentially or simultaneously, with classes taken at any campus yielding in-residence credit. The four campuses share a common course numbering system with similarly named courses also having common course content. Not every class is offered at every campus entity. It is assumed that students will generate audits at those district institutions at which they are matriculated, degree-seeking students. In such cases,

students will have one or more rad_goal_dtl (Degree records) for each campus at which they have declared an intended degree.

To reduce database maintenance while giving students and advisors optimum Degree Works services, Eloyce plans to configure Degree Works to share as much data as possible between the four colleges or campus entities. Data to be shared includes the course catalog, all student academic data, and UCX validation codes and literals. Each campus entity, however, will maintain its own set of degree requirements and wants the Degree Works audit worksheets to reflect each campus' needs and preferences. To accommodate sharing, Eloyce will want to allow common access to most of the RAD database tables. To assure individual maintenance of degree requirements and other audit needs, Eloyce will want to use the DAP database tables in the proper configuration for individual campus entities.

RAD tables

The student data imported from Eloyce's student system is housed in these tables:

```
rad_primary_mst  
rad_student_mst  
rad_noncrse_dtl  
rad_test_dtl  
rad_report_dtl  
rad_attr_dtl  
rad_applicnt_dtl  
rad_previnst_dtl  
rad_class_dtl  
rad_custom_dtl  
rad_transfer_dtl  
rad_term_dtl  
rad_aid_dtl
```

Configuring Degree Works to share these tables guarantees that all campus entities use the same set of classes, test scores, and other academic information for each student, regardless of the campus or campuses at which they matriculate. Eloyce does want to assure accurate auditing of academic information against the program requirements for each campus entity, but also wishes to conserve table maintenance and discspace. For that reason, it will share some RAD table information among campuses, but it will not share all institutional data.

To achieve the desired outcome, the District will not share the Degree records. These tables are struck through on the above list to demonstrate that it is not to be shared. The Degree record tables contain a student's degree, major, minor, catalog-year, and so on and really need to be linked to the degree-granting campus entity. In other words, this table should be unique to each campus entity or college so that if a student only matriculates at North College, the other campus entities would not run audits or review data for that student. If the student is matriculated at both the North and South campuses, each of those entities will have their own degree record for that student, and an audit may be generated for that student at each of those campuses.

Each campus will bridge data from the student system that includes any coursework from the District campuses, but each campus will only evaluate data for matriculated students with declared degrees at its campus. North College should be sending to Degree Works all of the students who have matriculated at North regardless of where within the four colleges students have actually taken coursework. A student matriculated at North College may have taken classes at North, South, East, and West, but only the student system extract for North should be pulling this

student's Degree records data. North will also extract the coursework taken at all four campuses. Extracts generated at South, East, and West will not extract this student at all, because the student is not a matriculated degree-seeking student at those campuses and the student's Degree records are not shared.

Some students may be matriculated at more than one campus, in which case their student data will be bridged to each matriculated campus and available for auditing. When extracts are run at each campus where a student is matriculated, that student's data will be extracted multiple times, but their shared data will only exist one time in Degree Works, while their unique campus Degree records will exist for each campus.

DAP tables

As stated above, DAP database tables will be configured to reflect the unique degree requirements for each campus. For example, the humanities requirement at North College is similar but somewhat different from the humanities requirement at East College. Eloyce will configure Degree Works so that the requirements tables are not shared among the campuses, and so that each entity has its own table.

```
dap_req_block  
dap_req_link_dtl  
dap_req_crs_dtl
```

Likewise, degree audits need to be retained by each campus entity individually, and therefore audit tables are not shared. Additionally, the exceptions and notes associated with those audits are not shared.

```
dap_student_mst  
dap_except_dtl  
dap_note_dtl  
dap_note_txt_dtl  
dap_audit_dtl  
dap_audit_tree  
dap_audtree_dtl
```

When using the Curriculum Planning Assistant tool the audits are extracted to the tables noted below. These tables would also be unique to each campus entity and remain unshared.

```
dap_result_dtl  
dap_resclass_dtl  
dap_resnoncr_dtl
```

Academic plans for students are linked to their academic requirements. That also dictates that planner tables should not be shared and that each campus entity would maintain unique planner data.

```
sep_plan_*  
sep_tmpl_*
```

For more information, see the [Multi-entity processing database tables](#) topic. Note that the list of UCX_* tables can be reviewed using Controller.

Multi-entity processing database tables

Updated: March 24, 2023

Several Degree Works tables may be configured for multi-entity processing at multiple campuses when creating the database.

The Degree Works database is created using the dbbuild script. For more information, see the [dbbuild script](#) topic. Be sure to consult with Ellucian staff to configure your share.xml and before running dbbuild.

This is a list of Degree Works tables that may be configured by institutions wishing to use multi-entity processing at multiple campuses. Sites considering use of multi-entity capabilities may want to schedule a technical consulting call with Ellucian to review their plans and particular needs before activating these features.

Table name	Description
dap_applicant_mst	Transfer Equivalency student applicant record
dap_audit_dtl	Stores a degree audit
dap_audit_tree	Stores the body of the degree audit - a true BLOB
dap_audtree_dtl	Stores the body of the degree audit - like a BLOB
dap_eqv_crs_mst	Stores a college equivalence records
dap_except_dtl	Stores audit exceptions/waivers
dap_map_cond_dtl	Transfer Equivalency articulation data
dap_mapping_dtl	Transfer Equivalency articulation data
dap_next_id_mst	Sequence numbers for scribe blocks, audits, and so on
dap_note_dtl	Stores notes for a student – when created, by whom, and so on
dap_note_txt_dtl	Stores the actual note text
dap_req_crs_dtl	Stores list of courses referenced in a scribe block
dap_req_link_dtl	Stores links from one scribe block to another
dap_req_block	Stores primary and secondary tags for each scribe block
dap_resclass_dtl	Stores audit CPA data
dap_resnoncr_dtl	Stores audit CPA data
dap_result_dtl	Stores audit CPA data
dap_student_mst	Stores date/time of last audit and locking information
dap_transfer_dtl	Transfer Equivalency articulation data

Table name	Description
dap_undecide_dtl	Transfer Equivalency articulation data
rad_aid_dtl	added in DW 4.0.0
rad_applicnt_dtl	Transfer Equivalency application data
rad_attr_dtl	Stores attributes about each class the student has taken - transfer_dtl and class_dtl
rad_class_dtl	Stores in-residence classes taken - historic and in-progress
rad_course_mst	Stores courses offered by the institution: title, credits, and so on
rad_crs_attr_dtl	Stores attributes about each class offered by the institution - associated with the course-mst
rad_custom_dtl	Stores other information about the student need by scribe requirements
rad_ets_mst	Transfer Equivalency list of transfer schools
rad_goal_dtl	Stores school, degree, student level, catalog year information
rad_goaldata_dtl	Stores fields of study, such as major, minor, concentration, and advisor information
rad_log_dtl	Log of bridge activity
rad_next_id_mst	Next-id information for courses, students, and ETS
rad_noncrse_dtl	Stores student non-course data
rad_previnst_dtl	Stores student's previous degree information
rad_primary_mst	Stores student name
rad_report_dtl	Stores other student data that needs to appear on the worksheet
rad_student_mst	Stores the student's active term
rad_swap_id_dtl	Stores a record of when a student ID was changed from one value to another through the bridge
rad_term_dtl	Stores student cum GPA/credits
rad_test_dtl	Stores student test score information
rad_transfer_dtl	Stores transfer class information
rad_hash_mst	Stores a record of what was loaded for this student; works with all rad_*_hsh tables
rad_aid_hsh	Stores hash value for the aid_dtl data
rad_applicnt_hsh	Stores hash value for the applicnt_dtl data
rad_attr_hsh	Stores hash value for the attr_dtl data
rad_class_hsh	Stores hash for the class_dtl data

Table name	Description
rad_custom_hsh	Stores hash for the custom_dtl data
rad_goal_hsh	Stores hash for the rad_goal_dtl
rad_goaldtdata_hsh	Stores hash for the rad_goaldtdata_dtl
rad_noncrse_hsh	Stores hash for the noncrse_dtl data
rad_previnst_hsh	Stores hash for the previnst_dtl data
rad_report_hsh	Stores hash for the report_dtl data
rad_student_hsh	Stores hash value for the primary (name), biog (birthdate and SSN), and student (active-term) data
rad_term_hsh	Stores hash for the term_dtl data
rad_test_hsh	Stores hash for the test_dtl data
rad_transfer_hsh	Stores hash for the transfer_dtl data
shp_group_mst	Loaded from SHP077; specifies default keys/access for each user class
shp_log_dtl	Stores web activity information
shp_settings_mst	Stores the Shepherd settings list
shp_user_mst	Stores user's ID and password and primary user class

Repeated classes

Updated: March 25, 2022

The Auditor needs to know what rules to follow when applying Repeated Classes to the audit. Repeated classes are identified using the rad_repeat_ptr and rad_repeat_plcy fields on the rad_class_dtl.

If the rad_repeat_ptr field is non-blank it is assumed to be a repeat. The rad_repeat_ptr field contains the discipline and number of the course it is repeating – regardless of which occurrence it is. The rad_repeat_plcy tells Degree Works how to handle the repeat set with regard to credit and GPA calculations. Your school's forgiveness policy is important in determining which repeat policy to use here.

Repeated Courses whose course numbers have changed

If a student originally took a course as MATH 150 and retakes it as MATH 153 or STAT 153, the Auditor needs to know that these two classes are associated. The rad_repeat_ptr field on the rad_class_dtl can record this.

Degree Works Repeat policies

Policy 1	The credits and grade points of the last (most recent) occurrence are counted by Degree Works. Earlier occurrences are forced to the
----------	--

"insufficient" section of the audit. Those in insufficient do not count in GPA calculations.

Most recent counts – others do not.

Policy 2	<p>The credits and grade points of the class with the best grade are counted by Degree Works. The other occurrences go to "insufficient". Those in insufficient do not count in GPA calculations. Best grade counts – others do not.</p> <p>However, when two classes have the same grade, Degree Works needs to know which class to keep and which should be placed in insufficient. The Additional Control Flag in AUD047 is used for this situation. You can set it to O to keep the Oldest class and N to keep the Newest class when two classes have the same grade.</p>
Policy 3	<p>All occurrences are counted by Degree Works. The audit should apply courses wherever they fit or in the "fallthrough" or "insufficient" section of the audit. All occurrences count and are treated separately.</p>
Policy 4	<p>All sets of grades and grade points count for GPA calculation. The credits from the last (most recent) occurrence are counted by Degree Works. The last occurrence is applied by Degree Works and all other occurrences go to "insufficient". Most recent apply to rules – others count in GPA calculation.</p>
Policy 5	<p>All occurrences of the class should be listed on the Degree Works audit where they could apply (that is, all the occurrences stay grouped together by Degree Works). All sets of grades and grade points are used in the GPA calculation, but only credits for the occurrence with the best grade are counted by Degree Works. The best grade is used by Degree Works when checking MINGRADE.</p> <p>All appear together on a rule and all count in the GPA calculation.</p>
Policy 6	<p>All occurrences with Repeat Policy 6 are applied by the Auditor wherever they fit or in the "fallthrough" section of the audit. Other occurrences of this course that are not tagged with policy 6 (should be tagged with 0) are forced to "insufficient". Those in insufficient do not count in GPA calculations.</p> <p>All with policy 6 apply to rules – all with policy 0 do not count in GPA calculations.</p>
Policy 7	<p>If the class is not found in this table the repeat limit is assumed to be 1. The class with the best grade is kept. If multiple classes have the same grade then the most recently taken class is kept – but a completed class is kept over an in-progress class. If a class is found in this table for the term range specified then the repeat limit is obeyed. Classes that are not kept are marked with an "insufficient reason" of "WG" (Worst Grade). These WG classes are moved from the insufficient section to the over-the-limit (Not Counted) section by the auditor but only after the auditor has</p>

calculated overall the major GPAs based on what was in the insufficient section. The fact that these insufficient classes appear in the over-the-limit section of the audit is a display issue only that is part of this policy.

Policy B	<p>Banner only.</p> <p>If the SHRTCKN_REPEAT_COURSE_IND is equal to an "E" and the Excluded class is NOT skipped the following rule shall apply: the rad_credits_earn, rad_gpa_credits and rad_grade_points will be set to "000000", the rad_insuff_flag will be set to Y and the rad_repeat_ptr and rad_repeat_plcy will be blanked out. This allows these special classes to be displayed in the insufficient section of the report, but with no impact to the credits earned or GPA.</p> <p>If the SHRTCKN_REPEAT_COURSE_IND is equal to "A" and the Averaged class is NOT skipped the following rule shall apply: the rad_insuff_flag will be set to Y and the rad_repeat_ptr and rad_repeat_plcy will be blanked out. This allows these repeated classes to be displayed in the insufficient section of the report, but they will still impact the GPA.</p> <p>If the SHRTCKN_REPEAT_COURSE_IND is equal to "I" the class will apply to rules as a normal class. The rad_repeat_ptr and rad_repeat_plcy will be blanked out for these classes.</p>
----------	---

Split credits

Updated: March 25, 2022

Split credits occur in situations where a course is valued at more credits than is required.

In that situation, what should happen to the "excess" credits? The default behavior of Degree Works is to move the course to another block or to the fall-through section of the audit if MAXTERM or MAXCREDITS is exceeded. The Scribe reserved words SPMAXTERM and SPMAXCREDIT can be used to force the excess credits to be applied by the Auditor to other places in the audit. Use the SPMAXTERM or SPMAXCREDIT block qualifiers when the Auditor Engine should split the credits automatically.

Syntax for split credits

Updated: March 25, 2022

The Parser Engine allows two block qualifiers for split credits: SPMAXTERM and SPMAXCREDIT. These two reserved words are extensions of the MAXTERM and MAXCREDIT qualifiers. The "SP" prefacing MAXTERM and MAXCREDIT signals to the Auditor Engine that the credits are to be split if the maximum is reached.

The format for MAXCREDIT is:

```
MAXCREDIT[S] real [IN | FROM] course_list
```

The format for SPMAXCREDIT follows:

```
SPMAXCREDIT[S] real [IN | FROM] course_list
```

The format for MAXTERM is:

```
MAXTERM {real CREDIT[S] | int CLASS[ES]} [IN | FROM] course_list
```

The format for SPMAXTERM follows:

```
SPMAXTERM {real CREDIT[S] [IN | FROM] course_list
```

where:

- course_list is a list of courses, for example, MATH 101, 102, CHEM 300:400, PHYS @
- int is a numeric value from 1 to 3 digits long
- real is a decimal value with up to 3 digits on each side of the decimal

SPMAXTERM and SPMAXCREDIT are only allowed as block qualifiers. SPMAXTERM and SPMAXCREDIT are NOT allowed as rule qualifiers. SPMAXTERM is only allowed with CREDITS.

Scribe Guided Edit Mode is not set up to add these qualifiers. While in guided edit mode, you may construct your qualifiers as if they were MAXCREDITS and MAXTERM qualifiers and manually enter the necessary "SP" prefix.

Auditor Engine processing of split credits

Updated: March 25, 2022

The Auditor Engine is able to determine where the excess should be applied by examining the block in which the SPMAXTERM or SPMAXCREDIT is found.

If this qualifier is in the starting block then the Auditor puts all excess credits and classes into the over-the-limit list. If the split credit qualifier is found in any block except the starting block, then the Auditor tries to apply the excess credits to rules in next or previous blocks. If the excess cannot be applied to other blocks then the excess goes to fall-through.

Class count

Updated: March 25, 2022

If a course is split across two blocks, BLOCK1 and BLOCK2, then the course is counted as a class in the class totals for BLOCK1 and is also counted as a class in the totals for BLOCK2. It does, however, only count one time in the overall class totals for all blocks.

```
BEGIN # BLOCK1
  SPMAXCREDITS 3 IN CHEM ª
  MAXCLASSES 2 IN CHEM ª
;
  9 CREDITS IN CHEM ª, PHYS ª, MATH ª LABEL "Science stuff";
END.

BEGIN # BLOCK2
  MAXCLASSES 2 IN CHEM ª
;
  7 CREDITS IN CHEM ª, GEOG ª LABEL "More science";
END.
```

If CHEM 105 was taken for 5 credits then the Auditor Engine may apply 3 credits to BLOCK1 and split the course so that the other 2 credits can be applied to BLOCK2. The CHEM course would be counted as a course in the MAXCLASSES qualifier in both blocks.

Exclusivity

Updated: March 25, 2022

Splitting a class between two blocks does not mean that the class is considered NONEXCLUSIVE. It will not be counted in a block's NONEXCLUSIVE count because the split course will be treated as if it is really two different courses.

```
BEGIN # BLOCK1    # not the starting block
  SPMAXCREDITS 3 IN CHEM ª
  NONEXCLUSIVE 1 CLASSES (ALLBLOCKS)
;
  9 CREDITS IN CHEM ª, PHYS ª, MATH ª LABEL "Science stuff";
END.

BEGIN # BLOCK2
;
  7 CREDITS IN CHEM ª, GEOG ª LABEL "More science";
  1 class in math 1ª label "low level math";
END.
```

If CHEM 120 is applied to BLOCK1 for 5 credits the Auditor Engine may keep three of the credits in BLOCK1 and apply the other 2 credits to BLOCK2. If MATH 156 is applied to BLOCK1, the Auditor Engine would see that it can also apply this class to BLOCK2 because of the NONEXCLUSIVE qualifier. CHEM 120 is not considered as a NONEXCLUSIVE class while the placing of MATH 156 into two blocks is. CHEM 120 could, however, be applied nonexclusively to another block for 3 credits instead of MATH 156. Only the portion of the course that is applied to BLOCK1 can be applied elsewhere as a nonexclusive course.

Note: It is recommended that you do not use NONEXCLUSIVE with either SPMAXCREDITS or SPMAXTERM in the same Scribe block. The example shown above is for the purpose of illustrating the exclusivity of classes that have been split between blocks and is not an endorsement of the use of these two header qualifiers

concurrently. The use of this combination of qualifiers may give unpredictable audit results.

Multiple splits

Updated: March 25, 2022

A class can be split as many times as needed. The course may have been taken for a non-integer credit amount or the SPMAXCREDITS could have specified a non-integer value number of credits.

```
BEGIN # BLOCK1 - not the starting block
      SPMAXCREDITS 3 IN CHEM @
      ;
      9 CREDITS IN CHEM @, PHYS @, MATH @ LABEL "Science stuff";
END.

BEGIN # BLOCK2 - not the starting block
      SPMAXCREDITS 1.5 IN CHEM @
      ;
      7 CREDITS IN CHEM @, GEOG @ LABEL "More science";
END.

BEGIN # BLOCK3
      ;
      4 CREDITS IN CHEM @, GEOG @, ANTH @ LABEL "Even More science";
END.
```

If CHEM 120 was taken for 6 credits then the Auditor Engine could apply 3 credits to BLOCK1, 1.5 credits to BLOCK2, and the remaining 1.5 credits to BLOCK3. If CHEM 120 was taken for 6.7 credits then the Auditor could apply 3 credits to BLOCK1, 1.5 credits to BLOCK2, and the remaining 2.2 credits to BLOCK3.

Output

Updated: March 25, 2022

On output, Degree Works on the Web shows the courses that were split with their modified number of credits on the rule to which they were applied. Any part of a split credit course that ended up in over-the-limit or fall-through is also reported with its modified number of credits.

Split power

Updated: March 25, 2022

One block has the power to tell the Auditor Engine that a course is to be split across blocks. The block to which the remaining credits may be applied does not have to indicate that it is OK to split credits.

```
BEGIN # BLOCK1 - not the starting block
    SPMAXCREDITS 3 IN CHEM @
;
9 CREDITS IN CHEM @, PHYS @, MATH @ LABEL "Science stuff";
END.

BEGIN # BLOCK2
;
7 CREDITS IN CHEM @, GEOG @ LABEL "More science";
END.
```

If CHEM 140 (5 credits) is applied to BLOCK1 then the Auditor Engine can split this course into BLOCK1 and BLOCK2 even though BLOCK2 does not explicitly give permission. As long as the Auditor Engine has permission from one of the blocks it can split the course as necessary.

Other qualifiers

Updated: March 25, 2022

The split credit qualifiers cannot change the behavior of other block or rule qualifiers. The maximum number of credits specified by another qualifier cannot be exceeded even if a split credit qualifier exists in the same block header. Another block qualifier may remove the whole course from a block without allowing the split credits qualifier to split a course.

```
BEGIN
    MAXPERDISC 2 CREDITS IN (CHEM, PHYS)
    SPMAXCREDITS 3 IN CHEM @
;
9 CREDITS IN CHEM @, PHYS @, MATH @ LABEL "Science stuff";
END.
```

If a 4 credit chemistry course fits on this block then the Auditor Engine will end up removing the whole course because of the MAXPERDISC qualifier. This qualifier does not have the capability of splitting courses and therefore does not do so. Blocks like the above should be avoided. The qualifiers that you insert into blocks should not conflict with each other.

The Auditor Engine processes the split credit qualifiers first, so courses will be split, if needed, before other qualifiers are encountered.

```
BEGIN
    MAXPERDISC 3 CREDITS IN (CHEM)
    SPMAXCREDITS 3 IN CHEM @
;
9 CREDITS IN CHEM @, PHYS @, MATH @ LABEL "Science stuff";
END.
```

If a 4 credit chemistry course fits on this block then the Auditor Engine first splits the course. It then continues to process the MAXPERDISC qualifier and sees that the maximum has not been exceeded.

If there are multiple split credit qualifiers in the block then the Auditor Engine must obey all qualifiers.

```
BEGIN
    SPMAXTERM 2 CREDITS IN CHEM @
    SPMAXCREDITS 3 IN CHEM @
    ;
    9 CREDITS IN CHEM @, PHYS @, MATH @ LABEL "Science stuff";
END.
```

If a 4 credit chemistry course fits on this block then the Auditor Engine first splits the course leaving 3 credits in this block because of SPMAXCREDITS. It then continues to process the SPMAXTERM qualifier and sees that the maximum has been exceeded and splits the course leaving only 2 credits in this block.

Best fit

Updated: March 25, 2022

Due to the best fit algorithm used by the Auditor Engine, a course may be placed on a rule without splitting it across multiple rules. The split credit qualifier is only used by the Auditor Engine after it places the course in the current block as the best fit.

```
BEGIN  # BLOCK1
;
10 CREDITS IN CHEM 102, 103, PHYS 113:120;
END.

BEGIN  # BLOCK2  # not the starting block
SPMAXCREDITS 2 IN CHEM 102
;
10 CREDITS IN CHEM @, PHYS @;
END.
```

If a student takes CHEM 102 for 5 credits, then there are two possible places that the course could fit. The Auditor Engine will see that the best fit for the course is in BLOCK1 (all else being equal, e.g., no PHYS courses were taken). The Auditor Engine will NOT try to apply 2 credits of CHEM 102 to BLOCK2 and the rest to BLOCK1. If the SPMAXCREDITS was on BLOCK1 then the Auditor Engine would indeed split the credits across the blocks.

Fall through

Updated: March 25, 2022

If a split course only fits or is only needed in one location then the remaining number of credits goes to fall-through if the split qualifier is not in the starting block.

```
BEGIN      # not the starting block
  SPMAXCREDITS 2 IN CHEM @
;
10 CREDITS IN CHEM @, PHYS @;
END.
```

If this is the only place where CHEM 102 can be applied then the Auditor Engine will split this 4 credit course leaving 2 here and putting the remaining 2 credits in fall-through.

Over-the-limit

Updated: March 25, 2022

The remaining number of credits goes to over-the-limit if the split qualifier is in the starting block.

```
BEGIN      # IS the starting block
  SPMAXCREDITS 2 IN CHEM @
;
1 block (major);
END.

BEGIN      # some other block
;
10 CREDITS IN CHEM @, PHYS @;
END.
```

The Auditor Engine will split this 4 credit course leaving 2 credits in the block where it is applied and putting the remaining 2 credits in over-the-limit.

Fall-through split

Updated: March 25, 2022

A split qualifier on the starting block counts those courses applied to all blocks and those in the fall-through list. When processing a split qualifier the Auditor Engine may split a course that is in fall-through by putting part of it into over-the-limit to satisfy the maximum.

Multiple split qualifiers

Updated: March 25, 2022

In a situation like that below the smallest number of the two will be kept on rules. If a 5 credit CHEM course is taken then 3 credits of CHEM will end up in Over-The-Limit.

```
BEGIN  * * Starting block
      SPMAXCREDITS 3 IN CHEM ª
      SPMAXTERM 2 CREDITS IN CHEM ª
;
...
...
...
END.
```

Course Link configuration

Updated: March 24, 2023

Degree Works users can click on courses listed in the worksheet advice or on courses in plans to see a description of the course.

The description may contain a configurable listing of course content, pre-requisites, course name changes, and so on. The RPT036 Create Course Link flag enables or disables this feature by worksheet. For each component of Course Link that can display, there is a Course Link Order field, which is used to control the order of each component on the page. To prevent an item from appearing, set the order value to 0 (zero).

In the RPT036 Course Link Order flags, you can define whether the window should show the TITLE, ATTRIBUTE, SECTIONS, and TRANSFER groups.

TITLE – (RPT050)

The BRIEF version shows only the course key and title, while the STANDARD and VERBOSE versions show the course key, credits, and title.

ATTRIBUTE – (RPT052)

The BRIEF and STANDARD modes list the attribute codes, while the VERBOSE mode shows both the attribute code and attribute description.

SECTIONS – (RPT054)

For the course sections, the extract checks for a start date newer than two weeks ago. This means the most current term will appear if it started within the last two weeks, in addition to all future terms also appearing. This logic may be altered by changing the CFG020 COURSELINK Sections Start Term and Sections Term Days Old to overwrite the two-week logic and setting the Sections Start Term to be the first term that should appear, or by changing the Sections Term Days Old to be more or fewer than 14 days of the two-week logic. You can also alter the SSBSECT3 SQL in the integration.banner.extract.config Shepherd setting to control exactly which sections appear. You may want to hide certain future terms, for example.

In all modes, the term, CRN, section, seats open, and meeting times appear. In Verbose mode, the course title also appears, which is useful for variable-title courses.

To hide the **Seats open** column, you can use Controller to localize these two properties in DashboardMessages.properties to use a tab character:

```
dash.courseLink.sections.header.seats=\t  
dash.courseLink.sections.header.seats.format=\t
```

See the other `dash.courseLink.sections` properties for other changes you may want to make.

TRANSFER – (RPT056)

The Transfer Equivalency product maintains a library of course mappings for articulation between the institution and feeder schools from which students transfer or attend classes not offered by the institution. The basic Transfer (often referenced as Course Finder) functionality is a feature of the Transfer Equivalency Self-Service component of Transfer Equivalency and is accessible through Course Link as an information group.

In Brief mode, the local course key, transfer school name, and transfer course key display in a tabular format with column headings. In Standard mode, the information shows in a pseudo-tabular format without column headings. In Verbose mode, a different tabular structure is used to include the transfer school's city and state information.

Degree Works accessibility compliance

Updated: March 25, 2022

Degree Works adheres to WCAG 2.1 AA guidelines for accessibility

Degree Works applications follow WCAG 2.1 AA guidelines for accessibility. Ellucian completes a Voluntary Product Accessibility Template (VPAT) for each application to better specify the level of compliance in the product. These are available from your Customer Service Manager.

Western character support

Updated: September 30, 2022

Degree Works does not currently support the UTF-8 database character set, but it does support 8-bit Western characters in student names, course titles, and other data bridged from the Student Information System (SIS).

When 8-bit Western character support is enabled (specifically, Windows-1252 character encoding), the additional letters and those with pre-composed diacritics commonly found in Latin alphabets like Spanish, French, Norwegian, and so on will display correctly in Degree Works applications (Responsive Dashboard, Controller, Transit, Transfer Equivalency Self-Service, and Transfer Equivalency Admin) in the following data:

- Student and advisor names
- Course titles
- Transfer course titles

- Transfer school names
- Student notes (audit and planner)
- UCX descriptions, such as goal data and terms
- Scribe Label, Remark, ProxyAdvice, and Display text
- Scribe disciplines, majors, and so on
- CourseLink data

Enabling 8-bit Western character support at your site requires configuration changes.

Note that this functionality was introduced with Release 5.1.0, so if Degree Works was installed on this version or later, this functionality should already be enabled.

Note: For MEP clients, each change starting with step 2 should be made in each MEP entity.

1. Change the NLS_CHARACTERSET in the Degree Works database to **WE8MSWIN1252**.

As an administrative user on your classic server, enter the following information to determine how the character set is configured in your database:

```
db
SELECT value FROM NLS_DATABASE_PARAMETERS WHERE parameter = 'NLS_
CHARACTERSET' ;
```

- If the NLS_CHARACTERSET parameter is not currently set to WE8MSWIN1252, it needs to be changed. Setting this parameter is a multi-step process that needs to be performed by a database administrator.
- If your Transit schema is in a database different from the Degree Works schema, this change needs to be made to the Transit database.
- No changes to the Banner database character set are required.
- No conversion of the Degree Works data is needed after this character set is changed. However, you will need to run the extract or bridge to refresh data with special characters.

2. Add the NLS_LANG environment variable to dwenv.config so Oracle uses the AMERICAN_AMERICA.WE8MSWIN1252 language/territory/character set.

- a. In the DATABASE Variables section of dwenv.config (located in \$DGWBASE), use a text editor to add the following variable, if it does not already exist:

```
# NLS_LANG - tell Oracle what language, territory and character
# set to use.
# Even if you are not in the United States please leave this setting as-is
# as it controls Oracle errors, date formats etc. Changing this to anything
# other than "AMERICAN_AMERICA" may cause problems.
# The WE8MSWIN1252 character set (8-bit, Western character set)
# allows
# non-ASCII characters to be used. This can be US7ASCII instead
```

```
d but then  
# only ASCII characters can be used.  
NLS_LANG="AMERICAN_AMERICA.WE8MSWIN1252"  
export NLS_LANG
```

- b. Exit your session and log in again so the environment variable gets set.
3. Change the classicConnector.serverCharset Shepherd setting to **Windows-1252**.

Before Release 5.1.0, the baseline value for the classicConnector.serverCharset Shepherd setting was ISO-8859-1. To enable 8-bit Western character support, use Controller to change this setting to **Windows-1252**.

4. Restart the daemons on your classic server.

```
cd $HOME  
daprestart && webrestart && tberestart
```

- If you use prerequisite checking, issue the following:
`prerestart`
- If you run the CPA daemons, issue the following:
`resrestart`
- If you use RAD08, issue the following:
`radrestart`

5. Restart the Degree Works applications on your java application server.

6. Run the extracts to update data in the Degree Works database.

Re-bridge data from your SIS to Degree Works so special characters are stored correctly in the database. Typically you will only need to run the student data extract, but depending where special characters are used at your site, you may need to run additional extracts such as the advisor and course extracts.

Because student data did not change in the SIS, you need to run the Banner student extract or the RAD11 job (for non-Banner schools) with the force flag set. Every student will get a new audit but this also ensures that the student names are written to the database using the updated character set.

- Banner schools - run RAD30 in Transit and select the **Force new audits** check box.
- Non-Banner schools - set the following export before running the RAD11JOB:
`export RAD11FORCE=TRUE`
`RAD11JOB filename`

Database tables

Updated: March 25, 2022

Degree Works stores and uses data in numerous database tables. This includes student, course, mapping and validation data from the student system, degree block, audit and results data, planner and template data and Transit job data.

DAP tables

Updated: March 24, 2023

Database tables with the DAP prefix store Degree Works-generated data such as Scribe blocks, audit trees, CPA results, exceptions, audit notes and petitions, and Transfer Equivalency applicant data.

Table	Description
dap_appdata_dtl	This table stores additional goal data for applicants processed in Transfer Equivalency.
dap_applicnt_mst	This table stores information on applicants processed in Transfer Equivalency.
dap_audit_dtl	This table contains the description of an audit. Each degree audit that is performed, other than What-If, has an entry. There is one dap_audit_dtl for each dap_audit_id. Audit results that may need to be used as selection criteria are stored here.
dap_audit_tree	This table stores the steno audit trees created by the Auditor Engine when the DW_AUDIT_BLOB environment variable is set to 1. The audit is stored in a BLOB field.
dap_audtree_dtl	This table stores the steno audit trees created by the Auditor Engine when the DW_AUDIT_BLOB environment variable is set to 0. Each entry is one line of an audit tree. The audit tree is binary data.
dap_eqv_crs_mst	This table tracks course equivalence through history across catalog years. As course numbers change or are reused, entries in this table map from the course the student took to the course for a specific catalog year. The key is a concatenated list of catalog year of student course plus discipline and number of student course plus catalog year being evaluated. For example, "1990 MATH 301 1995 " maps to "MATH 310 ", that is, MATH 301 taken in the 1990 catalog year is equivalent to MATH 310 in the 1995 catalog year.
dap_except_dtl	This table records manual overrides to the Auditor engine. Sample exceptions are waivers, substitutions, or lock-ins. There is one dap_except_dtl per exception, with a maximum of 9,999 exceptions per student/school/degree combination. Each exception is tied to a particular requirement in a requirements block. This table links the requirement to the exception and the exception to the student.
dap_gpa_history	This table stores overall and major GPA information by term from the student's audit. Used in the Tracking component of the new generation SEP.
dap_map_attr_dtl	This table stores the attributes related to the mappings in the dap_mapping_dtl.
dap_map_cond_dtl	This table stores the conditions related to the mappings in the dap_mapping_dtl.
dap_mapping_dtl	This table stores the mappings created in Transfer Equivalency.
dap_next_id_mst	This table contains the next available ID for Mappings, Requirement blocks and Audits. dap_next_key is "M" for mappings, "R" for

Table	Description
	requirements and "A" for audits. dap_next_id is of the form "Mxnnnnnn" for mappings, "Rxnnnnnn" for requirements and "Axnnnnnn" for audits. "x" is a letter, A-Z. "nnnnnn" is a number, sequentially assigned from 000001 to 999999. When 999999 is reached the "x" value is changed to the next letter. There is a maximum of 26 million requirement blocks or degree audits.
dap_note_dtl	This table contains the header information about a Degree Works note. It contains global information about a note for a student, such as note-status and note-type. There is one dap_note_dtl per student per note.
dap_note_txt_dtl	This table contains the text of a Degree Works note. Each entry contains one line of text per note. Sorting on note_num and note_seq displays the text of the note in order.
dap_req_block	This table describes a block of requirements. It associates a requirements ID with the database tags (Catalog Year, Degree, etc.) for the block.
dap_req_crs_dtl	This table houses the courses referenced in each of the requirement blocks. This table allows you to easily find out the blocks in which a course is referenced – for maintenance purposes.
dap_req_link_dtl	This table points to other blocks that were referenced in the requirements text using the BLOCK keyword. Degree Works uses this to know when a change to one block may affect another block and to ensure that a block is not circular (block A links to block B which links back to block A).
dap_resclass_dtl	This table stores class information used in an audit to be used by CPA.
dap_resnoncr_dtl	This table stores noncourse information used in an audit to be used by CPA.
dap_result_dtl	This table stores audit information to be used by CPA.
dap_student_mst	This table tracks the most recent activity for a student. An entry is created for each student who is audited or has exceptions or notes. There is one entry per student. This table also tracks whether or not the student is locked for processing.
dap_transfer_dtl	This table stores transfer and advanced placement exam information for Transfer Equivalency.
dap_undecide_dtl	This table stores information about unresolved articulation results.

RAD tables

Database tables with the RAD prefix store data bridged from the student system.

Table	Description
rad_aid_dtl	Stores financial aid data to be used in the Financial Aid audit worksheet.
rad_aid_hsh	Stores hash value for the aid_dtl data.
rad_applicant_dtl	Transfer Equivalency application data.
rad_applicant_hsh	Stores hash value for the applicant_dtl data.
rad_attr_dtl	Stores attributes about each class the student has taken – transfer_dtl and class_dtl.
rad_attr_hsh	Stores hash value for the attr_dtl data.
rad_class_dtl	Stores in-residence classes taken – historic and in-progress.
rad_class_hsh	Stores hash for the class_dtl data.
rad_course_mst	Stores courses offered by the institution: title, credits, and so on.
rad_crs_attr_dtl	Stores attributes about each class offered by the institution – associated with the course-mst.
rad_currrule_dtl	Stores curriculum rule data bridged from the SIS and used for What-If and Transfer Equivalency goal data filtering.
rad_custom_dtl	Stores other information about the student need by scribe requirements.
rad_custom_hsh	Stores hash for the custom_dtl data.
rad_ets_mst	Transfer Equivalency list of transfer schools.
rad_goal_dtl	Stores school, degree, student level, catalog year information.
rad_goal_hsh	Stores hash for the rad_goal_dtl.
rad_goaldtldata_dtl	Stores fields of study, such as major, minor, concentration, in addition to advisor information.
rad_goaldtldata_hsh	Stores hash for the rad_goaldtldata_dtl.
rad_hash_mst	Stores a record of what was loaded for this student; works with all rad_*_hsh tables.
rad_log_dtl	Log of bridge activity.
rad_next_id_mst	Next-id information for courses, students and ETS.
rad_noncrse_dtl	Stores student non-course data.
rad_noncrse_hsh	Stores hash for the noncrse_dtl data.
rad_previnst_dtl	Stores student's previous degree information.
rad_previnst_hsh	Stores hash for the previnst_dtl data.
rad_primary_mst	Stores student name.
rad_report_dtl	Stores other student data that needs to appear on the worksheet.
rad_report_hsh	Stores hash for the report_dtl data.
rad_student_hsh	Stores hash value for the primary (name), biog (birthdate and SSN) and student (active-term) data.
rad_student_mst	Stores the student's active term.

Table	Description
rad_swap_id_dtl	Stores a record of when a student ID was changed from one value to another through the bridge.
rad_term_dtl	Stores student cum GPA/credits.
rad_term_hsh	Stores hash for the term_dtl data.
rad_test_dtl	Stores student test score information.
rad_test_hsh	Stores hash for the test_dtl data.
rad_transfer_dtl	Stores transfer class information.
rad_transfer_hsh	Stores hash for the transfer_dtl data.

SHP tables

Updated: March 24, 2023

Database tables with the SHP prefix store Degree Works data used for user authorization and application localization.

Table	Description
shp_composer_mst	Stores baseline and localized Shepherd Script records.
shp_group_mst	Loaded from SHP077; specifies default keys/access for each user class.
shp_log_dtl	Stores web activity information.
shp_resource_mst	Stores baseline and localized application properties records.
shp_settings_mst	Stores environment and application configurations.
shp_user_mst	Stores user's ID and password and primary user class.

SEP tables

Database tables with the SEP prefix store Degree Works plan and template data for the Student Educational Planner.

Table	Description
sep_plan_class	Plan class information.
sep_plan_class_note	Plan class note.
sep_plan_gpa	Plan gpa information.
sep_plan_gpa_note	Plan gpa notes.
sep_plan_group	Plan group information.
sep_plan_noncourse	Plan non-course information.

Table	Description
sep_plan_noncourse_note	Plan non-course note.
sep_plan_note	Plan notes.
sep_plan_placeholder	Plan placeholder information.
sep_plan_placeholder_note	Plan placeholder note.
sep_plan_term	Plan term information.
sep_plan_term_note	Plan term note.
sep_plan_test	Plan test information.
sep_plan_test_note	Plan test notes.
sep_tmpl_class	Template class information.
sep_tmpl_class_note	Template class note.
sep_tmpl_gpa	Template gpa information
sep_tmpl_gpa_note	Template gpa notes.
sep_tmpl_group	Template group information.
sep_tmpl_noncourse	Template non-course information.
sep_tmpl_noncourse_note	Template non-course note.
sep_tmpl_note	Template notes.
sep_tmpl_placeholder	Template placeholder information.
sep_tmpl_placeholder_note	Template placeholder note.
sep_tmpl_term	Template term information.
sep_tmpl_term_note	Template term note.
sep_tmpl_test	Template test information.
sep_tmpl_test_note	Template test notes.

Transit tables

Updated: March 25, 2022

Database tables with the TRANSIT prefix store Transit jobs and artifacts.

The Transit tables are in a different schema than the rest of the Degree Works tables. On the classic server see the \$DB_LOGIN_TRANSIT variable to determine where the Transit tables reside. On the classic server you can run dbt to run sqlplus to connect to the Transit schema.

Table	Description
TRANSIT_JOB_INSTANCE	This table stores information on requested, running, scheduled, terminated, and completed jobs.

Table	Description
TRANSIT_ARTIFACT	This table hold one artifact of a job. An artifact is some kind of output from the job such as a report, stdout, or action report.
TRANSIT_INPUT	This table holds bulk input data for jobs. This is typically data intended to be loaded into the database, such as blocks to be imported using the DAP41 job. It is used only by certain jobs that ask for an input file when they are launched in Transit.
TRANSIT_SEED	Holds the next available job number.
DATABASECHANGELOG	Used by the database maintenance utility to track database changes. Not used by Degree Works applications directly.
DATABASECHANGELOGLOCK	Used by the database maintenance utility to track database changes. Not used by Degree Works applications directly.

Degree Works special scripts

Updated: September 29, 2023

The scripts that Degree Works uses will not always be accessed through a user interface. Most of the scripts are used within other scripts and processes.

Warning! Do not place scripts into the local/scripts directory as they are not used. Only those used in app/scripts are used.

Script	Description
bannerextract	Batch extract Banner data.
changepassword	Finds the shp_user_mst using the input Access ID (shp_access_id) and replaces the shp_access_code with the input Password. For more information, see the changepassword script topic.
dap16all	Reparses all the blocks in the database; it simply calls DAP16JOB.
dap22dbg	Runs and tars up information on the audit for this student. \$ dap22dbg 123456 mytarfile
dap22ids	Runs audits based on the student IDs in specified file – must be in data directory; use parameter file common/DAP22IDS: example: dap22ids MYSTUIDS
dapauditstopdffiles	Takes an input file of audit IDs and the name of a FOP stylesheet and creates a separate PDF file for each audit. See below for more details. For more information, see the dapauditstopdffiles script topic.

Script	Description
dapauditstoxmlfiles	Takes an input file of audit IDs and creates a separate XML file for each audit. See below for more details. For more information, see the dapauditstoxmlfiles script topic.
dapauditstoxml	Runs the getxmlaudit against the audits in the db and sends results to a single file; you can modify the script to select a subset of audits/students. For more information, see the dapauditstoxml script topic.
dapblockinsert	Inserts each of the blocks found in the admin(blocks directory into the db; see the script for the format of the header each block must have for it to find the block type and title; the script prompts the user for the start and stop catalog year. For more information, see the dapblockinsert script topic.
dapblockload	Loads the dap_req_block and dap_next_id_mst (domain of RA or RB) from files. You must specify a parameter of R to do all requirements or RB to only load in the blocks starting with RB or RA for the non-planner blocks. The RB blocks are those generated from plans for students. Note, you cannot use the load/unload scripts to copy between two environments on two different versions of Degree Works if the dap_req_block database tables are different in structure. Instead of this script, Ellucian recommends using DAP41 Load Scribe Blocks in Transit.
dapblocksget	<p>Use this to back-up your blocks to a text file named blocks.out. The output file will contain all primary and secondary tags and the block text.</p> <pre>\$ dapblocksget MAJOR 2015 # get the major blocks for this catalog year</pre> <pre>\$ dapblocksget MAJOR # get the major blocks for all catalog years</pre> <pre>\$ dapblocksget # get all blocks</pre>
dapblockunload	Unload the dap_req_block and dap_next_id_mst (domain of RA or RB) to files. The default is to unload only the RA blocks to files. You can specify a parameter of R to do all requirements or RB to only unload the blocks starting with RB. The RB blocks are those generated from plans for students. Note, you cannot use the load/unload scripts to copy between two environments on two different versions of Degree Works if the dap_req_block database tables are different in structure. Instead of this script, Ellucian recommends using DAP40 Unload Scribe Blocks in Transit.
dapdelaudits	Deletes audits older than the specified date. See the Freezing Audits section for more information about deleting frozen audits.
dapfindbadaudits	Displays a list of audits by student ID, audit ID and audit date that are believed to be corrupt. See the section below for more information. For more information, see the dapfindbadaudits script topic.

Script	Description
dapfindorphanedaudits	Displays a list of audits by student ID, audit ID, old degree and new degree that are associated with a student in the rad_goal_dtl that now has a different school/degree. For more information, see the dapfindorphanedaudits script topic.
dapmapload	Loads the dap_mapping_dtl, dap_map_cond_dtl, and dap_next_id_mst (domain of M) from files. Note, you cannot use the load/unload scripts to copy between two environments on two different versions of Degree Works if the dap_mapping_dtl or dap_map_cond_dtl database tables are different in structure. Instead of this script, Ellucian recommends using DAP43 Load Mappings in Transit.
dapmapupload	Unloads the dap_mapping_dtl, dap_map_cond_dtl, and dap_next_id_mst (domain of M) to files. Note, you cannot use the load/unload scripts to copy between two environments on two different versions of Degree Works if the dap_mapping_dtl or dap_map_cond_dtl database tables are different in structure. Instead of this script, Ellucian recommends using DAP42 Unload Mappings in Transit.
dapreqcrs	Finds the blocks the reference a particular course (used by SCR02JOB).
dapreqgrep	Searches through the text on the dap_req_tree for the string specified (used by SCR10JOB).
dapreqlist	Gets a listing of block type, block value, title, catalog years and parse status for all blocks. (used by SCR05JOB).
dapreq2ndlist	Gets a listing of block type, block value, title, catalog years and parse status and all of the secondary tags for all blocks. Used by SCR06JOB.
daprestart	Runs dapstop and then dapstart – and optionally with debugging on.
dapsendemail	Send an email to this address with the contents of this file as the body with this subject. Example: \$ dapsendemail myfriend@myschool.edu thisfile.log "Results of query"
dapshow	Shows the dap08 process.
dapstart	Starts dap08 for use with Scribe.
dapstop	Stops the dap08 process.
dapucx2eqv	Loads the equivalences from CFG070 into the dap-eqv-crs-mst.
dapucxload	Loads the UCX from a file. Instead of this script, Ellucian recommends using DAP45 Load UCX tables in Transit.
dapucxunload	Unloads all of the UCX to a file. Instead of this script, Ellucian recommends using DAP44 Unload UCX tables in Transit.
dapxpt	Shows the list of exceptions saved to the database.

Script	Description
dbbuild	Modify or create Degree Works database tables and objects. For more information, see the dbbuild script topic.
debugoff	Exports DWDEBUG=0; unsets DW_LOGDEBUG_PID.
debugon	Exports DWDEBUG=1; export DW_LOGDEBUG_PID to the 1 st param specified.
deleteall	Used to delete ALL student data from a Degree Works database. This script calls deleteall.sql which truncates the Degree Works database tables which store student data. Only students are affected, other users are not deleted.
deletestu	Used to delete student data from Degree Works database based on bridge date. Prompts user for date in YYYYMMDD format; all student data bridged <= that date will be deleted. Only Banner schools can use this script.
deleteunhooked	<p>Delete orphaned unhooked and unenforced exceptions. You can specify N or Y to control whether or not the exceptions will be deleted after they are shown. If no parameter is given the user is prompted for a decision.</p> <p>Orphaned exceptions are those that have no corresponding rad_goal_dtl. This happens when a student changes degrees.</p> <p>The vieworphaned option in ADMIN in Transit runs deleteunhooked with a parameter of N. The deleteorphaned option in Transit runs deleteunhooked with a parameter of Y.</p>
dgversion	Shows version information for Degree Works source code.
dwsettings	Import, Export, Delete, or Overwrite on the shp_settings_mst table. For more information, see the dwsettings script topic.
etssync	Synchronize the rad_ets_mst records from the global directory. For more information run "etssync --help".
exptable	<p>Exports the data for a given table to a file – to be later imported using importall;</p> <p>example:</p> <pre>exptable dap_next_id_mst myfile "dap_next_id like 'P%'"</pre> <p>Note: Only tables listed in dapdb, ucxdb, shpdb, or raddb in the schema directory can be used with this script.</p>
getxmlaudit	<p>Unloads an audit from the db and runs dapext to create an xml <Audit> tree.</p> <p>example: getxmlaudit AA000123</p>

Script	Description
	For more information, see the getxmlaudit script topic.
importall	Imports the data in the given file (created by exptable).
	example: importall SHP myfile
keycheck	Check the keys on all shp_group_mst and on non-student shp_user_mst records against SHP078. Report those groups/ accessIds that have keys that are obsolete or not valid and what the keys are.
packdebug	Reads debugging output from specified log files by sessionId and creates a zip file with an encrypted key. The script should be run in the target directory where log files exist, or should be provided complete log file path for the parameters. For more information, issue the packdebug --help command. For more information, see the packdebug script topic.
petsend	Sends an email to the given address notifying of WAITING or APPROVED petitions.
profiledbg	Analyzes a debug file that contains timing entries and creates a data file that can be imported into a spreadsheet for analysis. Requires knowledge of the internals of the Degree Works programs, and so is to be used under direction from the Degree Works Action Line. For more information, see the profiledbg script topic.
rad30dbg	Runs and tar up lots of good information on the Banner extract. The tar file created will always be rad30dbg.tar. It will overwrite a previously existing file. Examples:
	<pre>\$ rad30dbg student [student ID] \$ rad30dbg student 123456</pre>
	<pre>\$ rad30dbg student [student id-file] \$ rad30dbg student stuselect.ids</pre>
	<pre>\$ rad30dbg [any bannerextract mode] \$ rad30dbg advisor</pre>
radrestart	Runs radstop and then radstart
radshow	Shows the rad08 processes and its children
radstart	Starts rad08 – dynamic bridge daemon
radstop	Stops the rad08 processes
resrestart	Runs resstop and then resstart
resshow	Shows the dap25 parent and any running child processes
resstart	Starts dap25 – dynamic CPA daemon
resstop	Stops the dap25 process
rmoldfiles	A new script created to help you keep the logdebug, dgwspool and data directories clean of files building up. You can add rmoldfiles to your cron job running weekly or nightly. The 1st parameter is the

Script	Description
	directory name and the second parameter is the age of the file in days; if not supplied the default is 7 days:
Examples:	
<pre>\$ rmoldfiles logdebug \$ rmoldfiles /dw/admin/data \$ rmoldfiles dgwspool 7 \$ rmoldfiles /dw/admin/jobdata \$ rmoldfiles /dw/admin/dapaudit</pre>	
To add the rmoldfiles script to your cron job you can run crontab -e and edit the file to run the rmoldfiles script. The user that owns the crontab file must have proper permissions to delete the logdebug files.	
<pre>\$ crontab -e # min hr dm mo dw script # Run rmoldfiles every Sunday morning to delete # logdebug files older than 7 days \$ crontab -e # min hr dm mo dw script # Run rmoldfiles every Sunday morning to delete # logdebug files older than 7 days 0 5 * * 0 /dw/app/scripts/rmoldfiles /dw /admin/logdebug 7 >/tmp/rmlogdebug.log 2>&1 0 5 * * 0 /dw/app/scripts/rmoldfiles /dw /admin/dgwspool 15 >/tmp/rmdgwspool.log 2>&1 0 5 * * 0 /dw/app/scripts/rmoldfiles /dw /admin/data 20 >/tmp/rmdata.log 2>&1\</pre>	
sepdeleteplan	Delete a plan from the sep_plan table and all of its related tables by the student ID or the plan ID. Use “sepdeleteplan -h” for more information.
sepdeletetemplate	Delete a template from the sep_tmpl_mst table and all of its related tables by the template ID. Use “sepdeletetemplate -h” for more information.
setdbpasswords	Configures the values for the Degree Works database password, Student Information System database password and Transit database password. You may give the passwords either as option flags to the command or as positional parameters. If given as positional parameters, the Degree Works password is first. If neither of those is provided, the script will prompt for the values. The SIS password is optional. For more information, issue the setdbpasswords --help command.
showdbpasswords	Displays the encrypted string that represents the database passwords. This is typically used in the configuration of the data source for the java applications. For more information, issue the showdbpasswords --help command.

Script	Description
sharegen	Produces schema file(s) for the purpose of separating shared and unique tables for a Degree Works database that supports multiple institutions. Called by the dbbuild script. For more information, see the sharegen script topic.
shareinfo	Displays a list of all Degree Works tables with their associated owners. For more information, see the shareinfo script topic.
tableload	Load in a file that was created from a tableunload. Before running tableload you should empty the contents of the table in the target database. Use “tableload HELP” for more information. Note, you cannot use the load/unload scripts to copy between two environments on two different versions of Degree Works.
tableunload	Unload a database table to a file. This is useful for copy the contents of a table from your test environment to your production environment, for example. Use “tableunload HELP” for more information. Note, you cannot use the load/unload scripts to copy between two environments on two different versions of Degree Works.
ucxsync	Synchronize certain UCX tables loaded on different classic servers. For more information run “ucxsync --help”.
webanalyze	Script to analyze your web.log file. You can optionally email the results to someone; setting this up in cron to run daily is a good idea. Example: webanalyze - defaults to admin/logdebug/web.log Example: webanalyze myold.web.log Example: webanalyze web.log me@myschool.edu
weblogon	Get a list of keys or groups for a user.
webrestart	Runs webstop and then webstart – and optionally with debugging on.
webshow	Shows the web07 processes
webstart	Starts web07 for use with the web.
webstats	This is a tool to produce statistics on the performance of the web daemons by analyzing the web.log file. Its primary purpose is to produce a data file containing a time series of web daemon metrics. This includes the periodic measurements of the average transaction duration and maximum transaction count for each of web07. The comma-delimited file can be input into a spreadsheet or other statistical program for further analysis and graphing.
webstop	Stops web07 daemon processes.
webtime	Script to check how long web requests are taking to process. Example: webtime - defaults to admin/logdebug/web.log Example: webtime myold.web.log

changepassword script

Updated: March 25, 2022

The changepassword command changes the password stored in the shp_access_code on the shp_user_mst. It accepts a Shepherd Access ID (maximum of 14 contiguous characters) and Password (maximum of 64 contiguous characters) as input parameters.

The command must be executed from the system command prompt. The input Access ID and Password are passed onto the SHP31 program which then finds the shp_user_mst for the Access ID and updates the shp_access_code with the new password.

Format

```
changepassword <AccessId> <Password>
```

Example

```
$ changepassword 12345678901234 this-is-my-password-with-no-spaces
```

If you do not supply both the Access ID and a Password you will be prompted for them.

Note: No blanks are allowed in the Password. The first BLANK found will signify the end of the Password.

dap22dbg

Updated: March 25, 2022

Use the dap22dbg script to run and tar up lots of good information on the audit for this student.

```
$ dap22dbg 123456 mytarfile
```

Send the tar file created to Ellucian for analysis.

You can also run dap22dbg from Transit using the ADMIN report option. Simply supply a student ID and launch the job and a tar file will be created for you to download (as binary) and send to the Action Line on your open case.

dapauditstopdffiles script

Updated: March 25, 2022

This script allows you to create a PDF file for each audit ID specified in the input file. You can generate a file of audit IDs for students who have graduated or simply for archival purposes.

You run the script specifying two parameters. The first parameter is the file of audit IDs while the second parameter is the name of the FOP XSL stylesheet. For example:

```
$ dapauditstopdffiles gradstudents.txt fopaudits.xsl
```

Will use the fopaudits.xsl file pull from the shp_resource_mst to process each of the audit IDs found in gradstudents.txt.

You may also want to place a list of audits for some of your student athletes into a file and specify the athletic eligibility stylesheet.

```
$ dapauditstopdffiles stuathletes.txt fopaudits-athl.xsl
```

When creating PDF files you should run your athletic audits separate from your financial aid audits separate from your academic audits. You need to do this because you will want to specify a different FOP stylesheet for each type of audit.

When getting started, it is recommended you perform a test with 10 audit-ids in a file. Make sure the files get created without error. Also check the size of the files created. You can then extrapolate based on these 10 files to see how much free space you will need on the system when you run the script against your big list of audits.

The PDF creation process is not fast; it may take several seconds per audit. You may want to run the job in the background to allow it to run overnight in the background, as follows:

```
$ dapauditstopdffiles gradstudents.txt fopaudits.xsl > pdf.out 2>&1 &
```

When it completes you can review the pdf.out file for errors.

The format of each PDF file generated is: <stuid>~<school>~<degree>~<auditid>.pdf

Example: 9837631~UG~BA~AA000123.pdf

The files are placed in the current directory so you may want to create a special directory in which to run this script.

Note: You may want to archive particular frozen audits. You should decide which frozen audits you want to archive and then run SQL like that below to export these IDs to a file. Don't forget to specify the audit-type also to get either the Academic Audits (AA), Athletic Eligibility Audits (AE) or Financial Aid audits (FA). See UCX-AUD032 for a list of the freeze-type values.

```
select dap_audit_id from dap_audit_dtl  
where dap_freeze_type='FRZTYP' and dap_audit_type='AA'  
order by dap_audit_id
```

dapauditstoxmlfiles script

Updated: March 25, 2022

This script allows you to create an XML file for each audit ID specified in the input file. You can generate a file of audit IDs for students who have graduated or simply for archival purposes.

You run the script specifying a single parameter - the file of audit IDs. For example:

```
$ dapauditstoxmlfiles gradstudents.txt
```

Will process each of the audit IDs found in gradstudents.txt.

When getting started, it is recommended you perform a test with 10 audit-ids in a file. Make sure the files get created without error. Also check the size of the files created. You can then extrapolate based on these 10 files to see how much free space you will need on the system when you run the script against your big list of audits.

Unlike the corresponding PDF script, the XML creation process is fairly fast. However, you still may want to run the job in the background to allow it to run in the background – like this:

```
$ dapauditstoxmlfiles gradstudents.txt > xml.out 2>&1 &
```

When it completes you can review the xml.out file for errors.

The format of each XML file generated is: <stuid>~<school>~<degree>~<auditid>.xml

Example: 9837631~UG~BA~AA000123.xml

The files are placed in the current directory so you may want to create a special directory in which to run this script.

Note: You may want to archive particular frozen audits. You should decide which frozen audits you want to archive and then run SQL like that below to export these IDs to a file. Don't forget to specify the audit-type also to get either the Academic Audits (AA), Athletic Eligibility Audits (AE) or Financial Aid audits (FA). See UCX-AUD032 for a list of the freeze-type values.

```
select dap_audit_id from dap_audit_dtl  
where dap_freeze_type='FRZTYP' and dap_audit_type='AA'  
order by dap_audit_id
```

dapauditstoxml script

Updated: March 25, 2022

Extracts all audits and converts to xml using the getxmlaudit script. All audits are placed in a single XML file called allaudits.xml. Each audit is enclosed within start and end <Audit> xml tags as with Web XML audits.

The SQL WHERE clause in the CreateSQLFile function may be modified to select a subset of your audits based on date, audit-id, school, degree, student level, etc – anything on the dap-audit-dtl record. You may want to run this script several times using different criteria to keep the resulting xml file from getting too large. A single audit can easily be 100K meaning an extract of 100 audits would result in at least a 10MB file. You should conduct a few tests to see how big your files will be and adjust the WHERE clause accordingly – the size of audits differs from school to school.

To execute simply run the command at the UNIX prompt. The allaudits.xml file will be placed in the current directory. A “Processing audit” message will appear as each audit is processed.

```
$ dapauditstoxml  
Processing audit = [AE000268] ...  
Processing audit = [AE000273] ...  
Processing audit = [AE000269] ...  
Processing audit = [AE000259] ...
```

You may redirect the output to a file if you do not want to see the Processing messages:

```
$ dapauditstoxml > outputfile
```

You may want to compress the big xml file using some compression tool (like gzip) and move the file off the system. You can always uncompress the file and view and search through the xml data as needed.

dapblockinsert script

Updated: March 25, 2022

Place the blocks you want loaded into the admin(blocks directory).

Ensure that the blocks directory is empty before placing your new set of files there.

The script loads the dap_req_block table with the blocks found. The text of the block is loaded into a CLOB field in the table.

Run the script to load the blocks. Enter the appropriate catalog years to use for all blocks when prompted. These catalog years are used as the default in case a block is encountered that does not have the catalog years defined.

```
$ dapblockinsert  
Please enter the start catalog year > 20072008  
Please enter the stop catalog year > 99999999
```

You will see messages go to the screen. When it gets through all of the blocks it will process the SQL Insert statements – this may take a few minutes if you have a lot of blocks.

The script writes information to a dapblockinsert.log file in the logdebug directory. Please review it.

Be sure to run dap16 to parse the blocks and then fix the errors using Scribe.

Each file must contain header information and start with two #'s characters:

Line 1: can be the school name but it is ignored so it really can be anything

Line 2: must contain the block type and value

Line 3: must contain the block title

Line 4: optional – can contain the starting and ending catalog years separated by a hyphen.

Example:

```
##Ellucian University
##MAJOR=ACCT
##Major in Accounting
##2012-9999
```

```
BEGIN
```

```
;
```

```
END.
```

dapfindbadaudits script

Updated: March 25, 2022

Shows a list of audits by student ID, audit ID, and audit date that are believed to be corrupt.

The user is then given the opportunity to delete these bad audits and also is given directions on how to run new audits for the students with these corrupt audits. When new audits are created and saved to the database Degree Works may encounter a problem saving the new records to the dap_auditree_dtl table. This is usually caused by running out of space in the database. When this occurs Degree Works tags the dap_audit_dtl record and it is this tag that this script looks for to report the corrupted audits.

Regardless of whether the user answers Y or N to delete the audits a file of student IDs is saved to the admin/data directory. This file can then be run with the dap22ids script to create new audits for these students.

Note: The dapfindbadaudits script does not work on audits stored in the dap_audit_tree table. This occurs when the DW_AUDIT_BLOB environment variable is set to 1 and the audit is stored in a BLOB field.

```
$ dapfindbadaudits
These are the audits that are most likely corrupt.
This corruption is usually caused by running out of room in your dat
abase.
Here you see the student ID, audit ID and create date for each corru
pt audit.
```

```
Processing .. findbad.sql.5579
select dap_stu_id, dap_audit_id, dap_audit_date from    dap_audit_dtl
where
STUDENT_ID AUDIT_ID CREATE_DATE
===== ===== =====
N88665547  AA044158  20100527
N00010771  AA044159  20100527
```

```
N00010771 AA044160 20100527
N00011380 AA044162 20100527
N00011380 AA044163 20100527
N00011380 AA044164 20100527

6 rows selected
Do you want to delete these bad audits? (y/N) > y

Deleting bad/corrupt audits now...
Deleted 6 audits from the database
The list of students with bad audits has been saved to this file:
/dwprod/admin/data/studentswithbadaudits.ids
You may run "dap22ids studentswithbadaudits.ids" to create new audit
s for these students.
```

dapfindorphandedaudits script

Updated: March 25, 2022

This script shows a list of audits by student ID, audit ID, and old degree and new degree believed to be orphaned.

That is, these audits exist for students who have changed their degree from when the audit was generated. The user is then given the opportunity to delete these orphaned audits and the CPA. Note that frozen audits are ignored as are what-if audits.

Please note that this script removes all CPA data for the student identified with the orphaned audit; the script does not only delete the CPA records for the audit. You may want to recreate the CPA data for the students' newest audit after this script deletes the CPA data.

```
$ dapfindorphandedaudits
These are the audits that belong to students who changed their degree.
Here you see the student ID, audit ID and old and new degrees for each.
(Note that frozen audits are ignored; they are not considered orphaned.)
(Note that what-if audits are also ignored.)
```

```
Processing .. findorphanded.sql.17139
select dap_stu_id Student_Id, dap_audit_id Audit_id, dap_degree Old_Degree,
STUDENT_ID AUDIT_ID OLD_DEGREE      NEW_DEGREE
===== ===== ===== =====
210009206  AA043626 BA          BBA
210009301  AA043632 BA          DIPL
210009206  AA043703 BA          BBA
210009301  AA043709 BA          DIPL
210009206  AA043780 BA          BBA
210009301  AA043786 BA          DIPL
```

```
HERMIONE AA045100 MA BS
HERMIONE AA045101 BA BS
HERMIONE AA045103 MA BS
HERMIONE AA045104 BA BS
HERMIONE AA045351 GRAD_MA BS
HERMIONE AA045353 MA BS
HERMIONE AA045354 BA BS
HP AA045372 BA-ENGLISH BFA
HP AA045372 BA-ENGLISH BA
HP AA045372 BA-ENGLISH BS
```

16 rows selected

Do you want to delete these orphaned audits and associated CPA data?
(y/N) >
Orphaned audits have not been deleted.

dbbuild script

Updated: March 25, 2022

This script creates all the database objects needed by the Degree Works software.

It generates the table definitions from the dwschema.xml definition file, using the share.xml configuration file, which defines the relationship

between the various schema owners and their shared tables. It also loads all the necessary ancillary views and packages.

Options

- --debug - Output debugging messages to stderr.
- --help - Display the help screen.
- --pause - Pauses the script after each step.
- --share - Specifies the sharing configuration file.
- --usage - Display command syntax only.
- --verbose - Display progress information.
- --version - Display the version of this script.

The sharing configuration file must be configured for your particular sharing needs. It can be provided as the parameter to the --share option, or it will default to share.xml in the schema directory.

The --dbuser option can provide a database user that has the ability to create tables for all the schemas. If this option is omitted, then the value of the DB_LOGIN environment variable is used.

An experienced user can use the --pause option to step through specific parts of this command. At the beginning of each step, you will be prompted to either stop, continue, or skip just the next step.

The database, tablespaces, and users must be created before running this command. They are not created by the command.

Files

\$DGHOME/schema/dwschema.xml - is a file that contains the xml definition of all the required tables. This file is provided by Ellucian, and should not be modified by the client.

\$DGHOME/schema/share.xml - the default value for the file that defines the various schema owners and the tables they share. For multi-entity clients, this file must be configured and provided through the --share option.

debugon and debugoff

Updated: March 25, 2022

The debugon alias makes it easier for you to turn on debugging when requested by Ellucian.

Issuing "debugon" works in addition to "debugon frog" - the logdebug files created will have "frog" in their names so that you can easily find the ones that you created. Note that you must do a webrestart or daprestart within five minutes of issuing the debugon command; this helps prevent you from leaving debugging turned on accidentally.

The debugoff alias helps to easily turn debugging off again:

```
$ debugon frog
$ webrestart
    (do some testing on the dashboard)
$ cd logdebug
$ ll *frog.xml - these will be your debug files

$ env | grep DEBUG
DWDEBUG=1
DW_LOGDEBUG_DIR=/dw/dwprod/admin/logdebug
DW_LOGDEBUG_PID=frog
```

dgwversion

Updated: March 25, 2022

Use the dgwversion script to gather information about what versions of the Degree Works source code is currently in place.

```
$ dgwversion > verions.txt
```

dwsettings script

Updated: March 25, 2022

This script is used to import settings into the shp_settings_mst, export settings into a file, and delete settings from the shp_settings_mst. Additional options allow settings to be encrypted during an import or decrypted on an export.

Note: Settings can contain security sensitive data. Please guard their storage and transport adequately.

The syntax of the command is as follows:

```
dwsettings --export [--decrypt] [--key key[,key...]] file  
dwsettings --import [--encrypt] [--overwrite] file [file...]  
dwsettings --delete [--spec spec] --key key[,key...]  
dwsettings --delete [--spec spec] file.properties
```

Export

The --export option will produce an XML file of the selected keys. See below for the file format. The output can be filtered by a list of comma-delimited keys (with no spaces inbetween) by using --key option. The key is case sensitive. Entries will be selected whose keys start with one of the given keys. For example, “--key core” will select all settings with a key that starts with “core”. If no key is provided, all settings will be exported. Certain Shep settings, such as passwords, are encrypted in the database. Normally, the output would contain the encrypted value, but by using the --decrypt option, the values will be decrypted on export.

Import

The --import option will add settings to the database from one or more files. The files must be either XML (with a suffix “.xml”) or properties (with a suffix of “.properties”) files. There is no requirement that the entries in the file share any kind of key pattern. The properties file should contain one line for each setting in the format key=value. The --spec option may specify the spec value to be used for all entries, and if not provided, “default” will be used. The XML file format is shown below. It contains an attribute for the spec.

Normally, existing settings will not be overwritten on import. Using the --overwrite switch will force an overwrite of existing entries.

Certain Shep settings, such as passwords, are encrypted in the database. It is assumed that the import file contains already encrypted entries. Use the --encrypt option if the file contains plaintext entries that must be encrypted before insertion into the database.

Delete

The --delete option will delete settings from the database. Entries to be deleted may be specified either by listing them after the --key option or by listing them in a properties file. When using the --key option, you may list one or more keys separated by commas with no spaces in between. They keys must exactly match the keys of the settings to be deleted, not just the beginning of the keys, and they are case sensitive. The properties file should contain one line for each setting in the format key=value. The value portion is ignored and may be left blank (but the

equals sign is required). If the --spec value is not provided, it is assumed to be “default”. You may use a value of “@” to specify all specs of the provided key.

XML file format

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<DwConfigurations
    xsi:schemaLocation="urn:com:sungardhe:degreeworks:settings:DwConfigurations.xsd"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

    <DwConfigProperty spec="default"
        key="key">
        <! [CDATA[data value] ]>
    </DwConfigProperty>
</DwConfigurations>
```

The setting key and spec are attributes of the DwConfigProperty element. The data value should be placed in a CDATA enclosure as shown above. If the data value contains the string “\$RANDOM[nn]”, where the nn is a number, the value will be replaced with a randomly generated set of ASCII characters with a length of nn. For example, “\$RANDOM[32]” will produce a random literal 32 characters long.

Examples of how to run dwsettings

```
dwsettings --import infile.xml

<< specify infile.xml as the input file, existing settings will not be overwritten >>

dwsettings --import infile.properties
dwsettings --delete delete.properties

dwsettings --import --overwrite infile.xml
dwsettings --import --overwrite --encrypt infile.xml
dwsettings --import --encrypt infile.xml
dwsettings --import --overwrite delete.properties

dwsettings --export outfile.xml
dwsettings --export --key classicConnector outfile.xml
dwsettings --export --decrypt --key core outfile.xml
dwsettings --export --key core,studentPlanner --decrypt outfile.xml

dwsettings --delete --key my.complete.key
dwsettings --delete --spec test --key my.complete.key
dwsettings --delete --spec @ outfile.properties
```

getxmlaudit script

Updated: March 25, 2022

Extract specified audit from the db and convert to an xml file.

The file name created will be the audit-id with a .xml extension. The file will be placed in the current directory. The file will contain two xml commands.

```
$ getxmlaudit AA000123
```

This will create a file called AA000123.xml.

packdebug script

Updated: March 25, 2022

The packdebug script can be used to collect debug files from the server when a user has enabled debugging from the Java applications running on the Java Application Servers.

It reads debugging output from the specified log files by sessionId and creates a zip file with an encrypted key.

The script should be run in the target directory where log files exist, or should be provided the complete log file path in the parameters. If the input files do not exist, no files will be processed.

Options

- --key <mykey> - A user specified password for output zip file.
- --output <outputZipName> - A user specified file name for output zip file.
- --session <sessionId> - The debug sessionId provide in the UI when debug was enabled.

Examples

```
packdebug --key myKey --output myOutputFile --session 2669bb29-781e-4be4-8a5d-c348ffbd24bc scribe.log  
packdebug --key myKey --output myOutputFile --session 2669bb29-781e-4be4-8a5d-c348ffbd24bc /opt/apache-tomcat2/logs/scribe.log  
packdebug --key myKey --output myOutputFile --session 2669bb29-781e-4be4-8a5d-c348ffbd24bc /opt/apache-tomcat2/logs/ .log
```

preqstats

Updated: March 25, 2022

The preqstats tool produces statistics on the performance of the prerequisite daemons (DAP61 and DAP62) by analyzing the preq.log file.

Its primary purpose is to produce a data file containing a time series of daemon metrics. This includes the periodic measurements of the average transaction duration and maximum transaction counts. The comma-delimited file can be input into a spreadsheet or other statistical program for further analysis and graphing.

The tool can be run without any parameters to analyze your preq.log file. However, you can specify the file as a parameter if you have saved it under a different name. For example, preqstats preq.log.20140301

A summary report is printed to the screen.

```
Preq log file: preq.log.20140301 (3844198 lines)
Generated data file: preqstats.csv
Number of dap61s started = 50
Number of dap62s started = 10
SUMMARY:
Process      Total      Avg       Max
DAP61        494399    1.3632    10
DAP62        4302      0.1315    2
```

It displays the number of daemons originally started at the beginning of the report. It then gives, for each of the servers, the total number of transactions processed (Total), the average duration for a transaction (Avg), and the maximum number of concurrent transactions count (Max). The Max count is the maximum number of concurrent transactions occurring at the same time that were found for the server. For example, the summary above shows that, at some point covered by the preq.log file, there were 10 DAP61 processes processing transactions.

The data file is called preqstats.csv, and contains a series of snapshots at regular 1 minute intervals. Each snapshot contains a timestamp, an elapsed time, and for each daemon, an average duration of its transactions and the maximum concurrent transactions count (Max) in that period, first DAP61 and then DAP62. The average and maximum are calculated for a 5-minute period surrounding the snapshot time. For example, the preqstats.csv file might contain the following entries:

```
"01:54:00",840,1.148393,50,0.10293,8
"01:54:10",850,1.147676,45,0.11909,9
"01:54:20",860,1.145578,52,0.09293,5
"01:54:30",870,1.144776,46,0.10382,6
"01:54:40",880,1.149042,48,0.10292,8
```

At 1:54 (row 1, 1st column), 840 seconds as of the start of the log (2nd column), the average duration for a DAP61 transaction was 1.148393 seconds (3rd column), there was at most 50 DAP61 daemons running (4th column), the average duration for DAP62 was 0.10293 seconds (5th column), and there was at most 8 DAP62 daemons running (6th column) in that time period.

The numbers in the file are moving counts and averages. That means that a record is output every so often, at the value of the interval, by default 1 minute. The average and counts, however, are taken from transactions that occur in a window around that point. The window can be, and by default (5 minutes) is, larger than the interval. So, while we may be outputting records every minute, it will include, by default, the average duration of transactions occurring 2.5 minutes before to 2.5 minutes after that point.

The interval between snapshots can be changed using the --interval option, and the size of the period can be changed with the --window option. Both take parameters in seconds. The name of the data file can be changed with the --datafile option.

```
preqstats --interval=120 --window=600 --datafile mydata.txt
```

This would produce a snapshot every 2 minutes, and each snapshot would cover a 10 minute period centered on the snapshot. The first line of the file contains a heading entry.

The starting point for the first entry is calculated to put it on an even multiple of the interval and to include a full window of transactions, and the last entry is also calculated to contain only a full window. This means that there will be some transactions at the beginning and end that will not be included in the data. This is to prevent outlying data points at the beginning and end.

profiledbg script

Updated: March 25, 2022

Extract performance profiling data from a Degree Works classic debug file. This command is intended for use by the Degree Works development team, and should only be upon consultation with Ellucian.

Synopsis

```
profiledbg [--verbose] [--output <file>] file [file2 ...]
```

Description

This command creates a data file to be used for profile analysis. It uses a file created by the classic Degree Works debugging tool logdebug as input. It matches up LDTime benchmark entries in the file and outputs a comma delimited file with the mark and duration for each transaction. This can be imported into a spreadsheet to create a profile table giving counts average and total durations, etc. To get a debug file to use with this command, export DWBENCHMARK=1 before launching the program being profiled. It can also be used on debug files created with the DWDEBUG environment variable set. If multiple debug files are given, then the output of each will be concatenated into the one output file.

This command may output a message to stderr if it cannot find a matching LDTime begin mark for the end mark it is processing. This is generally the result of a programming error of some sort. For example, exiting a routine without issuing a DEBUG_MODULE_END. It may or may not throw off results. The errant mark is simply discarded.

If no output file is provided, the script will output to a file named profiledbg.out. If you use a single dash (" -") as the output file name, the output will be sent to stdout.

The --verbose switch provides progress notification as the script is executing, including progress dots every 100 lines of file input.

File output

The output file will be in the format:

```
Mark ID,0:0:1.234567890
```

The Mark ID column may have spaces or special characters, but it should not have commas, or that will impede importation into a spreadsheet. There are no guarantees as to the number of digits in the nanoseconds, although it will be 9 or less.

This file can be imported into Excel or another spreadsheet. A pivot table can then be constructed using the Mark as the row and, as columns, a count of the marks, an average of duration, max of duration, standard deviation of duration, sum of duration, and % of column total.

sharegen script

Updated: March 25, 2022

The sharegen tool is intended for use in creating and maintaining the Degree Works database.

It produces schema file(s) for the purpose of separating shared and unique tables for a Degree Works database that supports multiple institutions. As input, it takes an xml document describing the database structure (dwschema.xml), an xml document describing the sharing relationships between different institutions (share.xml), and the current database structure as divined from the native database. As output, it creates a set of sql ddl scripts to create or alter the database to fit the input specification. Output files can be generated in xml or sql format. In order to modify the database, sql files must be generated. The type and location of output files is defined in share.xml.

The sharegen script is called by the dbbuild script (it is never called directly). Before running dbbuild, any new schemas (users) defined by share.xml must be created in the database. The suggested procedure for setting up a schema/user named “dwschema” (for example) is to run the following commands using SQLPlus and substituting the mypassword and tablespace name dgw with a localized values.

1. Create user dwschema identified by mypassword default tablespace dgw quota unlimited on dgw temporary tablespace temp;
2. Grant connect to dwschema;
3. Grant dba to dwschema;
4. Verify the user was created: select * from all_users order by username;

dwschema.xml

This file is hand maintained and contains the general structural information for a Degree Works database, such as dapdb, raddb, etc. It defines the tables, keys, and indexes for the database. It uses the Torque/DdlUtils XML database schema DTD for the definition, which can be found at [H_{http://db.apache.org/ddlutils/schema}H](http://db.apache.org/ddlutils/schema). The file is located in \$DGWHOME/schema/dwschema.xml.

Note that dwschema.xml is provided and maintained by Ellucian and should not be modified.

share.xml

This file is hand maintained and contains the configuration data defining the sharing relationships between client entities. It defines each entity, and groups of these entities, and the tables they share. This XML schema is unique to Ellucian. The default file used for a single school is located in \$DGWHOME/schema/share.xml. An example file for multiple schools sharing some of the Degree Works tables located in \$DGWHOME/schema/mepshareexample.xml An xsd schema file is used to validate the structure and content of share.xml. It is located in \$DGWHOME/schema/dwshare.xsd.

More information on each of the important values in share.xml:

```
$ sharegen --sourceSchema dwschema.xml --shareInput share.xml
```

Depending on the configuration of share.xml this will produce one or more schema xml or sql files.

Usage

```
sharegen [<configfile>] --sourceSchema <sourceSchema>
--jdbcProperties <path/to/jdbc.properties>
[--shareInput <shareInput>] [--validateOnly <validateOnly>]
[--changeDatabase <changeDatabase>] [--createTables <createTables>]
[--dropTablesFirst <dropTablesFirst>] [--continueOnError <continueOnError>]

--sourceSchema <sourceSchema>
    The source schema file to use. Defaults to dwschema.xml.

[--shareInput <shareInput>]
    The schema for which to generate xml schema. Defaults to share.xml.

[--validateOnly <validateOnly>]
    Set validateonly=1 to validate share xml and schema xml. (default: 0)

[--changeDatabase <changeDatabase>]
    Set 1 to change/modify database immediately after creating output files.
    (default: 0)

[--createTables <createTables>]
    Set 1 to specify that CREATE TABLE statements will be generated. Set to
    0 if ALTER TABLE statements should be generated for objects that already
    exist in the database. (default: 0)

[--dropTablesFirst <dropTablesFirst>]
    Set 1 to output DROP TABLE statements in sql when createTables
    =1. This has no effect when createTables=0. (default: 0)

[--continueOnError <continueOnError>]
    Set 1 to continue processing live database changes even if errors occur.
    (default: 0)

[--jdbcProperties <path/to/jdbc.properties>]
    The path to the jdbc.properties file that contain the database connection parameters.
```

Default values

The sharegen script is not run directly; it is called by the dbbuild script. The sharegen script contains default values for dburl, sourceSchema and schemaInput, so those values do not normally need to be entered. When the dbbuild script is run, those values cannot be overridden on the command line.

shareinfo script

Updated: September 29, 2023

The shareinfo command generates a list of database owners and associated table names for all Degree Works tables. If the -s option is used, synonyms are listed instead of table names.

Format

```
shareinfo [-s]
```

Example

===== OWNER =====	===== TABLE =====
DGWDDBA	DAP_APPDATA_DTL
DGWDDBA	DAP_APPLICNT_MST
DGWDDBA	DAP_AUDIT_DTL
DGWDDBA	DAP_AUDIT_TREE
DGWDDBA	DAP_AUDTREE_DTL
DGWDDBA	DAP_EQV_CRS_MST
DGWDDBA	DAP_EXCEPT_DTL
... (and rest of associated tables and any additional users)	

webanalyze

Updated: March 25, 2022

Webanalyze is primarily a post-mortem tool to check to see how the system performed during the period covered by a web.log file.

You can run webanalyze against your current web.logfile or against a saved file. When webanalyze is run against an active web.log, it also displays information about the number of web07 daemons that are still running compared to the number you initiated. However, this number should always match because utl01 creates a new web07 process if one dies. The average time for all web07 requests is also displayed, allowing you to see if performance is getting better or worse over the day's activities.

You can e-mail the results of webanalyze, a useful feature when setting it up in cron to run on a daily or hourly basis. A quick check of the webanalyze results in an e-mail is a simple way to monitor system performance.

You can also run webanalyze by navigating to ADMIN in Transit and selecting webanalyze. The webanalyze results are also available on the Web under the Admin tab.

webstats

Updated: March 25, 2022

The webstats tool produces statistics on the performance of the web daemons by analyzing the web.log file.

Its primary purpose is to produce a data file containing a time series of web server metrics. This includes the periodic measurements of the average transaction duration and maximum transaction count for WEB07. The comma-delimited file can be input into a spreadsheet or other statistical program for further analysis and graphing.

The tool can be run without any parameters to analyze your web.log file. However, you can specify the file as a parameter if you have saved it under a different name. For example:

```
webstats web.log.20140301
```

A summary report is printed to the screen:

```
Web log file: web.log.20140301 (3844198 lines)
Generated data file: webstats.csv
```

```
Number of web07s started = 50
```

```
SUMMARY:
Process      Total        Avg      Max
WEB07      2003274    0.2605      50
```

It displays the number of daemons originally started at the beginning of the report. It then gives the total number of transactions processed (Total), the average duration for a transaction (Avg), and the maximum number of concurrent transaction count (Max). The Max count is the maximum number of concurrent transactions occurring at the same time that were found for the server. For example, the summary above shows that, at some point covered by the web.log file, there were 50 WEB07 processes processing transactions.

The data file is called webstats.csv, and contains a series of snapshots at regular 1 minute intervals. Each snapshot contains a timestamp, an elapsed time, and for WEB07, an average duration of its transactions and the maximum concurrent transaction count (Max) in that period. The average and maximum are calculated for a 5 minute period surrounding the snapshot time. For example, the webstats.csv file might contain the following entries:

```
"01:54:00",840,0.148393,50
"01:54:10",850,0.147676,50
"01:54:20",860,0.145578,50
```

```
"01:54:30",870,0.144776,50
"01:54:40",880,0.149042,50
```

At 1:54 (row 1, 1st column), 840 seconds as of the start of the log (2nd column), the average duration for a WEB07 transaction was 0.148393 seconds (3rd column) and there was at most 50 WEB07 daemons running in that time period (4th column), which lasted from 1:54:00 to 1:54:10.

The numbers in the file are moving counts and averages. That means that a record is output every so often, at the value of the interval, by default 1 minute. The average and counts, however, are taken from transactions that occur in a window around that point. The window can be, and by default (5 minutes) is, larger than the interval. So, while we may be outputting records every minute, it will include, by default, the average duration of transactions occurring 2.5 minutes before to 2.5 minutes after that point.

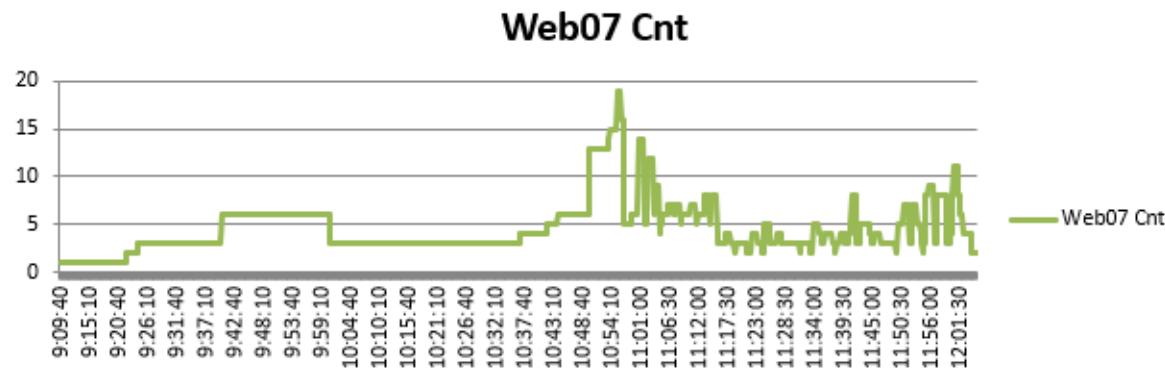
The interval between snapshots can be changed using the --interval option, and the size of the period can be changed with the --window option. Both take parameters in seconds. The name of the data file can be changed with the --datafile option.

```
webstats --interval=120 --window=600 --datafile mydata.txt
```

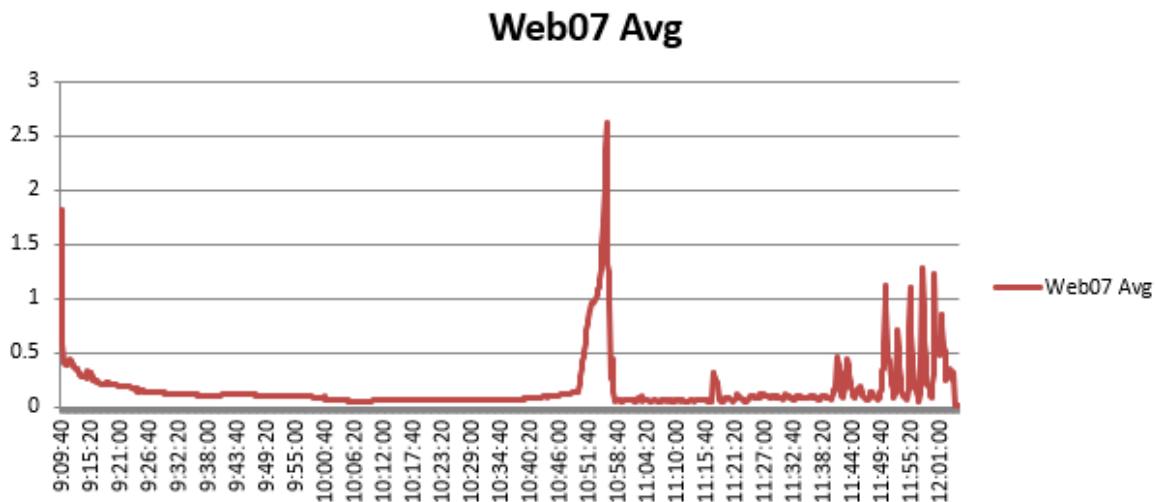
This would produce a snapshot every 2 minutes, and each snapshot would cover a 10 minute period centered on the snapshot. The first line of the file contains a heading entry.

The starting point for the first entry is calculated to put it on an even multiple of the interval and to include a full window of transactions, and the last entry is also calculated to contain only a full window. This means that there will be some transactions at the beginning and end that will not be included in the data. This is to prevent outlying data points at the beginning and end.

The following figures are examples of some graphs that can be produced from the file:



The above graph shows the count values (Cnt heading) from the data file plotted against the timestamp (Time). The graph below shows the average transaction duration (Avg heading) plotted against the timestamp.



You can see in the examples above that there was a period where WEB07 transaction time spiked dramatically. You should check the cpu usage during this time. Also look at the counts for WEB07. If they are approaching the configured maximums then you may want to increase that, so long as you have cpu overhead.

webtime

Updated: March 25, 2022

Webtime is a tool that provides information about the number of transactions.

You can run webtime against your current web.logfile or an older, saved web.logfile.

```
$ webtime  
$ webtime myold.web.log
```

The output provides the time it took for each request and the slowest, fastest and average times. See the sections that follow for more information.

Degree Works security

Updated: March 25, 2022

Security in Degree Works consists of logon authentication, service authorization, and user class assignment.

HTTPS/SSL security

Updated: September 30, 2022

Ellucian strongly recommends that all Java application servers running Degree Works be configured with SSL certificates and to allow only HTTPS/SSL access to applications.

Documentation for SSL certificate setup is widely available from each vendor or product website. Signed certificate providers provide additional documentation that relates to their products and is specific to many common platforms. Clients are responsible for setting this up as part of installation and configuration.

As a best practice, Ellucian recommends that Java applications be configured to not support weak SSL protocols such as SSLv2, SSLv3, TLS 1.0, and TLS 1.1. You should consider adding this environment variable to your deployment configuration to disable insecure protocols, leaving only TLS 1.1 and 1.2 enabled:

```
SERVER_SSL_CIPHERS=TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384,TLS_ECDHE_RSA_WI  
TH_AES_128_GCM_SHA256,TLS_DHE_RSA_WITH_AES_256_GCM_SHA384,TLS_DHE_RS  
A_WIT  
H_AES_128_GCM_SHA256,TLS_ECDHE_RSA_WITH_AES_256_CBC_SHA384,TLS_ECDHE  
_RSA_  
WITH_AES_128_CBC_SHA256,TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384,TLS_  
ECDHE  
_ECDSA_WITH_AES_128_CBC_SHA256
```

Authentication security

Updated: September 30, 2022

Logon authentication determines if the user is who they say they are.

The user typically enters an Access ID and Access Code (password), which is authenticated in a security database. The security database can either be the one supplied with Degree Works, one affiliated with the student system, or some institution-wide central service such as CAS. Logon authentication occurs before granting access to Degree Works.

One of several options for single sign-on (SSO) can be employed when Degree Works is integrated with other systems. Degree Works supports SSO for Luminis, Self-Service Banner, CAS, and SAML 2.0. External access managers such as Oracle Access Manager are also supported.

The changepassword script may be used to change the password (shp_access_code on the shp_user_mst) from the command line. For more information, see the [changepassword script](#) topic.

Banner Sites Only

The Banner extract programs generate default passwords for each student, advisor, and staff member processed. The password will be loaded with one of two forms:

- Password generated from custom SQL
- 10-byte random password

If custom SQL exists in the integration.banner.extract.config Shepherd setting for the appropriate keyword (PASSWORDSTU, PASSWORDADV or PASSWORDSTF), it will be used to read the Banner data and format it into a password (1 to 64-byte password) that will be loaded into the shp_access_code on the shp_user_mst for the individual. If the custom SQL is blank or the password generated from the custom SQL is blank, a 10-byte random alpha-numeric password will be generated.

A Change Password configuration flag, **Update password when bridging users**, exists in the CFG020 WEBPARAMS record. If this flag is set to **N**, the shp_access_code on the shp_user_mst will NOT be changed by the Banner extracts, only added when the student, advisor, or staff record is originally created.

For more information, see the [R171SHPU - SHP User Record](#) topic.

Multiple paths to authentication

Updated: March 25, 2022

When a non-authenticated user attempts to access a secured Degree Works application, the user will normally be redirected to a login page.

There are three possible scenarios for this:

- Native Degree Works login
- CAS single sign-on login
- SAML single sign-on login

With native Degree Works login, the user credentials can be validated against either the native Shepherd user database or an external LDAP server.

To configure the appropriate login scenario, change the core.security.authenticationType Shepherd setting to one of “SHP”, “CAS”, or “SAML”. The configuration requirements for each of these models are described below.

The authenticationType tells Degree Works where, if you are not already authenticated, to send you for authentication. If it is “SHP”, then you will get the built-in Degree Works login page. “CAS” will send you to a CAS server for login, and “SAML” will send you to a SAML server. When the type is set to CAS or SAML you, therefore, cannot use the built-in login pages with the typical Degree Works manager user. This authenticationType is used by all Degree Works applications unless you create separate shep settings for each application. For more information, see the [Multiple authentication entry points](#) topic.

As an alternative, you can configure an external access manager, such as Oracle Access Manager or Shibboleth, to handle authentication. In this scenario all requests are intercepted before they get to Degree Works, authentication is handled by the external access manager, and only authenticated requests are forwarded to Degree Works. For more information, see the [External access manager](#) topic.

Degree Works native login

Updated: March 25, 2022

Degree Works applications contain a login screen that can be used to collect users' credentials for access to the application.

Each application is independent, and usually requires a separate login for each one. We recommend that you look at CAS to provide a single sign-on experience. To use the native login screen, set the `core.security.authenticationType` Shepherd setting to "SHP".

The user's credentials can be validated either using the Degree Works Shepherd user database that is populated through the bridge, or they can be validated using an external LDAP server. If both are configured, then a valid entry in either database will authenticate the user.

Shepherd user database

Degree Works provides a basic user security database for logon authentication and service authorization. In this model, a `shp_access_id` and `shp_access_code` on the `shp_user_mst` is used for each student, advisor, faculty, staff, and administrator authentication.

This model requires that clients send the user class to the Bridge in the SHPU record. For Banner clients this is handled by the Banner extract.

To enable this type of authentication you must set several Shepherd settings in Controller:

- Set `core.security.authenticationType` to "SHP". This will direct users to the native Degree Works login page and authenticate their credentials against either the `shp_user_mst` or and LDAP database, depending on the following settings.
- Set `core.security.shp.authentication.enabled` to "true" to authenticate logon credentials against the `shp_user_mst`.
- Set `core.security.ldap.enable` to "true" to authenticate logon credentials against an LDAP database. See the *LDAP user database* topic for more configuration requirements. Either this setting or `core.security.shp.authentication.enabled` must be set to "true" when using "SHP" authentication. If both are set, valid credentials in either one will allow access to Degree Works.
- Set `core.security.passwordEncoding.enabled` to "true" so Controller and the Java bridges encrypt the password. Note, however, that the Banner Extract and the RAD extract still follows the Encrypt Password flag in the CFG020 WEBPARAMS record in Controller.
- Set `core.security.passwordCheck.sha1.enabled` to "true" to encoded the user's password with the SHA1 hash when comparing against the database password. If the `core.security.passwordEncoding.enabled` is set to "true", so should this setting.

- Set `core.security.passwordCheck.clearText.enabled` to “true” to allow the software to check the clear text (unencoded) password against the database. You would do this if you have passwords that have not been encoded. For example, if you turn on password encoding and do not reload all the users’ passwords, then you will have some passwords stored in the database in unencoded form. You would need to set this to “true” in order for those passwords to work.
- The `core.security.shp.maxLoginAttempts` setting specifies how many login attempts are allowed before further logins are ignored.
- The `core.security.shp.failLoginResetMinutes` specifies the amount of time in minutes after the maximum number of login attempts have been exceeded before the user is allowed to attempt another login.

LDAP user database

Updated: March 25, 2022

LDAP is frequently used as a central repository for user information and as an authentication service.

Degree Works can be configured to use an LDAP server instead of or supplementary to Degree Works native authentication (SHP database). LDAP authentication is enabled by changing the `core.security.ldap.enable` Shepherd setting to “true”. The `core.security.authenticationType` Shepherd setting must be set to “SHP”. Both LDAP and native SHP authentication can be enabled at the same time. In this case, valid credentials in either database will allow access.

No authorization (keys and services) information will be retrieved from LDAP. This is still controlled by Shepherd. An LDAP user must have an LDAP attribute that identifies their user records (`shp_user_mst`) in Degree Works.

LDAP can be configured for authentication in several different ways. You should be generally familiar with LDAP before configuring Degree Works to use it. These instructions do not cover the normal set up and use of an LDAP server.

You must tell the authentication software the location of the LDAP server by setting `core.security.ldap.serverUrl`. For example, this might be something like “`ldaps://ldap.myschool.edu:636/ou=users,dc=myschool,dc=edu`”. You should use the secure LDAP protocol for this.

An administrative user is required to find the user’s record and read attributes in the LDAP database. You configure this user’s dn in `core.security.ldap.adminDn` and their password in `core.security.ldap.adminPassword`. This user must have the ability to search for and return the user’s distinguished name (dn) and to read any users attributes. The password is stored in encrypted form in the settings database.

Degree Works authenticates the user by binding to LDAP with the user’s dn. It determines the dn in one of two possible ways. The first way is to use a pattern set in `core.security.ldap.userDnPattern`. The pattern should contain the token “{0}” that will be replaced by the user’s Access ID (login name). An example would be “`uid={0}`”. This is relative to the base given in the `core.security.ldap.serverUrl` setting. So, if a user with the id “bobstudent” logs in, and the `serverUrl` is set to “`ldaps://ldap.myschool.edu:636`”, and the

userDnPattern is set to “uid={0}”, then the software would expect the user’s unique dn to be “ou=users,dc=myschool,dc=edu,uid=bobstudent”.

The second way to retrieve the user’s dn is to do a search with an LDAP filter expression in the core.security.ldap.userSearchFilter setting. This follows the format specified in RFC 4515. You should include a special token, “{0}”, in your expression that will be replaced by the user’s Access ID (login user name). For example, “(uid={0})”. The scope of this search may be further limited by setting a search base in the setting core.security.ldap.userSearchBase. For more information about search filters see <https://tools.ietf.org/html/rfc4515>.

After the distinguished name is found, the user will be authenticated with a direct bind to the LDAP server using the password provided by the user in the login process.

Degree Works ID within LDAP location

When authentication is successful, then Degree Works needs to find out how to locate or “map” the LDAP user to Degree Works internal database to determine their authorities. The username provided may not be the Degree Works ID, which is the ID from the Student Information System, and it may not be the primary Access ID for the user in the Shp user database. We map an attribute in the LDAP server to the Alternate ID in the user’s shp_user_mst record through Controller. This should be bridged into Degree Works from the student system. You specify which LDAP attribute to use for this mapping in the setting core.security.ldap.studentId.attribute. It should be a unique attribute unless you are using the suffix, as described below.

Suppose that all Degree Works users have an attribute in LDAP named “employeeNumber” (an arbitrary example chosen from inetOrgPerson schema). You would need to populate that value into the Alternate ID field in the Shp User Record, usually using the bridge. Refer to the appropriate bridge document for your SIS for instructions on how to do this.

It is possible to have multiple attribute values for the user’s Degree Works ID. In this case, you must append a suffix to the Degree Works ID value in the LDAP record. You should use the same suffix for every user, and you would define this suffix in the setting core.security.ldap.studentId.suffix. So for example, you may use the attribute named “externalId” for multiple systems, with Degree Works being just one of many, and each system may need a different ID value. You would define a suffix, for example “::DGW” and append this suffix to each ID entered in LDAP for this attribute. For example, bobstudent may have an externalId attribute with a value of “12345668::DGW”. Degree Works will locate the value with the given suffix and remove the suffix before it looks up the students shp_user_mst record. If you use the suffix, every user must have this suffix appended to their attribute value.

CAS single sign-on

Updated: September 29, 2023

The Central Authentication Service (CAS) can be used to integrate Degree Works with portals and other Web applications.

CAS provides an open, well-documented protocol and an open-source Java server component. CAS supports LDAP, Active Directory, and other data sources for single sign-on. More information about CAS is available at <http://www.jasig.org/cas>.

Functionality

CAS provides single sign-on to applications by issuing a one-time-use ticket to an end user. The ticket can be validated by a client application and is also used to retrieve the identity of the end user for internal use. When configured for CAS authentication, Degree Works checks for a CAS ticket. If a ticket is not present, the request is redirected to the CAS server. After CAS authentication takes place, the request comes back to Degree Works with an authentication ticket. The end user's Degree Works record is retrieved by linking a CAS attribute obtained during ticket validation.

CAS installation and setup

The CAS installation, setup, and basic configurations are outside the scope of this document. As a prerequisite to installing CAS, you need to be familiar with the CAS server, and should be able to modify CAS.xml and jsp configuration files. Information about installing CAS and the supported authentication mechanisms is available at

<http://www.ja-sig.org/wiki/display/CASUM/Home>.

Configure Degree Works for CAS

CAS supports SSO by issuing a one-use ticket to an end user. The ticket can be validated by a client application and is also used to retrieve the identity of the end user for internal use. Degree Works uses different IDs to internally identify each end user and to log the ID into CAS. For example, an end user usually logs into CAS using a CAS username, while Degree Works internally uses the rad_id. As part of the CAS configuration, you must set up or choose an attribute in your CAS backing data store that will map to the Single sign-on ID field of the SHP user record. This Single sign-on ID is normally populated by the nightly student extract process. Alternatively, the attribute value could map to the SHP access ID of the user.

In Banner, the source of the Single sign-on ID it typically one of the following fields:

- SPRIDEN_ID - a common choice at Banner sites to map to a Degree Works rad_id. Usually this mapping takes place through the shp_user_mst.shp_access_id.
- UDCID - Degree Works must be configured to extract GOBUMAP_UDC_ID to the shp_user_mst.shp_sso_id field.

To configure the Degree Works, all configurations described in the following steps must be completed, including the configuration of the Shepherd settings.

1. Configure the Degree Works Banner Extract.

The Degree Works Banner Extract is configured to extract each user's GOBUMAP_UDC_ID and store it in Degree Works. Select access must be granted to the GOBUMAP table in Banner for the Degree Works user (typically dwmgr). When the Banner extract is run, if a GOBUMAP record is found for the individual, the GOBUMAP_UDC_ID will be loaded into the SHP_USER_MST.SHP_SSO_ID. If a GOBUMAP entry is not found, the individual's SPRIDEN_ID will be loaded into the SHP_USER_MST.SHP_SSO_ID.

2. Configure a CAS service for Degree Works. The CAS server needs to know that the Degree Works URL is protected for SSO. This is accomplished by configuring a CAS service for Degree Works. To configure the CAS service that protects Degree Works, follow these steps:

- a. Access your CAS server management page:
`https://<CAS host>:<CAS port>/<CAS context path>/services/manage.html`
 - b. Log in with a valid administrator user name and password (obtained from the CAS administrator).
 - c. Click the **Add New Service** tab.
 - d. Enter the following values:
 - Name: Your choice (for example, Degree Works appname).
 - Service URL: `https://<Degree Works application host>:< Degree Works application port>/<Degree Works application context path/>`
 - Description: Your choice (for example, Protecting Degree Works through CAS).
 - Status: Select **Enabled** and **SSO Participant**.
 - Attributes: Select **UDC_IDENTIFIER**.
 - e. Click **Save Changes**. The CAS server will allow Degree Works to make sign on requests.
 - f. Repeat steps **2.c** through **2.e** for each application: Controller, Composer, and Responsive Dashboard.
3. Configure the following Shep settings in Degree Works to enable CAS support:
 - core.security.authenticationType - Should be set to **CAS**. It directs Degree Works to redirect unauthenticated requests to the CAS login page.
 - core.security.cas.callbackUrl - This should be set to the URL of your Degree Works application. There should be one for each application with the appropriate specification value. It is used by CAS to callback to Degree Works after the user has logged in.
 - core.security.cas.idAttribute - This is the attribute configured in the CAS server in step **2** and holds the ID used to link the user to the Degree Works ID. For Banner clients, this is typically set to **UDC_IDENTIFIER**.
 - core.security.cas.loginUrl - This is the URL of the CAS login page. Degree Works will direct all users to this page when they have not yet authenticated.
 - core.security.cas.serverUrlPrefix - The CAS URL used by Degree Works to validate the authentication ticket sent with a request.

SAML single sign-on and sign-off

Updated: September 29, 2023

Degree Works can act as a service provider in a federated authentication environment that uses the SAML 2.0 protocols.

In a federated system, a user can log in to one of potentially several different identity providers - services that can authenticate the user. The user is then able to access Degree Works without having to sign in again. A trust relationship is established between the identity provider and Degree Works. A SAML ecosystem is complex and powerful, and a complete explanation is

beyond the scope of this document. A good overview can be found at www.oasis-open.org/committees/download.php/13525/sstc-saml-exec-overview-2.0-cd-01-2col.pdf.

When SAML authentication has been achieved, the Degree Works application generates and manages its own token with its own expiration time. The expiration of that token is controlled by the timeout maximum for the user, which is configurable. It is best if that expiration time corresponds to the expiration of the IDP session. For more information, see [Persistent authentication for a session](#).

SAML configuration

To enable SAML single sign-on, set the Shepherd setting `core.security.authenticationType` to **SAML**. You should also disable SHP authentication (logon form) by changing the `core.security.shp.authentication.enabled` Shepherd setting to `false`. Transfer Equivalency Self-Service will not use SAML so its authentication type should remain SHP. This may require creating a SHP setting for `core.security.authenticationType` with a spec of **treqss** and a value of **SHP** to override the default, if that is set to **SAML**.

You must configure each Degree Works application as a Service Provider (SP) with your SAML Identity Provider (IDP). The value that you use in your IDP SAML configuration for the Service Provider Issuer ID must be put in the value of the SHP setting `core.security.saml.entityId` with a setting spec value associated with that application (for example, `dashboard`).

Also in your IDP configuration is the Assertion Consumer URL value, which is the endpoint for the application. This should be set to `https://server:port/context/saml/SSO`

- server - the name of domain and subdomain of the application (for example, `degreeworks.myschool.edu`).
- port - the optional TCP/IP port of the endpoint.
- context - the optional deployment context, normally specified by the `SERVER_SERVLET_CONTEXT_PATH` in your application startup script.
- `/saml/SSO` - a constant that must be included as written.

You can add path fragments to the end of this URL. If you do, add that fragment to the SHP setting `core.security.saml.assertionConsumerUrl`. You need to configure this value with a different spec code for each application. If you need to specify the entire path for the URL, you should start the setting value with `https://`. In that case, the software will use the setting value exactly as it is entered and not try to prepend the server, port, and context.

Most ID providers supply a URL endpoint to retrieve descriptive metadata. This URL should be put in the SHP setting `core.security.saml.metadata.identityProviderUrl`. This URL must be accessible from the application server. If your IDP does not provide such a URL or it is not accessible, manually add the metadata to the `core.security.saml.metadata.xml.identityProvider` Shepherd setting. There are no special Degree Works modifications needed for this setting, and there needs to be only one with a "default" spec.

There are a number of settings related to the security key store. The `core.security.saml.keystore.location` Shepherd setting specifies the location on the server where the key store is located. It should begin with `file:`. For example: `file:/u01/jdk1.7.0_11/bin/keystore.jks`. The password associated with the key store is stored in

core.security.saml.keystore.password. The key used to sign the SAML request should be entered in the core.security.saml.keystore.keypair.signingKey Shepherd setting. This is usually the same value as the default key entered in core.security.saml.keystore.defaultKey. The core.security.saml.keystore.keypair.password contains the password for the key in the core.security.saml.keystore.keypair.signingKey.

After the user has authenticated, an assertion is provided to Degree Works that identifies the student. An attribute in the assertion must contain the ID located on the SHP_SSO_ID or SHP_ACCESS_ID field in the SHP_USER_MST. The SHP_SSO_ID is normally loaded by the extracts from the student system, but may be maintained in Controller. The name of attribute can be anything and is configured in the core.security.saml.idAttribute Shepherd setting.

Advanced SAML configuration

You may have a situation where the assertion needs to contain more than just the user's ID. This can be used to disambiguate users who might share the same ID at different institutions, for example. You can use the Shepherd setting core.security.ssold.assertionPattern. This contains a regular expression that must be matched by the incoming assertion. If the assertion does not match, the authentication will fail. The value for the setting follows the standard Java regular expression rules as explained at <https://docs.oracle.com/javase/tutorial/essential/regex/>.

When the expression has been successfully matched, the user's SSO ID is extracted from the assertion by looking for a named group with the ID of id. This is represented in the pattern by the string (?<id>). Whatever matches that named group will be used as the assertion ID.

Consider the following example: ^(?:CAMPUS1|ALL)-(?<id>.+)\$. This requires the incoming assertion to start with either CAMPUS1- or ALL- followed by the ID. If the assertion does not match that pattern (for example, starts with CAMPUS2), the authentication attempt fails. If it matches, the SSO ID is taken from the portion of the assertion after the hyphen.

If you do not need this level of pattern matching, this setting may be left empty, and the assertion is taken as the SSO_ID.

Single Sign-off

By default, when a users log off a Degree Works application, they are able to access that application again without re-authenticating, in addition to other applications. This is possible because Degree Works does not log them off of the central identity provider. This can be enabled by changing the SHEP setting core.security.singleSignoutEnabled to **true**. The application will then send a logout request assertion to the identity provider to log out the user. They will then not be able to run a Degree Works application nor any other application authenticated by that provider on that browser session. There is configuration required in your SAML server (asserting party). The logout URL will be your base URL plus **logout/saml2/slo**. For example: https://degreeworks.myschool.edu/dashboard/logout/saml2/slo. The assertion must be signed but not encrypted and use front-channel POST binding. We cannot use the same sign-on assertion URL for logoff. It must be configured separately in your IDP and be the value given above.

External access manager

Updated: March 25, 2022

An External Access Manager intercepts all requests to Degree Works, making sure they are authenticated.

Under this scenario, Degree Works has no responsibility for ensuring that the requests have been authenticated. Examples of an external access manager include Oracle Access Manager and Shibboleth. When configuring the external authentication manager, you must ensure that all Degree Works endpoints are covered.

The external access manager asserts the user's identity to Degree Works after authentication or SSO has taken place. The configuration allows the assertion to have a configurable name and to be delivered to Degree Works through an HTTP cookie or header.

Shepherd settings configuration

`core.security.externalAccessManager.enable`

Enable external access manager. Again, it is important to take care never to leave this flag enabled (true) unless an external access manager is in place, configured and operational. If enabled, other security mechanisms (for example, SHP, CAS) are disabled. In particular, the Shep setting `core.security.authenticationType` is ignored.

`core.security.externalAccessManager.assertionName`

The name of the external access assertion token. This name can be any value of your choosing except for "ASSERT_VALUE".

`core.security.externalAccessManager.assertionIsCookie`

If true, expect to find assertion value in a cookie. The name of the cookie should match the value in the setting `core.security.externalAccessManager.assertionName`. If false, it is expected to be in a header.

Multiple authentication entry points

Updated: September 30, 2022

Multiple types of authentication can be used, depending on the Degree Works application.

When the authenticationType is set to CAS or SAML, the basic SHP authentication is automatically disabled. However, you may decide, for example, that you want the dashboard to use CAS or SAML authentication, but you may want Transit or Controller to use basic SHP security and the built-in login pages. To do that, you can use Controller to create settings specific to each application.

For Transit, you can create two new settings with these values:

Key	Value	Specification
<code>core.security.authenticationType</code>	SHP	transit
<code>core.security.shp.authentication.enabled</code>	true	transit

For Controller, you can create two new settings with these values:

Key	Value	Specification
<code>core.security.authenticationType</code>	SHP	controller

Key	Value	Specification
core.security.shp.authentication.enabled	true	controller

Be sure to restart each application after making these changes in Controller. With these settings in place, you would then only need to have a link to the dashboard in your portal and users of Transit and Controller would use the built-in login.html pages.

Persistent authentication for a session

Updated: March 25, 2022

After it is authenticated, the user's browser receives stateless tokens conforming to the JSON Web Token (JWT) standard.

One is the “refresh” token which expires at the user’s maximum timeout and also identifies the increment timeout. The other token is the “access” token which expires at the user’s increment timeout and is recreated with each API call from the browser as long as the maximum timeout isn’t past. When a user logs out, token is cleared from the user’s browser and added to the TOKEN_REVOCATION_MST table. That table tracks revoked tokens to make sure they cannot be authenticated again.

This time limit has both an incremental and a maximum timeout. If the user continues to be active with their session, the expiration is extended by a configured increment. This is known as the timeout increment amount. There is also a maximum period, known as the timeout maximum, which is measured from the time the user signs on. Either or both of these values can be set to an unlimited time period.

The primary source for these values is the SHPCFG data. If the “TIMEINC” or “TIMEMAX” verbs are included in a rule the user qualifies for, then the timeout periods are set from these values. Otherwise, the values can come from the user’s SHP user record, or one of the groups to which the user belongs, or from the core.security.passport.timeoutIncrementDefault and core.security.passport.timeoutMaximumDefault Shepherd settings. Which one is used depends on the setting core.security.passport.timeoutPrecedence and core.security.passport.timeoutPreference.

API Authentication

Updated: March 24, 2023

The Degree Works API requires authentication through stateless token. All requests must be accompanied by an Authorization header. Requests that do not have an Authorization header will be rejected with an HTTP 403 status.

The authentication API is contained in the Transit and Transfer Equivalency Self-service applications.

Using stateless authentication in Degree Works begins with a request to authenticate user credentials. A valid authentication returns a token. That token can then be used on any number of subsequent requests until the token expires. The type of stateless token used in Degree Works API is a JSON Web Token (JWT).

The typical pattern for accessing a Degree Works API is a multi-step process. An example application or integration might look like:

1. Authenticate a Shepherd User to the authentication API /stateless-token URL and obtain the token from the response.
2. Access the desired API using the obtained token in the Authorization header.
3. Repeat or continue to use the same token on other API requests.

When a stateless token is created or validated by any API deployment in Degree Works, a shared secret is used to sign the token in to verify it is legitimate and has not been tampered with. Each deployment must have the same value for that shared secret. It is configured in the core.security.stateless.secretkey Shepherd setting.

Stateless token creation

Updated: September 29, 2023

You can obtain a stateless token from the Authentication API stateless-token endpoint.

Authenticated users

User credentials being authenticated must exist in the shp_user_mst database table. Users can be created using the Controller application.

Users can be authenticated with either their Access ID (shp_access_id) or Alternate username (shp_alt_id). One of these is passed, together with the password, to the API in a POST request, and the authentication token is returned. The Alternate username is typically used for Transfer Equivalency Self-Service users because this is where user emails are stored.

If the core.security.refererAllowableUrls Shepherd setting has been configured, you must provide a value in the request headers matching one of the values in that setting.

Only one of username (shp_access_id) or email (shp_alt_id) is required. If both are provided, username will be used.

```
POST api/stateless-token HTTP/1.0
Accept: application/net.hedtech.degeworks.stateless.token.v1+json
Content-Type: application/json
Referer: https://example.edu

{"username": "myuser",
 "email": "someone@example.edu",
 "password": "example"}
```

The response body contains a generated token in a JSON object. For example, the response looks like the following, but this token is truncated for brevity, where Bearer abchbGci0iJIUzI1NiJ9... is an example, truncated JSON Web Token (JWT) and not a real or valid token.

```
HTTP/1.1 200 OK
Content-Type: application/json
```

```
{"token": "Bearer abchbGci0iJIUzI1NiJ9...”}
```

Possible return HTTP statuses are:

Status code	Description
200	Authentication succeeded, a stateless token is returned.
401	Error - unauthorized. No data is returned.

Anonymous users

This endpoint in User API supports Transfer Equivalency Self-Service to generate a token for anonymous users who are given limited permission defined by the user specified in the core.security.referenceUser.username Shepherd setting.

The primary purpose of this is for the Transfer Equivalency Self-Service product. Anonymous user authorizations are determined by the list of keys and groups assigned to reference users, which can be configured in the Controller Users module like any other user. Keep in mind anonymous users are authorized to access anything permitted by the keys assigned to the reference users. Reference usernames are configured in core.security.referenceUser.username and passwords are configured in core.security.referenceUser.password.

If the core.security.referrerAllowableUrls Shepherd setting has been configured, you must provide a value in the request headers matching one of the values in that setting.

Request
<pre>GET api/stateless-token/anonymous HTTP /1.0 Accept: application/json Referer: https://example.edu</pre>

Response
<p>The response body contains a generated token in a JSON object. For example, the response looks like the following, but this token is truncated for brevity:</p> <pre>HTTP/1.1 200 OK Content-Type: application/json {"token": "Bearer abchbGci0iJIUzI1NiJ9...”}</pre>

Possible return statuses are:

Status code	Description
200	The request was valid, a stateless token is created and returned.
401	Authentication error. No data is returned.

Token utilization
Updated: March 24, 2023

After a user obtains a token, it should accompany requests to other API in an Authorization header. The value that was obtained from the API already includes the word Bearer followed by the token itself.

Example stateless token Authorization header

```
Authorization: {token}
```

User creation in User API for Transfer Equivalency Self-Service

Updated: March 24, 2023

To save their data in Transfer Equivalency Self-Service, users need to create a new account by using self-reported details.

Until a user creates an account or logs in with an existing one, the user is logged in using a special anonymous user access ID that is specified in the core.security.referenceUser.username Shepherd setting. To create an account, this anonymous user must be assigned DWTESELF key.

A shp_user_mst record is created upon success. The user role comes from the core.security.newUser.roles setting, and the user class is defined by core.security.newUser.userClass. Email is mapped to the shp_alt_id field, and a shp_access_id is generated for the user with a prefix defined by the core.idFactory.idPrefix setting.

POST to create a self-reported user

For a self-reported user, their email, name, and password come from the posted JSON.

```
POST api/users/ HTTP /1.0
Content-Type: application/json
Authorization: {token}

{"email": "XXX@XXX.com", "firstName": "XXX", "lastName": "XXX", "password": "XXX"}
```

POST payload:

email:	The e-mail id of the user.
firstName:	First name of the user.
lastName:	Last name of the user.
password:	The password for authenticating the user credentials.

Response

The response body contains a JSON representation of user and an Authorization header containing a new stateless token, which represents the new user's authentication.

```
HTTP/1.1 200 OK
Content-Type: application/json
Authorization: {token}
```

Response

```
{"firstName": "XXX", "lastName": "XXX", "email": "XXX@XXX.com", "id": "tXXX
X", "userId": "XXXXXXXXXXXXXX", "modifiedDate": XXXXXXXXXXXX, "socialUser": fa
lse, "username": null, "password": null, "name": "XXX, XXX"}
```

Possible return HTTP statuses are:

Status code	Description
200	Authentication succeeded. A stateless token is returned.
401	Error - unauthorized. No data is returned.
403	Authenticated but not allowed to create user due to missing DWTESELF key.
405	A method other than POST was used. No data is returned.
406	An unsupported media type was requested (for example, JSON).
500	User failed to be created because it already exists. This is only for self-reported user.

Authentication debugging in API Services

Updated: March 24, 2023

Debugging authentication problems can be challenging, because you are not able to send a debug header until you have been authenticated.

For this reason, in API Services, we have provided a way to turn on debugging with an environment variable, which will cause debug output to go to the log for all users so this can cause significant output in a production account.

To enable debug, you must set the DEBUG_BASIC_AUTHENTICATION environment variable to true before deploying the application.

Access control (authorization)

Updated: March 25, 2022

User access to functionality in Degree Works is controlled by keys and services.

Degree Works functionality is defined in terms of services. Every service has a lock with a corresponding key or keys. Keys can be organized and assigned into groups. The user's keyring is a set of keys which ultimately controls what services are allowed. The user's keyring is built during authentication. The user's keyring is put together based on attributes of the user, and keys are assigned through rules in the core.security.rules.shpcfg Shepherd setting or through Controller. Keys can be added or removed (hence services can be allowed or denied). Keys can be referenced through a group or individually.

Key assignment with SHPCFG

Updated: March 25, 2022

The SHPCFG data reside in the core.security.rules.shpcfg Shepherd settings and can be used to globally assign keys and timeouts at logon.

The core.security.rules.shpcfg Shepherd settings contains a series of if-statements which assign keys to a user's keyring based on the user's assigned role and other user attributes.

Keys and keyrings

Updated: March 25, 2022

Each user has a keyring with one or more keys.

These keyrings are stored in the database and give access to services. When users are authenticated at the time of logon, they acquire keys that are both explicitly and implicitly assigned.

Keys are needed to access "locked" services.

Keys are defined in SHP078, so you may view the available keys using Controller or review List of Services and associated Keys in the [Services](#) topic.

Groups are stored in the SHP_GROUP_MST and can be reviewed through Controller, or in the "List of standard Groups and Keys" in the [Groups](#) topic.

Explicit assignment

Explicit keys are those assigned to an individual by id either through Controller or in the core.security.rules.shpcfg Shepherd setting. This method is inefficient for assignments in bulk, but very effective for granular, specific control.

Implicit assignment

Implicit assignment of keys is accomplished through the use of authorization rules. It is very efficient for bulk assignments. This is accomplished by edits to the core.security.rules.shpcfg Shepherd setting, which contains the rules.

SHPCFG maintenance

Updated: March 24, 2023

SHPCFG is maintained in Controller in the core.security.rules.shpcfg setting.

The core.security.rules.shpcfg Shepherd setting is a collection of if-statements that must adhere to specific rules and syntax. The if-statements are in the following format:

```
if (expression) then  
    commands
```

An expression consists of a Code and Value, for example:

```
if (DGWUSERCLASS = "REG") then
    addgroup = SRNREG      # Standard keys for the registrar's office use
    rs
    addkey   = RSAUDIT     # Web service for audit/articulation
```

Users who have been authenticated are assigned a User Class (DGWUSERCLASS) and are granted keys based on assignment of groups or specific keys to their User Class. In the above example, users with User Class of REG will have all keys from the SRNREG group plus the RSAUDIT key added to their keyring.

There are two types of expression codes. Some are available to all users, while others are available only for use with students. Valid expression codes for use with all users are:

- DGWUSERCLASS - The USERCLASS assigned during authentication, defined in AUD012.
- DGWSHPACCID - The user's SHP_ACCESS_ID can be used to explicitly add or remove keys or groups.
- EVERYONE - Add or remove keys or groups to all users. This code is unique in that it does not require a value, for example, if (EVERYONE) then.

The following expression codes can be coded for students only. During the student extract process, if configured to do so, the following values will be stored in the SHP_USER_ATTRIB table, which makes this data available to SHPCFG.

- ACTIVETERM - RAD_STUDENT_MST.RAD_TERM
- CATALOGYEAR - RAD_GOAL_DTL.RAD_CATALOG_YR
- COLLEGE - RAD_GOALDATA_DTL.RAD_GOAL_VALUE where RAD_GOAL_CODE = COLLEGE
- DEGREE - RAD_GOAL_DTL.RAD_DEGREE_CODE
- DEGREESOURCE - RAD_GOAL_DTL.RAD_DEGREE_SRC
- MAJOR - RAD_GOALDATA_DTL.RAD_GOAL_VALUE where RAD_GOAL_CODE = MAJOR
- PROGRAM - RAD_GOALDATA_DTL.RAD_GOAL_VALUE where RAD_GOAL_CODE = PROGRAM
- SCHOOL - RAD_GOAL_DTL.RAD_SCHOOL
- STUDENTLEVEL - RAD_GOAL_DTL.RAD_STU_LEVEL
- Custom codes - Custom codes, generated by UCX_BAN080 (Banner) can also be used as SHPCFG expressions.

Valid commands in core.security.rules.shpcfg are as follows. They are not case-sensitive:

- addGroup - Adds a group to the user's keyring.
- remGroup - Removes a group from the user's keyring.
- addKey - Adds a specific key to the user's keyring.

- remKey - Removes a specific key from the user's keyring.
- deny - Denies access.
- timeInc - Sets a timeout increment for the user.
- timeMax - Sets a maximum login time for the user.

Expression operators that can be used are:

- = - Equal
- <> - Not Equal
- > - Greater than
- < - Less than
- >= - Greater than or equal to
- <= - Less than or equal to

Expressions can be combined with the conditional operators AND and OR. To ensure the results you expect, follow the best practice of using parentheses when combining expressions:

```
If (CODE1 = "VALUEA" or CODE2 = "VALUEB") then  
If (CODE1 = "VALUEA" and CODE2 = "VALUEB") then  
If (CODE1 = "VALUEA" and (CODE2 = "VALUEB" or CODE3 = "VALUEC")) the  
n
```

The use of ELSE is not supported. For example, the following is not valid:

```
If (DGWUSERCLASS <> "STU") then  
    addkey = SUPPORT # Support  
Else  
    addGroup = SRNSTU # Standard keys for students
```

Instead, code the above logic as follows in separate rules:

```
If (DGWUSERCLASS <> "STU") then  
    addkey = SUPPORT # Support  
  
If (DGWUSERCLASS = "STU") then  
    addGroup = SRNSTU # Standard keys for students
```

In the next example, all students will be granted all keys in the SRNSTU group, but only students whose RAD_SCHOOL value = UG will be granted the keys in the SEPSTUED group, except for the SEPPDELL key:

```
if (DGWUSERCLASS = "STU") then  
addGroup = SRNSTU # Standard keys for students  
  
if (DGWUSERCLASS = "STU" and SCHOOL = "UG") then
```

```
addGroup = SEPSTUED # Planner Edit/Create  
remkey   = SEPPDELL # Delete Plans
```

In the following example, the advisors are being assigned the SRNADV group. Advisors are also given the SDSTUANY key to allow them to search on any student. The SDSTUMY key is taken away so advisors are not tied to a specific list of advisees.

If your advisors are tied to certain advisees, the SRNADV group will give you what you need. If you are bridging students with different TERM values and want to see all students assigned to any advisor regardless of TERM value, set the UCX_CFG020 WEBPARAMS **Term used for filtering in student search** field to blanks to ignore the TERM field as part of the search criteria.

```
if (EVERYONE) then  
  
    TIMEINC = 0055          #55 Minutes of idle time  
  
    TIMEMAX = 0800          #Max 8 hrs before relogon needed  
  
    remKey = SDLOKAHD      #Remove look ahead  
  
#-----  
---  
### DegreeWorks keys for students  
  
#-----  
---  
if (DGWUSERCLASS = "STU") then  
  
    addGroup = SRNSTU  
  
    addKey   = SD2WIFHIS # allow student to view what-if history  
  
#-----  
---  
### DegreeWorks keys for applicants  
  
#-----  
---  
if (DGWUSERCLASS = "APP") then  
  
    addGroup = SRNAPP  
  
#-----  
---  
### DegreeWorks keys for advisors  
  
#-----  
---  
if (DGWUSERCLASS = "ADV") then  
  
    addGroup = SRNADV
```

```

remKey    = SDSTUMY # load my advisees only
addKey    = SDSTUANY # allow search on all students

#-----
#-----
#-- DegreeWorks keys for administrators

#-----
#-----
if (DGWUSERCLASS = "REG") then
    addGroup = SRNREG

```

The above example also illustrates use of the EVERYONE expression code. Here all users are assigned timeInc of 0055 and timeMax 0800. These time settings are in the format of hhmm, where timeInc is set to 55 minutes and timeMax is set to 8 hours.

While modifying core.security.rules.shpcfg, you can add comments that are preceded with a #. Comments can be added at the beginning of a line or after the assignment of a key or group, as illustrated above.

Running shpparse first will report any errors. Running daprestart (manually or through Transit) will also run shpparse, but you will not see the parse errors.

After modifying core.security.rules.shpcfg and doing a daprestart, you need to restart the Java applications for them to pick up the changes.

Services

Updated: September 29, 2023

Each component of business functionality is a service. Services may be broad (an entire web page or more) or narrow (a button that does something useful). Services are locked and keys are needed to access them.

Services and associated keys

Title	Key	Application
Advanced student filter - obey filters on user record for department, school, or college	ADFILTER	Responsive Dashboard
Audit Description	AUDDESCR	Worksheets
Freeze Audits	AUDFREEZ	Worksheets
Key to access Composer application	COMPOSER	Composer
Contact icon	CONTACT	Responsive Dashboard
In Contact dialog, also show student's advisors	CONTADVR	Responsive Dashboard
Application Access	CONTROL	Controller
Access to Configuration menu	CTLCONF	Controller

Title	Key	Application
Access to Groups tab in Authorization menu	CTLGROUP	Controller
Access to Add Groups	CTLGRPAD	Controller
Access to Delete Groups	CTLGRPDL	Controller
Access to Modify Groups	CTLGRPMD	Controller
Access to Properties menu	CTLPROP	Controller
Access to Add Shepherd Settings	CTLSETAD	Controller
Access to Delete Shepherd Settings	CTLSETDL	Controller
Access to Modify Shepherd Settings	CTLSETMD	Controller
Access to Add UCX Entries	CTLUCXAD	Controller
Access to UCX Bulk Operations	CTLUCXBK	Controller
Access to Delete UCX Entries	CTLUCXDL	Controller
Access to Modify UCX Entries	CTLUCXMD	Controller
Access to Users tab in Authorization menu	CTLUSERS	Controller
Access to Add Users	CTLUSRAD	Controller
Access to Delete Users	CTLUSRDL	Controller
Access to Modify Users	CTLUSRMD	Controller
Access to turn on and off debugging	DEBUG	All Java applications
Link to Ellucian's online documentation for Responsive Dashboard	DOCDASH	Responsive Dashboard
Transfer Equivalency Self-Service admin	DWTEADMN	Transfer Equivalency Self-Service
Transfer Equivalency Self-Service access.	DWTESELF	Transfer Equivalency Self-Service
Exceptions – Also Allow	EXPALLOW	Exceptions
Exceptions – Apply Here	EXPAPPLY	Exceptions
Exceptions – Change the Limit/ Remove Course	EXPCHANG	Exceptions
Exceptions – Force Complete	EXPFORCE	Exceptions
Exceptions – Substitute	EXPSUBST	Exceptions
Exceptions – view details on worksheet	EXPVWDTL	Exceptions
External Links	EXTLINKS	Links
Allows application access to Degree Works APIs	NOREFER	API Services
Banner Registration from Course Link	REGISTER	Responsive Dashboard
Retrieve audit/articulation	RSAUDIT	API Services
Retrieve classes by student ID	RSCLASS	API Services
Role for CourseInfo web services	RSCRSINF	API Services

Title	Key	Application
Retrieve Mapping through a Web Service	RSMAPPNG	API Services
Retrieve Plans through a Web Service	RSPLAN	API Services
Retrieve SHP Settings	RSSETTNG	API Services
Retrieve validation (UCX) tables	RSVALID	API Services
Run what-if audit	RSWHATIF	API Services
Access to modify all block types	SCRBLALL	Scribe
Access to modify ATHLETE blocks	SCRBLATH	Scribe
Access to modify AWARD blocks	SCRBLAWR	Scribe
Access to modify COLLEGE blocks	SCRBLCOL	Scribe
Access to modify CONC blocks	SCRBLCON	Scribe
Access to modify DEGREE blocks	SCRBLDEG	Scribe
Access to modify ID blocks	SCRBLID	Scribe
Access to modify LIBL blocks	SCRBLLIB	Scribe
Access to modify MAJOR blocks	SCRBLMAJ	Scribe
Access to modify MINOR blocks	SCRBLMIN	Scribe
Access to modify OTHER blocks	SCRBLOTH	Scribe
Access to modify PROGRAM blocks	SCRBLPRG	Scribe
Access to modify REQUISITE blocks	SCRBLREQ	Scribe
Access to modify SCHOOL blocks	SCRBLSCH	Scribe
Access to modify SPEC blocks	SCRBLSPC	Scribe
Access Scribe	SCRIBE	Scribe
Access to parse blocks in Scribe	SCRPARSE	Scribe
Admin tab/Theme editor	SDADMIN	Admin
Financial Aid Audits tab	SDAIDAUD	Worksheets
Delete Financial Aid Audits	SDAIDDEL	Worksheets
Financial Aid Audit History	SDAIDHIS	Worksheets
Review Financial Aid Audits	SDAIDREV	Worksheets
Run Financial Aid Audits	SDAIDRUN	Worksheets
Athletic Eligibility Audits tab	SDATHAUD	Worksheets
Delete Athletic Eligibility Audits	SDATHDEL	Worksheets
Athletic Eligibility Audit History	SDATHHIS	Worksheets
Review Athletic Eligibility Audits	SDATHREV	Worksheets
Run Athletic Eligibility Audits	SDATHRUN	Worksheets
Delete Academic Audits	SDAUDDEL	Worksheets
Audits – Worksheets tab	SDAUDIT	Worksheets

Title	Key	Application
Save as PDF option	SDAUDPDF	Worksheets
Review Audit	SDAUDREV	Worksheets
Run Audit	SDAUDRUN	Worksheets
Department user	SDDEPART	Responsive Dashboard
Exception Management Exceptions Search	SDEMEXSR	Exception Management
Delete applied petitions	SDEMPEAD	Exception Management
Exception Management Petitions Applied	SDEMPEAL	Exception Management
Exception Management Petitions Approved	SDEMPEAV	Exception Management
Exception Management Fix Petition Status	SDEMPEFX	Exception Management
Delete rejected petitions	SDEMPERD	Exception Management
Exception Management Petitions Rejected	SDEMPERJ	Exception Management
Exception Management Petitions Waiting	SDEMPEWA	Exception Management
Exceptions	SDEXCEPT	Exceptions
Add Exception	SDEXPADD	Exceptions
Delete Exception	SDEXPDEL	Exceptions
Exception Management Access	SDEPMGT	Exception Management
Student Search	SDFIND	Responsive Dashboard
Student Search by ID	SDFINDID	Responsive Dashboard
GPA Advice Calculator	SDGPAADV	GPA Calculator
GPA Calculator	SDGPACLC	GPA Calculator
GPA Graduation Calculator	SDGPAGRD	GPA Calculator
GPA Term Calculator	SDGPATRM	GPA Calculator
Audit History	SDHIST	Worksheets
Notes	SDNOTES	Audit Notes
Note Add	SDNTEADD	Audit Notes
Add free-form notes	SDNTECHG	Audit Notes
Delete Note	SDNTEDEL	Audit Notes
Modify Note	SDNTEMOD	Audit Notes
Run New Audit after saving note	SDNTERUN	Audit Notes

Title	Key	Application
View Note	SDNTEVUE	Audit Notes
Petition Add	SDPETADD	Petitions
Petitions delete	SDPETDEL	Petitions
Petitions Modify	SDPETMOD	Petitions
Petition View	SDPETVIEW	Petitions
Refresh Button	SDREFBTN	Responsive Dashboard
Show Refresh date/time (for Banner, also controls the refresh button in Dashboard)	SDREFRES	Responsive Dashboard
Registrar/Any Student – for searching	SDSTUANY	Responsive Dashboard
Student Services	SDSTUME	Responsive Dashboard
My Advisees – for searching	SDSTUMY	Responsive Dashboard
WEB30 Worksheet	SDWEB30	Worksheets
WEB31 Worksheet	SDWEB31	Worksheets
WEB32 Worksheet	SDWEB32	Worksheets
WEB33 Worksheet	SDWEB33	Worksheets
WEB34 Worksheet	SDWEB34	Worksheets
WEB35 Worksheet	SDWEB35	Worksheets
WEB36 Worksheet	SDWEB36	Worksheets
WEB37 Worksheet	SDWEB37	Worksheets
Financial Aid Report	SDWEB50	Worksheets
Aid and Academic Report	SDWEB51	Worksheets
Athletic Eligibility Report	SDWEB55	Worksheets
Athletic and Academic Report	SDWEB56	Worksheets
What-if Audit	SDWHATIF	What-If
What-If Delete button on History tab	SDWIFDEL	What-If
What-If History tab	SDWIFHIS	What-If
DegreeWorks Access	SDWORKS	Responsive Dashboard
Diagnostics Report	SDXML30	Responsive Dashboard
Class History report	SDXML31	Responsive Dashboard
Student Data report	SDXML32	Responsive Dashboard
Class Summary Report	SDXML33	Responsive Dashboard
Modify the critical indicator on plan and template requirements	SEPCRIT	Plans and Template Management
View, create, and modify internal notes on plans	SEPINOTE	Plans and Template Management
Set the active flag on a plan	SEPPACTV	Responsive Dashboard

Title	Key	Application
Create new plans and modify existing plans	SEPPADD	Plans
Generate planner audits	SEPPAUD	Plans
Permission to auto approve a plan	SEPPAUTO	Plans
Create a Scribe block from a plan	SEPPBLCK	Plans
Override plan course offering check (CFG072)	SEPPCOFF	Plans
Delete plans	SEPPDEL	Plans
Delete plans that are not locked/approved	SEPPDELL	Plans
Access to Plans tab	SEPPLAN	Plans
Lock a plan	SEPLOCK	Plans
Modify existing plans	SEPPMOD	Plans
Create new notes and modify existing notes on plans, terms and requirements	SEPPNADD	Plans
Delete plan, term and requirement notes created by the current user	SEPPNDEL	Plans
Modify existing plan, term and requirement notes created by the current user	SEPPNMOD	Plans
Delete plan, term and requirement notes that have been copied from a template	SEPPNTDT	Plans
Modify plan, term and requirement notes that have been copied from a template	SEPPNTET	Plans
Delete plan, term and requirement notes created by other users	SEPPNTGD	Plans
Modify plan, term and requirement notes created by other users	SEPPNTOW	Plans
Modify pointer field on choice requirements	SEPPPTR	Plans
Override plan pre- and co-requisite checking	SEPPRQTO	Plans
Select course choice requirement options on plans.	SEPPSEL	Plans
Create a new plan from a template	SEPTEMP	Plans
Access to generate planner What-If audits	SEPPWIF	Plans
Modify the plan scope field	SEPSCOPE	Plans
Create new templates and modify existing templates	SEPTADD	Template Management
Delete templates	SEPTDEL	Template Management
Modify existing templates	SEPTMOD	Template Management
Access to Template Management in the Admin menu	SEPTMGMT	Template Management
Create new notes and modify existing notes on templates, terms and requirements	SEPTNADD	Template Management

Title	Key	Application
Delete template, term and requirement notes	SEPTNDEL	Template Management
Modify template, term and requirement notes	SEPTNMOD	Template Management
Change template term scheme	SEPTTRMS	Template Management
Allows visibility of encrypted Shepherd settings	SHENCRPT	Controller
Access to Support Features	SUPPORT	Responsive Dashboard
Application access	TEAADMIN	Transfer Equivalency Admin
Access to Articulate tab	TEAARTIC	Transfer Equivalency Admin
Access to Audit tab	TEAAUDIT	Transfer Equivalency Admin
Access to Mappings tab	TEAMAP	Transfer Equivalency Admin
Access to Roll tab	TEAROLL	Transfer Equivalency Admin
Access to Transcript tab	TEATRANS	Transfer Equivalency Admin
Transfer Finder Categories from audit	TFCATGRY	Transfer Finder
Transfer Finder Tab	TFFINDER	Transfer Finder
Transfer Finder Don't Display Acknowledge Message	TFNOACK	Transfer Finder
Transfer Finder Transfer Audit	TFTRAUDT	Transfer Finder
Access to run ADMIN jobs and scripts	TRADMIN	Transit
Run all reports and processors	TRANALL	Transit
Access to upload job artifacts to the database (API only)	TRANART	Transit
Access to delete completed jobs	TRANDEL	Transit
Access to the Saved Jobs tab and to create, modify, delete saved job parameters	TRANPARM	Transit
Access to the Schedule button, Scheduled/ Saved Jobs tabs, recurring jobs, and saved parameters	TRANRCUR	Transit
Access to the Run Jobs tab	TRANRUN	Transit
Access to the Schedule button, Scheduled Jobs tab and to create, modify, delete scheduled jobs	TRANSCHD	Transit
Access to the Transit application	TRANSIT	Transit
Access to use SQL for select criteria	TRANSQL	Transit
Access to run AUD01	TRAUD01	Transit

Title	Key	Application
Access to run AUD02	TRAUD02	Transit
Access to run BAN62 (Banner clients)	TRBAN62	Transit
Access to run the DAP16 processor	TRDAP16	Transit
Access to run DAP21	TRDAP21	Transit
Access to run DAP22	TRDAP22	Transit
Access to run DAP27	TRDAP27	Transit
Access to run DAP28	TRDAP28	Transit
Access to run DAP40	TRDAP40	Transit
Access to run DAP41	TRDAP41	Transit
Access to run DAP42	TRDAP42	Transit
Access to run DAP43	TRDAP43	Transit
Access to run DAP44	TRDAP44	Transit
Access to run DAP45	TRDAP45	Transit
Access to run DAP54	TRDAP54	Transit
Access to run DAP58	TRDAP58	Transit
Access to run DAP59	TRDAP59	Transit
Access to run the RAD Bridge Processor (RAD clients only)	TRRAD11	Transit
Access to run the Student Extract (Banner clients)	TRRAD30	Transit
Access to run the Applicant Extract (Banner clients)	TRRAD32	Transit
Access to run the Staff Extract (Banner clients)	TRRAD33	Transit
Access to run the Banner Course Extract	TRRAD34	Transit
Access to run the Banner Curriculum Rules Extract	TRRAD35	Transit
Access to run the Banner Validation Extract (UCX)	TRRAD36	Transit
Access to run the Banner Transfer School Extract (ETS)	TRRAD37	Transit
Access to run the Banner Equivalencies Extract	TRRAD38	Transit
Access to run the Banner Transfer Equivalency Extract (Mappings)	TRRAD39	Transit
Access to run the delete student data processor (DELETEID)	TRRAD40	Transit
Access to run SCR02	TRSCR02	Transit
Access to run SCR05	TRSCR05	Transit

Title	Key	Application
Access to run SCR06	TRSCR06	Transit
Access to run SCR07	TRSCR07	Transit
Access to run SCR08	TRSCR08	Transit
Access to run SCR09	TRSCR09	Transit
Access to run SCR10	TRSCR10	Transit
Access to run SCR11	TRSCR11	Transit
Access to run SCR91 (test for Banner Prerequisite)	TRSCR91	Transit
Access to run SCR92 (test for Banner Prerequisite)	TRSCR92	Transit
Access to run SCR93	TRSCR93	Transit
Access to run SCR94	TRSCR94	Transit
Access to run SCR95	TRSCR95	Transit
Access to run UCX01	TRUCX01	Transit
What-if Audit Description	WIFDESCR	What-If
What-if Freeze Audits	WIFFREEZ	What-If

Groups

Updated: September 29, 2023

The keys assigned to a group can be modified through Controller.

A User Class will typically have a Group of Keys assigned. These groups are stored in SHPDB and can be viewed and modified using Controller. A user will inherit the group keys from their user class, which will be combined with other keys they may acquire from core.security.rules.shpcfg, or those assigned through Controller.

Standard groups and associated keys

Group	Key
CONTROL	CONTROL, CTLCONF, CTLGROUP, CTLGRPAD, CTLGRPDL, CTLGRPMD, CTLPROP, CTLSETAD, CTLSETDL, CTLSETMID, CTLUCXAD, CTLUCXBK, CTLUCXDL, CTLUCXMD, CTLUSERS, CTLUSRAD, CTLUSRDL, CTLUSRMD
SCRIBAEA	CONTROL, SCRIBE, SCRPARSE, SCRBLATH
SCRIBAID	CONTROL, SCRIBE, SCRPARSE, SCRBLAWR
SCRIBREG	CONTROL, SCRIBE, SCRPARSE, SCRBLALL
SEPADV	SEPCRIT, SEPINOTE, SEPPACTV, SEPPADD, SEPPAUD, SEPPBLCK, SEPPCOFF, SEPPDEL, SEPPPLAN, SEPPLOCK, SEPPNADD, SEPPNDEL, SEPPNTDT, SEPPNTET, SEPPPTR, SEPPRQTO, SEPPSEL, SEPPTEMP, SEPPWIF, SEPSCOPE

Group	Key
SEPREG	SEPCRIT, SEPINOTE, SEPPACTV, SEPPADD, SEPPAUD, SEPPAUTO, SEPPBLCK, SEPPCOFF, SEPPDEL, SEPLAN, SEPPLOCK, SEPNADD, SEPNDEL, SEPPNTDT, SEPPNTET, SEPPNTGD, SEPPNTOW, SEPPPTR, SEPRQTO, SEPPSEL, SEPTEMP, SEPPWIF, SEPSCOPE, SEPTADD, SEPTDEL, SEPTMGMT, SEPTNADD, SEPTNDEL, SEPTTRMS
SEPSTUED	SEPPADD, SEPPAUD, SEPPDELL, SEPPLAN, SEPPNADD, SEPPNDEL, SEPPSEL, SEPTEMP, SEPPWIF
SEPSTUVW	SEPPAUD, SEPPLAN, SEPPWIF
SRNADV	CONTACT, CONTADVR, EXPALLOW, EXPAPPLY, EXPCHANG, EXPFORCE, EXPsubst, EXPVWDTL, EXTLINKS, SDAUDIT, SDAUPDF, SDAUDREV, SDAUDRUN, SDEXCEPT, SDEXPADD, SDEXPDEL, SDFIND, SDFINDID, SDGPAADV, SDGPACLC, SDGPAGRD, SDGPATRM, SDNOTES, SDNTEADD, SDNTECHG, SDNTEDEL, SDNTEMOD, SDNTERUN, SDNTEVUE, SDSTUANY, SDSTUMY, SDWEB31, SDWEB36, SDWHATIF, SDWIFDEL, SDWIFHIS, SDWORKS, SDXML31, WIFDESCR, WIFFREEZ
SRNADGX	CONTACT, CONTADVR, EXTLINKS, SDAUDIT, SDAUPDF, SDAUDREV, SDAUDRUN, SDFIND, SDFINDID, SDGPAADV, SDGPACLC, SDGPAGRD, SDGPATRM, SDNOTES, SDNTEADD, SDNTECHG, SDNTEDEL, SDNTEMOD, SDNTERUN, SDNTEVUE, SDPETADD, SDPETDEL, SDPETMOD, SDPETVIEW, SDSTUANY, SDSTUMY, SDWEB31, SDWEB36, SDWHATIF, SDWIFDEL, SDWIFHIS, SDWORKS, SDXML31, WIFDESCR, WIFFREEZ
SRNAID	AUDDESCR, AUDFREEZ, CONTACT, CONTADVR, EXTLINKS, SDAIDAUD, SDAIDDEL, SDAIDHIS, SDAIDREV, SDAIDRUN, SDAUPDF, SDFIND, SDFINDID, SDSTUANY, SDWEB50, SDWEB51, SDWORKS, SDXML30
SRNAPP	CONTACT, EXTLINKS, SDAUPDF, SDAUDREV, SDSTUME, SDWEB31, SDWHATIF, SDWORKS, SDXML31
SRNATHL	AUDDESCR, AUDFREEZ, CONTACT, CONTADVR, EXTLINKS, SDATHAUD, SDATHDEL, SDATHHIS, SDATHREV, SDATHRUN, SDAUPDF, SDFIND, SDFINDID, SDSTUANY, SDWEB55, SDWEB56, SDWORKS, SDXML30, SDXML33
SRNREG	AUDDESCR, AUDFREEZ, CONTACT, CONTADVR, DEBUG, DOCdash, EXPALLOW, EXPAPPLY, EXPCHANG, EXPFORCE, EXPsubst, EXPVWDTL, EXTLINKS, SDADMIN, SDAUDDEL, SDAUDIT, SDAUPDF, SDAUDREV, SDAUDRUN, SDEMEXSR, SDEMPEAD, SDEMPEAL, SDEMPEAV, SDEMPEFX, SDEMPERD, SDEMPERJ, SDEMPEWA, SDEXCEPT, SDEXPADD, SDEXPDEL, SDEXPMGT, SDFIND, SDFINDID, SDGPAADV, SDGPACLC, SDGPAGRD, SDGPATRM, SDHIST, SDNOTES, SDNTEADD, SDNTECHG, SDNTEDEL, SDNTEMOD, SDNTERUN, SDNTEVUE, SDSTUANY, SDWEB30, SDWEB31, SDWEB36, SDWEB37, SDWHATIF, SDWIFDEL, SDWIFHIS, SDWORKS, SDXML30, SDXML31, SDXML32, SDXML33, SUPPORT, WIFDESCR, WIFFREEZ

Group	Key
SRNSTU	CONTACT, EXLINKS, SDAUDIT, SDAUDPDF, SDAUDREV, SDGPAADV, SDGPACLC, SDGPAGRD, SDGPATRM, SDSTUME, SDWEB31, SDWEB36, SDWHATIF, SDWORKS, SDXML31
TFADV	TFCATGRY, TFNOACK, TFTRAUDT, TFFINDER
TFREG	TFCATGRY, TFNOACK, TFTRAUDT, TFFINDER
TFSTU	TFCATGRY, TFTRAUDT, TFFINDER
TRANBAN	TRBAN62, TRRAD30, TRRAD32, TRRAD33, TRRAD34, TRRAD35, TRRAD36, TRRAD37, TRRAD38, TRRAD39, TRSCR91, TRSCR92, TRSCR93, TRSCR94, TRSCR95
TRANREG	TRADMIN, TRANDEL, TRANRUN, TRANSCHD, TRANSIT, TRANSQL, TRAUD01, TRAUD02, TRDAP16, TRDAP21, TRDAP22, TRDAP27, TRDAP28, TRDAP40, TRDAP41, TRDAP42, TRDAP43, TRDAP44, TRDAP45, TRDAP54, TRDAP58, TRDAP59, TRRAD11, TRSCR02, TRSCR05, TRSCR06, TRSCR07, TRSCR08, TRSCR09, TRSCR10, TRSCR11, TRUCX01
TEAREG	TEAADMIN, TEAMAP, TEATRANS, TEAARTIC, TEAAUDIT, TEAROLL

Users

Updated: March 25, 2022

Authenticated users may request services based on their user class assignment.

User class assignment associates the user with similar users for the purpose of controlling read-write access to notes and access to menu options in Degree Works. User class is sent to Degree Works on the SHPU record when each user is loaded through the bridge. For Banner clients this is handled by the Banner extracts and their associated configuration tables.

User class

The valid user class codes are stored in AUD012. The user class determines which shp_group (collection of keys) is granted to the user, thereby determining which services the user can access.

The user class is stored on the SHP_USER_MST in the database. A user can have only one user class.

The following set of groups is normally assigned in core.security.rules.shpcfg based on the user class:

User Class	Description	Group (shp_group_mst)
ADV	Advisor	SRNADV, SEPADV, TFADV
ADVX	Advisor without exceptions	SRNADVX, SEPADV
AID	Financial Aid Office	SRNAID, SCRIBAID
APP	Applicant	SRNAPP, SEPSTUVW or SEPSTUED, TFSTU

User Class	Description	Group (shp_group_mst)
ATHL	Athletic Department	SRNATHL, SCRIBAEA
REG	Registrar	SRNREG, SRNAID, SRNATHL, SEPREG, TFREG, SCRIBREG, CONTROL, TRANREG
STU	Student	SRNSTU, SEPSTUVW or SEPSTUED, TFSTU

Database privileges

Updated: March 25, 2022

Ellucian recommends that you give the Degree Works database user account DBA privileges while you are processing an update to avoid any complications.

However, when the update is complete, you may choose to restrict the account to the following minimum required privileges:

```
grant create table to dwschema;
grant unlimited tablespace to dwschema;
grant create session on dwschema;
alter user dwschema quota unlimited on dgw;
alter user dwschema quota unlimited on pseudotemp;
grant SELECT_CATALOG_ROLE to dwschema;
```

If your user account is something other than dwschema, replace it in the commands listed above.

Encrypted data

Updated: March 25, 2022

Shepherd user passwords and Shepherd settings are stored in an encrypted form in the database.

Shepherd user passwords

For users defined in the native Shepherd user database, the passwords can be optionally stored in an encrypted format. This can be enabled in Controller in the CFG020 record with a key of WEBPARAMS. Set the “Encrypt Password” field to “Y”. If this field is modified, then all passwords must be reset. The encryption is a one-way digest using the SHA-1 algorithm. The password cannot be decrypted.

Shepherd settings

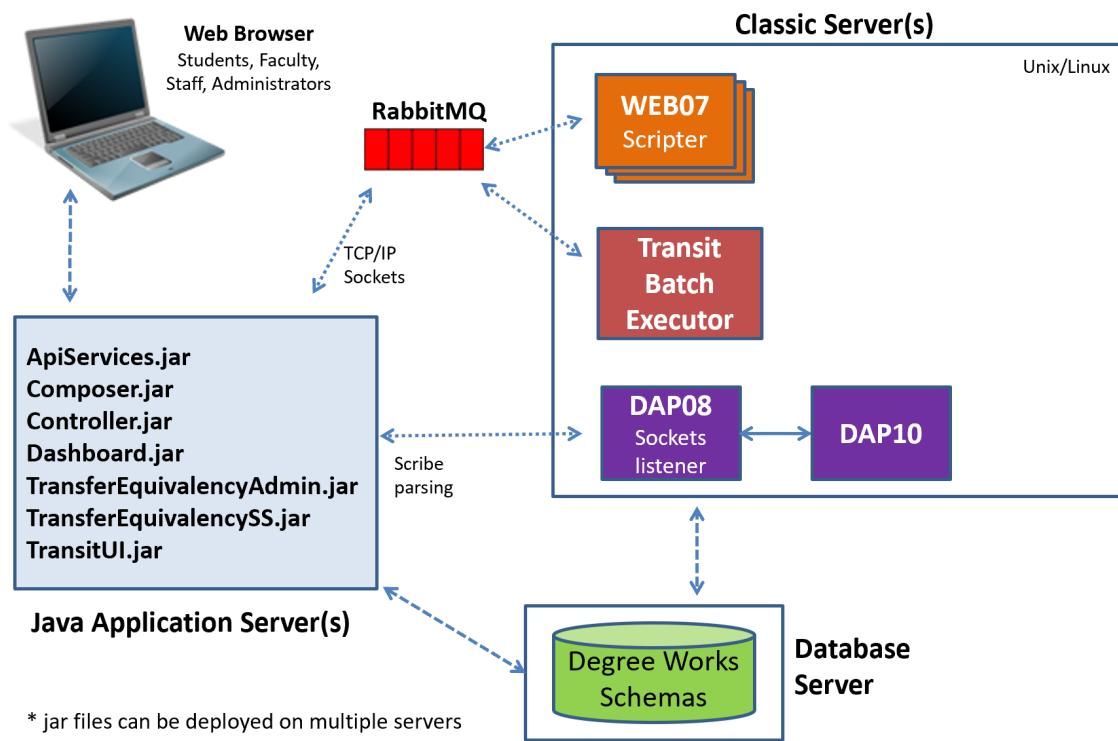
There are several settings configured in the database that contain sensitive information, such as passwords. These settings are encrypted as they are maintained using Controller. The encryption is a 2-way (symmetric) encryption using a 128-bit AES block cipher. This means that the clear text unencoded value can be viewed in Controller. In order to view and edit these values, the user must be assigned an access key of SHENCRPT. Without this key, the encrypted entries will not be displayed in Controller. Before editing any encrypted entry, you must first enter the encryption key.

This is stored in the core.shpSetting.encryptionKey Shepherd setting. This entry is not encrypted and can only be seen or edited only by someone with the SHENCRPT key.

If any password is modified, the applications that rely on that setting should be restarted.

Degree Works applications

A typical Degree Works deployment includes classic, database, RabbitMQ and Java application servers.



Deployment considerations

Updated: March 25, 2022

There is great flexibility in the deployment of Degree Works applications.

The Java applications are all standalone applications and can be partitioned however is most practical. They can all be deployed on one server or they can be split up on multiple servers. It is

common, for example, to put administrative applications on one server and self-service applications on another. The administrative applications have lower and more predictable loads while the self-service applications may have spikes of high demand that require more overhead and or adaptive load balancing strategy.

The classic applications require a more monolithic supportive environment and so are typically all run on the same server, although it may be load balanced by cloning the entire system. One or more java applications can also be deployed on this server. There is no direct communication by a user to the classic applications – all requests first go through one of the Java applications (e.g. Responsive Dashboard).

The RabbitMQ broker can be deployed on one of the other servers or on its own server. It generally does not consume significant server resources.

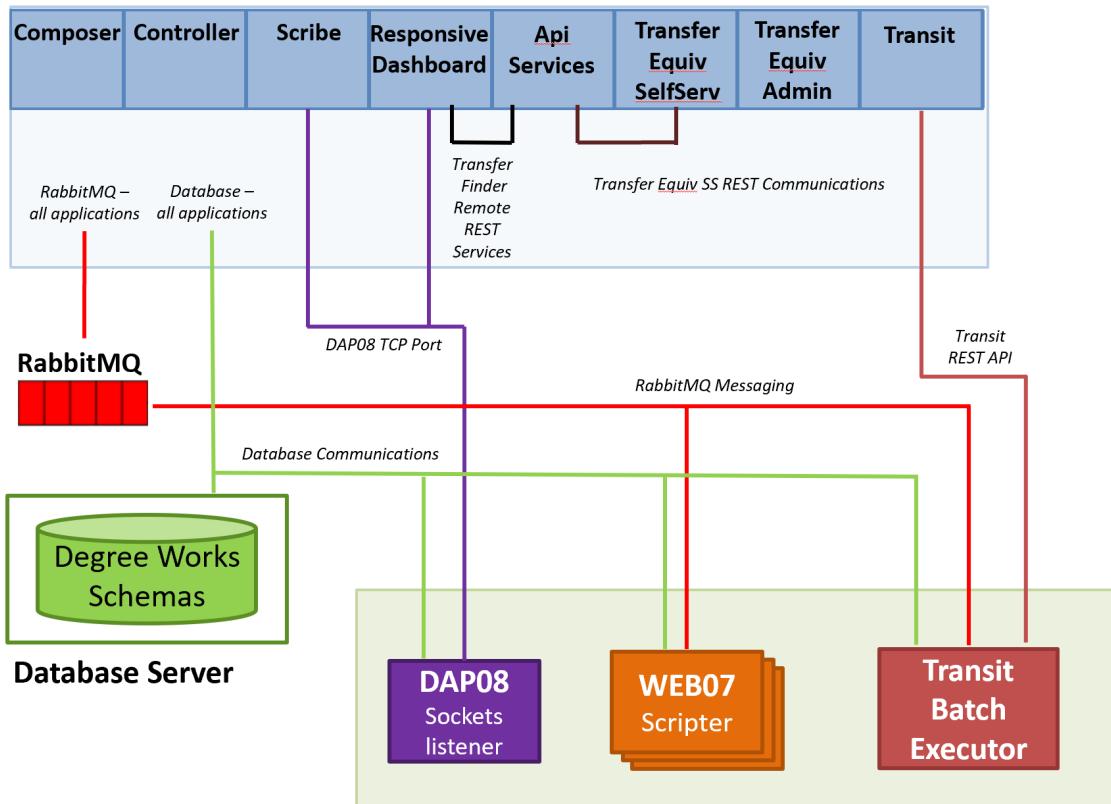
It is also possible to put the database server on one of the existing servers.

When considering deployment, bear in mind that it is the Java applications that receive and respond to requests. The classic applications, RabbitMQ and the database only communicate with internal applications.

Degree Works intercommunication

There are many communication pathways between the various components in a Degree Works environment.

Degree Works Communication Pathways



The above diagram shows the communication pathways within the Degree Works application ecosystem. They use one of the following protocols:

- Database – All applications access the Degree Works database.
- RabbitMQ – A message queuing and delivery system used to:
 - communicate UCX and Shepherd setting changes to all applications.
 - request audit services from WEB07 by the Responsive Dashboard, Transfer Equivalency Admin, Transfer Equivalency Self-Service and API Services.
 - request certain other data used by the Responsive Dashboard.
 - transmit job requests to the Transit Batch Executors.
- DAP08 TCP Socket – Used to request parsing of blocks. Used by Scribe, and by the Student Planner when creating blocks based on plans.
- REST APIs – Multiple uses:

- By the Transit Batch Executor to get information about requested jobs, provide job status updates and upload artifacts from the Transit API.
- Transfer Equivalency Self-service also uses API Services to provide audits.
- The Responsive Dashboard Transfer Finder functionality uses API Services at remote institutions to retrieve classwork history and provide audits. It also uses API Services at the central installation to obtain global configurations.

Degree Works external communication requirements

Updated: March 25, 2022

Transfer Equivalency Self-service can produce PDFs of its audits.

To do this, it requires access to an external content delivery network address to retrieve fonts and other assets. You must allow access from the Transfer Equivalency Self-service server to cdn.elluciancloud.com on port 443 for this to work.

Request-response

Updated: March 25, 2022

When a user makes a request in a Degree Works application, there are several different paths that the response can take.

The following is a step-by-step description of the operation of a Degree Works Web session, and the fulfillment of a service request.

1. When the user clicks a button in their browser a request is sent to one of the Java applications.
2. The application either processes the request itself or requests additional data from one of the other applications.
3. If an audit or certain other student data is required, a request is sent over a RabbitMQ exchange. The next free web07 daemon takes the request from the queue and then calls the scripter subroutine to process the request. The scripter handles the request and then passes the completed script back through the RabbitMQ message queue.
4. If a Scribe block needs parsing or saving, a request is sent over a TCP/IP socket to DAP08. It processes the request and passes a response back to the calling application.
5. If a job launch has been requested in Transit, the request is logged to the database and a request is posted on a RabbitMQ exchange for the next Transit Batch Executor to pick up and run. The executor will update the job status and upload any completed artifacts by using a REST call to the Transit API.
6. Occasionally an application will request additional services from one of the other Java APIs. This is done using a REST call.

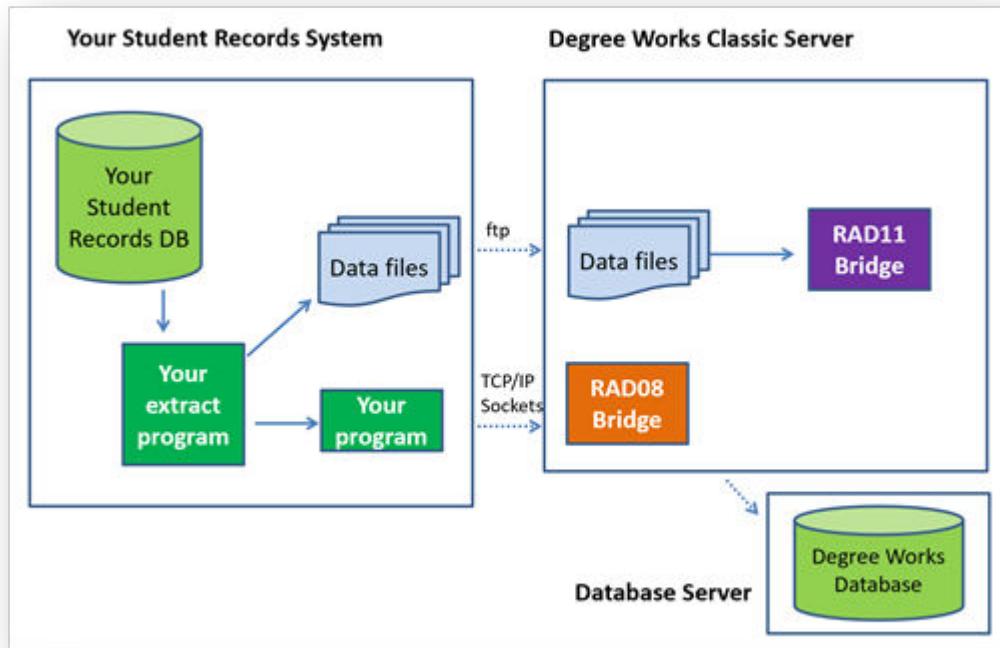
Degree Works and your student data

Updated: March 25, 2022

Degree Works relies on data extracted from your student system and bridged into its tables.

The following diagram shows the bridging of your records to Degree Works. This flow is used by non-Banner schools that write their own extract program and use RAD11 to bridge data into Degree Works.

Although the diagram shows your Student System on a machine different from where Degree Works resides, both systems can be on the same machine.



Bridge method #1 - nightly batch loading of student and other data

1. Your Extract Program reads from your Student Records System and writes the data to one or more data files.
2. The data files are FTP'd to the Degree Works system.
3. The RAD11 batch program is run pointing to the specific BIF file to be loaded.
4. RAD11 removes all data from the RAD database for a given ID and inserts the new, bridged data.

Bridge method #2 - used for dynamically sending data for one student

1. Your program pushes data to the RAD08 Bridge Listener. The program may be asked to send this data because of change to student data within or outside of the context of a user accessing Degree Works.
2. The RAD08 Bridge Listener receives the student data and writes it to the Degree Works database.
3. After RAD08 returns a FINISHED message, your program may proceed with sending a run audit request to Degree Works to be sure the latest audit for the student reflects the data changes.

Degree Works maintenance

Updated: March 25, 2022

Degree Works software requires that certain system management tasks be performed on a regular basis.

Some critical tasks must be restarted after a system failure. Other tasks need to be done on a regular daily, weekly, monthly, or yearly basis. And still other tasks must be performed as needed.

This section is a checklist of the system management tasks associated with the Degree Works software package.

Applications restart

Updated: March 24, 2023

There are times when you will need to restart the Degree Works applications.

You can restart the daemons running in classic by running webrestart and daprestart in Transit or by running those commands directly on the classic server's Unix console. To restart the Java applications, use the start and stop scripts you created on your application server.

When you make changes to UCX records through Controller or if you bridge in new UCX values, you do not need to restart any of the applications. Messages are broadcast over RabbitMQ to notify each of the applications of a change to the UCX. The software recognizes the changes and acts appropriately when new requests are received.

When you make changes to Shepherd Settings through Controller, save some exceptions, you do not need to restart any of the applications. Messages are broadcast over RabbitMQ to notify each of the applications of a change to the settings. The software recognizes the changes and acts appropriately when new requests are received. However, when you make changes to these settings you will need to restart the classic and Java applications:

- articulation.*

- classicConnector.amqp*
- core.amqp.*
- core.security.authenticationType
- core.security.cas.*
- core.security.externalAccessManager.enable
- core.security.ldap.*
- transferFinder.central.*

When you make changes to properties through Controller you do not need to restart the Java applications. The properties will be refreshed by the applications after about 20 seconds. However; a user currently logged into the application may need to sign out and sign on again before the changes will be visible.

Cron setup for Degree Works

Updated: March 25, 2022

Clients can configure cron to schedule reports or processes to run on a daily basis.

This is most simply done by scheduling a script to run in cron, where the script runs your daily processes.

As the Degree Works administrative user, save your daily maintenance script on your classic server and configure cron to run it. Use the crontab -e command to create or update cron.

```
$ crontab -e
# Run daily extract and maintenance jobs at 2:00 am on weekdays
0 2 * * 2-6 /home/dwadmin/cron/NightlyMaintenance.sh
```

In the example daily maintenance script below, a file named dw_maint_yymmdd.log will be created to contain messages from each process that is to be run. You must create the directory to contain these log files, for example, /home/dwadmin/cron/logs.

Be sure to modify the path to the dwenv and dwenv.config scripts, so that the Degree Works environment is sourced and all of the scripts and processors can be located.

If you want to run your student extract job through cron, you will need to run launchjob and provide a Transit parameter file in JSON format. For more information please see [The launchjob script](#) topic.

```
#!/usr/bin/ksh
#
# Run this script daily via cron
#
# Source the Degree Works environment
. /dworks/app/scripts/dwenv /dworks/dwenv.config
```

```
# create log file and name with date
export LOGFILE="/home/dwamin/cron/logs/`/bin/date +dw_maint_%y%m%d.log`"

echo "Starting daily extracts"                                >$LOGFILE
date >>$LOGFILE

echo "Running student extract via transit"                  >>$LOGFILE
launchjob $ADMIN_HOME/myjobs/rad30Stu.json                >>$LOGFILE 2>&1
echo "Daily extracts are complete"                         >>$LOGFILE

# remove all output older than 7 days
rmoldfiles $ADMIN_HOME/dgwspool 7                          >> $LOGFILE
rmoldfiles $ADMIN_HOME/logdebug 7                           >> $LOGFILE
rmoldfiles $ADMIN_HOME/jobdata 7                            >> $LOGFILE

echo "Restarting dap daemons"                             >> $LOGFILE
daprestart                                                 >> $LOGFILE
echo "Restarting web daemons"                            >> $LOGFILE
webrestart                                                >> $LOGFILE

# email the logfile results
mailx -s DW_extract_results admin@mySchool.edu < $LOGFILE
```

After system failure

Updated: March 25, 2022

It is recommended that these tasks be accomplished after a system failure.

It is suggested that you add appropriate symlinks in your /etc/rc3.d directory (or wherever is appropriate for your operating system) to the dw* scripts in the /etc/init.d directory so that the Degree Works daemon processes startup when your machine is booted. Only create symlinks for the daemons you want started. For example, if you are not using the Banner prerequisite daemons then do not create a symlink do the dw*.preq script in /etc/init.d.

You will need to execute the .profile in the same shell.

Launch other regularly scheduled Degree Works jobs.

Some Degree Works jobs are scheduled to run on a regular basis and should be launched if they are not in the job queue after recovery from a system failure. Your users may have other such jobs (the Bridge program), or you may have regular system management jobs (the backup) that need to be launched after the machine is restarted.

Classic server operating system change

Updated: March 25, 2022

Additional Steno libraries are required when moving your Degree Works environment to a different operating system.

When you want to create a new Degree Works environment on another machine you may lose your audit history. When the source machine has a different OS from the target machine the historic audits created on the source machine and stored in the database cannot be accessed on the target machine. Here the machine that is of concern is the classic server; the database server is not relevant. Specifically, when moving from a 32-bit to a 64-bit machine the audits will not be readable on the new machine. When moving from HP-UX, Sun Solaris or AIX to Linux the audits will also be lost because Linux is a little-endian machine while the other three are big-endian. However, in this second scenario where the old and new machines are both 64-bit your audits can still work on the Linux machine but only if a new version of the Steno library (/opt/steno/lib/libsteno64.a) is put into place on the Degree Works classic server. This new library can be obtained from the Action Line.

If you are moving from a 32-bit machine to a 64-bit machine regardless of the OS, your old audits will be lost. You should truncate the dap_audittree_dtl table and then the dap_audit_dtl table as these audits that were created on your 32-bit machine cannot be read on your new 64-bit machine.

Unlike with audits, your Scribe blocks are safe when moving to a new OS or are recompiling in 64-bit mode. However, you do need to delete the contents of your admin/daptrees directory in the new environment (or before recompiling in 64-bit mode) and run a dap16all. You will need to do this regardless of getting the new Steno library installed.

In summary, if you are moving from HP-UX, Sun Solaris or AIX to Linux or if you are moving from a 32-bit to a 64-bit machine you will need new binaries, in addition to the Steno library mentioned above for retaining audits. Please contact the Action Line to obtain these new binary files and install instructions.

Daily maintenance tasks

Updated: March 25, 2022

There are several maintenance tasks that should be done on a daily or weekly basis to keep Degree Works stable.

1. Check disk space for free space and fragmentation.
2. Run webrestart and daprestart as part of the nightly job. Also run preqrestart and radrestart if you are using them. If you are always wanting CPA data to be generated for new audits run resstop before running the bridge/extract and restart when it is done.
3. Run student extract. See the Cron setup section above.
4. Other daily jobs. Some Degree Works jobs are scheduled to run on a regular basis and should be launched if they are not in the job queue. Your users may have other such jobs (the

Bridge program), or you may have regular system management jobs (the backup) that need to be launched.

5. Check execution reports for Degree Works jobs. Examine the .log files in the admin/logdebug directory for the daemons. Use Transit to examine the jobs that were launched.
6. A daily partial backup (data only) of the Production account is important for safety and recovery.

Monthly maintenance tasks

Updated: March 24, 2023

On a monthly basis, copy test data from databases in Production to TEST environment using a database tool.

Note: You may not want to copy all audits (and associated CPA data) from Production to TEST, as that data is not needed in TEST and is a lot of data to copy. You should consider excluding these tables from your copy process: dap_audit_dtl, dap_audit_tree, dap_audtree_dtl, dap_result_dtl, dap_resclass_dtl, dap_resnoncr_dtl.

After copying your data into TEST, follow these steps:

1. Copy the files from daptrees in Production to TEST, overwriting those in TEST.
2. If you don't want to keep the production notes and exceptions in TEST, you can remove the contents of these tables: dap_note_txt_dtl, dap_note_dtl and dap_except_dtl.
3. Remove old rad_log_dtl records using the dapdelradlogs script. For example, \$ dapdelradlogs 20101231. All records created before the date specified are deleted. In this example records created before December 31, 2010 are deleted. This script should be run about one time per month to help clean up the Degree Works tablespace.

Semi-yearly or yearly maintenance tasks

Updated: March 24, 2023

On a semi-yearly or yearly basis, update your current term defaults in Degree Works.

Change the **Term used for filtering in student search** field in UCX-CFG020 WEBPARAMS.

This should be done anytime the bridged term changes. This term is used as the default.

Code patches between releases

Updated: March 25, 2022

Sometimes you may encounter problems and require an immediate fix from the ActionLine.

When you are provided with a new file with the fix you should archive your existing file using the “archive” script.

For example, if the ActionLine gives you a new version of dap43s.c you should first archive your old file by doing one of the following:

```
$ cd c  
$ archive dap43s.c  
$ archive $DGWHOME/src/c/dap43s.c
```

The archive script will copy the file into the \$DGWHOME/archive directory keeping the same directory structure (archive/src/c in this case). It is important to archive your existing version in case the new version you receive has bigger problems and you need to revert back to your archived version.

At any time you can view your archived files by doing the following:

```
$ ll -R $DGWHOME/archive
```

The archive script cannot be used to archive files under \$LOCAL_HOME or \$ADMIN_HOME – it can only be used to archive files under \$DGWHOME.

Update internal Tomcat libraries in standalone Java applications

Updated: September 30, 2022

The standalone Java applications contain embedded Tomcat libraries (jar files).

The tomcatUpdater script allows you to update these libraries should you want to use newer versions as they become available. The tomcatUpdater script can be found in /app/scripts on your classic server. It needs to be copied to your application server, if it is different from your classic server, to update embedded Tomcat libraries in these applications.

You will need to locate and download the appropriate Tomcat jar files for the version you choose. The latest version can be found at <https://mvnrepository.com/>. You should not update past the major version already in the jar file. There are 3 jar files that must be updated together:

- tomcat-embed-core
- tomcat-embed-el
- tomcat-embed-websocket

To see the version used in the current application jars, use the --show option of the tomcatUpdater:

```
tomcatUpdater --show transit.jar
```

The files should be loaded onto the system with the jar to be updated, although they do not need to be in the same directory. The command to update a jar file is:

```
tomcatUpdater --source {sourceDirectory} {dwJarFile} ...
```

The sourceDirectory is the directory containing the new Tomcat jar files and dwJarFile is the Degree Works application jar file to be updated. If the three new Tomcat jar files are in your current directory, you do not need to specify the --source option. More than one Degree Works application jar can be updated at a time by specifying a space-delimited list. For example:

```
tomcatUpdater --source /usr/lib/tomcatlibs transit.jar controller.ja  
r
```

Every given jar will be expanded into a temporary directory named after the jar but without the .jar extension. This directory must not exist before running the command. These directories will be removed automatically by the script.

As-needed tasks

Updated: March 25, 2022

This section describes Degree Works maintenance tasks that should be done as-needed.

Degree Works updates processing

Updated: March 25, 2022

Degree Works updates are not optional. If you do not process the updates, your software becomes increasingly outdated. At some point, Ellucian will stop supporting old versions of Degree Works at your site if you have received an update but not processed it.

Degree Works user addition

Updated: March 25, 2022

When personnel turnover takes place, you may need to add a new user for the Degree Works system.

Instructions for adding new users are in this document. Consult that section for the details of how to add a Degree Works user. Ellucian strongly recommends that the user be added first into the TEST account and then, after testing and training, be added into the Production account.

Blocks transfer between two different environments

Updated: March 25, 2022

Scribe block data can be copied between environments to keep data current and in sync.

Transferring blocks between environments consists of three steps:

- Use DAP40 in Transit to unload the Scribe blocks to a zip file.
- Use DAP41 in Transit to load the Scribe blocks from a zip file.
- Run DAP16 in Transit to reparse your new set of blocks.

DAP40 - Unload Scribe Blocks

You can specify that all blocks be unloaded, just the RA blocks, or just the RB blocks (the RB blocks are those generated from student plans). The default is to unload all blocks.

The contents of the dap_req_block table are unloaded along with the appropriate dap_next_id_mst records.

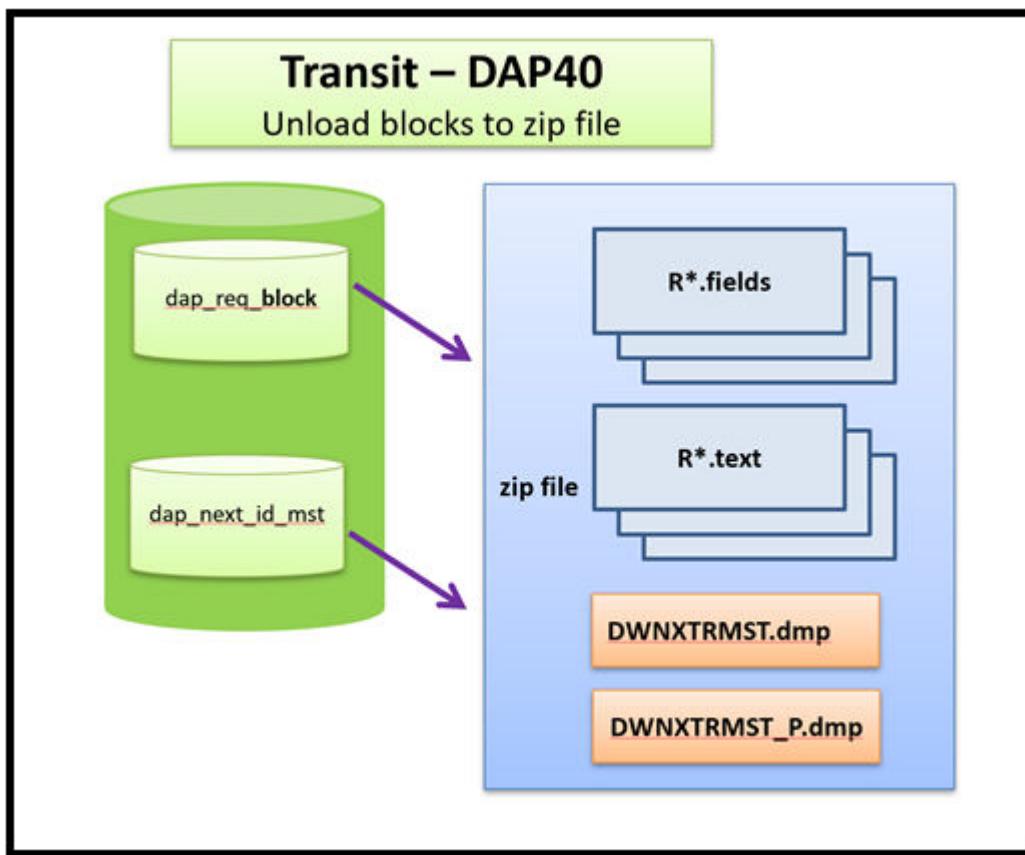
- The block meta data is unloaded to R*.fields files.
- The block text is unloaded to R*.text files.
- The dap_next_id_mst records are unloaded to DWNXTRMST.dmp (next RA block ID) and DWNXTRMST_P.dmp (next RB block ID).

These files are then combined into a single zip file.

When the job completes, an artifact of dapblocks.zip is attached to the job. When you download the zip file, it will have a default name of Jnnn_dapblocks.zip where nnn is the job number. Depending on your browser settings, the zip file will be automatically placed into your download folder, or you will be asked where you want to save the zip file. Regardless of how you download the file, you should consider creating a “Degree Works blocks” folder on your local machine to store these zip files. You may choose to have subfolders for each of your environments (for example, prod, test, dev, and so on), or you may choose to place FromProd, for example, in the zip file name so you can manage your zip files. Sometimes you will run the unload job to create an archive of your blocks, perhaps before making modifications per curriculum changes. Placing “ProdArchive” in the zip file name is something you might consider. You may also consider copying the zip files to a file server that is shared with your team. To help you remember whether this zip file contains RA, RB, or all blocks, you may want to add either “RA”, “RB” or “all” as part of your file name. This will help you later when you upload this zip file using DAP41 to load the blocks into the target environment. You will know exactly which blocks your zip file contains. You may also use a zip tool on your local machine to inspect the contents of the zip file.

Note:

- You cannot use the load/unload jobs to copy between two environments on two different versions of Degree Works if the dap_req_block database tables are different in any way. Do a SQL "desc dap_req_block" in both the source and target schemas to ensure the table descriptions are identical. If the tables are identical, you are safe to copy the blocks between the two environments.
- The unload job does not remove the blocks from the current environment. They are merely copied to files.



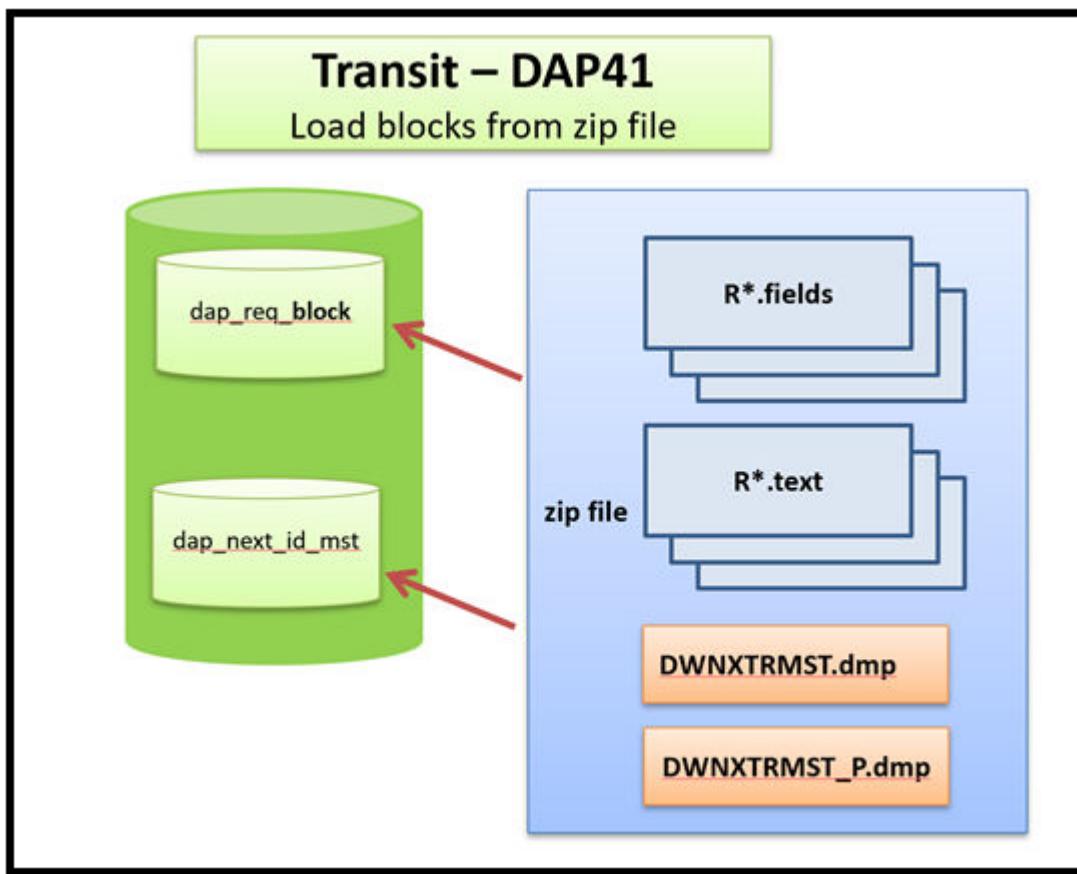
DAP41 - Load Scribe Blocks

When running DAP41, you need to upload the zip file containing the blocks you want to load into a target environment. The zip file you upload must be from an artifact you downloaded from a DAP40 job you ran in this or some other environment.

Unlike in the dapblockload script, the DAP41 job does not archive the existing blocks before loading in the new blocks. If you want to archive the existing blocks, you should first run DAP40 in the target environment and save the zip file artifact placing it into an archive folder or with "archive" in its name.

Before the new blocks are loaded, the contents of the dap_req_block, dap_req_crs_dtl, and dap_req_link_dtl database tables are deleted along with the RA and RB dap_next_id_mst records. However, if the user specifies that only the RA blocks are to be loaded, even if the zip file contains both RA and RB blocks, then only the RA blocks are deleted before being loaded. The same logic applies if you want to load only RB blocks.

The R*.fields and R*.text files from the zip file are then processed to create insert and update sql commands. Only the files matching the parameter of RB or RA will be processed. The dmp files are then loaded into the dap_next_id_mst so the creation of new blocks in Scribe will start with the next available block ID.



The `dapblockunload` and `dapblockload` scripts may be used instead of DAP40 and DAP41, but they require access to the classic server. Additionally, `dapblockunload` and `dapblockload` use tar files, while DAP40 and DAP41 use zip files. You cannot upload tar files created from `dapblockunload` into DAP41.

You can run DAP40 using `launchjob`, but the zip file artifact must be downloaded using Transit. You cannot, however, run DAP41 using `launchjob`, as there is not a way to supply the zip file of blocks to load.

Run DAP16 to reparse your new set of blocks

When the new blocks are loaded, you must use Transit to run DAP16 to reparse all of the blocks. Be sure to review the DAP16 report to examine any errors found in the parsing process, and check the log to ensure you don't have an unexpected number of unhooked exceptions. Fix all errors found. You can run DAP16 from Transit or run the `dap16all` script on the classic server to reparse all of your blocks.

Mappings transfer between two different environments
Updated: March 24, 2023

You can transfer mappings between environments by unloading them to a zip file with Transit DAP2 and then loading them from a zip file using Transit DAP43.

DAP42 Unload Mappings

DAP42 unloads all mappings to a zip file.

The contents of the dap_mapping_dtl, dap_map_cond_dtl, and dap_map_attr_dtl tables are unloaded along with the appropriate dap_next_id_mst record.

- The DAP_MAPPING_DTL data is unloaded DAP_MAPPING_DTL.dmp.
- The DAP_MAP_COND_DTL data is unloaded DAP_MAP_COND_DTL.dmp.
- The DAP_MAP_ATTR_DTL data is unloaded DAP_MAP_ATTR_DTL.dmp.
- The DAP_NEXT_ID_MST M record is unloaded to DAP_NEXT_ID_MST.dmp.

These files are then combined into a single zip file.

When the job completes, an artifact of mappingsunload.zip is attached to the job. When you download the zip file, it will have a default name of Jnnn_mappingsunload.zip, where nnn is the job number. Depending on your browser settings, the zip file will be automatically placed into your download folder, or you will be asked where you want to save the zip file. Regardless of how you download the file, you should consider creating a Degree Works folder on your local machine to store these zip files. You may choose to have subfolders for each of your environments (for example, prod, test, dev, and so on), or you may choose to place FromProd, for example, in the zip file name so you can manage your zip files. Sometimes you will run the unload job to create an archive of your blocks, perhaps before making modifications per curriculum changes. Placing ProdArchive in the zip file name is something you might consider. You can also use a zip tool on your local machine to inspect the contents of the zip file.

Note:

- You cannot use the load/unload jobs to copy between two environments on two different versions of Degree Works if the dap_mapping_dtl, dap_map_cond_dtl, or dap_map_attr_dtl database tables are different in any way. Do a SQL "desc <tablename>" on all four tables in both the source and target schemas to ensure the table descriptions are identical. If the tables are identical, you are safe to copy the blocks between the two environments.
- The unload job does not remove the blocks from the current environment. They are merely copied to files.

DAP43 Load Mappings

When running DAP43, you need to upload the zip file containing the mappings you want to load into a target environment. The zip file you upload must be from an artifact you downloaded from a DAP42 job you ran in this or some other environment.

Unlike in the dapmapload script, the DAP43 job does not archive the existing mappings before loading in the new mappings. If you want to archive the existing mappings, you should first run

DAP42 in the target environment and save the zip file artifact placing it into an archive folder or with "archive" in its name.

Before the new mappings are loaded, the contents of the dap_mapping_dtl, dap_map_cond_dtl, and dap_map_attr_dtl database tables are deleted along with the M dap_next_id_mst record.

The dmp files found in the zip file are loaded into the mappings table. Lastly the next-id dmp file is then loaded into the dap_next_id_mst so the creation of new mappings will start with the next available mapping ID.

The dapmapunload and dapmapload scripts can be used instead of DAP42 and DAP43, but they require access to the classic server. Additionally, dapmapunload and dapmapload use tar files, while DAP42 and DAP43 use zip files. You cannot upload tar files created from dapmapunload into DAP43.

You can run DAP42 using launchjob, but the zip file artifact must be downloaded using Transit. You cannot, however, run DAP43 using launchjob, as there is not a way to supply the zip file of mappings to load.

UCX tables transfer between two different environments

Updated: September 30, 2022

You can transfer UCX tables between environments by unloading them to a zip file with Transit DAP44 and then loading them from a zip file using Transit DAP45.

DAP44 Unload UCX tables

DAP44 unloads all UCX tables to a zip file.

The contents of the all UCX tables listed in SYS001 are unloaded. These files are then combined into a single zip file.

When the job completes, an artifact of ucxunload.zip is attached to the job. When you download the zip file, it will have the default name Jnnn_ucxunload.zip, where nnn is the job number. Depending on your browser settings, the zip file will be automatically placed into your download folder, or you will be asked where you want to save the zip file. Regardless of how you download the file, you should consider creating a Degree Works folder on your local machine to store these zip files. You may choose to have subfolders for each of your environments (for example, prod, test, dev, and so on), or you may choose to place FromProd, for example, in the zip file name so you can manage your zip files. Sometimes you will run the unload job to create an archive of your UCX tables. Placing ProdArchive in the zip file name is something you might consider. You can also use a zip tool on your local machine to inspect the contents of the zip file.

Note:

- Do not use the load and unload jobs to copy the UCX tables between environments that are on different versions of the Degree Works software. Some of the tables have entries that are specific to the software version, such as SYS935 and others. You should instead use the bulk options feature in Controller to load and unload a table at a time.

-
- The unload job does not remove the UCX tables from the current environment.
They are merely copied to files.

DAP45 Load UCX tables

When running DAP45, you need to upload the zip file containing the UCX tables you want to load into a target environment. The zip file you upload must be from an artifact you downloaded from a DAP44 job you ran in this or some other environment.

Unlike in the dapucxload script, the DAP45 job does not archive the existing UCX tables before loading in the new UCX tables. If you want to archive the existing UCX tables, you should first run DAP44 in the target environment and save the zip file artifact placing it into an archive folder or with "archive" in its name.

The dmp files found in the zip file are loaded into the UCX tables.

The dapucxunload and dapucxload scripts may be used instead of DAP44 and DAP45, but they require access to the classic server. Additionally, dapucxunload and dapucxload use tar files, while DAP44 and DAP45 use zip files. You cannot upload tar files created from dapucxunload into DAP45.

You can run DAP44 using launchjob, but the zip file artifact must be downloaded using Transit. You cannot, however, run DAP45 using launchjob, as there is not a way to supply the zip file of UCX tables to load.

Old student data removal from your Degree Works database

Updated: September 29, 2023

After bridging students into Degree Works for many years, you may want to remove inactive student data from the Degree Works database.

The easiest method is to run the RAD40 - Student Data Delete Processor in Transit. For more information, see the [RAD40 - Student Data Delete Processor](#) topic.

Banner schools can also run the deletestu script to delete students by bridge date. To run this script, issue the following command, where YYYYMMDD will be used to select students who have been bridged on or before this date. The script will create a file of the selected student IDs and store it in the \$ADMIN_HOME/data directory as deletestu.ids. The bannerextract deleteid process will then be run on the deletestu.ids file, removing these students from the Degree Works database.

```
deletestu YYYYMMDD
```

If needed, you can edit the deletestu.ids file before the delete process takes place. The script has two prompts, one to confirm the cut-off date and one to confirm the deletion. When the script displays the number of students to be deleted and you are prompted **Confirm delete?**, respond with **N**. This will terminate deletestu, but you can then edit deletestu.ids to modify the student IDs to be deleted. After modifying the file, issue the command:

```
bannerextract deleteid deletestu.ids
```

Database cloning from test to production

Updated: March 25, 2022

When you are ready to go live with Degree Works, you may want to clone the Degree Works database from your test environment to your production environment.

You will need to review certain Shepherd settings (shp_settings_mst table). These settings should be different between environments but others may need to be different also based on your setup.

```
core.security.cas.callbackUrl  
core.classicUrl.serverDomain  
classicConnector.dap08.port  
classicConnector.serverNameOrIp
```

If you can't get into Controller to modify these settings you can use the shpsettingsset script to change the value. For example:

```
$ shpsettingsset classicConnector.dap08.port 1934
```

You may also use shpsettingsshow to view the settings. Specify any part of the setting:

```
$ shpsettingsshow classicConnector
```

Note: Do not use these scripts to copy the UCX tables between environments that are on different versions of the Degree Works software. For example, if PROD is on the 5.0.5 release and TEST is on the 5.0.7 release you should not use these scripts. Some of the tables have entries that are specific to the software version, such as SYS935 and others. You can instead use tableunload/tableload or the bulk options feature in Controller to load/unload a table at a time.

Email notification configuration maintenance

Updated: March 25, 2022

Degree Works batch processes support email notification of job status.

In Controller, you can go to the Configuration tab and search for **email** to find the email settings.

You can optionally set this email address value to give the address to be used when a user clicks **Reply**. If this value is not set, the To address is used as the reply-to address.

```
core.notification.fromEmail
```

You can optionally set a CC email address used for all of the processors listed above. This might be useful for copying the registrar or an office assistant.

```
core.notification.ccEmail
```

Ensure the SMTP_MAIL_SERVER variable is defined in your environment. If it is not, you can set it in dwenv.config. If this is not set to a correctly configured mail server, e-mail notifications will not work.

Set this to your machine's mail server if not already set. For example, mailhost.mymachine.edu.

```
export SMTP_MAIL_SERVER=mailhost.mymachine.com
```

Database credentials changes

Updated: September 30, 2022

When you change the database user or password for the configured Degree Works user (see the DB_LOGIN environment variable) or the SIS database user (see the DB_LOGIN_BANNER environment variable), you must configure the new credentials in three different areas.

In all the configurations, an attempt has been made to hide the database passwords as much as possible. That is, they do not appear unencrypted in files, or in a user's environment.

Warning! This does not prevent a user who is logged in to the server from seeing the plain password. It is still critical to secure access to the server to only those individuals who would be authorized to have full access to the database.

Classic applications

The database connections for the classic software are configured in the file dwenv.config in the \$DGWBASE directory. This file, which is sourced in during the logon process, sets the DB_LOGIN environment variable. This variable contains the user name (schema) and connection. For example:

```
DB_LOGIN="dgwmgr@dwdevl"
export DB_LOGIN
```

In the above example, **dgwmgr** is the user name and **dwdevl** is the connection name.

The database password is configured using the **setdbpasswords** command. For example,

```
setdbpasswords --password mydwpwpassword --transitpassword mytransitpa
ssword --sispassword mysispassword
```

In the above example, **mydwpwpassword** is the Degree Works database password associated with the DB_LOGIN user, **mytransitpassword** is the Transit database password associated with the DB_LOGIN_TRANSIT user, and **mysispassword** is the password associated with the SIS password.

To prevent your database passwords from being visible in server files, you can generate an encrypted string for a password. To get the encrypted string, you should log in to the classic environment that uses this database (has the same user), and issue the following command:

```
showdbpasswords -p -t
```

You must have already run the **setdbpasswords** command, as described above, in that environment. It will produce the following output (example):

Substitute the following for the DW password in the server.xml data source and jdbc.properties files:

```
ENC(Smf/7G6r2Erw5c2YIkCIJw==)
```

The encrypted value for the TRANSIT password is:

```
ENC(ELG7VoJoykLrIHnCqGfljA==)
```

Copy those values (including the ENC and parentheses) and paste them into your jdbc.properties, start scripts, and so on, in place of the text value of your passwords.

Java applications on the classic server

There are a few Java applications run on the classic server. These require the username and password to be configured in a jdbc.properties file in the \$ADMIN_HOME/common directory. See the following abridged example:

```
dw.jdbc.driverClassName=oracle.jdbc.driver.OracleDriver
dw.jdbc.url=jdbc:oracle:thin:@dbhost.myschool.edu:1521/dwprod
dw.jdbc.username=dgwmgr
dw.jdbc.password=ENC(Smf/7G6r2Erw5c2YIkCIJw==)
```

Java self-contained web applications

Java applications such as Responsive Dashboard, Controller, Transit, and Transfer Equivalency Self-Service are started with a shell script. See the following abridged example for Transit:

```
export DW_DATASOURCE_URL="jdbc:oracle:thin:@dbhost.myschool.edu:1521
/dwprod"
export DW_DATASOURCE_USERNAME="dgwmgr"
export DW_DATASOURCE_PASSWORD="ENC(Smf/7G6r2Erw5c2YIkCIJw==)"
export TRANSIT_DATASOURCE_URL="jdbc:oracle:thin:@dbhost.myschool.edu
:1521/dwprod"
export TRANSIT_DATASOURCE_USERNAME="dgwtransit"
export TRANSIT_DATASOURCE_PASSWORD="ENC(ELG7VoJoykLrIHnCqGfljA==)"
export DEPLOY_LOCATION="/u01/dw/webapps"

...
/usr/java/latest/bin/java -jar $DEPLOY_LOCATION/transit.jar \
> /usr/tomcat/default/logs/startup-transit.log 2>&1 &
```

The DW_DATASOURCE_PASSWORD and TRANSIT_DATASOURCE_PASSWORD variables should contain the same encrypted strings as the example above.

Degree Works standing daemons

Updated: March 24, 2023

There are several daemon processes that run on the Degree Works classic server that handle service requests.

The following daemons should always be running on the classic server for the Degree Works software to function properly:

- dap08 - Scribe sockets listener daemon
- web07 - Dashboard daemons
- transitexecutor.jar - Transit Batch Executor

Optionally you can choose to run these daemons:

- dap25 - Create CPA results
- rad08 - Dynamic Bridge (non-Banner)
- dap61 - Banner pre-requisite checker
- dap62 - Banner pre-requisite descriptions

These jobs can be controlled by using the scripts which can be entered at the system prompt. The restart scripts can also be run through Transit. However, if you have multiple classic servers running the restarts will only occur on the machines where the Transit Batch Executor is running. If it is running on multiple machines the restart request could go to any of the machines.

When you have multiple classic servers it is important to know which daemons should be or should not be run on each server:

- dap08 - run on only one classic server
- dap25 - run on only one classic server
- web07 - run on one or more servers as needed
- rad08 - run on one or more servers as needed
- dap61/dap62 - run on one or more servers as needed
- transitexecutor - run on one or more servers as needed

webshow

Updated: March 25, 2022

The webshow script shows the running web07 processes and the parent utl01 process with scope=web. Also shown is the status of the message queue used to communicate between the web daemons.

```
$ webshow
OWNER      PID      PPID      STARTED      CPUTIME COMMAND
=====  =====  =====  =====  =====  =====
=====
dwadmin    8785      1 Mar 7      00:00:00 /dw/dwprod/app/bin/utl01x db=
dwadmin_proda scope=web

dwadmin    8830    8785 Mar 7      00:00:00 web07x db=dwadmin_proda
```

```
dwadmin 8831 8785 Mar 7 00:00:00 web07x db=dwadmin_prod&
dwadmin 8832 8785 Mar 7 00:00:00 web07x db=dwadmin_prod&
```

```
===== Degree Works Web Message Queue: Key: 603 = 0x25b =====
MsgQ Key MsgId Owner Perms #Bytes #Msgs
0x0000025b 31555591 dwadmin 666 0 0
```

webstart

Updated: March 25, 2022

The webstart script uses the dwinit.web script located in \$DGWHOME/initd.

The webstart script starts up a utl01 process and tells it to startup a series of web07 child processes. You control the number of web07 processes that are started by changing the classic.daemons.web07.count setting in Controller.

The web07 count is usually a high number like 30 or 50; you will need to test to determine what works best for your system.

If the daemons are already running a webstop is performed first. This happens for all “start” scripts.

webrestart

Updated: March 25, 2022

The webrestart script runs webstop followed by webstart.

webstop

Updated: March 25, 2022

The webstop script stops the utl01 process with scope=web causing the child web07 processes to also stop.

dapshow

Updated: March 24, 2023

The dapshow script shows the running dap08 process and the parent utl01 process with scope=dap.

```
$ dapshow
OWNER PID PPID STARTED CPUTIME COMMAND
```

```
===== ===== ===== ===== ===== ===== ===== ===== ===== ===== =====
=====
dwadmin    9785      1 Mar 7      00:00:00 /dw/dwprod/app/bin/utl01x db=
dwadmin_prod@ scope=dap

dwadmin    9786  9785 Mar 7      00:00:00 dap08x -p7701 db=dwadmin_prod
@
```

dapstart

Updated: March 24, 2023

The dapstart script is similar to webstart.

The difference is that dwinit.dap is used.

Only one dap08 daemon will be started. For each parse request from Scribe, a new child dap08 process is forked, but it dies after the request is processed.

daprestart

Updated: March 25, 2022

The daprestart script runs dapstop followed by dapstart.

dapstop

Updated: March 24, 2023

The dapstop script stops the utl01 process with scope=dap causing the child dap08 process to also stop.

radshow

Updated: March 25, 2022

The radshow script shows the running rad08 processes and the parent utl01 process with scope=rad.

Note that the number of rad08 processes running will always be one more than the number specified in `classic.daemons.rad08.count`. This is because utl01 always spawns a single rad08 and this rad08 parent then spawns the number of children specified by `classic.daemons.rad08.count`. In the example below the `classic.daemons.rad08.count` was 4.

```
$ radshow
OWNER      PID      PPID      STARTED      CPUTIME      COMMAND
=====  =====  =====  =====  =====  =====
=====
dwadmin      856      1 11:33      00:00:00 /dw/dwprod/app/bin/utl01x db=
dwadmin_prod@ scope=rad

dwadmin      858      856 11:33      00:00:00 rad08x db=dwadmin_prod@ -p800
1 -c3
dwadmin      859      858 11:33      00:00:00 rad08x db=dwadmin_prod@ -p800
1 -c3
dwadmin      860      858 11:33      00:00:00 rad08x db=dwadmin_prod@ -p800
1 -c3
dwadmin      861      858 11:33      00:00:00 rad08x db=dwadmin_prod@ -p800
1 -c3

dwadmin      861      858 11:33      00:00:00 rad08x db=dwadmin_prod@ -p800
1 -c3
```

radstart

Updated: March 25, 2022

The radstart script is similar to webstart.

The difference is that dwinit.rad is used and the Controller setting is classic.daemons.rad08.count. This should be a low number if you don't send Degree Works many simultaneous dynamic bridge requests your system. This can be higher if you have many simultaneous pushes of data. Most schools do not use rad08 and this is only used by non-Banner schools at that.

radrestart

Updated: March 25, 2022

The radrestart script runs radstop followed by radstart.

radstop

Updated: March 25, 2022

The radstop script stops the utl01 process with scope=rad causing the child rad08 processes to also stop.

resstart

Updated: March 25, 2022

The resstart script is similar to webstart.

The difference is that dwinit.res is used and these Controller settings are used:

```
classic.daemons.dap25.auditMaximum  
classic.daemons.dap25.auditsPerChild  
classic.daemons.dap25.loopMaximum  
classic.daemons.dap25.sleepSeconds
```

See [Advanced Reporting](#) for more details on these settings.

resrestart

Updated: March 25, 2022

The resrestart script runs resstop followed by resstart.

resstop

Updated: March 25, 2022

The resstop script stops the util01 process with scope=res causing the child dap25 processes to also stop.

tbestart

Updated: March 25, 2022

The tbestart script starts the transitexecutor.jar running the Transit Batch Executor. This is needed to allow Transit to launch jobs. It is also needed for the launchjob script to function.

tberestart

Updated: March 25, 2022

The tberestart script runs tbestop followed by tbestart. Unlike with daprestart and webrestart, you do not need to run tberestart every night in your cron script.

tbestop

Updated: September 29, 2023

The tbestop stops the transitexecutor.jar.

For more information, see the [The tbestop script](#) topic.

preqstart

Updated: March 25, 2022

The preqstart script is similar to webstart.

The difference is that dwinit.preq is used and the Controller settings are classic.daemons.dap61.count and classic.daemons.dap62.count.

The dap61 and dap62 count can be higher if you have many requests coming from Banner. Running preqshow can show if the queue is getting backed up due to not having enough daemons running.

preqrestart

Updated: March 25, 2022

The preqrestart script runs preqstop followed by preqstart.

preqstop

Updated: March 25, 2022

The preqstop script stops the utl01 process with scope=preq causing the child dap61 and dap62 processes to also stop.

Load balancing and proxies

Updated: March 25, 2022

Load balancing is the technique of creating multiple physical or virtual servers that run the same application. Reverse proxies are used for many reasons, including caching content and offloading SSL encryption. Both load balancers and proxies involve intercepting traffic going to and from the Degree Works server.

Load balancing may be done for performance reasons – allowing you to spread the load to different network segments. It may also be easier to respond to changing demand levels by adding or removing servers, thereby optimizing your server resources. It may be done for reliability reasons, allowing failing systems to be removed from service while not disrupting overall responsiveness. Many factors go into architecting a load balanced system, and it is not in the scope of this document to cover them all. Instead, this section discusses the unique aspects of the Degree Works set of applications as they apply to load balancing.

Degree Works began as a monolithic application that was intended to be run on one server. As it has evolved, it incorporated newer technologies that lent themselves to load balancing. It is now possible to load balance most of the Degree Works applications, with some caveats.

The load balancing capabilities of any piece of application code depend on the technology used in its construction. Ellucian uses our classic applications that run daemon processes on a Unix system, and Java administrative applications that run standalone.

The nature of the application should also factor into the load balancing design. Administrative applications are used by a limited number of users, even at large institutions. It may not be desirable to load balance these at all, depending on the objectives you are trying to meet.

Classic load balancing

Updated: September 30, 2022

The classic Degree Works server, which runs the web daemons (started through webstart) use a RabbitMQ queue to communicate with the Responsive Dashboard and other applications. As such, it can be load balanced with a few caveats.

There are several configuration and localization files still on the classic server. Any changes to these files must be kept in sync between all the load balanced servers. Examples of these include dwenv.config. Generally, any time you need to modify a file on one of the classic servers, you must make those changes to all other servers as well, or copy the files to all the other servers.

Each server will have its own web.log file, so if you use the webanalyze or webstats tools, you will need to do that on each server, and then combine the results. The webanalyze that you might run from Transit will only report from the server that the Transit Batch Executor is running.

Java application load balancing

Updated: March 25, 2022

Load balancing Java applications such as Responsive Dashboard and Transfer Equivalency Self-Service is straightforward, as generally nothing special is required.

If your load balancer or proxy terminates the SSL connection, you must add a few additional environment variables to your startup script:

```
export SERVER_FORWARD_HEADERS_STRATEGY=framework  
export SERVER_TOMCAT_PROTOCOL_HEADER="x-forwarded-proto"
```

```
export SERVER_TOMCAT_REMOTE_IP_HEADER="x-forwarded-for"
export SERVER_TOMCAT_REMOTE_IP_PROXIES_HEADER="x-forwarded-by"
```

System performance

Updated: March 25, 2022

The best time to begin thinking about performance is before performance issues are reported.

You should become familiar with the tools used to monitor performance and collect performance metrics from your systems at different times of the day and year. Find out how the system performs during periods of light and heavy usage. When issues occur, you can compare current metrics against these baselines and that will help you identify the components that are causing the issue.

Troubleshooting

Define the problem by gathering data about when the issue occurred and the task the user was trying to accomplish when the issue occurred. Usually users think of "poor performance" as "poor response time". Check if that is the case, or if the user meant that a batch job takes too long to run.

Try and determine the programs that were running when the issue occurred. Find out which users report the issue, and if the issue is constant or intermittent. If response time is defined as the time from clicking Run Audit to seeing something returned, then find out that duration. Ask your users to keep a log of the service, ID, time-of-day, and response time whenever the issue occurs.

Ask the user to call you immediately when the issue occurs. You can run some simple diagnostics to gather system-wide information.

If you suspect an issue with the Degree Works software, call Ellucian and send us the information you have gathered. Ellucian will work with you to further analyze the problem and, if needed, will ask you to call your computer hardware vendor.

System management and performance

Updated: March 25, 2022

In many ways the performance of Degree Works is linked to your overall system performance.

If your computer is overloaded for other tasks, then it may be overloaded for Degree Works processes as well. Ellucian and your computer hardware vendor can help you analyze if you have the right hardware for the tasks you want to perform.

Sufficient disk space and database management often help improve software performance.

Check the available disk space on the system, regularly. Check the degree of fragmentation as well, if you have only small clusters of disk space, you may need to perform some disk space maintenance.

Database server configuration

Updated: March 25, 2022

There are several best practices and techniques that DBAs can use to optimize an Oracle database for best performance.

A complete discussion of those is beyond the scope of this document, this document focuses on those items that apply to Degree Works.

The following configuration parameter is the most important configuration parameter specific to Degree Works:

```
cursor_sharing=FORCE
```

If this parameter is not set, system performance will be adversely affected.

The database character set should be set to WE8MSWIN1252. Currently, UTF-8 is not supported.

Some additional initialization parameters might help with performance of the Degree Works databases, but optimum values will depend on the RDBMS version and database size; consult your Oracle documentation. Oracle is largely self tuning, so care should be taken when setting any non-default values.

The DBA can refer to Oracle performance documentation for more information.

Java database pooling configuration

Updated: September 29, 2023

Java programs obtain a database connection from a pool of open connections.

If the pool is too small, performance may suffer, and the application may even stop working entirely. The default size is adequate for a test account but will need to be increased for production.

You can adjust this value by exporting an environment variable in the application start script. The exact variable differs depending on the application.

The following applications use the DW_DATASOURCE_MAX_ACTIVE environment variable:

- Composer
- Responsive Dashboard
- Transfer Equivalency Self-Service

The following use the DW_DATAPOOL_MAX_ACTIVE environment variable:

- API Services

-
- Controller
 - Transfer Equivalency Admin
 - Transit User Interface and APIs (for access to the Degree Works database)

In addition, Transit User Interface and APIs also access the Transit database and use the TRANSIT_DATAPool_MAX_ACTIVE environment variable for that pool.

Example:

```
export DW_DATASOURCE_MAX_ACTIVE=100
```

The same attributes you can change in the Tomcat server.xml file can be set in these startup scripts by exporting environment variables with names beginning with DW_DATASOURCE_ or DW_DATAPool_ plus the attribute name (for example, export dw_datasource_maxIdle=50).

Example:

```
Replace {poolsource} with DATAPool or DATASOURCE depending on the application:
```

```
export DW_{poolsource}_INITIAL_SIZE=10
export DW_{poolsource}_MIN_IDLE=10
export DW_{poolsource}_MAX_IDLE=10
export DW_{poolsource}_MAX_ACTIVE=300
export DW_{poolsource}_TIME_BETWEEN_EVICTION_RUNS_MILLIS=1800000
export DW_{poolsource}_REMOVE_ABANDONED=true
export DW_{poolsource}_REMOVE_ABANDONED_TIMEOUT=120
export DW_{poolsource}_VALIDATION_QUERY="select 1 from dual"
export DW_{poolsource}_TEST_WHILE_IDLE=true
export DW_{poolsource}_TEST_ON_BORROW=true
export DW_{poolsource}_VALIDATION_QUERY_TIMEOUT=10
```

Database pool monitoring

The database pool exposes metrics through the Java Management Extensions (JMX). There are many tools that can monitor these metrics. JConsole is a free application that comes with Java that will display these values. For Tomcat deployed apps, these metrics can be viewed under the MBean Catalina/DataSource/javax.sql.DataSource. For microservice apps, they can be viewed under net.hedtech.degeworks/JmxBasicDataSource/datasource. There is a getState operation that will report all the relevant values for the datasource.

Degree Works web server daemon configuration

Updated: March 24, 2023

The primary configurations for the Degree Works daemons are the instance counts for the various daemons.

There is only one daemon that can be configured: web07. The number of these is controlled by this Controller setting:

```
classic.daemons.web07.count
```

There is no formula for setting this configuration. Too few servers will choke off responses to requests. Too many will use up system resources (memory, process space).

You can use the webtime script to analyze a web.log script to determine how many requests were handled during the time period covered by the log. There are no exact guidelines for the number of web07 daemons. Try different values for this parameter and check if it has any impact on performance.

If you have under-configured these settings, your response times will suffer, yet neither the Degree Works classic server, the application server, nor the database server will be taxed.

The web performance is significantly affected by a refresh or audit. Normally, a student will simply view audits that have previously been calculated, generally during the last student extract batch run. However, there are settings that can be enabled which would cause new refreshes and audits to be generated. These are set in the UCX-CFG020 REFRESH record. Be careful with these settings because they have a very significant effect on performance. For similar reasons, what-if audits can also have an impact on performance.

There are also several operating system parameters that may limit the number of servers that can be run. The ulimit command lists the important ones:

```
$ ulimit -aS
time(cpu-seconds)    unlimited
file(blocks)          unlimited
coredump(blocks)      0
data(kbytes)          unlimited
stack(kbytes)          10240
lockedmem(kbytes)     32
memory(kbytes)        unlimited
nofiles(descriptors) 4096
processes              77824
$ ulimit -aH
time(cpu-seconds)    unlimited
file(blocks)          unlimited
coredump(blocks)      unlimited
data(kbytes)          unlimited
stack(kbytes)          unlimited
lockedmem(kbytes)     32
memory(kbytes)        unlimited
nofiles(descriptors) 4096
processes              77824
```

The first example details the soft limits (`ulimit -aS`), which can be changed for any particular session, and the second example (`ulimit -aH`) details the hard limits, which can only be changed by the system manager. The commands may be slightly different on your system, as may the means to set the limits and the defaults. The limits of particular concern are the `nofiles` and `processes` values. The `nofiles` value limits the number of open files, and the `processes` value limits the number of running processes. As these limits were meant primarily to prevent “runaway”

processes from consuming the entire machine, there is little negative impact in making them very large. If they are too small you may see error messages in your log files, such as “Too many open files” or “Too many processes”. You may also notice that you do not have as many servers running as you configured. On most systems, a nofiles setting of 4096 and processes setting of 77824 should be sufficient.

Batch processing performance

Updated: March 25, 2022

The primary configuration used to adjust batch performance is the transit.*.workerCount settings. If the batch job is running on the classic server, setting this variable to more than the CPU count can decrease the overall time to completion. You will need to test to determine the optimal setting for your environment for each job.

When running the resstart to build CPA results be sure to consider the classic.daemons.dap25.auditMaximum and the classic.daemons.dap25.auditsPerChild settings. They control the maximum number new audits that will be processed together and how many audits will be assigned to each spawned dap25 child process. The number of child processes spawned is based on these two settings. For example, if the max is set to 1,000 and the per-child is 100 then 10 child processes will be spawned; each child will handle 100 audits. You may need to alter these settings if you are finding that the building of CPA results is taking too long.

Another factor that can significantly affect the building of CPA results is the flag settings in the UCX-CFG020 RESULTS record. Each flag set to “Y” will increase the time it takes to process a student. You should only turn on those flags for data that you will be using.

Custom indexes

Updated: March 25, 2022

When you create your own indexes into the Degree Works database tables you cannot create indexes with names longer than 21 characters as they are not supported.

Transit Jobs

Updated: March 25, 2022

Review Transit job reference information as you work with Transit and Degree Works.

ADMIN Administrative tasks

Updated: March 24, 2023

You can use Transit to run certain administrative tasks without IT assistance.

Access to run ADMIN jobs in Transit is granted with the TRADMIN Shepherd key.

The ADMIN job allows you to run tasks on your Degree Works classic server.

Command	Description
dap22dbg	Run audit and capture debug files
rad30dbg	Run Banner extract and capture debug
logview-dap	View the 1st 100 and last 1000 lines of dap.log
logview-prq	View the 1st 100 and last 1000 lines of preq.log
logview-rad	View the 1st 100 and last 1000 lines of rad08.log
logview-res	View the 1st 100 and last 1000 lines of dap25.log
logview-web	View the 1st 100 and last 1000 lines of web.log
daprestart	Restart the dap08 daemons
prerestart	Restart the Banner prereq daemons
radrestart	Restart the rad08 daemons
resrestart	Restart the res/CPA daemons
webrestart	Restart the web07 daemons
dapshow	Show the dap08 daemons currently running
preqshow	Show the Banner prereq daemons currently running
radshow	Show the rad08 daemons currently running
resshow	Show the results/CPA daemons currently running
webshow	Show the web07 daemons currently running
dapstop	Stop the dap08 daemons
preqstop	Stop the Banner prereq daemons
radstop	Stop the rad08 daemons
resstop	Stop the results/CPA daemons

Command	Description
webstop	Stop the web07 daemons
viewlocals	View a list of localized resources
webanalyze	Analyze the active logdebug/web.log
weblogon	Test a user's logon to view key list
vieworphaned	View orphaned unhooked/unenforced exceptions
deleteorphaned	Delete orphaned unhooked/unenforced exceptions
envcheck	Check the Degree Works environment
updateverify	Verify the previous update
get weblog	Get the full web.log
gettransitlog	View the full transitexecutor.log
dapucx2eqv	Copy CFG070 to the dap_eqv_crs_mst
viewrules	Export all rad_currrule_dtl records into a zip file.

The dap22dbg (run an audit) and rad30dbg (run the Banner extract) scripts are useful when working with the Action Line. You need to supply the student ID in the field provided when running these commands. When the job is complete you can view the log file and tar file on the Manage Jobs tab in Transit.

The logview tasks are a great way to peek at the log files on the classic server without having to contact IT. These are extremely helpful when working with the Ellucian Action Line.

The restart tasks should be scheduled to run every night but you may have a need to restart the servers in the middle of the day. This is useful when you do not have command prompt access on the Degree Works classic server.

The show tasks are a handy way to check on the status of the running daemons on the classic server.

The stop tasks should be used rarely and with care. These will stop the running daemons on the classic server. Ellucian recommends using the restart commands instead, which first stop then start the daemons.

The viewlocals task is a great way to remind you what resources you have localized in Composer. Your team will find this very useful and it is a great way to also communicate to the Action Line what resources you have localized.

The webanalyze task allows you to get a summary of activity from the active web.log on the classic server.

The weblogon script is useful for examining the security keys that will be given to a particular user when they logon to Degree Works. You need to supply the user's logon ID in the field provided. You do not have to provide the password as this special tool does a pseudo logon that does not require the password.

The vieworphaned task allows you to view a list of orphaned unhooked and unenforced exceptions. Orphaned exceptions are those that have no corresponding rad_goal_dtl. After

viewing the orphaned exceptions you may choose to run the deleteorphaned task. (The vieworphaned task runs the deleteunhooked classic script with a parameter of N.)

The deleteorphaned task allows you to delete the orphaned unhooked and unenforced exceptions. You should first run the vieworphaned task to view the orphaned exceptions before running this delete task. (The deleteorphaned task runs the deleteunhooked classic script with a parameter of Y.)

The envcheck, gettransitlog, and getweblog tasks are helpful when working with the Action Line. They may ask you to run these and send the resulting log file.

When you manually update CFG070 in Controller, you should run dapucx2eqv to copy the CFG070 records into the dap_eqv_crs_mst. When you run the equiv Banner extract, you do not need to run this script as the extract updates both CFG070 and the dap_eqv_crs_mst.

To view all of the curriculum rule data bridged into Degree Works, use the viewrules task. When you run viewrules, all rad_currule_dtl records will be loaded into a .csv file and then zipped up. You can then download the. zip file, unzip it and view your curriculum rules.

AUD01 - List unhooked and unenforced exceptions

Updated: March 25, 2022

The AUD01 report will show all unhooked and all unenforced exceptions.

Access to run AUD01 in Transit is granted with the TRAUD01 Shepherd key.

It is suggested that this report be run on a regular basis to identify if exceptions have become unhooked. However, if you use label-tags on all of your labels exceptions should rarely become unhooked. See the [Label Tags](#) topic for more information.

This report is run against all records, as there are no selection options from Transit.

After running this report you might consider running the vieworphaned task under the ADMIN report. This allows you to view the list of orphaned unhooked and unenforced exceptions; this orphaned list will be a subset of the list produced by AUD01. You may then choose to run the deleteorphaned task to remove those that are no longer associated with a student's goal.

For example:

Unhooked exceptions					
6789	RR	BS	RA000112	UN	testi
ng the RR except BH					
6789	FC	BS	RA000112	UN	testi
ng fop exceptions					
1633092319	FC	BA	RA000112	UN	Force
complete this rule					
1646953703	FC	BA	RA000235	UN	force
complete					
1960	RR	BS	RA000263	UN	Repla

```

ce MATH 1305 with BUS 2397
1493208187    NN    BS          RA000515      UN      REM C
IS3281 AND REQ 2 CLAS NOT 1
1493208187    AH    BS          RA000515      UN      apply
hist 3500 from elec
1493208187    AA    BS          RA000515      UN      allow
psyc 1100 wth 5 cre, stu has 4 cred in elec
1493208187    FC    BS          RA000515      UN      testi
ng force complete
1493208187    RR    BS          RA000515      UN      repla
ce cs 3660 w/ engl3003 from electives
001234        FC    BS          RA000515      UN      Force
complete this rule
001234        FC    BS          RA000515      UN      Force
complete this rule
Unenforced exceptions
6789         RR    BS          RA000112      R2      Test-
-replace COMM 1000 with ART 1020
1969         AA    BS          RA000901      R2      Allow
hist 3400 to apply here.
B123456       NN    BA          RA001363      R2      testi
ng 1-3H96G8
B123456       FC    BA          RA001363      R2      Macro
econ course taken in high school
B003          FC    BA          RA001399      R2      Pleas
e force this rule complete - per Stephen's

```

AUD02 - Delete audits by freeze type and date

Updated: March 25, 2022

The AUD02 processor allows you to delete frozen audits older than a certain date.

Access to run AUD02 in Transit is granted with the TRAUD02 Shepherd key.

The freeze type is a required field to prevent you from deleting all audits by accident. The primary use of this is to clean out certain frozen audits periodically. This is especially handy when frozen audits are created using DAP27 to run batch what-if audits.

For example:

```

Deleting audits older than 20130609 with a freeze-type of WHATIF
DAPDELAUDITS deleted 7 audits from the database

```

BAN62 - Satisfactory Academic Progress Processor

Updated: March 25, 2022

BAN62 populates the Banner tables SHRSAPP and SHRSARJ with Degree Works audit results.

Access to run BAN62 in Transit is granted with the TRBAN62 Shepherd key.

The run-time questions for BAN62 are:

Create new audit? - When this box is not checked, BAN 62 will process the most recent audits for the selected pool of students, which is based on the selection criteria entered in the Selection Tab. None of the other check boxes will be available. When this box is checked, new audits will be created for the selected pool of students. In addition, the following options become available:

- Refresh data from Banner before running audits. - when checked, the student data will be refreshed from Banner before a new degree audit is processed.
- Select Level - the selected pool of students will be filtered to the School Level (Banner Level) selected in the picklist.
- Include In-progress classes? - in-progress classes will be included in the newly generated audit.
- Include Preregistered classes? - preregistered classes will be included in the newly generated audit.
- Freeze these new audits? - the newly generated audits can be frozen with a new freeze type "SAPFRZ".
- Audit Description - it is optional to include a freeze description, but it is recommended that when running the BAN62 processor that the associated audits are kept for reference.
- Select Term - the current term in which the user is running the BAN62 processor. This term value will be written to the SHRSAPP_TERM_CODE_EVAL column in Banner. This is a required field that reflects the term for which academic progress is being evaluated.

For additional information about this processor, see the [Satisfactory academic progress](#) topic.

DAP16 - Parse Requirements Processor

Updated: March 25, 2022

DAP16 parses all blocks in the database and updates the dap-req-crs-dtl, dap-req-link-dtl and dap-result-dtl tables.

Access to run DAP16 in Transit is granted with the TRDAP16 Shepherd key.

DAP16 processes all blocks (dap_req_block) found in the database. Each block is reparsed and the dap-req-crs-dtl, dap-req-link-dtl and dap-result-dtl tables are updated. In addition, the syntax and remarks trees in the daptrees directory are replaced. If any errors were found during the parsing phase, the parse status is updated with "NO"; if no errors were found the status value is set to "OK".

DAP16 can also be run from the host prompt using the “dap16all” script if needed. In addition, a subset of blocks can be run by setting the DAP36_CRITERIA variable to choose only the blocks you want. You may also override the default order-by clause, for example:

```
$ export DAP36_CRITERIA="dap_block_type = 'MAJOR' "
$ export DAP36_ORDER_BY="dap_block_title"
$ dap16all
```

The default order-by used by DAP16 is: dap_block_type, dap_block_value, dap_cat_yr_start, dap_cat_yr_stop.

For each block that DAP16 parses a line will appear in the log file showing the block type, value, cat-yrs, req-id and parse status. For example:

Type	Value	Catalog	Years	ID	Status
COLLEGE	SCI	19941995	:99999999	RA001267	Parsed
okay					
COLLEGE	SCI	20042005	:99999999	RA001269	Parsed
okay					
COLLEGE	SCI	20102011	:99999999	RA001268	Parsed
okay					
CONC	ACCT	19941995	:19951996	RA000600	Parsed
with errors	ACCT	19961997	:19992000	RA000203	Parsed
CONC	ACCT	20002001	:20022003	RA000506	Parsed
okay					

When DAP16 is done, a summary line will be shown in the log file:

```
Blocks that Parsed Without Errors: 794
Blocks that Parsed With Errors: 134
Total Requirement Blocks Parsed: 928
```

When DAP16 is finished, the dap16action script will run and create a list of all the blocks in the database that have a NO parse status. In addition, the same summary line shown in the log file will appear in the action file. Both the action file and log file can be viewed in Transit.

To view and fix the parse errors Scribe should be used. In Scribe you should search on blocks with parse errors and fix each block one by one.

Review and configure the following settings:

- parser.batch.notifications.toEmail
- transit.dap16.workerCount

DAP21 - Extract Articulated Transfers

Updated: March 25, 2022

The DAP21 batch program is the processor used to create an XML file of articulated transfer data from DAPDB.

Access to run DAP21 in Transit is granted with the TRDAP21 Shepherd key.

DAP21 is intended for use by sites using Degree Works Transfer Equivalency and wishing to FTP the articulated transfer data back to the student system. DAP21 is launched from Transit after selecting the students to be processed.

Selection criteria from the dap_applcnt_mst, dap_appdata_dtl, dap_college_dtl and dap_transfer_dtl must be used to define the desired group of students. If no selection criteria are used then only the TreqStatus of "AR" and "RO" (If the Re-roll flag is "Y") will be used to screen students. Run-time questions for DAP21 will be:

Name of output file?

The XML file name will be no longer than 8 bytes long. The file will be output to the \$ADMIN_HOME/data directory. If the file name is not supplied then it will default to dap21nnnn.xml, where nnnn is the last 4 digits of the Process Identification Number (PID). For example, if the PID is "27865" for DAP21 the output file name would be dap217865.xml.

Re-roll students who were already rolled?

Y = Selected students with a dap_applcnt_mst treq_status = "RO" or "AR" will be included.

N = Only selected students with a dap_applcnt_mst treq_status = "AR" will be included.

If left blank then "N" is used as the default Re-roll Student parameter.

DAP21 will extract the data for the selected students, screened by the Re-roll Student run-time response, to an XML flat file. DAP21 will also change the dap_applcnt_mst treq_status from "AR" to "RO", set the dap_roll_date to the current system date, set the dap_roll_time to the current system time and set the dap_what to "TREQEXPT". DAP21 will not produce a printed report but will display a tally of the number of records processed in the dap21nnnnA.act (summary) and dap21nnnnL.log (detail) files.

Email message capability

Updated: March 25, 2022

The DAP21 job may also send email notification that includes the contents of the Action File if the "DAP21_EMAIL_ADDRESS" environment variable has been set in dwenv.config file for the appropriate staff member(s).

For example:

```
export DAP21_EMAIL_ADDRESS=some.staff@sungardhe.com
```

When DAP21 finishes an email message is sent to the appropriate individuals if this configuration has been set appropriately.

DAP21 errors, warnings, and success

Updated: March 25, 2022

DAP21 does not validate data so it does not give errors about specific records.

However, it does filter records based on the Treq Roll Status from the dap_applicnt_mst. Edits:

- If the Parameter Re-roll Student flag = 'N' and the dap_treq_status = 'RO' (already rolled) the student will be Skipped and NO records will be written to the output file.
- If the Parameter Re-roll Student flag = 'Y' and the dap_treq_status is NOT = 'RO' (already rolled) and NOT = 'AR' (Articulation Resolved) the student will be Skipped and NO records will be written to the output file.

Warnings

Updated: March 25, 2022

DAP21 does not emit very many warnings.

The ID code of the students skipped along with one of the following two messages will be displayed on the DAP21 execution log for the student ID (dap21nnnnL.log file in the \$ADMIN_HOME/dgwspool directory):

Skipping Student ID [N00010961] because TreQ data has already been rolled!
Skipping Student ID [N00010961] because the TreqStatus shows the data has not been articulated!

Fatal errors

Updated: March 25, 2022

DAP21 will fail if the output file cannot be created, if the Parameter record cannot be found or read successfully, if the DAP database cannot be opened, or if one of the Transfer Equivalency tables being used (rad_primary_mst, dap_applicnt_mst, dap_appdata_dtl, dap_college_dtl or dap_transfer_dtl) cannot be read.

Success

Updated: March 25, 2022

DAP21 writes the number of records considered, output and skipped to the dap21nnnnA.act Action File.

```
===== DAP21 ACTION FILE =====
=====
```

```
STUDENTS CONSIDERED: 3
```

```
STUDENTS OUTPUT: 2
```

```
STUDENTS SKIPPED: 1
```

```
NUMBER OF LINES WRITTEN TO XML FILE: 485
```

```
Output XML File = /home/dpayne/WorkSpaces/main/admin/data/DRP21OUT.xml
```

```
DAP21JOB Started Wed Jun 15 16:40:52 EDT 2011
DAP21JOB Ended   Wed Jun 15 16:41:00 EDT 2011
```

```
Time elapsed to process students:          0 hours, 0 minutes, 12 s
econds.
```

Note that the XML output file will be created in the \$ADMIN_HOME/data directory.

DAP21 writes detailed information from the "ArticulationExport" script to the dap21nnnnnL.log Log File:

```
DAP21JOB Started Wed Jun 15 16:32:44 EDT 2011
Working directory = /home/dpayne/WorkSpaces/main/admin/logdebug/dap2
1drp
DegreeWorks Release = DW4.0.8
-rw-rw---- 1 dpayne users 405 Jun 15 16:32 DAP21M01
ArticulationExport === Input Parameters ===
DAP21A      User, Re12345678           280950021 DAP
```

```
DAP21P0001DRP21OUT          N
DAP21S00012470 =8080      Student ID        001
          01
DAP21S00022470 =SRI       Student ID        001
          01
DAP21S00032470 =N00010961 Student ID        001
          01

ArticulationExport === End Parameters ===
ArticulationExport === Running utl39 to select ID Codes ===
Runing UTL39 to select IDS to an ID file
The UTL39_IDFILE variable is not set - must run utl39
ArticulationExport === Begin UTL39 === Wed Jun 15 16:32:44 EDT 201
1

UTL39: 3 students were selected
ArticulationExport === End UTL39 === Wed Jun 15 16:32:44 EDT 201
1
```

```
-rwxrwx--- 1 dpayne users 33 Jun 15 16:32 utl68.ids
Our dap21.students file of IDs:
-rwxrwx--- 1 dpayne users 33 Jun 15 16:32 dap21.students
ArticulationExport === Setting java_classpath ===
ArticulationExport === Begin Articulation Export === Wed Jun 15 16
:32:44 EDT
2011

DAP21 Parameters = /home/dpayne/WorkSpaces/main/admin/logdebug/dap2
1drp/DAP21M01

ArticulationExport === ExportExecutor Parameters ===
-----
Input ID File =
/home/dpayne/WorkSpaces/main/admin/logdebug/dap21drp/dap21.students
Output XML File = /home/dpayne/WorkSpaces/main/admin/data/DRP21OUT
.xml
Reroll Student = N
-----

ArticulationExport === Running ExportExecutor to create XML Output
File ===
JAVA:06/15/11 16:32:45 Version and build number of this jar package
is 4.0.MAIN-SNAPSHOT.1206
JAVA:06/15/11 16:32:45 Initializing jndi contex
JAVA:06/15/11 16:32:45 Found datasources list
JAVA:06/15/11 16:32:45 Creating a datasource for dw, jdbc.properties
should cont
ain values for dw.jdbc.driverClassName, dw.jdbc.url, dw.jdbc.username,
dw.jdbc.password and dw.jdbc.jndiname
JAVA:06/15/11 16:32:45 Creating a datasource for ops, jdbc.propertie
s should contain values for ops.jdbc.driverClassName, ops.jdbc.url,
ops.jdbc.username, ops.jdbc.password and ops.jdbc.jndiname
JAVA:06/15/11 16:32:45 Initializing jndi contex
JAVA:06/15/11 16:32:45 Found datasources list
JAVA:06/15/11 16:32:45 Creating a datasource for dw, jdbc.properties
should contain values for dw.jdbc.driverClassName, dw.jdbc.url, dw.
jdbc.username, dw.jdbc.password and dw.jdbc.jndiname
JAVA:06/15/11 16:32:45 Creating a datasource for ops, jdbc.propertie
s should contain values for ops.jdbc.driverClassName, ops.jdbc.url,
ops.jdbc.username, ops.jdbc.password and ops.jdbc.jndiname
JAVA:06/15/11 16:32:45 Initializing Spring
JAVA:06/15/11 16:32:54 Finished initializing Spring
JAVA:06/15/11 16:32:54 STUDENTS CONSIDERED: 3
JAVA:06/16/11 14:42:55 ***** <MESSAGES> ****
*****
JAVA:06/16/11 14:43:07 Exporting Student ID [8080]
JAVA:06/15/11 16:32:55 Exporting Student ID [N00010961]
JAVA:06/15/11 16:32:56 Skipping Student ID [SRI] because TreQ data h
as already been rolled!
JAVA:06/16/11 14:42:55 ***** <END MESSAGES> ****
*****
```

```
JAVA:06/15/11 16:32:56 0 hours 0 minutes 11 seconds
ArticulationExport === Completed ExportExecutor for the selected students ===
ArticulationExport === XML Output File: ===
-rw-rw---- 1 dpayne users 270 Jun 15 16:32 /home/dpayne/WorkSpaces/
main/admin/data/DRP21OUT.xml
ArticulationExport === Total Counts: ===
STUDENTS OUTPUT: 2
NUMBER OF LINES WRITTEN TO XML FILE: 485
ArticulationExport === End ExportExecutor === Wed Jun 15 16:32:56
EDT 2011
ArticulationExport === Cleanup ===
Time elapsed to process students: 0 hours, 0 minutes, 12 seconds.
DAP21JOB Ended Wed Jun 15 16:32:56 EDT 2011
```

The XML file produced by DAP21 will contain 5 types of records defined in the AcademicRecord_v1.5.0.xsd PESC schema – “custom” Transfer Equivalency elements have been added for the Applicant Master data, Applicant Detail data, School data and Course data where no corresponding element exists:

<Student> <Person>	ID code and name from the rad_primary_mst
<Student> <ApplicantMasterData>	data from the dap_applicant_mst
<Student> <ApplicantDetailData>	major/minor/concentration data from the dap_appdata_dtl
<Student> <AcademicRecord> <School>	transfer school data from the dap_college_dtl
<Student> <AcademicRecord> <Course>	transfer course data and tests from the dap_transfer_dtl

All records from each table for each student selected from DAPDB/RADDB will be included in the XML output file. It is up to each site to determine which elements out of the generated XML file are to be imported into the local student system.

DAP21 will not generate any errors. No data validation will occur. It is assumed that the data in the database has already been validated. The output file will be named dap21nnnn.xml, where nnnn is the last 4 digits of the job number, unless a file name was supplied in response to run-time question. The XML output file will be written to the \$ADMIN_HOME/data directory.

Sample output XML data file

Updated: March 25, 2022

DAP21 uses the AcademicRecord_v1.5.0.xsd PESC schema for its output XML definitions.

The “ArticulationExportEnvelope” has been created to export all selected students and include them in the same output XML file. Many custom elements have been added for the Degree Works Transfer Equivalency database tables being exported. Most of the items defined below are custom except where they are noted as standard.

```

<xs:complexType name="StudentType">
  <xs:annotation>
    <xs:documentation>Student Type</xs:documentation>
  </xs:annotation>
  <xs:sequence>
    <xs:element name="Person" type="AcRec:PersonType">
      <xs:element name="ApplicantMasterData" type="AcRec:ApplicantMasterData">
        <xs:element name="ApplicantDetailData" type="AcRec:ApplicantDetailData"
          maxOccurs="unbounded">
          <xs:element name="AcademicRecord" type="AcRec:AcademicRecordType"
            maxOccurs="unbounded">
            </xs:sequence>
        </xs:element>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

```

These are the first three lines of the XML output file:

```

<!--Sungardhe Treq Articulation Export--><SungardheTreqArticulationExport
  xmlns:ns3="urn:org:pesc:core:CoreMain:v1.8.0"
  xmlns="urn:com:sungardhe:degreeworks:treq:ArticulationExportEnvelope
  :v1.0.0">

```

These are the nodes used for each student:

```

<ns3_:CollegeTranscript xmlns:ns3_="urn:org:pesc:message:CollegeTranscript:v1.2.0" xmlns="">
  <Student>
    <Person>
      <ApplicantMasterData> </ApplicantMasterData>
      <ApplicantDetailData> </ApplicantDetailData>
      <AcademicRecord>
        <School> </School>
        <AcademicSummary> </AcademicSummary>
        <Course> </Course>
      </AcademicRecord>
    </Student>
  </ns3_:CollegeTranscript>

```

This is the last line of the XML output file:

```
</SungardheTreqArticulationExport>
```

Person XML

Selected fields from the rad_primary_mst will be written to the XML output file, one record per each data element in an XML format as described below.

These standard elements are from the AcademicRecord_v.1.5.0.xsd “Person” node.

XML Tag	Len	DAPDB	Comments
<SchoolAssignedPersonID>	10	dap_stu_id	This element is the universal ID code assigned to a student at the time of admission. This ID must be valid in the rad_primary_mst.
<FirstName>	180	rad_name (first name after the comma stripped out from the rad_name)	Student's first name.
<MiddleName>	180	rad_name (middle name after the comma and a blank stripped out from the rad_name)	Student's middle name.
<LastName>	180	rad_name (last name before the comma stripped out from the rad_name)	Student's last name.
<CompositeName>	180	rad_name	The student's name in the format: last, first middle.

Applicant Master XML

Selected fields from the dap_applcnt_mst will be written to the XML output file, one record per each data element in an XML format as described below.

These elements are from the AcademicRecord_v.1.5.0.xsd “Applicant Master Data” node.

```

<!----->
<!--Custom ApplicantMasterData-->
<!----->
<xss:complexType name="ApplicantMasterData">
    <xss:annotation>
        <xss:documentation>Applicant Master Data from the dap_applcnt_mst<xss:documentation>
    <xss:annotation>
    <xss:sequence>

        <!-- Custom DegreeWorks Elements from the DAP_APPLICN
T_MST -->

```

All elements from this table are “custom” for Ellucian.

Tag Name	Len	DAPDB	Comments
<ApplicantSchool>	12	dap_school (UCX-STU350)	This element defines the school (campus) to which the student has applied (for example, "UG" for Undergraduate School, "GR" for Graduate School, "LW" for Law School).
<ApplicantDegreeInterest>	2	dap_deg_interest (UCX_STU564)	This element is a code to define the type of degree interest associated with this applicant (for example, "DS" for Degree Seeking, "ND" for Non-Degree Seeking, etc.).
<ApplicantDegree>	12	dap_degree (UCX-STU307)	This element contains the code that identifies the student's intended degree (for example, "BS" for Bachelor of Science, "MA" for Master of Arts, "BFA" for Bachelor of Fine Arts).
<ApplicantCatalogYear>	12	dap_catalog_yr (UCX-STU035)	This element defines the catalog year in effect for the student's degree program. The catalog year determines which set of degree requirement definitions should be used when evaluating the student's progress towards completing the degree.
<ApplicantTreqStatus>	2	dap_treq_status (UCX-TRQ062)	The Transfer Equivalency articulation status, should be "AR" for Articulated or "RO" for Rolled Previously.
<ApplicantArticulationDate>	8	dap_artic_date	The Transfer Equivalency articulation date,

Tag Name	Len	DAPDB	Comments
			formatted CCYYMMDD
<ApplicantArticulationTime>	6	dap_artic_time	The Transfer Equivalency articulation time, formatted HHMMSS
<Condition1>	12	dap_condition1	This element stores additional articulation condition as defined by your site.
<Condition2>	12	dap_condition2	This element stores additional articulation condition as defined by your site.
<Condition3>	12	dap_condition3	This element stores additional articulation condition as defined by your site.
<Condition4>	12	dap_condition4	This element stores additional articulation condition as defined by your site.
<Condition5>	12	dap_condition5	This element stores additional articulation condition as defined by your site.
<Condition6>	12	dap_condition6	This element stores additional articulation condition as defined by your site.
<Condition7>	12	dap_condition7	This element stores additional articulation condition as defined by your site.
<Condition8>	12	dap_condition8	This element stores additional articulation condition as defined by your site.
<Condition9>	12	dap_condition9	This element stores additional articulation condition as defined by your site.
<Condition10>	12	dap_condition10	This element stores additional articulation condition as defined by your site.

Applicant Detail XML

Selected fields from the dap_appdata_dtl will be written to the XML output file, one record per each data element in an XML format as described below. This is a repeating table. A student may have several dap_appdata_dtl records, one for each Goal Code defined for a student.

These elements are from the AcademicRecord_v.1.5.0.xsd “Applicant Detail” node.

```
<!-- Custom ApplicantDetailData-->
<!-- Custom DegreeWorks Elements from the DAP_APPDATA
_DTL -->
```

All elements from this table are “custom” for Ellucian.

XML Tag	Len	DAPDB	Comments
<GoalCode>	12	dap_goal_code	Must be one of the valid codes: COLLEGE CONC MAJOR MINOR PROGRAM
<GoalCodeDescription>	30		Description of the above valid code used.
<GoalValue>	12	dap_goal_value	If the Goal Code = COLLEGE, the goal value will contain a college code from UCX_STU560. If the Goal Code = CONC, the goal value will contain a concentration from UCX_STU563.

XML Tag	Len	DAPDB	Comments
			If the Goal Code = MAJOR, the goal value will contain a major from UCX_STU023.
			If the Goal Code = MINOR, the goal value will contain a minor from UCX_STU024.
			If the Goal Code = PROGRAM, the goal value will contain a program code from UCX_STU316.
<GoalValueDescription>	30	UCX_STU???	The UCX_VALUE description for each Goal Value:
			UCX_STU560: COLLEGE
			UCX_STU563: CONC
			UCX_STU023: MAJOR
			UCX_STU024: MINOR
			UCX_STU316: PROGRAM

Academic Record - School Type XML

Selected fields from the dap_college_dtl will be written to the XML output file, one record per each data element in an XML format as described below. This is a repeating table. A student may have several dap_college_dtl records, one for each transfer school attended.

These elements are from the AcademicRecord_v.1.5.0.xsd “SchoolType” node.

```

<! ----->
<! --AcademicSession Types-->
<! ----->
<xs:complexType name="SchoolType">
  <xs:sequence>
    <xs:element name="OrganizationName">
      <xs:group ref="core:OrganizationIDGroup">

```

```

<xss:element name="LocalOrganizationID">
<xss:element name="SchoolOverrideCode">
<xss:element name="SchoolLevel">
<xss:element name="Contacts">
<xss:element name="NoteMessage">

<!-- Custom DegreeWorks Elements from the DAP_COLLEGE
    _DTL -->

```

All of the elements from this table are “custom” for Ellucian except for the LocalOrganizationIDCode which is part of the standard schema.

Field Name	Len	DAPDB	Comments
<LocalOrganizationIDCode>	10	dap_school_id	This element identifies the transfer institution where this transfer class was taken. It is validated against the rad_ets_mst.
<TransferDegree>	12	dap_tr_degree (UCX-STU379)	This element contains the code that identifies the student's previous degree (for example, "BA" for Bachelor of Arts, "AA" for Associate of Arts Degree, "MS" for Master of Science, etc).
<TransferConferFlag>	2	dap_confer_flag	This element is a Y/N flag indicating if this institution conferred a degree.
<TransferMajor>	12	dap_tr_major (UCX-STU382)	This element is used to store the major associated with this degree (for example, "ART" for Art Major, "BUS" for Business Major, "ENGL" for English Major").
<TransferStartDate>	8	dap_tr_start	This element is the date the student began attending this transfer institution. Formatted CCYY-MM-DD.
<TransferStopDate>	8	dap_tr_stop	This element is the date the student stopped attending this

Field Name	Len	DAPDB	Comments
			transfer institution. Formatted CCYY-MM-DD.
<TransferTranscriptDate>	8	dap_trnscpt_date	This element is the date of the transfer transcript. Formatted CCYY-MM-DD.
<Condition1>	12	dap_condition1	This element stores additional articulation condition as defined by your site.
<Condition2>	12	dap_condition2	This element stores additional articulation condition as defined by your site.
<Condition3>	12	dap_condition3	This element stores additional articulation condition as defined by your site.
<Condition4>	12	dap_condition4	This element stores additional articulation condition as defined by your site.
<Condition5>	12	dap_condition5	This element stores additional articulation condition as defined by your site.
<Condition6>	12	dap_condition6	This element stores additional articulation condition as defined by your site.
<Condition7>	12	dap_condition7	This element stores additional articulation condition as defined by your site.
<Condition8>	12	dap_condition8	This element stores additional articulation condition as defined by your site.
<Condition9>	12	dap_condition9	This element stores additional articulation condition as defined by your site.
<Condition10>	12	dap_condition10	This element stores additional articulation

Field Name	Len	DAPDB	Comments
			condition as defined by your site.

Academic Record - Course Type XML

Selected fields from the dap_transfer_dtl will be written to the XML output file, one record per each data element in an XML format as described below. This is a repeating table. A student may have several dap_transfer_dtl records, one for each transfer class taken.

These elements are from the AcademicRecord_v.1.5.0.xsd “Course Type” node. The “standard” Transfer Course items that are extracted by DAP21 are as follows:

```

<xs:complexType name="CourseType">
    <xs:sequence>
        <xs:element name="CourseCreditUnits" />
        <xs:element name="CourseCreditEarned" />
        <xs:element name="CourseAcademicGrade" />
        <xs:element name="CourseRepeatCode" />
        <xs:element name="CourseQualityPointsEarned" />
        <xs:element name="CourseLevel" />
        <xs:element name="CourseSubjectAbbreviation" />
        <xs:element name="CourseNumber" />
        <xs:element name="CourseSectionNumber" />
        <xs:element name="CourseTitle" />
        <xs:element name="CourseBeginDate" />
        <xs:element name="CourseEndDate" />

        <!-- Custom DegreeWorks Elements from the DAP_TRANSFER_DTL -->

```

Field Name	Len	DAPDB	Comments
<CourseSchool>	12	dap_school (UCX-STU350)	This element is the school code within which the transfer class is to be associated (for example, "UG" for Undergraduate School, "LW" for Law School, "GR" for Graduate School).
<CourseDegreeInterest>	2	dap_deg_interest (UCX_STU564)	This element is a code to define the type of degree interest associated with this transfer class (for example, "DS" for Degree Seeking, "ND" for Non-Degree Seeking, etc.)..

Field Name	Len	DAPDB	Comments
<LocalOrganizationIDCode> (standard)	10	dap_school_id	This element identifies the transfer institution where this transfer class was taken. It is validated against the rad_ets_mst.
<TransferSubjectAbbreviation>	12	dap_tr_disc	This element is the discipline code as it was identified at the transfer institution.
<TransferCourseNumber>	12	dap_tr_crse_num	This element is the course number as it was identified at the transfer institution.
<TransferCourseTitle>	30	dap_tr_title	This element stores the title of the course as it was identified at the transfer institution.
<TransferCourseCredits>	7	dap_tr_credits	This element is the number of credits taken at the transfer institution. It is a decimal field in the database, but will only include a decimal if necessary. For example, a >3< will be written for 3.0 credit class while a >2.5< will be written for a 2.5 credit class.
<CourseBeginDate> (standard)	8	dap_tr_start	This element is the start date for a particular transfer class. Format CCYY-MM-DD.
<CourseEndDate> (standard)	8	dap_tr_stop	This element is the stop date for a particular transfer class. Format CCYY-MM-DD.
<CourseCreditUnits> (standard)	2	dap_tr_cr_method (UCX-STU346)	This element defines the calendar structure for the transfer institution's academic year (for example, "S"

Field Name	Len	DAPDB	Comments
			= Semester, "Q" = Quarter).
<TransferAcademicGrade>	6	dap_tr_grade	Grade from the transfer institution. (UCX-STU398)
<TransferQualityPointsEarned>	7	dap_tr_grade_pts	This element is the number of grade points awarded at the transfer institution. It is a decimal field in the database, but will only include a decimal if necessary. For example, a >12< will be written for a class with 12.0 quality points while a >10.5< will be written for a class with 10.5 quality points.
<TransferArticulation>	4	dap_articulation (UCX-TRQ065)	The articulation code indicating one-to-one, one-to-many, many-to-one, and many-to-many. Assigned by Transfer Equivalency (DAP12).
<CourseSubjectAbbreviation> (standard)	12	dap_discipline	The discipline code of the resident course articulated by Transfer Equivalency.
<CourseNumber> (standard)	12	dap_course_num	The course number of the resident course articulated by Transfer Equivalency.
<CourseSection> (standard)	2	dap_section	The course section of the resident course articulated by Transfer Equivalency.
<CourseTitle> (standard)	30	dap_course_title	The course title of the resident course articulated by Transfer Equivalency.

Field Name	Len	DAPDB	Comments
<CourseDepartment>	12	dap_dept (UCX-STU362)	The academic department code of the resident course articulated by Transfer Equivalency.
<CourseDivision>	12	dap_division (UCX-STU351)	The academic division code of the resident course articulated by Transfer Equivalency.
<CourseCreditEarned> (standard)	7	dap_cr_earn	The credits earned at your institution as articulated by Transfer Equivalency. It is a decimal field in the database, but will only include a decimal if necessary. For example, a >3< will be written for a class worth 3.0 credits while a >2.5< will be written for a class worth 2.5 credits.
<CourseCreditType>	2	dap_credit_type (UCX-STU355)	The type of credit, for example, academic, granted for the course articulated by Transfer Equivalency.
<CourseGradeType>	2	dap_grade_type (UCX-STU356)	The type of grade, for example, A-F or Pass-Fail, granted for the course articulated by Transfer Equivalency.
<CourseAcademicGrade> (standard)	6	dap_final_grade (UCX-STU385)	The grade granted for the course articulated by Transfer Equivalency.
<CourseQualityPointsEarned> (standard)	7	dap_grade_points	The grade points granted for the course articulated by Transfer Equivalency. It must be numeric. Calculated by Transfer Equivalency based on FINAL-GRADE and CR-EARN.

Field Name	Len	DAPDB	Comments
<CourseClassStatus>	2	dap_class_status (UCX-STU380)	The class status code for the course articulated by Transfer Equivalency, "A" = Add, "RP" = first instance of repeated course, "RT" = subsequent instances of repeated course.
<CourseRepeatSubjectAbbreviation>	12	dap_repeat_disc	The course discipline of the resident course repeated by the resident course articulated by Transfer Equivalency. For example, if MATH 115 is articulated but it is a repeat of PHIL 111 then repeat_disc would be PHIL. This field is not calculated by Transfer Equivalency and will only be filled in if it was passed from the student system by some external process.
<CourseRepeatNumber>	12	dap_repeat_num	The course number of the resident course repeated by the resident course articulated by Transfer Equivalency. For example, if MATH 115 is articulated but it is a repeat of PHIL 111 then repeat_num would be 111. This field is not calculated by Transfer Equivalency and will only be filled in if it was passed from the student system by some external process.
<CourseRepeatPolicy>	2	dap_repeat_plcy	The repeat policy governing how Degree

Field Name	Len	DAPDB	Comments
		(UCX-AUD047)	Works will apply the repeated class. Valid repeat policies are documented in UCX-AUD047. This field is not calculated by Transfer Equivalency and will only be filled in if it was passed from the student system by some external process.
<CourseAcadVotech>	6	dap_acad_votech (UCX-XXX376)	The academic-vocational code of the resident course articulated by Transfer Equivalency.
<CourseClassType>	2	dap_class_type (UCX-XXX377)	The class type code of the resident course articulated by Transfer Equivalency.
<TransferArticulationDate>	8	dap_artic_date	The Transfer Equivalency articulation date, formatted CCYYMMDD
<TransferArticulationStatus>	2	dap_artic_status	The Transfer Equivalency articulation status.
<Authorizer>	30	dap_authorizer	The name of the person who authorized the articulation of this transfer course.
<UserDefined1>	12	dap_user_def1	This element stores additional data as defined by your site.
<UserDefined2>	12	dap_user_def2	This element stores additional data as defined by your site.
<UserDefined3>	12	dap_user_def3	This element stores additional data as defined by your site.
<UserDefined4>	12	dap_user_def4	This element stores additional data as defined by your site.

Field Name	Len	DAPDB	Comments
<UserDefined5>	12	dap_user_def5	This element stores additional data as defined by your site.
<UserDefined6>	12	dap_user_def6	This element stores additional data as defined by your site.
<UserDefined7>	12	dap_user_def7	This element stores additional data as defined by your site.
<UserDefined8>	12	dap_user_def8	This element stores additional data as defined by your site.
<UserDefined9>	12	dap_user_def9	This element stores additional data as defined by your site.
<UserDefined10>	12	dap_user_def10	This element stores additional data as defined by your site.
<Condition1>	12	dap_condition1	This element stores additional data as defined by your site that can be used by Transfer Equivalency as a condition to be evaluated during transfer articulation.
<Condition2>	12	dap_condition2	This element stores additional data as defined by your site that can be used by Transfer Equivalency as a condition to be evaluated during transfer articulation.
<Condition3>	12	dap_condition3	This element stores additional data as defined by your site that can be used by Transfer Equivalency as a condition to be evaluated during transfer articulation.
<Condition4>	12	dap_condition4	This element stores additional data as defined by your site that can be used by Transfer Equivalency

Field Name	Len	DAPDB	Comments
			as a condition to be evaluated during transfer articulation.
<CourseMapId>	8	dap_map_id	The identifier of the mapping rule used during Transfer Equivalency articulation. Filled in by Transfer Equivalency.
<CourseStudentSequence>	4	dap_tr_stu_seq	Sequence number to uniquely identify this transfer course for this student. It is an integer field in the database. For example, if the sequence number is 4 a >4< will be written to the XML file. There will be no leading zeroes.
<CourseTerm>	12	trm_sort (UCX-STU016)	This element stores the term associated with the transfer class (for example, "201120" for Spring 2011) .

Example output

The data file output by DAP21 is an XML file. The data file contains records described above and resides in the \$ADMIN_HOME/data sub-directory of the test or production directory. The output XML data file contains the Transfer Equivalency Articulation data for all students selected.

DAP21 sorts data according to ID and School-ID.

Sample records in the DAP21 output XML file for one student:

```

<?xml version="1.0" encoding="UTF-8"?>
<!--Sungardhe Treq Articulation Export--><SungardheTreqArticulationEx
port xmlns:ns3="urn:org:pesc:core:CoreMain:v1.8.0" xmlns="urn:com:su
ngardhe:degreeworks:treq:ArticulationExportEnvelope:v1.0.0">
    <ns3_:CollegeTranscript xmlns:ns3_="urn:org:pesc:message:CollegeTr
anscript:v1.2.0" xmlns="">
        <Student>
            <Person>
                <SchoolAssignedPersonID>N00010961<SchoolAssignedPersonID>
                <Name>
                    <FirstName>Kathleen<FirstName>
                    <LastName>Weber<LastName>
                    <CompositeName>Weber, Kathleen<LastName>
                </Name>
            </Person>
        </Student>
    </CollegeTranscript>
</SungardheTreqArticulationExport>

```

```

<Person>
<ApplicantMasterData>
    <ApplicantSchool>U<ApplicantSchool>
    <ApplicantDegree>BA<ApplicantDegree>
    <ApplicantCatalogYear>1978<ApplicantCatalogYear>
    <ApplicantTreqStatus>R0<ApplicantTreqStatus>
    <ApplicantArticulationDate>2011-01-19<ApplicantArticulationD
ate>
    <ApplicantArticulationTime>131020<ApplicantArticulationTime>
    <Condition1>testcond1<Condition1>
    <Condition2>testcond2<Condition2>
    <Condition3>testcond3<Condition3>
    <Condition4>testcond4<Condition4>
    <Condition5>testcond5<Condition5>
    <Condition6>testcond6<Condition6>
    <Condition7>testcond7<Condition7>
    <Condition8>testcond8<Condition8>
    <Condition9>testcond9<Condition9>
    <Condition10>testcond10<Condition10>
<ApplicantMasterData>
<ApplicantDetailData>
    <GoalCode>MAJOR      <GoalCode>
    <GoalCodeDescription>Major<GoalCodeDescription>
    <GoalValue>ANSC      <GoalValue>
    <GoalValueDescription>Animal Science<GoalValueDescription>
<ApplicantDetailData>
<ApplicantDetailData>
    <GoalCode>MINOR<GoalCode>
    <GoalCodeDescription>Minor<GoalCodeDescription>
    <GoalValue>ART-MINOR  <GoalValue>
    <GoalValueDescription>Art Minor<GoalValueDescription>
<ApplicantDetailData>
<ApplicantDetailData>
    <GoalCode>COLLEGE<GoalCode>
    <GoalCodeDescription>College<GoalCodeDescription>
    <GoalValue>00<GoalValue>
    <GoalValueDescription>No College Designated<GoalValueDescrip
tion>
    <ApplicantDetailData>
    <ApplicantDetailData>
        <GoalCode>MINOR<GoalCode>
        <GoalCodeDescription>Minor<GoalCodeDescription>
        <GoalValue>ARTHIST-MN<GoalValue>
        <GoalValueDescription>Art History Minor<GoalValueDescription
>
    <ApplicantDetailData>
    <AcademicRecord>
        <School>
            <OrganizationName>Generic Community College<OrganizationNa
me>
            <LocalOrganizationID>
                <LocalOrganizationIDCode>1111<LocalOrganizationIDCode>

```

```
<LocalOrganizationID>
<TransferDegree>AA<TransferDegree>
<TransferConferFlag>N <TransferConferFlag>
<TransferMajor>MATH<TransferMajor>
<TransferStartDate>2008-09-01<TransferStartDate>
<TransferStopDate>2008-12-15<TransferStopDate>
<TransferTranscriptDate>2008-12-20<TransferTranscriptDate>
<Condition1>testcond1<Condition1>
<Condition2>testcond2<Condition2>
<Condition3>testcond3<Condition3>
<Condition4>testcond4<Condition4>
<Condition5>testcond5<Condition5>
<Condition6>testcond6<Condition6>
<Condition7>testcond7<Condition7>
<Condition8>testcond8<Condition8>
<Condition9>testcond9<Condition9>
<Condition10>testcond10<Condition10>
<School>
<AcademicSummary>
  <AcademicSummaryType>TransferOnly<AcademicSummaryType>
<AcademicSummary>
<Course>
  <CourseCreditValue>3<CourseCreditValue>
  <CourseCreditEarned>3<CourseCreditEarned>
  <CourseAcademicGrade>A<CourseAcademicGrade>
  <CourseQualityPointsEarned>12<CourseQualityPointsEarned>
  <CourseSubjectAbbreviation>ENGELECT<CourseSubjectAbbreviat
ion>
  <CourseNumber>150<CourseNumber>
  <CourseSectionNumber>X<CourseSectionNumber>
  <CourseTitle>19th Century American Lit<CourseTitle>
  <CourseSchool>U<CourseSchool>
  <CourseDepartment>ENGL<CourseDepartment>
  <CourseDivision>AS<CourseDivision>
  <CourseCreditType>TR<CourseCreditType>
  <CourseGradeType>AF<CourseGradeType>
  <CourseRepeatSubjectAbbreviation>ENGL<CourseRepeatSubjectA
bbreviation>
  <CourseRepeatCourseNumber>144<CourseRepeatCourseNumber>
  <CourseRepeatPolicy>1 <CourseRepeatPolicy>
  <CourseAcadVotech>AC<CourseAcadVotech>
  <CourseClassType>CL<CourseClassType>
  <TransferSubjectAbbreviation>ENGL<TransferSubjectAbbreviat
ion>
  <TransferCourseNumber>130<TransferCourseNumber>
  <TransferCourseTitle>American Literature of the 19<Transfe
rCourseTitle>
  <TransferAcademicGrade>A-<TransferAcademicGrade>
  <TransferQualityPointsEarned>10.5<TransferQualityPointsEar
ned>
  <TransferArticulation>1T01<TransferArticulation>
  <TransferArticulationDate>2011-05-15<TransferArticulationD
```

```
ate>
    <Authorizer>DRP</Authorizer>
    <UserDefined1>testudef1</UserDefined1>
    <UserDefined2>testudef2</UserDefined2>
    <UserDefined3>testudef3</UserDefined3>
    <UserDefined4>testudef4</UserDefined4>
    <UserDefined5>testudef5</UserDefined5>
    <UserDefined6>testudef6</UserDefined6>
    <UserDefined7>testudef7</UserDefined7>
    <UserDefined8>testudef8</UserDefined8>
    <UserDefined9>testudef9</UserDefined9>
    <UserDefined10>testudef10</UserDefined10>
    <Condition1>testcond1</Condition1>
    <Condition2>testcond2</Condition2>
    <Condition3>testcond3</Condition3>
    <Condition4>testcond4</Condition4>
    <CourseMapId>MA147444</CourseMapId>
    <CourseTerm>200820</CourseTerm>
    </Course>
    <Course>
        <CourseCreditUnits>S</CourseCreditUnits>
        <CourseCreditValue>3</CourseCreditValue>
        <CourseCreditEarned>3</CourseCreditEarned>
        <CourseAcademicGrade>B</CourseAcademicGrade>
        <CourseQualityPointsEarned>9</CourseQualityPointsEarned>
        <CourseSubjectAbbreviation>MATH</CourseSubjectAbbreviation>
        <CourseNumber>121</CourseNumber>
        <CourseTitle>Algebra I</CourseTitle>
        <CourseSchool>UG</CourseSchool>
        <CourseCreditType>TR</CourseCreditType>
        <CourseGradeType>AF</CourseGradeType>
        <TransferSubjectAbbreviation>MTH</TransferSubjectAbbreviation>
    on>
        <TransferCourseNumber>110</TransferCourseNumber>
        <TransferCourseTitle>College Algebra</TransferCourseTitle>
        <TransferAcademicGrade>B</TransferAcademicGrade>
        <TransferQualityPointsEarned>9</TransferQualityPointsEarned>
    >
        <TransferArticulation>1T01</TransferArticulation>
        <TransferArticulationDate>2011-01-19</TransferArticulationDate>
    ate>
        <CourseMapId>MA147382</CourseMapId>
        <CourseTerm>200820</CourseTerm>
        </Course>
        <AcademicRecord>
            <Student>
                <ns3_:CollegeTranscript>
            <SungardheTreqArticulationExport>
```

DAP22 - Generate Audits

Updated: March 24, 2023

DAP22 generates Degree Works audits in batch.

Access to run DAP22 in Transit is granted with the TRDAP22 Shepherd key.

Students accessing Degree Works from the web may not have the capability to run new audits. Instead, they view the most recent audit stored for them in the Degree Works database. For this reason, the Degree Works database must contain an audit for every active student. Registrar and Advisors typically can run new audits, but these audits are performed for one student at a time. The DAP22 processor will run audits in batch, which means it will produce an audit for every student, for every active student or for a group of students selected by some criteria. However, it is rare you will need to do this from Transit because batch audits are run after RAD11 completes, meaning you should have updated audits for your students as their data changes.

Typically when DAP22 is run, the option to create a new audit is selected. However, DAP22 can be run without generating a new audit just to generate PDF or XML output from the students' most recent audit by not selecting the **Create new audit** check box. Note that in this scenario, if the core.audit.process.runOnScribeChanges.enabled Shepherd setting is true, the auditor engine will check to see if any changes were made to the Scribe blocks in the student's audit after the last audit was generated. A new audit will be run if changes are detected.

DAP22 can be used to create PDF files of student audits in batch. When **Create PDF file** is selected as the job output, the user will be prompted for the page dimensions and the language properties locale to be used when generating the audit output. The page dimensions drop-down is populated with values from SYS100 while the language properties locale drop-down is populated with values from SHP080.

You can run simultaneous DAP22 processes to take advantage of the multiple processors you have on your machine. You may want to run a batch on seniors, another on juniors, another on sophomores, and yet another on first-year students. Running batch audit processes like this in parallel should result in better throughput if you have more than one CPU on your system.

Set the transit.dap22.workerCount setting to tell DAP22 how many processes should be used to run the audits. The DAP22 job launched from Transit and after RAD11 completes will run multiple DAP22 processes in parallel against the list of student IDs selected. Running multiple processes should result in better throughput, but will also consume more resources than running a single DAP22. If your workerCount is too high, the job may also run slower than if the workerCount was a smaller number. If you select the Create PDF file or raw XML output types in Transit, the DAP22 job will run only one dap22 process regardless of the workerCount value.

When running new audits, creating output, or building CPA results records, be sure you chose the correct audit type. Normally you will want this set to Academic Audit, but if you want to generate financial aid audits, you can set the audit type to Financial Aid Audit. You must specify an aid term when running a financial aid audit, however.

The following report formats will appear in the **Select audit report** drop-down list when you select Create PDF file as the output format for DAP22:

Warning! Be sure to select the report format that corresponds to the chosen audit type.

- RPT30 – Registrar Report
- RPT31 – Student View
- RPT33 – Graduation Checklist
- RPT36 – Registration Checklist
- RPT50 – Financial Aid Report
- RPT51 – Aid and Academic Report
- RPT55 – Athletic Eligibility Report
- RPT56 – Athletic and Academic Report

The flags in UCX-RPT036 for these reports determine how audit data will be displayed in that particular format. Depending on the report you chose, you can create either a PDF file of audits or a raw XML file that you can process with your own tools later.

The **Select sort** drop-down determines how the audits should be sorted in the output file.

Report outputs

Updated: March 25, 2022

In Transit you can specify the type of output.

Output	Output action	Description
PDF	Create PDF file	When PDF is specified a .pdf file will be created using the stylesheet specified in UCX-RPT036. The .pdf file will be placed in the admin/pdfreports directory. This PDF file can then be downloaded, viewed, and printed as needed. The dap22done script (found in app/scripts) can be modified to move the files from the pdf reports directory to the final destination.
		When the “Create individual output” option is checked, the files will placed in the admin/pdfreports/indivNNNN directory, where NNN is the job

Output	Output action	Description
		<p>number returned in Transit. Each PDF file is named student_<id>_<school>_<degree>.pdf</p> <p>Example: student_123456_UG_BS.pdf</p> <p>Note: You must also select an audit report if PDF is selected. This tells Degree Works what stylesheet is needed in addition to other UCX-RPT036 configuration options.</p>
XML	Create raw xml	<p>The XML file created will be placed in the admin/xmltrees directory. You may use this raw XML as input to one of your own tools as needed.</p> <p>When the “Create individual output” option is checked, the files will be placed in the admin/xmltrees/indivNNNN directory, where NNN is the job number returned in Transit. Each XML file is named student_<id>_<school>_<degree>.xml</p> <p>Example: student_123456_UG_BS.xml</p>

dwfop

Updated: March 25, 2022

The dwfop script is used by Degree Works to take in an XML file and create a PDF file or printed output.

Warning! There is a 2GB limit to any file created on the system. Because of this limit you need to be careful of the number of students you run in any one particular batch.

Each audit's xml can be 50K, 100K, or more thus limiting the number of students that can be commingled into a single XML file. Whether you desire PDF output, printed output or the raw XML file, an XML document is created along the way containing an audit for each of the students in your pool. You may have to split up your pool of students into multiple runs to get the output you want. The DAP22 log file shows a line for each student processed and shows you how complete the audit is for each student. Here you can see that one student is 100% complete with the requirements for their BS degree while the other two students still have some work to do. You may want to copy and paste this information into Excel and pull out just those students who are close to completion. This percent complete information appears only when new audits are being created.

```
Processing student 19719874: UG/BA... - 71% complete
Processing student 19843213: UG/BS... - 100% complete
Processing student 82347211: UG/BA... - 93% complete
```

PDF

When creating a PDF file from XML you need to supply the XML filename, "PDF" as the 2nd parameter, the XSL filename and the PDF filename.

Format: dwfop xml-filename pdf xsl-filename pdf-filename

Example: dwfop myfile.xml pdf /path/to/xsl/some.xsl myfile.pdf

Print

When printing a file from XML you need to supply the XML filename, "PRINT" as the 2nd parameter, the XSL filename, the printer and the number of copies.

Format: dwfop xml-filename print xsl-filename device num-copies

Example: dwfop myfile.xml print /path/to/xsl/someother.xsl 0952 1

Note:

- The xml file is looked for in xmltrees if the filename given is not found in current directory.
- The xsl file and the other xsl files it uses can be any directory. You need to specify where the xsl files reside.
- Images referenced in the xsl should be housed in the same localization as the xsl files.
- The pdf file name must be absolute (basename or with directory specified).
- The fop file is created in the admin tmp directory and saved if debugging is on (DWDEBUG=1). Otherwise it is deleted.

DAP27 - What-if Audits

Updated: March 24, 2023

DAP27 generates what-if audits on a set of selected students in batch.

Access to run DAP27 in Transit is granted with the TRDAP27 Shepherd key.

You may want to do this to determine if some students have completed a certificate or have completed another program.

You must select the Catalog Year, Level, and Degree. You can then select at most one of each of the following: program, college, major, minor, concentration, specialization and liberal learning. Because of this, you cannot run audits for double-majors, for example.

If you choose to freeze the audits you will be guaranteed that they will remain in the database until you choose to delete them. (See the notes below on deleting frozen audits.) This allows you to view the audits on the web in the What-if History tab or to run SQL against the dap_audit_dtl to examine the high-level results. (See the notes below on running SQL against the audit results.) If you do not freeze the audits it is possible the audits will get deleted before you get a chance to review them. It is recommended that you setup a least one freeze type in AUD032 to be used for these batch what-if audits. You may choose to freeze all of these audits using a freeze type of WHATIF or you may setup different codes for the different what-if batches you will be running.

Instead of, or in addition to, freezing the audits, you can print or create a PDF file. Review the notes under DAP22 about printing and creating a PDF.

The DAP27 log file shows a line for each student processed, much like DAP22. The log file shows you how complete the audit is for each student. Here you can see that one student is 100% complete with the requirements for the BUS_CERT degree while the other two students still have some work to do. You may want to copy and paste this information into Excel and pull out just those students who are close to completion.

```
Processing student 19719874: UG/BUS_CERT... - 71% complete
Processing student 19843213: UG/BUS_CERT... - 100% complete
Processing student 82347211: UG/BUS_CERT... - 93% complete
```

Note that the majors are pulled from AUD027 instead of STU023 and that the minors are pulled from AUD029 instead of STU024.

Frozen audit deleting

If you chose to freeze the audits you can delete them using the freeze type you specified. The easiest way to do this is to use AUD02 in Transit. This tool makes use of the dapdelaudits script on the server to delete all audits created before a specified date with the freeze type you used. For this reason, you should consider if using a single freeze type of WHATIF is sufficient or if you need to create several different freeze types for the different batches you will be running.

SQL run against audit results

In order to have the audits saved to the database you must have the CFG020 DAP14 What-if History Count flag set to a value of 01 or greater – even if you are freezing them. Freezing them ensures that they will be there until you delete them while relying on the History Count only guarantees that they will be saved for some period of time before they will eventually be deleted when new what-if audits are generated – in batch or on the web. When the audit is saved to the database, a DAP_AUDIT_DTL record with high level information is recorded. You can find out how complete the degree is for each student by running a simple query such as the following:

```
Select dap_stu_id, dap_audit_pct from dap_audit_dtl  
where dap_freeze_type='WHATIF'  
Order by dap_stu_id;
```

This assumes you only have one set of batch what-if audits housed in the database at a time and have frozen the audits using the WHATIF freeze type. You can also use the dap_degree field as part of your WHERE clause or use more than one freeze type when running your audits to help differentiate the different sets of audits you are running.

Note that the dap_audit_pct (audit percent complete) field is a CHAR field and will contain values such as 0037, 0098, 0100. Please keep this in mind when trying to select on specific values and using greater-than and less-than. It might be best to export this information to Excel and use the tools in Excel to format/filter/etc as needed.

The CPA dap_result_dtl records are not generated as part of DAP27 so you cannot query the details of the audit results. The details of the audit are stored in the dap_audit_tree, but this table cannot be queried because the data within is all binary.

Review and configure the transit.dap27.workerCount setting.

DAP28 - Alternate What-if Audits

Updated: March 24, 2023

DAP28 generates what-if audits in batch on a set of selected students based on the alternate curriculum you have bridged for each student.

Access to run DAP28 in Transit is granted with the TRDAP28 Shepherd key.

You may have an alternate set of curriculum for some set of students for which you want to generate audits.

The curriculum for each student is found by retrieving from the rad_custom_dtl the alternate school, degree and catalog year. Each of these items must have been bridged with the names specified. For each of the other goal values, you may bridge multiple majors, minors, etc by specifying a number after each. For example, A-MAJOR1 could be bridge for CHEM and A-MAJOR2 could be bridged for BIOL. See the chart below for more information and how to bridge catalog year values for each goal value.

You can choose to freeze the audits and you can choose to print or create a PDF file out of them. If you choose to freeze the audits, you will be guaranteed they will remain in the database until you choose to delete them. (See the notes below on deleting frozen audits.) This allows you to view the audits on the web in the What-if History tab or to run SQL against the dap_audit_dtl to examine the high-level results. (See the notes below on running SQL against the audit results.) If you do not freeze the audits it is possible the audits will get deleted before you get a chance to review them. It is recommended that you setup a least one freeze type in UCX-AUD032 to be used for these batch what-if audits. You may choose to freeze all of these audits using a freeze type of WIFALT or you may setup different codes for the different what-if batches you will be running.

Instead of, or in addition to, freezing the audits, you can create a PDF file. Review the notes under DAP22 about creating a PDF.

The DAP27 log file shows a line for each student processed, much like DAP22. The log file shows you how complete the audit is for each student. Here you can see that one student is 100% complete with the requirements for the UG/BA degree while the other two students still have some work to do in their degrees. You may want to copy-n-paste this information into Excel and pull out just those students who are close to completion.

```
Processing student 19719874: UG/BS    ... - 71% complete
Processing student 19843213: UG/BA    ... - 100% complete
Processing student 82347211: GR/MBA   ... - 93% complete
```

One use of this is for Banner schools to bridge the outcome curricula as custom records. You will then essentially be running what-if audits on the outcome records. You may want to create a UCX-AUD032 freeze type of WIFOUT or OUTCOM to help you better identify these audits.

You will need to bridge the alternate curricula to the rad_custom_dtl by using the method appropriate for your student extract:

- Banner schools will setup UCX-BAN080 records.
- Other schools will create BIF records as needed using the R121CUST record layouts.

Custom code	Meaning	Example code and value
A-SCHOOL	School (aka Level)	A-SCHOOL=UG
A-DEGREE	Degree	A-DEGREE=BS
A-CATYEAR	Overall Catalog Year	A-CATYEAR=2014
A-PROGRAMx	xth Program – x is 1, 2, 3, and so on	A-PROGRAM1=BS_CHEM
A-PROGRAMxCY	xth Program Catalog Year	A-PROGRAM1CY=2015
A-MAJORx	xth Major – x is 1, 2, 3, and so on	A-MAJOR1=CHEM
A-MAJORxCY	xth Major Catalog Year	A-MAJOR1CY=2013

The same codes are supported for COLLEGE, MINOR, CONC, SPEC, and LIBL following the pattern you see above for PROGRAM and MAJOR.

The CY catalog year code and value are optional for any given goal. If it is not found the overall catalog year value is used when determining the correct Scribe block to use for the given goal code.

For Banner schools, if your school is using the Program-as-degree feature, you will instead bridge the program code (example: BA_HIST) as the A-DEGREE and optionally bridge the degree code (example: BA) as A-PROGRAM1.

Frozen audit deleting

If you chose to freeze the audits, you can delete them using the freeze type you specified. The easiest way to do this is to use AUD02 in Transit. This tool makes use of the dapdelaudits script on the server to delete all audits created before a specified date with the freeze type you used. For this reason, you should consider if using a single freeze type of WIFALT is sufficient or if you need to create several different freeze types for the different batches you will be running.

SQL run against audit results

In order to have the audits saved to the database you must have the UCX-CFG020 DAP14 What-if History Count flag set to a value of 01 or greater – even if you are freezing them. Freezing them ensures that they will be there until you delete them while relying on the History Count only guarantees that they will be saved for some period of time before they will eventually be deleted when new what-if audits are generated – in batch or on the web. When the audit is saved to the database, a DAP_AUDIT_DTL record with high level information is recorded. You can find out how complete the degree is for each student by running a simple query such as the following:

```
Select dap_stu_id, dap_audit_pct from dap_audit_dtl  
where dap_freeze_type='WIFALT'  
Order by dap_stu_id;
```

This assumes you only have one set of batch what-if audits housed in the database at a time and have frozen the audits using the WIFALT freeze type. You can also use the dap_degree field as part of your WHERE clause or use more than one freeze type when running your audits to help differentiate the different sets of audits you are running.

Note that the dap_audit_pct (audit percent complete) field is a CHAR field and will contain values such as 0037, 0098, 0100. Please keep this in mind when trying to select on specific values and using greater-than and less-than. It might be best to export this information to Excel and use the tools in Excel to format, filter, and so on as needed.

The CPA dap_result_dtl records are not generated as part of DAP28 so you cannot query the details of the audit results. The details of the audit are stored in the dap_audit_tree, but this table cannot be queried because the data within is all binary.

Review and configure the transit.dap28.workerCount setting.

DAP40 Unload Scribe Blocks

Updated: March 25, 2022

The DAP40 batch program unloads your Scribe blocks to a zip file that you can download from Transit.

Access to run DAP40 in Transit is granted with the TRDAP40 Shepherd key.

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.

There are no selection or sort choices, although you can restrict the output to certain types of blocks using the one question for the job:

When you select blocks to unload, your choices are:

- 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.

DAP40 will extract the desired blocks, one per file, named with the block ID and a .text extension (for example, RA000291.text). The block metadata (for example, title, period) will be extracted into a separate file named with the block ID and an extension of .fields (for example, RA000291.fields). The job will also unload the DAP_NEXT_ID_MST table entries for the extracted blocks. They are named DWNXTRMST.dmp (for the RA next ID) and DWNXTRMST_Pdmp (for the RB next ID). All of the files will be zipped into one job artifact called dapblocks.zip. This file can then be downloaded to your local machine. The file may be quite large depending on the number and size of your blocks.

For more information about using DAP40 and DAP41 to manage blocks, see [Blocks transfer between two different environments](#).

DAP40 errors, warnings, and success

Updated: March 25, 2022

The action file will normally contain only the start and end times and a count of the total number of blocks unloaded.

If there are errors, an error count and error messages will also show. The log file will show additional information related to the errors. These will usually be database errors that require a DBA to resolve.

DAP41 Load Scribe Blocks

Updated: March 25, 2022

The DAP41 batch program loads your Scribe blocks from a specified zip file that you upload to Transit.

Access to run DAP41 in Transit is granted with the TRDAP41 Shepherd key.

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.

There are no selection or sort choices. The required questions prompt the user for the zip file containing the blocks to load and what type of blocks should be loaded.

DAP41 will check to see if zip file contains academic blocks, planner blocks, or both. The zip file may contain both types of blocks but you can specify to only load one type. However, if the zip file contains only one type of blocks and you specify you want to load the other set of blocks, the job output will show an error. Before the new blocks are loaded the existing academic and planner blocks are deleted from the database. The zip file should contain the complete set of blocks and not merely a subset for the given type. The blocks are then loaded from the .text and .fields files contained within the zip file. The DWNXTRMST.dmp (for the academic blocks) and DWNXTRMST_P.dmp (for the planner blocks) are then loaded into the dap_next_id_mst. This ensures that new blocks added in Scribe will start off with the next available block ID.

For more information about using DAP40 and DAP41 to manage blocks, see [Blocks transfer between two different environments](#).

DAP41 errors, warnings, and success

Updated: March 25, 2022

The action file will normally contain just the start and end times and a count of the total number of blocks loaded.

If there are errors, an error count and error messages will also show. The log file will show additional information related to the errors. These will usually be database errors that require a DBA to resolve.

DAP42 Unload Mappings

Updated: March 24, 2023

The DAP42 batch program unloads your transfer mappings to a zip file that you can download from Transit.

Access to run DAP42 in Transit is granted with the TRDAP42 Shepherd key.

You can then use the DAP43 job to upload this file into a different environment. This allows you to move blocks from your test to production environment, for example.

There are no selection or sort choices for the job.

DAP42 will extract all mapping data from the DAP_MAPPING_DTL, DAP_MAP_COND_DTL, and DAP_MAP_ATTR_DTL. Each set of records is placed into a dmp file with the name of the table. The job will also unload the DAP_NEXT_ID_MST table M entry. All of the files will be zipped into one job artifact called mappingsunload.zip. This file can then be downloaded to your local machine. The file may be quite large depending on the number mappings.

For more information about using DAP42 and DAP43 to manage blocks, see [Mappings transfer between two different environments](#).

DAP43 Load Mappings

Updated: June 17, 2022

The DAP43 batch program loads your mappings from a specified zip file that you upload to Transit.

Access to run DAP43 in Transit is granted with the TRDAP43 Shepherd key.

The zip file must have come from you running DAP42 to unload mappings from the same or another environment. Ensuring this allows you to move mappings from your test to production environment, for example.

There are no selection or sort choices. The required question prompts the user for the zip file containing the mappings to load.

Before the new mappings are loaded, the existing mappings are deleted from the database. The zip file should contain the complete set of mappings and not merely a subset. The mappings are then loaded from the dmp files contained within the zip file. The DAP_NEXT_ID_MST.dmp is then loaded into the dap_next_id_mst. This ensures that new mappings added will start off with the next available mapping ID.

For more information about using DAP42 and DAP43 to manage blocks, see [Mappings transfer between two different environments](#).

DAP44 Unload UCX tables

Updated: September 30, 2022

The DAP44 batch program unloads your UCX tables to a zip file that you can download from Transit.

Access to run DAP44 in Transit is granted with the TRDAP44 Shepherd key.

You can then use the DAP45 job to upload the zip file into a different environment. This allows you to move UCX tables from your test to production environment, for example.

There are no selection or sort choices for the job.

DAP44 will unload each table listed in SYS001 into its own file with the name of the table. All of the files will be zipped into one job artifact called ucxunload.zip. This file can then be downloaded to your local machine. You can selectively decide to load certain UCX tables instead of all of them by editing the zip file and removing the tables you do not want loaded through DAP45. When editing the zip file, do not remove the SYS001 file or you will get an error when running DAP45.

For more information about using DAP44 and DAP45 to manage UCX tables, see [UCX tables transfer between two different environments](#).

DAP45 Load UCX tables

Updated: September 30, 2022

The DAP45 batch program loads your UCX tables from a specified zip file that you upload to Transit.

Access to run DAP45 in Transit is granted with the TRDAP45 Shepherd key.

The zip file must have come from you running DAP44 to unload UCX tables from the same or another environment. This allows you to move UCX tables from your test to production environment, for example.

There are no selection or sort choices. The required question prompts the user for the zip file containing the UCX tables to load.

Each UCX table found in the zip file is loaded into its respective database table replacing the previous contents.

For more information about using DAP44 and DAP45 to manage UCX tables, see [UCX tables transfer between two different environments](#).

DAP54 - Template processor plan creation

Updated: March 25, 2022

You can assign plans in a batch to students using DAP54 in Transit.

Access to run DAP54 in Transit is granted with the TRDAP54 Shepherd key.

You can specify the template that is to be used for all students in your pool or you may let the processor determine the best template for the student based on their curriculum. Note: If you use the template ID option, the template you specify must have a DEGREE tag (either optional or required). If there isn't a DEGREE tag, plans will not be created for any student.

You must also select the starting term for the plans created because the template only specifies the term type – not the actual terms. If you do specify a template be sure the starting term you specify has the same term type as the first term type on the template.

For the students selected, you can create the plans as temporary ones. When the “Are these plans temporary” check box is selected, the plans that are created will have the sep_plan.create_what set to TEMPORARY.

If you encounter an “insufficient memory for the Java Runtime Environment to continue” error message, you may need to increase the memory allocation for DAP54. You can do this by increasing the DAP54_MAX_HEAP variable in dwenv.config.

The steps used to find an appropriate template for each student are as follows:

If the template is specified:

- The template specified will be retrieved from the database
- It is assumed that the template has a degree tag
- For each student, the student's degree will be compared to the template degree to be sure they match
- If the degree on the template does not match the student's degree, no plan is created
- If a student has multiple degrees, the degree matching the template will be chosen and saved on the new plan

If the template is not specified:

- Try to find a template based on all of the student's data (school, degree, major, minor, conc, etc.) This is done for each of the student's degrees.
- If more than one matching template is found, the template with the most tags is chosen. If multiple templates have the most matching tags, then the majors on the matching templates are examined. The template for the primary major will be chosen. If multiple templates match as a result of multiple concentrations for a single major, then no template is selected.
 - For example, a template with the student's minor may be found and another with the student's conc may be found. Because both are valid for the student, no decision can be made as to which to use.
 - Another example, templates are found for CHEM and MATH majors. Because CHEM is the student's primary major, that template is selected.
 - Another example, templates are found for the VIOLIN and PIANO concentrations for the music major. In this case a decision is not made; no template is selected.
- If any template found has a template scheme whose first term type does not match the term type of the starting term specified, the template will be discarded.

The DAP54 log files viewable in Transit gives a summary of how many plans were, and were not, created for your pool of students.

```
--- Plan Creator has completed  
91 students were selected for plan creation  
83 students received a new plan
```

The batch assignment processor does not attempt to assign a plan to a student if the student already has a plan recorded for the degree.

When building templates you should be sure to create one for every type of student – those that do not have a major on their degree record and perhaps those that don't even have a degree code recorded.

Temporary plans are only created for students who do not have an active, approved plan for the given degree. When the check box is not selected, no plan will be created for the student if they already have a plan – even if it is inactive/unapproved. However, multiple temporary plans for the same degree will be created if DAP54 is run more than one time without deleting existing temporary plans from the database. To identify temporary plans, "TEMPORARY" is recorded on

the sep_plan.create_what field in the database. You can delete temporary plans later by doing the following in the Degree Works database:

```
execute sep_helper.delete_plans_create_what ('TEMPORARY');
```

Degree template search

Updated: March 25, 2022

The assumption for all examples is that UCX-SEP001 has a specific setup.

SCHOOL	Required=Y	MatchStudentGoalData=Y
DEGREE	Required=Y	MatchStudentGoalData=Y
MAJOR	Required=Y	MatchStudentGoalData=Y
MINOR	Required=N	MatchStudentGoalData=Y
CONC	Required=N	MatchStudentGoalData=Y
COOP	Required=N	MatchStudentGoalData=N

Example 1

Templates for CHEM major

1	SCHOOL=UG CONC=BIOCHEM	DEGREE=BS	MAJOR=CHEM	MINOR=ART
2	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	MINOR=ART
3	SCHOOL=UG CONC=BIOCHEM	DEGREE=BS	MAJOR=CHEM	MAJOR=CHEM
4	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	MAJOR=CHEM

Student's degree: SCHOOL=UG, DEGREE=BS, MAJOR=CHEM

Template 1 is not chosen because the student does not have the minor or concentration.

Template 2 is not chosen because the student does not have the minor.

Template 3 is not chosen because the student does not have the concentration.

Template 4 is chosen because the student has the matching school, degree and major.

Example 2

Templates for CHEM major – with COOP tag

1	SCHOOL=UG COOP=SPRING3	DEGREE=BS	MAJOR=CHEM	MINOR=ART
2	SCHOOL=UG COOP=SPRING3	DEGREE=BS	MAJOR=CHEM	MAJOR=CHEM

Student's degree: SCHOOL=UG, DEGREE=BS, MAJOR=CHEM

Template 1 is not chosen because the student does not have the minor.

Template 2 is chosen because the student has the matching school, degree, and major. Because the co-op has the MatchStudentGoalData flat set to N in UCX-SEP001 this is not compared against the student's degree.

Example 3

Templates for CHEM major; student has minor and conc also

1	SCHOOL=UG CONC=BIOCHEM	DEGREE=BS	MAJOR=CHEM	MINOR=ART
2	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	

Student's degree: SCHOOL=UG, DEGREE=BS, MAJOR=CHEM; MINOR=ART, CONC=BIOCHEM

The first template is selected because we search on all of the student's goal information (school, degree, major, minor, and conc in this situation).

Template 1 is valid because both the minor and concentration match.

Template 1 is chosen.

Example 4

Templates for CHEM major; student has minor and concentration – but templates have no concentration

1	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	MINOR=ART
2	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	

Student's degree: SCHOOL=UG, DEGREE=BS, MAJOR=CHEM; MINOR=ART, CONC=BIOCHEM

In this example, no template was found matching all of the student's data but two templates were found matching some of the student's data. Because the first template matches on more data than the second one, template 1 is chosen.

Example 5

Templates for CHEM major; student has minor and conc also

1	SCHOOL=UG CONC=BIOCHEM	DEGREE=BS	MAJOR=CHEM	MINOR=ART
2	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	MINOR=ART
3	SCHOOL=UG CONC=BIOCHEM	DEGREE=BS	MAJOR=CHEM	
4	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	

Student's degree: SCHOOL=UG, DEGREE=BS, MAJOR=CHEM; MINOR=ART, CONC=BIOCHEM

These templates are selected because we search on all of the student's goal information (school, degree, major, minor, and conc in this situation).

Template 1 is valid because both the minor and concentration match.

Template 2 is valid because the minor matches.

Template 3 is valid because the concentration matches.

Template 4 is also valid because all of its tags match the student's data.

However, template 1 is chosen because it has more tags that match the student's data.

Example 6

Templates for CHEM major; student has minor and conc also

1	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	MINOR=ART
2	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	
	CONC=BIOCHEM			
3	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM	

Student's degree: SCHOOL=UG, DEGREE=BS, MAJOR=CHEM; MINOR=ART, CONC=BIOCHEM

These templates are selected because we search on all of the student's goal information (school, degree, major, minor and conc in this situation).

Template 1 is valid because the minor matches.

Template 2 is valid because the concentration matches.

Template 3 is also valid because all of its tags match the student's data.

Both templates 1 and 2 have the same number of matching tags but it cannot be determined which one should take precedence so template 3 is chosen because it is the best match of required template tags to the student's goal information.

Example 7

Templates for CHEM and MATH major

1	SCHOOL=UG	DEGREE=BS	MAJOR=CHEM
2	SCHOOL=UG	DEGREE=BS	MAJOR=MATH

Student's degree: SCHOOL=UG, DEGREE=BS, MAJOR=CHEM; MAJOR=MATH

These templates are selected because we search on all of the student's goal information (school, degree, major, minor, and conc in this situation).

Template 1 is valid because the major matches.

Template 2 is valid because the major matches.

Template 1 is chosen because CHEM is the student's primary major.

Example 8

Templates for MUSIC major:

1	SCHOOL=UG	DEGREE=BA	MAJOR=MUSIC	CONC=PIANO
2	SCHOOL=UG	DEGREE=BA	MAJOR=MUSIC	CONC=VIOLIN

Student's degree: SCHOOL=UG, DEGREE=BS, MAJOR=MUSIC; CONC=PIANO; CONC=VIOLIN

These templates are selected because we search on all of the student's goal information (school, degree, major, minor, and conc in this situation).

Template 1 is valid because the major and conc matches.

Template 2 is valid because the major and conc matches.

Because PIANO is the student's primary concentration, template 1 will be chosen.

DAP58 - Batch tracking processor

Updated: March 25, 2022

The DAP58 processor allows you to track whether or not your students are following their educational plans.

Access to run DAP58 in Transit is granted with the TRDAP58 Shepherd key.

The tracking processor checks each plan to see if the specified courses were taken on or before the planned term, checks GPA requirements, checks if minimum test scores were achieved, and if non course requirements were taken. Each requirement, term, and plan has its tracking status updated allowing you to run reports against the data.

If you encounter an "insufficient memory for the Java Runtime Environment to continue" error message, you may need to increase the memory allocation for DAP58. You can do this by increasing the DAP58_MAX_HEAP variable in dwenv.config.

When you run DAP58, you specify whether you want to run it in the official or unofficial mode. On the plan (sep_plan) and term (sep_plan_term) tables this determines whether the OFFICIAL_TRACKING_STATUS or the UNOFFICIAL_TRACKING_STATUS is updated by the processor. The requirements' TRACKING_STATUS field is updated regardless of the mode being used. The tracking status values are ON_TRACK, OFF_TRACK, NOT_EVALUATED, and NOT_FOUND. Newly added requirements and those in the future have a status of NOT_EVALUATED while certain GPA requirements may have a NOT_FOUND status if the GPA for the past term cannot be determined.

You also must specify a cutoff term when running the processor. Planned terms following this cutoff term are ignored by the processor. Only those terms from UCX-STU016 with Show in SEP Picklist set to Y will appear in this picklist.

The DAP58 log files viewable in Transit give a summary of how many plans were processed for your pool of students.

241 students were selected for tracking status update
241 students were successfully updated

When you run the processor in the unofficial mode, the updated tracking status is not reflected on the web interface for plans. The unofficial mode is meant for reporting purposes. To see the updated tracking status in the Responsive Dashboard, run the processor in the official mode.

DAP59 - Batch timetabling processor

Updated: March 25, 2022

The DAP59 processor allows you to run audits against students' educational plans.

Access to run DAP59 in Transit is granted with the TRDAP59 Shepherd key.

The timetabling processor runs an audit for each of the student's active and locked/approved plans for the given school/level specified. If an active and locked/approved plan cannot be found for a student, the processor will look for a TEMPORARY plan created from a template by DAP54. All classes after the student's active term up to and including the cutoff term specified will be included in the audit. If the student has preregistered for the same classes that appear on the plan, the duplicate classes appearing on the plan will not be included, redundantly in the audit. This special planner audit is then saved to the CPA tables allowing you to find out how each of the planned classes applied in the audit.

If you encounter an "insufficient memory for the Java Runtime Environment to continue" error message, you may need to increase the memory allocation for DAP59. You can do this by increasing the DAP59_MAX_HEAP variable in dwenv.config.

For more information, see the [Planner audits in CPA](#) topic.

You must specify the school/level of the students and plans you want to process.

You also must specify a cutoff term when running the processor. Planned terms following this cutoff term are ignored by the processor.

RAD11 - The Traditional Bridge Batch Processor

Updated: March 25, 2022

The "traditional" approach to updating the Degree Works data repository (RAD) from the Student Information System (SIS) involves "bridging" information from the SIS database to the RAD database using the "RADBRIDGE" batch program as the processor.

Access to run RAD11 in Transit is granted with the TRRAD11 Shepherd key.

It takes as input a file containing a list of data files which have been prepared by a client-written program operating on the SIS side that prepares new information for transmission to Degree Works. This list file is called RAD11M01 and resides in the data sub-directory of the test or

production environment. The RAD11M01 file is a record oriented file (there is a newline character on each line). Each record in RAD11M01 contains the name of a data file containing data to be processed.

As the “classic” approach, this methodology may always be used to move data between the SIS and RAD irrespective of other methods that may be developed by Ellucian for specific Student Information systems, such as an integrated data extract, which uses RAD30 (see next section).

RAD11 loads extracted data from the student information systems (SIS) into the Degree Works database.

For more information, see the [Bridge program](#) topic.

After selecting RAD11 - Radbridge Batch Processor, in the space provided, enter the name of the applicable BIF file.

Note that the BIF file must already exist in your admin/data directory on the classic server.

Review and configure the following settings:

- parser.batch.notifications.toEmail
- transit.rad11.workerCount

Data files

Updated: March 25, 2022

The data files used by RAD11 to process data from the SIS (Student Information System) are fixed length files 1000 bytes in length.

Each data file contains records described in the [Record layout](#) topic, and they also reside in the data sub-directory of the test or production environment. A data file can contain the data for multiple students, ETS schools, courses, UCX tables and Degree Works course equivalency information. It is very important that data for a particular ID (student or staff member) not be split across multiple files.

RAD11 identifies a record with a data file according to its indicator. RAD11 sorts data according to ID, indicator and term if appropriate. For this reason it is essential that indicators be exact. See the [Bridge program](#) topic for a list of indicators and the data set in the Degree Works data repository to which the record is applied.

FTP process

Updated: March 25, 2022

The data files used by RAD11 to process data from the SIS (Student Information System) should be delivered to the admin/data sub-directory of the test or production environment through FTP or a similar process.

The details of this process should be worked out with Ellucian. The basic syntax for delivering a data file through FTP is:

```
> ascii  
> put datafile  
> put RAD11M01
```

An example RAD11M01 file might be:

```
students1  
students2
```

Where students1 is a set of BIF records for one set of students and students2 is another file for a separate set of students. Normally the RAD11M01 file has a single file with all of the students to be bridged within it – but you have the option to list multiple files in RAD11M01. The RAD11M01 file also optional. You can simply bridge the datafiles themselves and refer to them directly when you run RAD11JOB.

Run RAD11 processor

Updated: March 25, 2022

RAD11 can be scheduled to run nightly, weekly, or on some other regular basis. It is recommended that RAD11 be run at least one time a week to refresh student data for the current term or semester.

You can run simultaneous RAD11 processes to take advantage of the multiple processors you have on your machine by splitting your data up into multiple files and listing these files in the RAD11M01 file. For example, if your site has 4 CPUs, split your bridge data files into four separate files and add these files to the RAD11M01. When RAD11JOB runs, it will spawn one process for each of the data files listed in RAD11M01. You may want to run a batch on seniors, another on juniors, another on sophomores and yet another on first-year students. Running batch bridge processes like this in parallel should result in better throughput if you have more than one CPU on your system.

Instead of creating multiple files, you can also deliver a single file to Degree Works. When one file of students is listed in RAD11M01, the transit.rad11.workerCount setting (set in Controller) comes into play. Based on this setting, the student data file will be split up into multiple files, allowing each file to be run by RAD11 simultaneously, having the same effect as having supplied RAD11 with multiple files to begin with.

Each time RAD11 processes data for an ID all records for that ID should be included in the data file. RAD11 will delete old records completely before adding new ones from information provided in the data file. A total refresh of student data is done each time RAD11 processes an ID.

If RAD11M01 does not exist or is empty then RAD11 will do no processing and reschedule its next run, if configured to do so.

The RAD Bridge program can be executed either on demand from Transit or scheduled on a recurring basis using the UNIX "at" utility. Therefore, there are 2 scripts for executing RAD11:

RAD11JOB and rad11sch. Both reside in the batch directory. RAD11JOB is used by Transit, while rad11sch is used with "at" to schedule a recurring bridge.

The rad11sch script is used to schedule RAD11JOB on a periodic basis using "at". Specifically, rad11sch passes the time interval for "at" to the RAD11JOB script. The time interval is configured by you upon installation of Degree Works or whenever you are ready to schedule a recurring bridge of student data. Modify the setting in rad11sch to run RAD11JOB every x minutes, hours or days beginning at y time. You may want to run RAD11JOB every 30 minutes to see if your system has sent it new files to process. If RAD11JOB finds no files, it will remain idle.

```
$ cat rad11sch

#!/usr/bin/sh
# rad11sch is a script that executes RAD11JOB using the "at" utility
#
# It is used to schedule the RAD Bridge program (RAD11) to run at
# recurring intervals.
#
#####
####
# Run rad11job telling it to reschedule rad11sch tomorrow at 3 a.m.
RAD11JOB $Pid -t "03:00am tomorrow"
#####
####
# Other examples
# RAD11JOB $Pid -t "now + 30 minutes" - 30 minutes from now
# RAD11JOB $Pid -t "now + 2 hours"      - 2 hours from now
# RAD11JOB $Pid -t "now + 1 day"        - one day from now
# RAD11JOB $Pid -t "0300 tomorrow"      - at 3am tomorrow
# RAD11JOB $Pid -t "03:00am + 1 week"   - at 3am 1 week from now
# RAD11JOB $Pid -t "0300"                - 3am tomorrow as long as 3am
# has already
# passed
#
#           If no date is given, today is assumed if the given
#           time is greater than the current time, and tomorrow is
#           assumed if it is less. If the given month is less than
#           the current month (and no year is given), next year is
#           assumed.
```

RAD11JOB takes in this optional time parameter and runs rad11sch using "at".

For more information on the "at" utility read the man pages on your system.

RAD11JOB is executed from Transit without the optional time parameter. It executes on demand, as opposed to rad11sch which executes as scheduled.

Errors, warning, and success

RAD11 will write fatal errors, warnings and success to the rad_log_dtl of the Degree Works data repository. RAD11 will also extract these messages to its log at the end of its run. The log file can be viewed using the View Jobs button in Transit.

Fatal errors

RAD11 requires two records to successfully process an ID: PRIM and DEGR. If either of these records does not exist a fatal error is written to the rad_log_dtl and processing continues with the next ID. In addition, essential data items in these records are validated against UCX tables.

Invalid data will also generate a fatal error. The TERM field in the PRIM record is validated against UCX-ST016, the school field in the rad_goal_dtl file is validated against UCX-STU350 and catalog_yr from the rad_goal_dtl file is validated against UCX-STU035. ID numbers in each of these data files are validated against the ID number in each header record.

If an ID is a staff member then only a PRIM record is required.

Warnings

RAD11 will write warnings to the rad_log_dtl if configured to do so. Warnings are produced if ID numbers in the remaining non-required data files do not match the ID in the header record. When a warning is produced for one of these data files the record in question is skipped and processing continues with the next record for that ID. The first 200 bytes of the record are written to the rad_log_dtl for easy identification.

Success

RAD11 will write success to the rad_log_dtl for each successful processing of an ID, if configured to do so. "OKAY" is the message written along with the ID number.

Additional modes

RAD11 can be run to update ETS schools, courses, UCX tables, Degree Works equivalency information, delete all for an ID or to exchange ID numbers. Data for these additional modes can be included in the same data file for an ID or placed in other files as long as the file is listed in RAD11M01.

Delete ID

Delete ID deletes all data for an ID from the Degree Works data repository. See [UCX-CFG020 RADBRIDGE](#) to set a configuration flag if DAPDB data such as notes should be deleted as well. For more information about the data layout, see [Bridge Interface Format](#).

Change ID

If an ID number for a student changes for any reason, RAD11 can be used to change the ID number in all tables that an ID has data for. For more information about the data layout, see [Bridge Interface Format](#).

UCX update

RAD11 can be used to perform a batch update of UCX tables. For more information about the data layout, see [Bridge Interface Format](#).

ETS update

RAD11 can be used to perform a batch update of ETS data. For more information about the data layout, see [Bridge Interface Format](#). RAD11 will not delete all ETS data in the Degree Works data repository. It is necessary to include only those records you want to add or update.

Course update

RAD11 can be used to perform a batch update of Course data. For more information about the data layout, see [Bridge Interface Format](#). RAD11 will not delete all Course data in the Degree Works data repository. It is necessary to include only those records you want to add or update.

dap_eqv_crs_mst update

RAD11 can be used to perform a batch update of Course equivalency data. For more information about the data layout, see [Bridge Interface Format](#). RAD11 will delete all Course equivalency data in the Degree Works data repository. It is therefore necessary to include all equivalency records when updating the dap_eqv_crs_mst table using RAD11. To update/add a single dap_eqv_crs_mst record, use Controller to edit/add the record to the UCX-CFG070 table and then run the dapucx2eqv script to load the records to the dap_eqv_crs_mst table.

Changed data

RAD11 stores a value in the rad_hash_mst representing the last set of student data bridged. When RAD11 receives a new set of data for a student the hash value derived from the new set of data is compared to the value stored in the rad_hash_mst. If the values are the same RAD11 recognizes that the new student data is exactly what is already in the database and does not save this new data. If the old hash value differs from the new value, RAD11 writes a date along with the new hash value to the rad_hash_mst; the new set of data is also saved to the database. In performing this hash check, RAD11 will not spend time processing student data that does not need to be bridged.

rad_hash_mst override

While the rad_hash_mst allows the RAD11 processor to skip student records for which the data has not changed, it is sometimes necessary to ensure that new records are added to the database regardless of the rad_hash_mst value. This is particularly true when conducting performance benchmark testing. You may tell RAD11 to ignore the rad_hash_mst by using the "FORCE" option:

```
$ RAD11JOB FORCE
```

RAD11 simply knows to ignore the value in the current rad_hash_mst and bridge the student regardless of not having changed data.

DAP22 after RAD11 completes

After RAD11 completes, the radhashid script extracts the student IDs from the rad_hash_mst that were just bridged by RAD11. Only those students who actually had changed data will be selected from the rad_hash_mst. After the list of student IDs has been created, the batch/dap22r11 script launches DAP22 to run a new audit for each of the students. In doing this, we are ensuring that an up-to-date audit exists based on the latest bridged data for each student. A batch-id and date on the rad_hash_mst ensures that DAP22 is run on those students just bridged by RAD11; students bridged earlier in the day by another RAD11 job will not be considered.

Notifications

The RAD11JOB executes a rad11started when it starts up and a rad11ended script when it finishes. These scripts may be modified to send an email to your IT department or perform some other duty as needed. The default scripts don't perform any activity.

Do not intermingle different types of records into the same data file. For example, course data should be in one file, UCX data should be another, equivalence records in another and student data in a separate file. In addition, do not mix different file types in one RAD11M01 like this:

```
COURSE  
UCX  
EQUIV  
STUDENT
```

Instead run each type of data in separate runs of RAD11 like this:

```
$ RAD11JOB COURSE  
$ RAD11JOB UCX  
$ RAD11JOB EQUIV  
$ RAD11JOB STUDENT
```

In these examples, we are specifying the actual name of the files in the admin/data directory instead of the RAD11M01 file. You can actually refer to the bridge file directly instead of specifying the RAD11M01. Above the COURSE file has all of the course BIF records, the UCX file has all of the BIF UCX records, and so on.

RAD3x - Banner Extract and Bridge

Updated: March 25, 2022

The RAD3x jobs are used to extract data from the Banner database and bridge it into the Degree Works database.

For more information, see [Banner Integration](#).

RAD30 - Banner Student Extract and Bridge

Updated: March 24, 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.

Access to run RAD30 in Transit is granted with the TRRAD30 Shepherd key.

RAD30 generates new audits for students with data changes. However, if the **Force new audits** check box is selected, new audits will be generated for all students, even if there are no data changes.

You can select the students to bridge either by specifying a list of student IDs or by using the default SQL in the integration.banner.extract.student.sql.daily setting.

Be sure your UCX-CFG020 BANNER settings are correct before running the extract.

Review and configure the following settings:

- parser.batch.notifications.toEmail
- transit.rad30.workerCount

RAD32 - Banner Applicant Extract and Bridge

Updated: March 24, 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.

Access to run RAD32 in Transit is granted with the TRRAD32 Shepherd key.

The job generates new audits for applicants with data changes.

You can select the applicants to bridge either by specifying a list of applicant IDs or by using the default SQL file in the integration.banner.extract.applicant.sql.daily setting.

Be sure your UCX-CFG020 BANNER settings are correct before running the extract.

RAD33 - Banner Non-Student Extract and Bridge

Updated: September 29, 2023

You can use RAD33 to run the Banner extract for non-students, which creates access records for staff who require access to Degree Works.

Access to run RAD33 in Transit is granted with the TRRAD33 Shepherd key.

You must select the user class of the users you are extracting. You can use the **Show in RAD33** flag in UCX-AUD012 User Class Codes to configure which user classes display in this drop-down.

You can select the staff IDs to bridge either by specifying a list of IDs or use the default SQL file that corresponds to the user class specified in the jobs. For example, if ADV is the selected user class, the setting integration.banner.extract.adv.sql.daily will be used to select the set of advisors. If a daily.sql setting does not exist for the selected user class, you need to create the new setting in Controller.

Be sure your UCX-CFG020 BANNER settings are correct before running the extract.

RAD34 - Banner Course Extract

Updated: March 25, 2022

You can use RAD34 to run the Banner COURSE extract, which extracts all current courses from your course catalog in Banner to Degree Works.

Access to run RAD34 in Transit is granted with the TRRAD34 Shepherd key.

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.

RAD35 - Banner Curriculum Rules Extract

Updated: March 24, 2023

You can use RAD35 to run the Banner CURRRULE extract, which bridges curriculum rule data from Banner to Degree Works.

Access to run RAD35 in Transit is granted with the TRRAD35 Shepherd key.

This job first deletes the old curriculum rules and then re-adds the new curriculum rules. To manage how curriculum rule records for rules that have been deactivated for a period of time and then reactivated are bridged into Degree Works, review the integration.banner.extract.curriculumRules.multipleRanges.enabled Shepherd setting in the [integration.banner](#) topic.

Note: You can use the viewrules command under ADMIN in Transit to extract and view the curriculum rules bridged to Degree Works.

RAD36 - Banner Validation Table Extract (UCX)

Updated: March 25, 2022

You can use RAD36 to run the Banner UCX extract, which creates Degree Works validation tables using data from the Banner validation tables.

Access to run RAD36 in Transit is granted with the TRRAD36 Shepherd key.

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).

RAD37 - Banner Transfer School Extract (ETS)

Updated: March 25, 2022

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.

Access to run RAD37 in Transit is granted with the TRRAD37 Shepherd key.

RAD37 only adds or updates rad_ets_mst records; it does not delete.

RAD38 - Banner Equivalencies Extract

Updated: March 25, 2022

You can use RAD38 to run the Banner EQUIV extract, which bridges historic courses and their current equivalent courses from Banner to Degree Works.

Access to run RAD38 in Transit is granted with the TRRAD38 Shepherd key.

The job first deletes both the dap_eqv_crs_mst and UCX-CFG070 and re-adds all equivalencies.

RAD39 - Banner Transfer Equivalency Extract (Mappings)

Updated: March 25, 2022

You can use RAD39 to run the Banner MAPPINGS extract, which bridges transfer equivalent rules from Banner to Degree Works.

Access to run RAD39 in Transit is granted with the TRRAD39 Shepherd key.

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 integration.banner.extract.equiv.* Shepherd settings are correct before running the extract.

RAD40 - Student Data Delete Processor

Updated: March 25, 2022

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.

Access to run RAD40 in Transit is granted with the TRRAD40 Shepherd key.

The default action of RAD40 is to delete only the student's RAD and SHP records, leaving the DAP and SEP data untouched. This is useful if you need to re-extract the student's data from the SIS but do not want to lose the student's historical audits, exceptions, and plans. A rad_ids file of all the deleted students will be created as an artifact when the job completes. This can be downloaded and used to run the student extract.

If desired, you can 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.

SCR02 - Find blocks where this COURSE is referenced

Updated: March 25, 2022

SCR02 identifies blocks that reference specified courses.

Access to run SCR02 in Transit is granted with the TRSCR02 Shepherd key.

SCR02 job makes use of the dapreqcrs script.

When running the job you need to enter a discipline and optionally enter a course number.

When the job runs the output will be sent to a Report file. The report will list the blocks that reference this course.

When UCX-CFG020 DAP13 Process equivalent courses is Y, SCR02 will not find blocks that are scribed with a course ID that has been renumbered (CFG070) or split (CFG078). This is because SCR02 does not search the actual block text in the dap_req_block table; it searches the dap_req_crs_dtl view which stores the parsed equivalent of the course ID in the text. For example, if BIOL 4100 is scribed in the block but it has an equivalency to BIOL 4103 then this report will find that BIOL 4103 is in the block but it will not report that BIOL 4100 is in the block.

SCR05 - List blocks changed by date range

Updated: March 25, 2022

SCR05 generates a list of all of the blocks you have in Degree Works.

Access to run SCR05 in Transit is granted with the TRSCR05 Shepherd key.

SCR05 generates a simple report listing all of your blocks – sorted by the block type and value. This job makes use of the dapreqlist script.

You can optionally specify a date range for changes made in Scribe to the blocks. You can specify the start date and leave the end date blank or simply click on "Today" for the ending date. To set

both dates to blank so that no filtering is done on the modify date click on Blank on both fields. You may also click Set As Defaults to tell Transit to use these settings the next time you attempt to run the report. If you do specify a date range the report will additionally list the ModifyDate for each block selected.

SCR06 - List block primary and secondary tags

Updated: March 25, 2022

SCR06 generates a list of all of the blocks showing their primary and secondary tags. In doing this you may easily see which blocks use a Program or Student ID secondary tag for example.

Access to run SCR06 in Transit is granted with the TRSCR06 Shepherd key.

You may use the SCR06 job in Transit to get a report listing all of your blocks – sorted by the block type and value. Being there is so much data delivered in the report it is best to copy the data into Excel for sorting and viewing. This job makes use of the dapreq2ndlist script.

No questions appear in Transit when SCR06 is chosen.

The report always lists all blocks.

The secondary tags listed are School, Degree, College, Major1, Major2, Conc, Minor, Libl, Spec, Program, and Student ID, and can be viewed by using the horizontal scroll bar.

SCR07 - List block text from Scribe

Updated: March 25, 2022

SCR07 creates a report of the block text for your Scribe blocks based on a particular catalog year so you may save them as a backup or view on your PC in bulk.

Access to run SCR07 in Transit is granted with the TRSCR07 Shepherd key.

You may use the SCR07 job in Transit to get a report listing all of your blocks with all tags and text lines – sorted by the block type value. This job makes use of the dapblocksget script.

Transit prompts the user for a catalog year and a block type. This tool will find blocks whose catalog year range encompasses the catalog year specified for the block type specified. The script checks to see if the starting catalog-year is less than or equal to this value and whose ending catalog-year is greater than or equal to this value. Normally users should select the current catalog year because they don't want old blocks.

The resulting report file may be very large and thus will take a long time to open. Because the resulting report can be very large, the block type is a required field to help reduce the size of any one report listing. You may want to run one report for DEGREE blocks, another for MAJOR blocks and so on.

SCR08 - List LOG entries from Scribe text

Updated: March 25, 2022

SCR08 generates a list of all of the LOG entries in your Scribe text so you may review what has changed in your blocks.

Access to run SCR08 in Transit is granted with the TRSCR08 Shepherd key.

You may use the SCR08 job in Transit to get a report listing all of your blocks with LOG text lines – sorted by the block type and value and text sequence number. This job makes use of the dapreqgrep script but it only looks for lines with LOG in column 1-3 of the text line.

No questions appear in Transit when SCR08 is chosen.

The report always lists all text lines from requirement blocks containing LOG starting in column 1.

SCR09 - List TODO entries from Scribe text

Updated: March 25, 2022

SCR09 generates a list of all of the TODO entries in your Scribe text so you may review what work still needs to be done.

Access to run SCR09 in Transit is granted with the TRSCR09 Shepherd key.

You may use the SCR09 job in Transit to get a report listing all of your blocks with TODO text lines – sorted by the block type and value and text sequence number. This job makes use of the dapreqgrep script but it only looks for lines with TODO in column 1-4 of the text line.

No questions appear in Transit when SCR09 is chosen.

The report always lists all text lines from requirement blocks containing TODO starting in column 1.

SCR10 - Find blocks where this text is referenced

Updated: March 25, 2022

SCR10 generates a list of the blocks that make use of remarks or proxy-advice or use certain values.

Access to run SCR10 in Transit is granted with the TRSCR10 Shepherd key.

You can do this through the SCR10 job – this job makes use of the dapreqgrep script.

When running the job you need to enter a text string. Remember that this is similar to doing a Find in Notepad with the “Match case” option selected; the search is case sensitive. If you type “Remark” it will not find “remark” or “REMARK” or any other instances of the remarks.

When the job runs, the output will be sent to a Report file. The report will list the text lines containing the specified string in addition to the block information.

SCR11 - Find and Replace block text

Updated: March 25, 2022

SCR11 allows you to find any string of text up to 30 characters in blocks and replace it with a new string.

Access to run SCR11 in Transit is granted with the TRSCR11 Shepherd key.

The new text can be shorter or longer than the old text but the new text is also limited to 30 characters. This report makes use of the dapreqsearchnreplace script.

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. Just as with SCR10, for example, if you search for "zebra" it will not find any lines with "Zebra" or "ZEBRA" because SCR11 is also case sensitive. You may have to run SCR11 again for "Zebra" and then again for "ZEBRA" if you want to replace all "zebras" with something else.

This report searches for the text exactly as you see it in Scribe. This report should not be used 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 HIST 106 would not be found. The find-and-replace does not understand the Scribe language and does not do any more than what the find-and-replace does in Notepad.

Because this is such a powerful tool, it is possible you will 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, just 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.

SCR91 - Test Banner Prerequisite Checker Service

Updated: March 25, 2022

Using this form, you can test the data Degree Works will return given mock Banner registration data.

Access to run SCR91 in Transit is granted with the TRSCR91 Shepherd key.

At the very least you need to supply the student's SPRIDEN ID (not the PIDM) and the course discipline/subject and course number. To correctly mimic a registration request you should send the CRN and term values but they are not required using this testing tool. The student's level and degree are optional but maybe needed to give correct results if the student has multiple degrees on their academic record.

If you recently made changes to this student's record in Banner you may want to refresh the student before testing here in Transit to make sure Degree Works has the most up-to-date information.

In the log file information like this will be reported based on the test that was processed. There "here is why" messages are taken from the ProxyAdvice on the REQUISITE block found.

```
Test results for Student 300000001; Course ENGL 222; CRN 12345 Term  
20094 ...
```

The requisite requirements were not satisfied - here is why:

```
Engl 100 is a prereq for Engl 222  
You did not meet this requirement.
```

In Transit you can also view the Diagnostics XML to help understand the results of the prereq audit. If you don't want to view the raw XML you may choose to send this XML to the Action Line with your question. However, for this Diagnostics XML to be generated you must launch the SCR91 job with debugging enabled.

SCR92 - Test Banner Prerequisite Description Service

Updated: March 25, 2022

Using this form you can test the data Degree Works will return given a mock Banner catalog request.

Access to run SCR92 in Transit is granted with the TRSCR92 Shepherd key.

At the very least you need to supply the student's SPRIDEN ID (not the PIDM) and the course discipline/subject and course number. The CRN and term values are optional as often this information is not known. The student's level and degree are optional but maybe needed to give correct results if the student has multiple degrees on their academic record.

If you recently made changes to this student's record in Banner you may want to refresh the student before testing here in Transit to make sure Degree Works has the most up-to-date information.

In the log file information like this will be reported based on the test that was processed. There "prereq description" messages are taken from the Remarks on the REQUISITE block found. The "prereq courses" are the list of courses found in the REQUISITE block in all rules and qualifiers.

```
The prereq description = .....
```

```
This is the 1st line of the remark for block RA001464.
```

```
This is the 2nd line - the course is MATH 101
```

```
The prereq courses = .....
```

```
ACCT 111
```

MATH 100
MATH 099

SCR93 - Report by CATALOG

Updated: March 25, 2022

All SCBCRSE records for the specified term are selected that have the Prereq-in-Degree Works flag enabled.

Access to run SCR93 in Transit is granted with the TRSCR93 Shepherd key.

The subject+number value for each record is then looked up in Degree Works to see if a REQUISITE block exists. All matching blocks for the REQUISITE block are displayed. If no match is found an error is displayed instead.

You can use the check box in Transit to suppress non-error conditions. In doing this the report will only report on those records that may need to be fixed or reviewed further.

You can choose to show the Scribe block text for each of the blocks found but this will make the report listing much longer. In addition, this block text flag is ignored if you choose to only show error conditions.

SCR94 - Report by SCHEDULE

Updated: March 25, 2022

All SSBSECT records for the specified term are selected that have the Prereq-in-Degree Works flag enabled.

Access to run SCR94 in Transit is granted with the TRSCR94 Shepherd key.

The term+CRN value for each record is then looked up in Degree Works to see if a REQUISITE block exists. All matching blocks for the REQUISITE block are displayed. If no match is found an error is displayed instead.

You can use the check box in Transit to suppress non-error conditions. In doing this the report will only report on those records that may need to be fixed or reviewed further.

You can choose to show the Scribe block text for each of the blocks found but this will make the report listing much longer. In addition, this block text flag is ignored if you choose to only show error conditions.

SCR95 - Report by REQUISITE Block

Updated: March 25, 2022

If the first 6-bytes are valid in UCX-STU016 we know it is a valid term and thus the REQUISITE value is a term+CRN value. The term+CRN is looked up in SSBSECT to see if the Degree Works prereq flag is turned on.

Access to run SCR95 in Transit is granted with the TRSCR95 Shepherd key.

If no record is found or if the flag is disabled, an error is displayed for the REQUISITE block.

If the REQUISITE block is a subject+number, the SCBCRSE table is looked up to see if the course is enabled. If no record is found or if the flag is disabled, an error is displayed for the REQUISITE block.

You can use the check box in Transit to suppress non-error conditions. The report will only report on those records that may need to be fixed or reviewed further.

UCX01 - UCX Records Modified

Updated: March 25, 2022

UCX01 generates a list of all UCX records that were modified between a specific date range.

Access to run UCX01 in Transit is granted with the TRUCX01 Shepherd key.

The report generated will list the UCX table number, UCX key, and modify date. For example:

UCX_AUD034	RG	09-MAR-11
UCX_AUD034	ST	09-MAR-11
UCX_AUD034	TM	09-MAR-11
UCX_AUD047	1	09-MAR-11
UCX_AUD047	2	09-MAR-11
UCX_AUD047	3	09-MAR-11
UCX_AUD047	4	09-MAR-11
UCX_AUD047	5	09-MAR-11
UCX_AUD047	6	09-MAR-11
UCX_CFG020	BANNER	15-APR-11
UCX_CFG020	DAP13	06-JAN-11
UCX_CFG020	DAP14	15-APR-11
UCX_CFG020	PLANNER	25-MAR-11

Schema

Updated: September 29, 2023

The Schema text file provides the structure of the Degree Works database.

Click Attachments  on this page to download the file.