ellucian.

Degree Works

Banner Considerations

Technical Guide

Release 5.0.3
December 2019

Notices

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

Banner Attributes

Banner Class Attributes

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

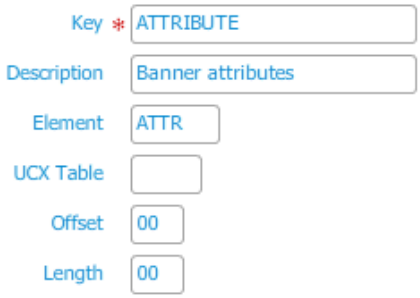
Historic class attributes are stored in two tables: SHRATTR and SHRATTC. A UCX-CFG020 BANNER flag “Always Process SHRATTC” controls whether attributes from SHRATTC are extracted if attributes from both tables exist. SHRATTR class attributes are processed first. If any SHRATTR attributes are found for 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”. See the UCX-SCR044 screen below.



Transfer Classes Applied to Program

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 which have been applied to a Program in Banner. Such classes are identified by linking the Banner Transfer Equivalence record, SHRTRCE, to the Transfer Course Degree Applied table SHRTRCD where SHRTRCD_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 flag "Transfer Program Attr" 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

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) which have been applied to a Program in Banner. To enable this feature, set the UCX-CFG020BANNER flag "Curr/Hist Program Attr" 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 a few rad_attr_dtl records for a given student:

rad_attr_key	rad_attr_code	rad_attr_value
SWK430-016	ATTRIBUTE	1501SH
LAW211-017	ATTRIBUTE	1501SH
WEL116-018	ATTRIBUTE	1501SH
HRM540-019	ATTRIBUTE	2001AA
LAW516-020	ATTRIBUTE	2001AA
POL01C-021	ATTRIBUTE	1501SW
POL01C-022	ATTRIBUTE	1501SW
PSY01C-023	ATTRIBUTE	1501SW

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 “1501SW” program could be scribed as:

```
MaxClasses 0 in @ (With Attribute <> 1501SW )
```

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 “1160TH” 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 <> 1160TH)

;
```

Banner Course Attributes

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=”.

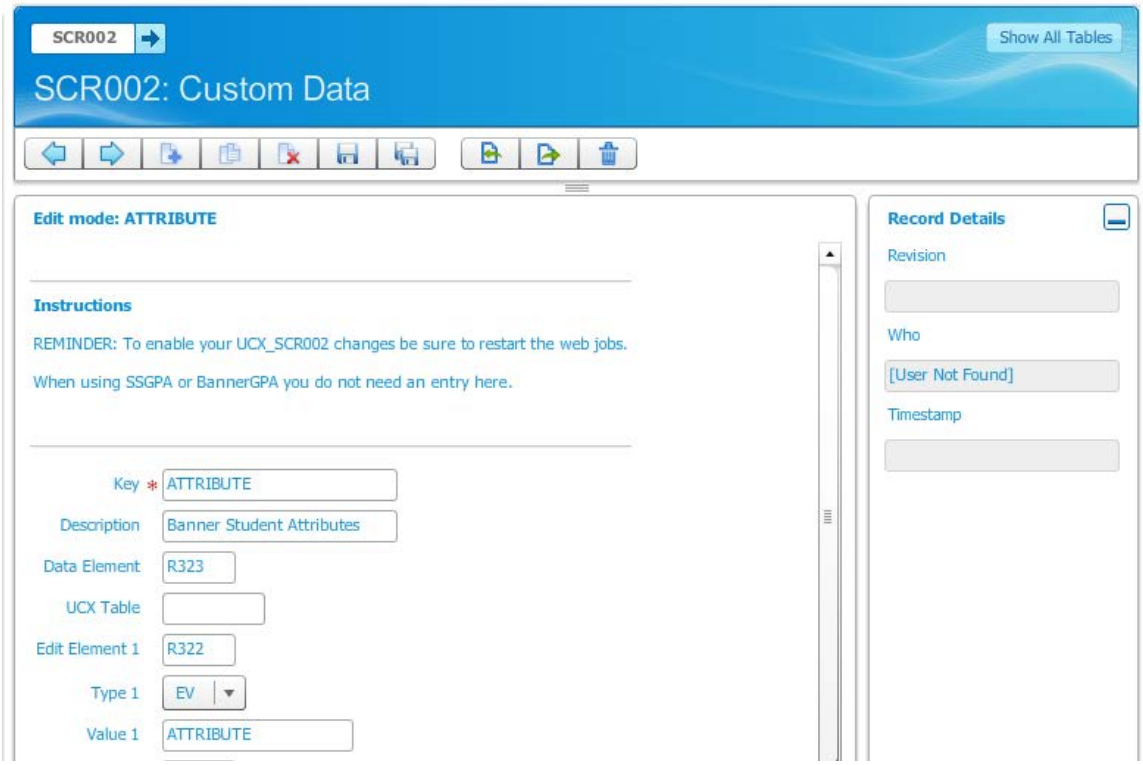
Banner Student Attributes

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” 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.



Banner Setup Checklist

For a more detailed explanation of the items in the Setup Checklist, see the *Degree Works Data Extract* Section of this document.

UCX

Review and change UCX settings using Controller (see the *Degree Works Configuration Technical Guide* for more information).

UCX	Action
UCX-SCR001 STATUS	change Description to “Student Type”
UCX-SCR001 SCHOOL	change Description to “Level”
UCX-SCR001 LEVEL	change Description to “Student Class Level”
UCX-CFG020 BANNER	Be sure the Banner Site flag is Y. You can turn on the Search in Banner flag now but it is suggested that you perform all searches against the bridged data at first while getting everything setup. Leave this flag as ‘N’ for now but switch it to ‘Y’ later in your implementation. See the documentation regarding the other flags on this record.
UCX-CFG020 REFRESH	Set the flags to appropriately configure the refresh parameters.
UCX-CFG020 SEARCH	Set the Show flags to ‘N’ for items you are not using. Show Specialization and Show Liberal Learning should be ‘N’ since these are not used in Banner. Note - School in Degree Works is actually the same as Level in Banner. See the documentation regarding the other flags on this record.
UCX-STU352 DISCIPLINE STATUS	Set the Discipline Status flag to ‘I’ for those disciplines that are Inactive. If a Discipline is Inactive then the class (current, historic and transfer) will be skipped and NOT bridged to the Degree Works rad_class_dtl.

UCX-STU385 IN PROGRESS Flag	The default In-Progress flag is 'N' for historical classes found in SHRTCKN. If an historical grade combination in UCX-STU385 (key = School/GradeType/Grade) is considered 'In-Progress' then set this In-Progress flag to 'Y' so that the rad_inprog_flag on the rad_class_dtl gets built appropriately with a 'Y' value.
UCX-STU385 OVERRIDE Flags	There is a "master" Override flag and 8 Override values. See the Technical Guide UCX documentation for details on how each of these Override fields is used.
UCX-STU385 Transfer Repeat Flags	There are three Transfer Repeat fields that may be used to override the standard Transfer Repeat Pointer and Repeat Policy used for repeated transfer classes. See the Technical Guide UCX documentation for details on how to load these fields.
UCX-BAN080 Custom Data	For special "if" statements and for any special data desired on the worksheet setup UCX-BAN080. See the Technical Guide UCX documentation for details on how to setup UCX-BAN080.

Banner Data Extracts

Selecting People

Modify the listed SQL scripts as required, to select the desired students, applicants and advisors you want to bridge to Degree Works. The standard files listed here are delivered to *app/sql* and must be copied to *local/sql*.

```
local/sql/bannerstudents.sql
local/sql/bannerapplicants.sql
local/sql/banneradvisors.sql
```

The queries in these files should only select the IDs of the individuals you want to bridge into Degree Works. When the student extract is run, the `bannerstudents.sql` query is executed and the resulting IDs are then processed by the extract. You should test the select statements that you use in these files through sqlplus, to verify that the count you get is as expected.

Deleting People

Create an ID file of the individuals that should be deleted from Degree Works and run the extract. The ID file must reside in the *local/sql* directory.

```
$ bannerextract deleteid deleteFile.ids
```

Selecting UCX Entries

You can run RAD36 in Transit to populate the UCX tables but it will run the extract on all validation tables; there is no way to select a subset of tables.

If selected UCX tables are to be re-extracted from Banner to Degree Works, on the Classic server, create a UCX file containing a list of Degree Works tables. For example, write one table per line:

```
STU356
STU385
STU560
```

Add a ".ucx" extension to this file in the local/sql directory. The actual file name can be any valid Unix file name. However, it MUST have the ".ucx" extension. For example, the file "bantables.ucx" may be created as follows: local/sql/bantables.ucx. The banner extract command would be:

```
bannerextract ucx bantables.ucx
```

When the command above is executed then only the Degree Works tables listed in that file will be re-loaded with Banner data. Make sure to check the setting for the UCX-CFG020 RADBRIDGE "Add UCX Entries Only" flag. Make sure it is set to "N" if ALL entries are to be reloaded from Banner. If only NEW entries are to be loaded from Banner make sure this UCX-CFG020 flag is set to "Y". It is recommended that this flag always be set to "Y".

Selecting Different Records for Your Institution

Modify the `integration.banner.extract.config` setting as needed to select a different set of records than those Degree Works is selecting by default. Make sure to review this setting and make all appropriate changes for your site.

If your Banner database has been configured for MEP (Multiple-Entity Processing), see instructions in the *MEP (Multiple-Entity Processing) in Banner* section of this document.

Nightly extracts

It is recommended to run a nightly student extract to ensure your student data is current. Set up the "launchjob" script in cron (see the *Transit Administration Guide* for more information on running the launchjob script via cron). For example:

```
launchjob $ADMIN_HOME/myjobs/rad30.standardSql.json
launchjob $ADMIN_HOME/myjobs/rad32.standardSql.json
launchjob $ADMIN_HOME/myjobs/rad31.standardSql.json
```

It is recommended that you only setup the student, applicant and advisor extracts to run each night in cron. It is best to run the other extracts on an as-needed basis in Transit – or run once a month/term via cron. These are the other extracts:

RAD33 Banner Staff extract
RAD34 Banner Course extract
RAD35 Banner Curriculum Rules extract
RAD36 Banner Validation extract (UCX)
RAD37 Banner Transfer School extract (ETS)
RAD38 Banner Equivalencies extract
RAD39 Banner Transfer Equivalency extract (Mappings)

Post extract

After UCX has been extracted from Banner review each of these tables and setup as needed.

Be sure not to run UCX extract again - unless the UCX-CFG020 RADBRIDGE *Add UCX Entries Only* is set to Y so that none of the records you changed will be deleted– only new records in Banner will be added to the UCX.

Table	Fields to setup
UCX-AUD027	Filter fields
UCX-STU016	Planner and Show in Transfer Equivalency Self-Service flags and the Financial Aid Year and Term Type fields
UCX-STU035	Planner flag
UCX-STU307	Short degree field
UCX-STU346	Calendar codes used in Transfer Equivalency Self-Service
UCX-STU352	Discipline Status ("I" – Inactive)
UCX-STU385	GPA Calc flag

UCX-STU385	In-Progress flag
UCX-STU385	Override flags
UCX-STU385	Override Transfer Repeat flag/fields

Degree Works Data Extract

Configuration and Installation

The Degree Works Data Extract process for Banner clients was developed using Oracle's Pro*C embedded SQL tools. These programs execute on the Degree Works application server – Linux, UNIX, etc. Ellucian has developed several extract programs to not only extract the required data but to also convert the data to the Bridge Interface Format (BIF) so that it is ready to load into Degree Works.

Verify or perform the following steps to configure your Degree Works Server to perform the Banner Degree Works Data Extract:

STEP 1 – Banner Database login

Ensure that the DB_LOGIN_BANNER environment variable is set correctly:

```
$ env | grep DB_LOGIN_BANNER
```

If this does not show the correct Banner database login you need to edit this variable in `$DGWBASE/dwenv.config`, run setdbpasswords and log back into your host session. Here are examples of how to set this variable:

```
export DB_LOGIN_BANNER=userid
export DB_LOGIN_BANNER=userid@some.other.machine.edu
```

If the database is on a remote machine you may use the "@" option to specify where the database resides. You must have the remote database server configured in Oracle's tnsnames.ora file.

To set the password for your Banner user run setdbpasswords:

```
setdbpasswords --sispassword thebnrpassword
```

Ensure that the ORACLE_SID environment variable is set correctly:

```
$ env | grep ORACLE_SID
```

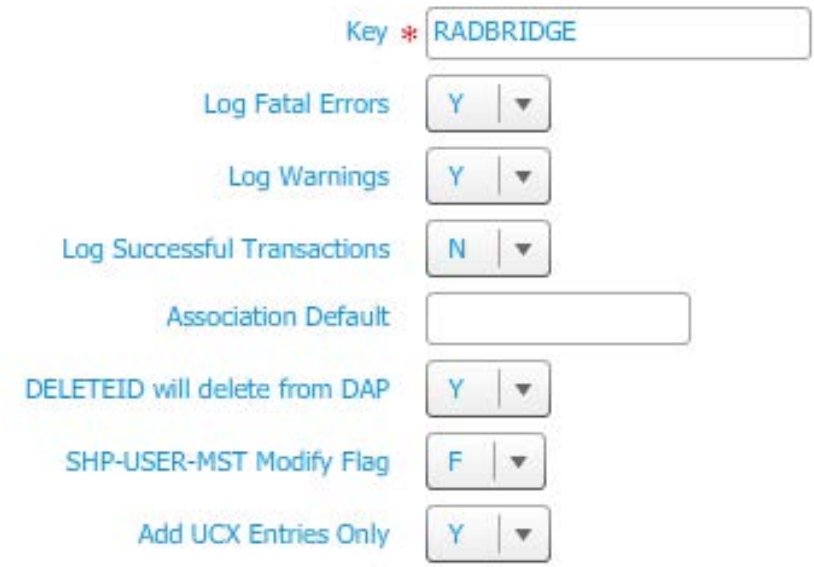
If this is not set correctly, review your setup of `$DGWBASE/dwenv.config` – see config_SetupMenuItems.

Changes required for MEP (Multiple-Entity Processing)

If you are connecting to a Banner database which has been configured as MEP (Multi-Entity Processing), you must create a DB_LOGIN_BANNER user for each MEP entity established in Degree Works. See instructions in the *MEP (Multiple-Entity Processing) in Banner* section of this document.

STEP 2 – Setup UCX-CFG020 RADBRIDGE

Review all settings BEFORE running the UCX extract (see the *Degree Works Configuration Technical Guide* for more information). Specifically, make sure the Add UCX Entries Only is set to Y before running the UCX extract (RAD36).



STEP 3 – Extract UCX and courses

Before attempting to bridge student data from Banner into Degree Works, it is recommended to first extract Validation codes (UCX) and course information. These data are required before the Scribe tool can be used to add or modify requirement blocks in your Degree Works database.
In Transit, launch these jobs:

RAD34 course extract
RAD36 UCX extract

Be sure the review the log files to be sure the jobs ran successfully.

STEP 4 – Setup UCX-CFG020 BANNER

Review all settings BEFORE running the student extract.

Repeat Policy

Although you may use repeat policies 1-6 telling Degree Works to use the class with the best grade or the most recently taken class etc it is recommended that you set the Repeat Policy flags to ‘B’ for all Repeat Indicators.

When ‘B’ is used this behavior will occur:

- **Excluded** classes will end up in the Insufficient section of the audit but they will not affect the overall GPA or credits.

- **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 degree.
- **Included** classes will apply to rules as normal classes affecting the GPA and total credits.

By using the 'B' setting you are telling Degree Works to handle each class based on the indicator without regard to grades or terms taken as the decision about which classes should be counted and not counted has already been made and recorded using the indicator. When 'B' is in use the normal Degree Works repeat logic is skipped simplifying the auditing process greatly.

The six different Repeat Policy fields should be set to the same value. Only in rare scenarios does it make sense for these to be different.

STEP 5 – Setup SQL select files

Review, modify and/or create the ".sql" files in the local/sql directory using your Degree Works user before launching the RAD3x processors. Only those sql files that are localized at your institution need to be in the /local/sql directory. It is recommended that you copy the sql statement into sqlplus and make sure it executes properly. Also, make sure that the number of ID codes selected by the sql statement is what is expected and that they are the correct ID codes. In addition, ensure that you use SELECT DISTINCT(SPRIDEN_ID) when selecting students so that the same ID does not appear more than once in the results.

Note: If a Banner table is used in one of the sql files identified below that is not found in the "Required Access to Banner Tables" list located at the end of this document make sure to have your database administrator add the appropriate access. Otherwise the ID codes will not be extracted correctly and the banner extract will fail.

Review and modify as needed the file used to select the students you want bridged to Degree Works:

```
local/sql/bannerstudents.sql
```

Review and modify as needed the file used to select the applicants you want bridged to Degree Works:

```
local/sql/bannerapplicants.sql
```

Review and modify as needed the file used to select the advisors you want bridged to Degree Works:

```
local/sql/banneradvisors.sql
```

STEP 6 – Setup integration.banner.extract.config setting

Using Controller, review and modify as needed the `integration.banner.extract.config` setting to select a different set of records based on your particular needs. The SQL FROM/WHERE clauses for every Banner table used by the Banner extract programs are included in this configuration file.

The config should look like the example you see here – the FROM/WHERE text for your school may need to be changed.

```
# SGBSTDN must be a; AND is required at the end of the WHERE
SGBSTDN-from:  FROM SGBSTDN a
SGBSTDN-where: WHERE a.SGBSTDN_TERM_CODE_EFF =
SGBSTDN-where:       (SELECT MAX(b.SGBSTDN_TERM_CODE_EFF)
SGBSTDN-where:        FROM SGBSTDN b WHERE b.SGBSTDN_PIDM = a.SGBSTDN_PIDM)
AND
#                    a.SGBSTDN_PIDM = <students-pidm>
```

A special set of records with keys of “CALCFCN-from:” and “CALCFCN-where:” have been created for the special Banner function: “F_CLASS_CALC_FCN”. This function call is made by the banner student extract using the PIDM, LEVL_CODE and TERM_CODE specified in this setting to generate the student’s Class Standing code that is loaded into the rad_stu_level on the rad_goal_dtl. A default set of special records with a key of “CALCFCN” are included in this configuration file and are loaded with the FROM and WHERE clauses from the SGBSTDN default entry. Change this set of “CALCFCN” records as appropriate for your site.

Here is an example of the SORLCUR/SORLFOS entries in this configuration file:

```
################### SORLCUR - LEARNER CURRICULUM (ban40) ###############

# SORLCUR must be a; AND is required at the end of the WHERE
SORLCUR-from:  FROM SORLCUR a
SORLCUR-where: WHERE a.SORLCUR_CACT_CODE = 'ACTIVE'
SORLCUR-where:   AND a.SORLCUR_SEQNO     =
SORLCUR-where:    (SELECT MAX(b.SORLCUR_SEQNO) FROM SORLCUR b
SORLCUR-where:      WHERE b.SORLCUR_PIDM        = a.SORLCUR_PIDM
SORLCUR-where:        AND b.SORLCUR_PRIORITY_NO = a.SORLCUR_PRIORITY_NO
SORLCUR-where:        AND b.SORLCUR_LMOD_CODE   = 'LEARNER')
SORLCUR-where:   AND
#                    a.SORLCUR_PIDM = <students-pidm>

################### SORLFOS - STUDENT FIELD OF STUDY (ban40) ###########

# SORLFOS must be a; AND is required at the end of the WHERE
SORLFOS-from:  FROM SORLFOS a, SORLCUR b
SORLFOS-where: WHERE b.SORLCUR_CACT_CODE  = 'ACTIVE'
SORLFOS-where:   AND b.SORLCUR_SEQNO =
SORLFOS-where:    (SELECT MAX(c.SORLCUR_SEQNO) FROM SORLCUR c
SORLFOS-where:      WHERE c.SORLCUR_PIDM        = b.SORLCUR_PIDM
SORLFOS-where:        AND c.SORLCUR_PRIORITY_NO = b.SORLCUR_PRIORITY_NO
SORLFOS-where:        AND c.SORLCUR_LMOD_CODE   = 'LEARNER')
SORLFOS-where:   AND a.SORLFOS_CSTS_CODE  = 'INPROGRESS'
SORLFOS-where:   AND a.SORLFOS_CACT_CODE  = 'ACTIVE'
SORLFOS-where:   AND a.SORLFOS_PIDM       = b.SORLCUR_PIDM
```

```
SORLFOS-where:   AND a.SORLFOS_LCUR_SEQNO = b.SORLCUR_SEQNO
SORLFOS-where:   AND
#                    a.SORLFOS_PIDM = <students-pidm>
```

Batch Data Extract Process

The batch extract programs are executed one of two ways: a script named *launchjob* run via cron**,** or via Transit's RAD3x family of jobs. When extracting students Degree Works automatically runs a new degree audit for each of the students that has changed data since the last time they were extracted. In doing this you will be sure that each student's degree audit reflects any changes made to their student record.

Student and/or applicant academic data is required for Degree Works to generate a Degree Audit. Ellucian extracts student data, applicant data, advisor information, your course catalog and the list of transfer institutions from the Banner database and stores this data in the Repository for Audit Data (RAD) database tables. This procedure is known as the Degree Works Bridge process and consists of the following data extraction processes:

Students – active students' academic and basic biographic data are extracted based on the SQL specified in the `integration.banner.extract.config` setting. Student data can also be extracted individually by the SPRIDEN ID.

Applicants – admissions applicants academic and basic biographic data may be extracted based on the SQL specified in the `integration.banner.extract.config` setting. However, only unique combinations of the Level (School) and Degree may be extracted (this assumes the UCX-CFG020 BANNER configuration flags are set appropriately and/or the APPLICANT data extract is performed). Applicant data may also be extracted individually by the SPRIDEN ID.

Advisors – access records are created for advisors who require access to Degree Works.

Staff – access records are created for staff who require access to Degree Works.

Course Catalog – all current courses from your course catalog.

Course Equivalents – historic courses and their current equivalent courses.

Curriculum Rules – Curriculum Rules used by What-if audits.

ETS – Transfer institution ETS codes, names and identification data.

UCX Validation Codes – Validation tables consisting of data from Banner STV tables.

Mappings – Transfer Articulation Mappings for import into Transfer Equivalency Admin and Transfer Equivalency Self-Service. This mapping information is also used by a Course Link display.

The ETS extract must be run before the mappings extract.

bannerextract and launchjob

The bannerextract script is being replaced with the newer *launchjob* script.. However, there are some cases when the bannerextract script may still need to be used; those cases are noted in this document.

Note that launchjob requires a parameter file in JSON format. The launchjob script can be used to schedule the extract to run using cron or at or it can be run directly at any time during the day as needed. When running any of the extracts manually you should use Transit; the launchjob script is intended to be used when scheduling jobs in cron.

Warning

Ensure that you are not running bannerextract from the local/sql directory.

After running *launchjob* the resulting log file can be viewed in Transit. After running *bannerextract* the log file can be viewed in $ADMIN_HOME/dgwspool.

Student

As mentioned in the *Transit Administration Guide*, the sample JSON job files in app/samples should be copied to a new $ADMIN_HOME/myjobs directory that you create. You then modify the files in your myjobs directory as needed.

To run the student extract using the default `bannerstudents.sql` file in the local/sql directory your JSON file needs to be setup like this:

```
{
      "name": "RAD30",
      "parameters": [{
            "type": "SELECTDEFAULTQUERY",
            "usingDefaultQuery": "true"
      }]
}
```

You then specify your JSON file when running launchjob:

```
$ launchjob $ADMIN_HOME/myjobs/rad30.defaultStudentSql.json
```

You can also specify a list of student IDs to be extracted. If you are using a JSON file it would look like this:

```
{
      "name": "RAD30",
      "parameters": [{
         "type": "SELECTIDLIST",
         "ids": [
           "studentId1",
           "studentId2",
           "studentId3",
           "etc"
    ]
      }, {
```

```
            "type": "QUESTIONS",
            "answers": []
        }]
}
```

However, if you have a file of student IDs then it is probably easier to use Transit to import the IDs instead of creating a JSON file like the one above. You would only need to create a JSON file if, for some reason, you needed to load the IDs in your cron job or if you could not access Transit.

You would then specify your JSON file containing the IDs when running launchjob. For example:

```
$ launchjob $ADMIN_HOME/myjobs/rad30.myids.json
```

To extract a single student ID it is easiest to use Transit or the refresh button on the dashboard.

Sometimes a need arises to force a student to be bridged from Banner into Degree Works thereby ignoring or overriding the hash value that is normally checked. An environment variable can be set to force this extract. This variable must be set at the command line. The extract must also be run from the command line. Once the user does not want to force extracting, the user can either log off or reset the environment variable. Make sure the command is typed exactly as below (case does matter). Only extracts performed from the command line by this user will be affected by this environment variable. Transit extracts will not be affected. Extracts done during a cron job will not be affected unless explicitly denoted in the cron job.

For Banner sites the commands are:

```
$ export RAD11FORCE=ALL
$ bannerextract student listofstudents.ids
```

Applicant

To run the applicant extract using the default `bannerapplicants.sql` file in the local/sql directory your JSON file needs to be setup like the one above for students using the default SQL file except change the “name” to “**RAD32**”.

You then specify your JSON file when running launchjob. For example:

```
$ launchjob $ADMIN_HOME/myjobs/rad32.defaultApplicantSql.json
```

Please see the student example above on creating a JSON file to run a list of IDs. To run a list of applicant IDs you use the same JSON file format except you need to change the “name” property to “**RAD32**”.

Advisor

To run the advisor extract using the default `banneradvisors.sql` file in the local/sql directory your JSON file needs to be setup like the one above for students using the default SQL file except change the “name” to “**RAD31**”.

You then specify your JSON file when running launchjob. For example:

```
$ launchjob $ADMIN_HOME/myjobs/rad31.defaultAdvisorsSql.json
```

Please see the student example above on creating a JSON file to run a list of IDs. To run a list of advisor IDs you use the same JSON file format except you need to change the “name” property to “**RAD31**”.

Staff

Please see the student example above on creating a JSON file to run a list of IDs. To run a list of staff IDs you use the same JSON file format except you need to change the “name” property to “**RAD33**”.

Deleting IDs

To delete unwanted IDs from the Degree Works database, you can create a query to select those individuals, and save it in a file named `local/sql/bannerdeleteids.sql`. To run the delete function, issue the following command after exiting the local/sql directory **(do NOT use with cron)**:

```
$ bannerextract deleteid
```

You can also specify a different sql file in the local/sql directory:

```
$ bannerextract deleteid somedeletes.sql
```

You can also specify a file of IDs in the local/sql, admin/data or current directory. With all ids files, be sure to upload your file to the server as ASCII or Text, not Binary.

```
$ bannerextract deleteid somedeletes.ids
```

You can also run the script to delete a single ID:

```
$ bannerextract deleteid 12345
```

(where “12345” is the student ID.)

Also review the UCX-CFG020 RADBRDIGE DELETEID setting; set to Y to delete the audits, notes and exceptions for a student in addition to all bridged data.

Selected UCX Tables

To run the ucx extract using a list of Degree Works UCX tables listed in a file in the local/sql directory with an “.ucx” extension **(do NOT use with cron)**:

```
$ bannerextract ucx someucxtables.ucx
```

Warning

DO NOT put the UCX_ prefix on the table names even though the actual database table names are UCX_STU352, UCX_STU560 and UCX_STU563.

For example, the “local/sql/someucxtables.ucx” file might contain tables:

```
STU352
STU560
```

```
STU563
```

In this case ONLY these 3 UCX tables will be re-extracted from Banner.

Note: The UCX-STU563 table (Concentrations) is re-created from the STVMAJR Banner table which also is used to recreate major tables (UCX-STU023 and UCX-AUD027) and minor tables (UCX-STU024 and UCX-AUD029). However, only UCX-STU563 will be extracted with this banner extract.

Make sure to check the setting for the UCX-CFG020 BANNER “Add UCX Entries Only” flag BEFORE running the UCX bannerextract. Make sure it is set to “N” if ALL entries are to be reloaded from Banner. If only NEW entries are to be loaded from Banner make sure this UCX-CFG020 flag is set to “Y”.

In the above example, UCX-STU352 is being re-extracted. It has a “Discipline Status” flag in it which is manually input using Controller. These updates will need to be made again if the entire UCX-STU352 is re-extracted.

Other Modes

The other extract modes do not have associated sql files – these modes extract ALL records from the Banner database and load them into the associated Degree Works database tables. However, you may modify the relevant sections in the `integration.banner.extract.config` setting to control exactly which records are bridged into Degree Works.

These other modes do not have parameters so the JSON file you would use with the launchjob script would look like this with the appropriate RAD3x name specified:

```
{
  "name": "RAD3x"
}
```

Each job can of course be run manually in Transit; the JSON file is required when running these jobs in cron.

Course – RAD34

Only adds/updates rad_course_mst records, but deletes and re-adds rad_crs_attr_dtl records.

Curriculum Rules – RAD35

The old curriculum rules are first deleted and then new curriculum rules are re-added.

Equivalences – RAD38

The dap_eqv_crs_mst and UCX-CFG070 are first both deleted and all equivalencies re-added.

ETS (transfer schools) – RAD37

Only adds/updates rad_ets_mst records (no deletes).

Mappings – RAD39

The old mappings are first deleted and then new mappings are re-added.

UCX (Validation tables) – RAD36

If UCX-CFG020 RADBRIDGE Add UCX Entries Only = “N” all records are deleted and re-added.

If UCX-CFG020 RADBRIDGE Add UCX Entries Only = “Y” only new records will be added (no updates).

Transit – RAD30

Transit allows you to run the RAD30 processor by ID or by SQL selection. For students you may use the default bannerstudents.sql query or you can enter individual IDs or use an ID file containing the pool of students that you wish to import into Degree Works.

There is also a method to submit RAD30 via cron, using the launchjob script – as explained above. The launchjob script will submit your job to Transit so that the resulting reports or debug files can be retrieved via the TransitUI. For more information on Transit and launchjob, see the *Transit Administration Guide*.

transit.rad30.workerCount

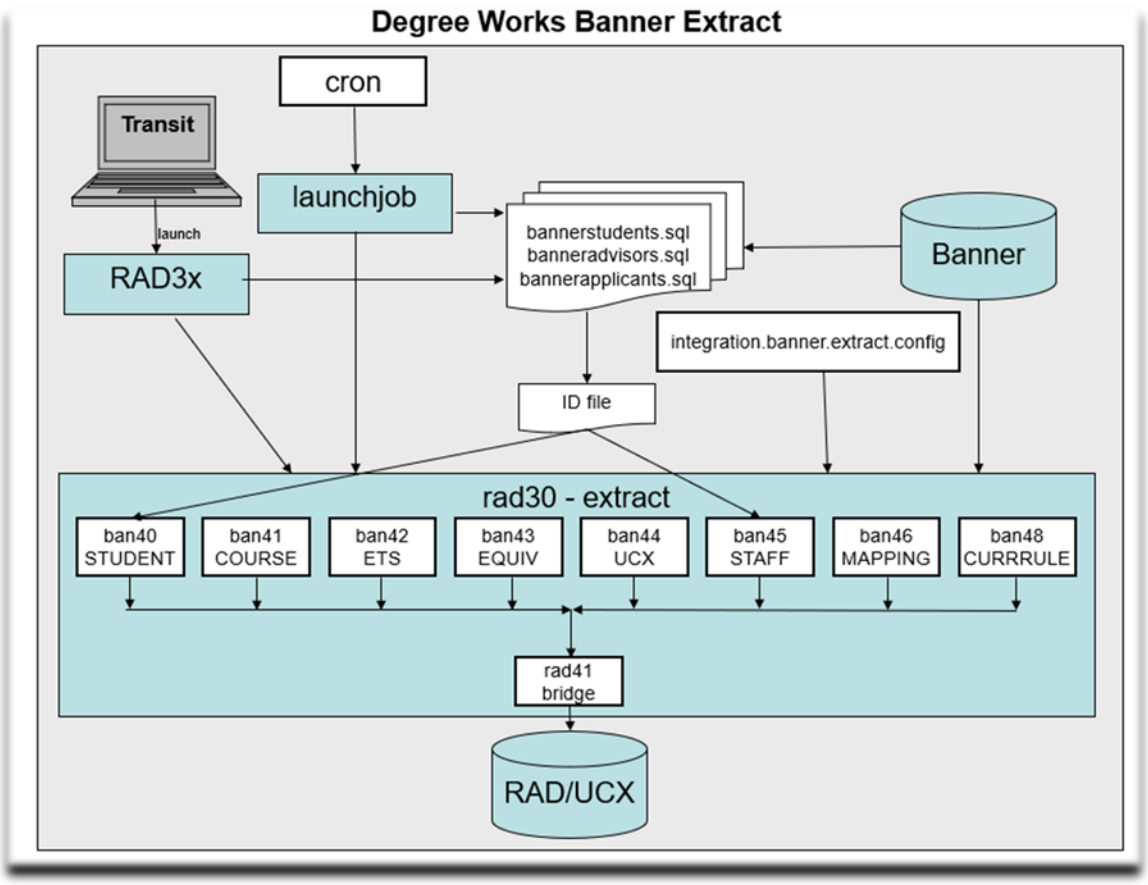
The RAD30, RAD31, RAD32 and RAD33 jobs will use the **transit.rad30.workerCount** setting to split up the file of student, applicant, advisor or staff IDs into multiple files. Once multiple ID files are created, the scripts are then able to launch multiple processes on each of the files. In theory, with this workerCount set to 4 the task gets done in a quarter of the time.

Use Controller to change your setting. You need to determine what value works best for the resources available on your system.

Bridge Audits

When the extract finishes loading student data a list of students with changed data 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. Please see the *Degree Works Configuration Technical Guide* for information on these settings. A separate dap22 log file is created for this process and can be accessed via Transit. You can review it to see how many students had data changes and to check the status of those audits.

Batch Extract Flow Diagram



Dynamic Data Extract Process

In addition to extracting student data via the batch extract you may extract data for a single student using the dynamic method – there are two ways in which a dynamic extract may take place: the first is available from within the Degree Works web application as a button, and the second occurs whenever a triggering event takes place.

The first way of performing a dynamic extract from Banner is to push a button. Users with the SDREFRES key will be shown the Last Refresh field in the student context area at the top of the main web page. When the user also has the SDREFBTN key they will be allowed to click a button to refresh the student.

The Last Refresh date and time indicate the last time the student's academic data was copied from Banner into Degree Works. At any time users may click the on-demand refresh button to once again pull in this student's data. Users may want to do this after a major or grade change was made in Banner and they don't want to wait for the nightly batch extract. Once the extract process completes the user will get a confirmation message and the Last Refresh date/time will be updated. In the Dashboard the user will need to run *Process New* to run a new audit on the refreshed data. However, in the Responsive Dashboard a new audit is automatically created.

The second way of performing a dynamic extract involves a triggering event, which may occur when any user requests an audit from the Worksheets, What If, Planner, Exceptions or Look Ahead tabs. The UCX-CFG020 REFRESH record in Controller is used to control this behavior.

Running Audits

Turning on the Dynamic Refresh with all or some of the other flags turned on will affect performance – both for all users of the system and the particular users attempting to run an audit. Before Degree Works runs the requested audit it refreshes the student's academic data from Banner – this takes extra time and will consume additional system resources. When a refresh occurs through running an audit the user is not notified – but the Last Refresh date/time is updated.

The Refresh Timeout is a mechanism the client may use to inhibit repetitive dynamic refreshes for students that are incidental and not necessary, for instance when multiple What-Ifs are being launched within a short time span. The client can specify a length of time during which no additional refreshes of data would be appropriate. This timeout setting is used in conjunction with the refresh date/time stored on the rad-primary-mst to tell Degree Works if it should re-read the student's data in Banner.

As with the batch extract process, the banstudent and radbridge routines are used to perform the dynamic refresh. The radbridge does check for changed data and will skip the insertion of records if no changes exist – but will update the refresh date/time on the rad-primary-mst and that shown on the web page either way.

Viewing Audits

With the View Audit Refresh flag set to Y the system will check the timeout setting as with running an audit and will execute the extract as needed. If there are data changes a new audit will be run. In

addition, the audit and refresh date/times will be updated in the student context area on the web page reflecting what has just occurred. When an extract is processed while in View mode the bridge date/time are not updated if there are no real data changes – this is different from when in Run mode.

When the View Audit Refresh flag is N and one of the other refresh flags is Y the system compares the extract/bridged date/time to that on the most recent audit for the given school/degree. If it is determined that the audit is stale a new audit will be processed.

Banner Degree Coding Structure

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 utilize 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.

Example A

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 SORLFOS - sorlfos_majr_code =CHEM; lcur_seq_no=1 SORLFOS - sorlfos_majr_code =BIOL; lcur_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.

Example B

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 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	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.

Example C

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 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	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.

Example D

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 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	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. □ 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.

Example E

two different BS degrees (same level) - two majors – Not recommended

Definition of degree and location of data, in Banner	Results in Degree Works
SORLCUR - sorlcur_degc_code=BSMATH; seq-no=1 SORLFOS - sorlfos_majr_code =MATH; lcur_seq_no=1 SORLCUR - sorlcur_degc_code =BSPHYS; seq-no=2 SORLFOS - sorlfos_majr_code =PHYS; lcur_seq_no=2	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. 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. 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.

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

Example F

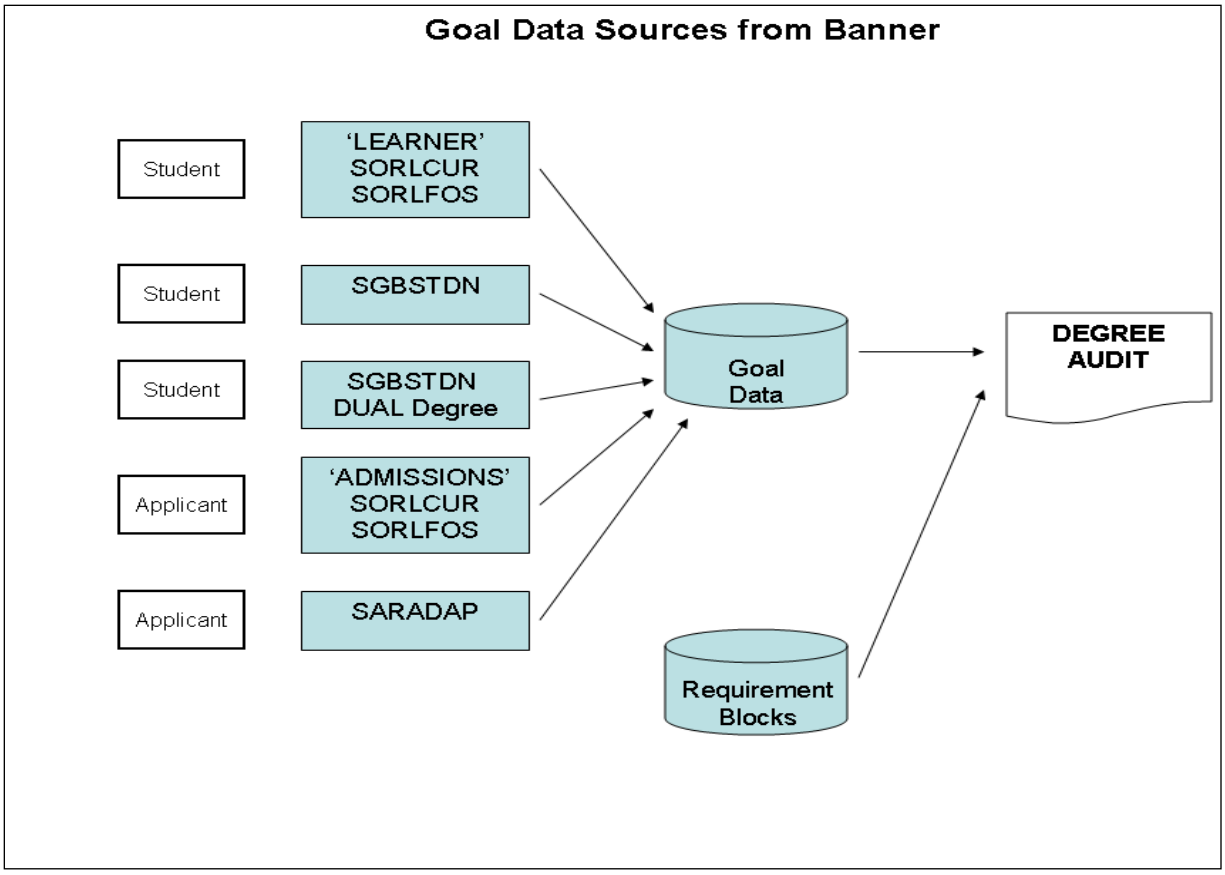
one BS degree (same level) – with two different programs and Program as Degree = Y

Definition of degree, and location of data, in Banner	Results in Degree Works
SORLCUR - sorlcur_degc_code=BS; program=ANTH SORLCUR - sorlcur_degc_code=BS; program=CHEM	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. 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.

Banner Applicant Processing

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 job RAD31 may be run to extract applicant data to create the appropriate Goal and Goal Data records for each unique Level (school) and Degree combination (as well as extract all other appropriate Banner data for an applicant that may be used by Degree Works). The diagram below outlines how the Goal Data may be loaded into Degree Works by the RA30 and RAD31 jobs with three STUDENT paths and two APPLICANT paths:



There are 6 paths that can be taken to load student/applicant Goal data:

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 for unique Level (School)/Degree combinations.

6. Applicant Goal Data may be loaded from SARADAP curriculum data for unique Level (School) codes.

The rules for determining how RAD31 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) as well as 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 as well as 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 (pre-DW4.1.0 functionality).

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:

1. 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 then set LEARNER_FOUND = "Y".

2. 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”.

3. 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”.

4. 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”.

Once 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:

1. If LEARNER_FOUND = “Y” then the appropriate rad_goal_dtl and rad_goaldata_dtl records will be created from LEARNER SORLCUR/SORLFOS.

2. If LEARNER_FOUND = “N” and ADMISSIONS_FOUND = “N” and SGBSTDN_FOUND = “Y” then the appropriate rad_goal_dtl and rad_goaldata_dtl records will be created from SGBSTDN.

3. 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_goaldata_dtl.records will be created from the Dual Degree.

4. 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_goaldata_dtl records will be created from the ADMISSIONS SORLCUR/SORLFOS data.

5. 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_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)/Degree combination is unique (does not match any LEARNER or SGBSTDN combinations previously extracted) then the appropriate rad_goal_dtl and rad_goaldata_dtl records will be created from SARADAP.

6. 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 (e.g., AP tests), etc., 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_ENTRY
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_ENTRY
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(l.SORLFOS_SEQNO) FROM SORLFOS l
SORLFOS2-where: WHERE l.SORLFOS_PIDM = b.SORLCUR_PIDM
SORLFOS2-where:   AND l.SORLFOS_PRIORITY_NO = a.SORLFOS_PRIORITY_NO
SORLFOS2-where:   AND l.sorlfos_csts_code = 'INPROGRESS'
SORLFOS2-where:   AND l.SORLFOS_LCUR_SEQNO = b.SORLCUR_SEQNO)
SORLFOS2-where: AND
#                    a.SORLFOS_PIDM = <students-pidm>
```

Required Access to Banner

Database Table Access

Access to your Banner database is required for the extract process, therefore read access must be provided to the tables listed in the chart below. 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 via 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.

As of Banner 8.5.3, 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.

Note: As of the 4.1.1 release, TreQ is now known as Transfer Equivalency and WebTreQer is now known as Transfer Equivalency Self-Service.

Banner Table	Course Catalog	Course Equivalents	Curriculum Rules	ETS (transfer)	Advisors / Staff	Students	Applicants	UCX	Banner General	Transfer Equivalency Admin / Self-Service	Description
GOBUMAP					X	X	X				UDC_ID/SPRIDEN_PIDM Mapping
GORADID					X	X	X				ADDITIONAL_ID – moved to shp_user_mst
GOREMAL					X	X	X				Email Address Data
SARADAP							X				Applicant Degree Data
SCBCRKY		X									Course Start/End Dates
SCBCRSE	X	X									Course Master Data
SCBDESC	X										Course Description
SCRATTR	X										Course Attributes
SCRLEVL	X										Course School Attributes (DW-SCHOOL)
SCREQIV		X									Course Equivalents
SCRRTST	X										Course Prerequisites
SCRTEXT	X										Course Description
SFRSTCR						X	X				Current Class Data

Banner Table	Course Catalog	Course Equivalents	Curriculum Rules	ETS (transfer)	Advisors / Staff	Students	Applicants	UCX	Banner General	Transfer Equivalency Admin / Self-Service	Description
SGBSTDN						X	X				General Student Degree Data
SGRADVR					X	X	X				Advisor Data
SGRSATT						X	X				Student Attributes by PIDM and Current Term
SGRATHE						X					Student Athlete – first date of attendance for athletic eligibility
SGRCATT						X					Class Attribute Repeating Table
SGRCLSR						X					Student Classification Table
SHBRPTS						X					Title Indicator
SHBTATC										X	Transfer Institution Transfer Catalog Data
SHRATTC						X	X				Student Attributes by CRN and Historic Term
SHRATTR						X	X				Student Attributes by PIDM and Historic Sequence Number
SHRDGMR						X					Student Degree table

Banner Table	Course Catalog	Course Equivalents	Curriculum Rules	ETS (transfer)	Advisors / Staff	Students	Applicants	UCX	Banner General	Transfer Equivalency Admin / Self-Service	Description
SHRGRDE						X	X	X			Grade Codes (UCX-STU385) Grade Types (UCX-STU356)
SHRGRDO						X	X	X			Grade Codes – valid combo of level/gmod/grade (UCX-STU385)Grade Types (UCX-STU356)
SHRICMT										X	Transfer Articulation Institution Course Comment
SHRLGPA						X	X				Summary GPA/Credits Data
SHRNCRS						X	X				Non Course Data
SHRQPNM						X	X				Non Course Data – Papers and Exams
SHRTATC										X	Transfer Institution Catalog Equivalent Data
SHRTATT						X	X				Student Attributes by PIDM and Transfer Sequence Number
SHRTCKD						X	X				Institutional Course Term Degree Applied Repeating Table
SHRTCKG						X	X				Historic Grade Values

Banner Table	Course Catalog	Course Equivalents	Curriculum Rules	ETS (transfer)	Advisors / Staff	Students	Applicants	UCX	Banner General	Transfer Equivalency Admin / Self-Service	Description
SHRTCKL						X	X				Historic Level (School) Data
SHRTCKN						X	X				Historic Class Data
SHRTGPA						X	X				Detail GPA/Credits by Term
SHRTRCD						X					Transfer Course Degree Applied table
SHRTRCE						X	X				Transfer Equivalent Data
SHRTRCR						X	X				Transfer Class Data
SHRTRIT						X	X				Transfer School Data
SHRTRAT										X	Transfer Attributes
SOBCACT						X	X				SORLCUR active indicator validation
SOBCURR			X								Curriculum Rules Base Table
SOBSBGI				X							ETS Address Data
SORBTAG				X							ETS Calendar Data

Banner Table	Course Catalog	Course Equivalents	Curriculum Rules	ETS (transfer)	Advisors / Staff	Students	Applicants	UCX	Banner General	Transfer Equivalency Admin / Self-Service	Description
SORCCON			X								Curriculum Rules Concentration Data
SORCMJR			X								Curriculum Rules Major Data
SORCMNR			X								Curriculum Rules Minor Data
SORDEGR						X	X				Previous Degree Data
SORLCUR						X	X				Concurrent Degree Data
SORLFOS						X	X				Field of Study Data
SORMCRL			X								Curriculum Rules Control Table
SORTEST						X	X				Test Score Data
SMRPRLE						X	X				Program codes
SPRIDEN					X	X	X				Primary Name Data
SSBSECT						X	X				Schedule Master Data
SSBXLST	X										Cross Listing Table – seat count

Banner Table	Course Catalog	Course Equivalents	Curriculum Rules	ETS (transfer)	Advisors / Staff	Students	Applicants	UCX	Banner General	Transfer Equivalency Admin / Self-Service	Description
SSRATTR						X	X				Student Attributes by CRN and Current Term
SSRMEET	X										Section meeting times
SSRXLST	X										Cross Listing table – grouping
STVACCL								X			Calendar Codes (UCX-STU346) used by Transfer Equivalency Self-Service
STVACYR											Catalog Year Codes (UCX-STU035)
STVATTR	X							X			Attribute Codes (UCX-STU050) and used by Course Link
STVCLAS								X			Student Level Codes (UCX-STU305)
STVCOLL								X			College Codes (UCX-STU560)
STVCSTA	X	X	X								Course Status Codes
STVDEGC								X			Degree Codes (UCX-STU307)
STVGMOD								X			Grade Types (UCX-STU356)

Banner Table	Course Catalog	Course Equivalents	Curriculum Rules	ETS (transfer)	Advisors / Staff	Students	Applicants	UCX	Banner General	Transfer Equivalency Admin / Self-Service	Description
STVLEVL								X			School Codes (UCX-STU350)
STVMAJR								X			Major Codes (UCX-STU023), Minor Codes (UCX-STU024), Concentration Codes (UCX-STU563)
STVNATN				X							ETS Foreign Country Codes
STVNCRQ						X	X				Non Course Codes
STVNCST						X	X				Non Course Status Codes
STVQPTP						X	X				Non Course Exam/Paper Codes
STVRSTS						X	X				Course Registration Status
STVSBGI				X		X	X	X			ETS School Names
STVSSTS	X										Course section status (active indicator)
STVSTST						X	X				Student Status (DW student type)
STVSTYP						X	X	X			Student Type (DW Student Status Codes UCX-STU306)

Banner Table	Course Catalog	Course Equivalents	Curriculum Rules	ETS (transfer)	Advisors / Staff	Students	Applicants	UCX	Banner General	Transfer Equivalency Admin / Self-Service	Description
STVSUBJ								X			Discipline Codes (UCX-STU352)
STVTAST										X	Transfer Articulation Course Status
STVTERM	X	X	X			X	X	X			Term Codes (UCX-STU016)
SURVERS									X		Version of Oracle being used
TWGBWSES									X		Banner Self-Service

Transfer Equivalency Customers: If you have purchased 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

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

If 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 in order to install `dwssbfaculty.sql` and `dwssbstudent.sql`.

(See the *Self-Service Banner* section in this document).

TWBKWBIS

TWBKFRMT

BWLKOIDS

BWLKOSTM

BWCKFRMT

BWLKILIB

BWCKLIBS

Banner Database Issues

Pointing to a different Banner database

The following are the steps to take in order to point to a different Banner database from Degree Works.

1. Create a **dwmgr** user in the new Banner database with select privileges to the list of tables identified in this document. You must run the two grants scripts, bannergrants.sql and bannergrants2.sql in the Banner database (both are in $DGWHOME/sql.) (bannergrants2 must be run by sysdba as it grants access to Oracle queues required for pre-requisite checking).

   Note: Typically the username is dwmgr but you can actually use any name you like. If you choose a different name, then use that name in the instructions below instead of dwmgr.

2. On the Degree Works server, update the `$ORACLE_HOME/network/admin/tnsnames.ora` entry and add the new Banner database.

3. On the Degree Works server, edit the `dwenv.config` file and modify the following line:
   `export DB_LOGIN_BANNER=`
   and set the value to
   `export DB_LOGIN_BANNER=dwmgr@service`

   where dwmgr is the user defined in the new Banner database and service is the service name from tnsnames.ora. Keep this change by saving the `dwenv.config` file.

   To set the password for your Banner user run setdbpasswords:

   ```
   setdbpasswords --sispassword thebnrpassword
   ```

4. Log in again into the Degree Works server so that the new DB_LOGIN_BANNER variable is set. To check, issue the following command:
   `env | grep DB_LOGIN_BANNER`

   and you should see the new entry from dwenv.config.

5. Test your new Banner connection by typing (as the dwadmin user logged into the Degree Works server):
   `dbb`

   and SQL*Plus should be launched in the Banner database. To verify that you are looking at the correct Banner database issue:
   `SQL> select * from global_name;`

6. Restart the web and dap daemons using the `webrestart` and `daprestart` commands.

7. Determine if the data in the following tables in the new Banner database are different from the Banner database you originally used to populate Degree Works. If so, rerun the associated processes so data from the new Banner database is used to populate Degree Works.

SCBCRSE, SCRATTR, SCRRTST	rerun course extract
SCREQIV, SCBCRKY, STVCSTA	rerun equiv extract
SHRGRDO, SHRGRDE	rerun ucx extract for UCX-STU385 rerun ucx extract for UCX-STU356
STVCLAS	rerun ucx extract for UCX-STU305
STVCOLL	rerun ucx extract for UCX-STU560
STVDEGC	rerun ucx extract for UCX-STU307
STVGMOD	rerun ucx extract for UCX-STU356
STVLEVL	rerun ucx extract for UCX-STU350
STVMAJR	rerun ucx extract for UCX-STU023, UCX-STU024, UCX-AUD027, UCX-AUD029, UCX-STU563
STVSTYP	rerun ucx extract for UCX-STU306
STVTERM	rerun ucx extract for UCX-STU016, UCX-STU035
STVACYR	rerun ucx extract for UCX-STU035

Before running the UCX extract you should review the 'Add UCX entries only' flag on UCX-CFG020 RADBRIDGE. If you already have entries in these UCX tables that you do not want to lose you should set this flag to Y so that only new entries that do not exist in the UCX tables are added but existing entries are not touched.

After the ucx extract has been executed, issue a `daprestart` and `webrestart`.

Creating the Banner Database Link in Degree Works

Once you have finished all of the steps above and confirmed that you can connect to Banner with the dbb command, you should recreate the database link to Banner in the DW database.

If the Banner and Degree Works databases are on separate servers, modify `tnsnames.ora` on the Banner database server so there is a connection to the Degree Works database. This is required for the Banner database link, created in the Degree Works database, to connect to Banner.

1. On the Degree Works server, log in as the administrative user (typically dwadmin) and determine the Banner login information by issuing the following command:
   ```
   echo $DB_LOGIN_BANNER
   ```

   The result will be similar to the following:
   ```
   DB_LOGIN_BANNER=banneruser@bannerservice
   ```

   To view the Banner database password run this command:
   ```
   showdbpasswords --sispassword --noencrypt
   ```

2. cd to the $DGWHOME/sql directory. If your site has modified the bannerlink.sql script and placed it in $LOCAL_HOME/sql, cd to that directory before creating the database link.
3. Log into SQL*Plus in the Degree Works database by issuing the db command. Locate and remove the existing database link if one exists:
   ```
   db
   SQL> select db_link from user_db_links;

   DB_LINK
   --------------------------------------------------------------------------------
   MYDBLINK

   SQL> drop database link MYDBLINK;
   ```

4. Execute the bannerlink.sql script to add the new database link:
   ```
   db
   SQL> @bannerlink BANNER_SERVICE BANNER_USER BANNER_PW
   SQL> exit
   ```

Note: The bannerlink script will attempt to drop the database link, so you can ignore the “ORA-02024: database link not found” Oracle warning.

MEP (Multiple-Entity Processing) in Banner

Banner sites wishing to enable multi-entity processing must first configure table sharing in the **dwschema.xml** and **share.xml** files which reside on the Degree Works administrative server. For more information please see the *Multi-Entity Processing* section of the *Degree Works Technical Guide*. Once your multi-entity table sharing has been established, you must then ensure that each campus entity in Degree Works has been configured to extract the correct data from Banner.

Each of your Degree Works campus entities will have unique $ADMIN_HOME and $LOCAL_HOME directories on the Degree Works administrative server. You should be able to use the "newenv" command to switch from one MEP entity to another on the Degree Works server. The following changes must be made to each campus entity environment:

1. **bannerstudents.sql.** This file resides in $LOCAL_HOME/sql and should be configured by each campus entity to select the Banner IDs of only their matriculated students. For example, the CAMP_CODE column on the SORLCUR table is a very common way to find active students from a particular campus. Writing SQL to select only those students matriculated at a specific campus entity will assure that degree audits are generated for only these students and not for any student within the multi-campus institution. For more information on editing bannerstudents.sql, see *Setup SQL Select Files* in the *Degree Works Data Extract* section of this document.

2. **`integration.banner.extract.config`.** Each campus entity must also maintain its own setting, maintained using Controller. For each student selected using bannerstudents.sql, the SQL in this setting will be run to select only those records of interest to this college. For example, if a student is matriculated at multiple campuses, the setting should be modified so that only the data for a specific campus are read; this results in the correct degree information for that particular campus. For more information on editing this setting, see *Setup integration.banner.extract.config setting* in the *Degree Works Data Extract* section of this document.

3. If you are connecting to a Banner database which has been configured as MEP (Multi-Entity Processing), you must create a unique Degree Works user in the Banner database and associate it to a Banner VPDI code. In the Banner database, an individual MEP environment is identified by the VPDI code which is stored on the VPDI_CODE column of each table. In Degree Works, you must create a unique user in the Banner database for each MEP entity. This is the same user which is stored in the DB_LOGIN_BANNER variable in dwenv.config.

    a. In Banner, enter the GSAVPDI form and click on the User Assignment tab. Here you need to associate each Oracle user you have created in Banner for Degree Works (DB_LOGIN_BANNER) with its Banner VPDI Institution Code. Once your DB_LOGIN_BANNER users are established in the Banner database, they should be available in the User ID picklist in GSAVPDI.

    b. On your Degree Works server, for each of your campus entities, edit the `$ADMIN_HOME/dwenv.config` file and place the following environment variable definition after the DB_LOGIN_BANNER variable:
    `export BANNER_VPDI=MYINST`

    where MYINST is the Banner VPDI Code which you have associated with your

Degree Works Oracle user DB_LOGIN_BANNER. Do not place spaces between the equal sign (=) and the VPDI Code value. The Maximum size of the VPDI code is 6-bytes.

Note: You must log out and back into your host session after making this change.

Warning

Make sure the correct VPDI Code is input as NO error checking is done! If an invalid VPDI code is used, the database queries will not return any records for tables that are linked to a particular VPDI code, and you will NOT get the correct data extracted into Degree Works.

Banner Workflow Integration

Installation Guide

Follow these steps in order to install and enable Banner Workflow integration with Degree Works for the sample models: exception petition process and planner approval process.

Note: These steps are optional.

We provide guidance and suggestions on Banner Workflow configuration here. But, it is assumed you are familiar, in general, with setting up Models, Business Events, Business Processes and Business Components in Workflow and you should refer to Banner Workflow documentation for more information.

Once the sample Workflow models are installed and set up you should inspect and customize the models before testing or using them. In particular you should inspect the activities, notifications and emails defined in each of these workflow models to customize Roles, Performers and email addresses. The “from” address in all notifications and emails used in these models is the email set for the Workflow “admin” user. These samples also use Workflow Roles: Approver, Academic Advisor and Academic Dean. You should be aware of the users and email addresses associated with these Roles because they will be the ones notified to approve Degree Works requests if you do not modify the models. Most likely, you will want to customize the models to change which Roles/Users are responsible for performing the approval actions and receiving Workflow notifications.

As of the 4.1.0 release, there are two versions of the Student Educational Planner. The new version based on java/ZK technologies uses new tables that are named beginning with “sep_plan”, so in this document it is known as the sep_plan version. The old version is based on the dap_planner_dtl table, so it is known as dap_planner.

Log in to the Degree Works host server

1. Load Workflow related packages into the Degree Works database.

    a. `cd app/sql`

    b. `db`

    c. `@wf_parameters_pkg.sql`

    This file contains package wf_parameters and procedures F_PetitionParams and F_PlannerParams. This package can be customized in order to change the parameters that are sent to Banner Workflow model DW_PETITION and DW_PLANNER. The parameters that are extracted from the Degree Works database by these procedures needs to match up with the parameters that the Workflow models expect. So, if the “context parameters” of the model are customized, then these procedures probably

need to be customized too. For the new SepPlan workflow, this package is not used since the data parameters are handled in java.

d. `@wf_updates_pkg.sql`
This file contains a package named wf_updates and procedures P_UpdatePetition, P_UpdatePlanner, and P_UpdateSepPlanner. These procedures would probably not need to be customized.

2. Enable Banner Workflow integration using Controller.
For exception petitions: CFG020 WORKFLOWPETITION “Enable Petition Workflow”
For old dap_planner: CFG020 WORKFLOWPLANNER “Enable Planner Workflow”
For new sep_plan: studentPlanner.planner.planApprovalMethod=W

3. For new sep_plan, review shpSettings core.workflow.* and studentPlanner.planner.workflow.*. Please see the *Degree Works Configuration Technical Guide* for information on these settings.

4. For exception petitions and the old dap_planner, review settings in CFG020 WORKFLOWWSDL to set the URL to your Banner Workflow server WSDL for web services location.

5. For exception petitions and the old dap_planner, review settings in CFG020 WORKFLOWCREDENTIALS to set a username and password for your Banner Workflow server. Commonly, “wfwebservices” would be the username.

Log in to the Banner Workflow host server

These tasks need to be completed by a system administrator who can reconfigure and restart the Workflow server.

6. Upload these three files from your Degree Works environment to your Workflow server.

   a. `dw_petition.zip`

   b. `dw_planner.zip`

   c. dw_sep_planner.zip

   They will be located in the *sis_Banner/export/Workflow* directory under the updates (installer root) directory. Locate the updates directory from your Degree Works installation (at *$HOME/updates*) or from your most recent Degree Works update (*$DGWHOME/updates*). Locate the files by issuing this command from within the updates directory:

   ```
   find . -name dw_petition.zip 2>/dev/null
   ```

   cd to that location to access the Workflow zip files. Transfer them as binary to your Workflow server.

7. Execute these two commands to import the sample Degree Works workflow models. Substitute a valid workflow username and password (wfroot for example) on your system as well the actual location of the files you uploaded in the previous step. After importing these files,

workflow will have the Workflow Models and Business Components under category "+DegreeWorks".

a. `$WFHOME/bin/import username password dw_petition.zip`

b. For old dap_planner only: `$WFHOME/bin/import username password dw_planner.zip`

c. For new sep_plan only: `$WFHOME/bin/import username password dw_sep_plan.zip`

8. Edit `$WFHOME/config/configuration.xml` to add a DataSource for the Degree Works database to the "DataSources" section. An example DataSource setup follows in which you should substitute: your actual Degree Works server url for "your.degreeworks.server.edu", the actual port number the Degree Works Oracle listener is running on for "1521" and your actual oracle listener handler for "YOUR_DEGREEWORKS_SID". The username and password here must match what you see in the $DB_LOGIN variable; this is the Degree Works user login to the Degree Works schema.

```
<DataSources>
 <!-- ... other existing data sources should not be modified... -->
 <DataSource name="DegreeWorks">

<Url>jdbc:oracle:thin:@your.degreeworks.server.edu:1521:YOUR_DEGREEWORKS_
SID</Url>
     <Username>yourusername</Username>
     <Password>yourpassword</Password>
 </DataSource>
</DataSources>
```

a. After editing `configuration.xml`, run: `$WORKFLOW_HOME/bin/wftool uploadconfig`

b. Use the `engineconsole` and `startengine` scripts to restart the engine node(s).

c. Restart the OC4J instance(s).

Log in to the Banner Workflow web environment as admin user

9. Create the Product Type needed for connecting to the Degree Works database. As an administrator using Workflow in your web browser, navigate to Workflow System Administration > Product Types. Click "Add Product Type". Give it the name "DegreeWorks", Version 1 (one), and choose the Data Source "DegreeWorks" from the drop-down list. This depends on the naming of the DataSource and successful restart of the Workflow server.

10. Create/modify business events for the three Degree Works models: Business Events > Business Event Definitions > Click "Add Business Event Definition".

a. Set “Name:” to anything you like, we suggest “DW_PETITION”. Whatever name you set, it will also have to be set to the same name on CFG020 WORKFLOWPETITION “Petition Event Name”. Click “Save”.

b. Add Event Parameters for “DW_PETITION” as follows and set them all to Type=”Text” and Guaranteed to Yes or No as shown below. On the topic of “Guaranteed” options: if you change any data that you want to be “optional” in wf_parameters_pkg.sql, then set those parameters to Guaranteed=”No”.

CREATE_DATE	Text	Yes
DESCRIPTION	Text	No
NOTE_TEXT	Text	Yes
STUDENT_EMAIL	Text	No
STUDENT_GOALS	Text	No
STUDENT_ID	Text	Yes
STUDENT_NAME	Text	Yes
UNIQUE_ID	Text	Yes
USER_EMAIL	Text	No
USER_NAME	Text	Yes

c. There are two optional parameters that you can define and modify wf_parameters_pkg.sql to implement. They control whether either or both approvers in the sample model should be ignored. Define and set APPROVAL1_REQUIRED=0 to disable approver1. Define and set APPROVAL2_REQUIRED=0 to disable approver2.

d. Still on Business Event Definition, click “Add Workflow Association”. Select the value from the drop-down list provided for your workflow model and version.

e. After you click “Save” you will then map all of the business event parameters to the context parameters within that version of the workflow model. You must map all “guaranteed” parameters. When finished click “Save Parameter Mappings”.

11. If you are using the old dap_planner, repeat step 10 for another event named “DW_PLANNER” which has to match on CFG020 WORKFLOWPLANNER “Planner Event Name” and have parameters: DESCRIPTION, STUDENT_EMAIL, STUDENT_GOALS, USER_EMAIL, CREATE_DATE, NOTE_TEXT, STUDENT_ID, STUDENT_NAME, UNIQUE_ID, USER_NAME. Again, parameters named APPROVAL1_REQUIRED and APPROVAL2_REQUIRED are optional and work the same way as explained above.

12. If you are using the new sep_plan, repeat step 10 for another event named “DW_SEP_PLAN_APPROVAL” which has to match on shpSetting “studentPlanner.planner.workflow.plannerEventName” and have parameters: DESCRIPTION, STUDENT_EMAIL, STUDENT_GOALS, USER_EMAIL, CREATE_DATE, NOTE_TEXT, STUDENT_ID, STUDENT_NAME, UNIQUE_ID, USER_NAME. Again, parameters named APPROVAL1_REQUIRED and APPROVAL2_REQUIRED are optional and work the same way as explained above.

13. Create/modify business processes for the three Degree Works models: Enterprise Management > “Add Business Process”. We suggest you name the business processes using

the same token as the Business Events: DW_PETITION and DW_PLANNER (for dap_planner) and DW_SEP_PLAN_APPROVAL (for sep_plan).

a. Set Status=Active.

b. Click “Add Workflow Association” and choose the appropriate workflow model/version.

c. Click “Add Event Association” and choose the Business Event name you defined in steps 11/12.

d. Authorized Initiators can be left empty, it is optional.

e. Click “Save Process”.

14. Verify that you have these three models installed by opening them in the Workflow Modeler: DW_PETITION, DW_PLANNER (for dap_planner), and DW_SEP_PLAN_APPROVAL (for sep_plan).

15. If you want to customize the models immediately, use the modeler to make a “new version” or “create a copy”. Refer to the Banner Workflow documentation for more information on using the modeler.

a. When customizing the models, if you change any “context parameters” then you will have to update the pl/sql where those parameters are extracted from Degree Works in `app/sql/wf_parameters_pkg.sql` (and reload that package into the Degree Works database).

b. You will also have to modify the business event parameters you defined later in steps 11-12 to match the parameters names given in the model context parameters and in the pl/sql package.

Managing Plan/Petition Approval – two tools

Degree Works also provides tools to allow your users to approve/reject plans and petitions. However, if you are using Banner Workflow to manage plans and petition approval it would be best to not use those tools in Degree Works. Using both mechanisms to approve/reject plans will most likely lead to confusion on campus about the approval process. Specifically, the Manage tab under the Planner tab can be used to approve/reject submitted plans and Exception Management supports a way to approve/reject petitions. Both of these should be turned off if Banner Workflow is being used to manage plan and petitions approval.

See the *Exception Management* and *Student Educational Planner* sections in the *Web User Guide* for more information.

Integration with Portals

End-User Access to Degree Works by Banner customers is typically accommodated through either Banner Self-Service or the Luminis products.

The following sections provide more information about both these products, and their use with Degree Works.

Luminis

Single Sign-on for Degree Works

This is a guide for integrating Degree Works with Luminis via single sign-on (SSO). The only option for single sign-on that is supported by both Degree Works and Luminis is CAS.

Support for Generic Connector Framework (GCF) has been desupported.

Integration using a Single Sign-On Provider

CAS is the recommended single sign-on provider. For guidance configuring CAS, see the relevant sections of Ellucian CAS Handbook and Degree Works Settings/Security documentation.

SAML 2.0 direct single sign-on is supported by Degree Works and is expected to be supported by Luminis in the future. When using Ellucian Identity Service (EIS), SAML and additional single sign-on providers can also be used to bridge applications that are using CAS.

The most common Degree Works application for linking from a portal is Dashboard Servlet. Links to the full Degree Works dashboard and to individual views within the dashboard are possible. One example is a link to a student's own audit. See the Example Degree Works URLs section below. Other applications which support single sign-on links include Scribe and Administrative Shell. Student Planner and Transfer Finder cannot be linked directly from a portal; they have to be accessed through Dashboard Servlet.

Example Degree Works URLs for single sign-on links

When Degree Works applications and Luminis are configured to use the same single sign-on provider, linking between the applications is simple. Formulate a URL which includes any desired instructions for Degree Works and implement it using a Luminis feature like a bookmark link. For Dashboard Servlet, no special instructions are needed in the URL to indicate that CAS should be used. If one of those providers is configured and enabled, single sign-on will take effect when a user requests a URL for that application.

Here are some example URLs that can be used for single sign-on links from Luminis to Degree Works applications. None of these are specific to Luminis, the same examples would work for any portal and they can be bookmarked by end users. Curly braces indicate placeholders for site-specific choices.

Link to the Worksheets tab in Responsive Dashboard:

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

Link to the Exceptions tab in Responsive Dashboard:

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

Link to the Plans tab in Responsive Dashboard:

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

Link to the main Dashboard Servlet frameset:

```
https://{server}/{deployment-path}/dashboard?SERVICE=SCRIPTER&SCRIPT=SD2WORKS
```

Retrieve a student's own audit from Dashboard Servlet:

```
https://{server}/dashboard?SERVICE=SCRIPTER&SCRIPT=SD2GETMYAUDIT&ACTION=REVAUDIT&
REPORT=WEB31&ContentType=xml
```

Linking to Scribe or Administrative Shell is as simple as linking to the root of their deployments. Make sure to consider these administrative applications should be visible only to certain users:

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

Self-Service Banner (Banner 8)

Single Sign-on for Degree Works

Introduction

These instructions will guide the user through configuration of Self-Service Banner (SSB) and Degree Works (DW) for single sign-on integration.

Overview of single sign-on process

The purpose of these installation steps is to make a new menu item available in Self-Service under the tab where it is configured. When that menu item is selected the link will send a single sign-on request to Degree Works then cause the Degree Works application to be displayed if the single sign-on request is validated. If it is not validated then an error will be displayed.

The recommended means of accomplishing single sign is using Ellucian Identity Services (EIS) which supports SAML and CAS in Dashboard Servlet.

Menu setup for Student role

This guide covers how to set up a PL/SQL package to be called when a user selects a menu item. The package generates HTML markup which defines how the page looks and works in the browser. The packages provided with Degree Works are meant to be used as an example, so it would be best to decide on how these links are desired to function and appear before proceeding. The persons who implement this should be familiar with PL/SQL, HTML, Javascript in the browser, and have access to WebTailor Administration. Code modifications will be necessary depending on if you are using native SSB single sign-on, or an external single sign-on provider like CAS.

When modifying these sample SQL files, first copy to a different location and maintain your changes separate from the Degree Works installation package because these files could be updated in future release. Consider using a source control system to maintain your code.

1. No extra grants are needed for these packages. You already ran the bannergrants.sql script to run the necessary grants.

2. Locate the dwssbstudent.sql file in the app/sql directory on the DW host server (cd app/sql). If your institution uses something other than WWW_USER for Banner Self-Service connections, edit the grant statement in dwssbstudent.sql and dwssbfaculty accordingly. Use the shortcut script dbb to connect to your Banner database in sqlplus, and then run the file by issuing @dwssbstudent.sql. This will create the package called “DW_Student” in the database.

3. Log in to SSB with a user who has access to WebTailor Administration. Select the WebTailor tab, then “Web Menus and Procedures” from the menu.

4. Click the “Create” button to add a new web menu or procedure. Enter data for the following required fields. Enter data for the other fields according to your preference.

    a. Page Name: DW_Student.P_SignOn.

    b. Description: Degree Works. Or, enter whatever text you would like to appear as the description.

    c. Module: Student Self-Service. Or, select the appropriate module for your site.

    d. Enter page title, header text, and header graphic as you prefer.

    e. Enter back link settings as you prefer. If you are adding the Degree Works menu item to the student main menu then the back link url would be:
    `twbkwbis.P_GenMenu?name=bmenu.P_StuMainMnu`

    f. Choose Associated Roles according to your preference.

5. Steps 5 and 6 are options dependent on how you want to display or call the DW package from within Self-Service. Display the Degree Works link on a main menu by calling the new web procedure from your main menu package: DW_Student.P_SignOn (term, pidm, 0). The first two arguments, term and pidm, can be sent as null. The third argument, show_headers, should

equal 0 (zero) when you are calling this procedure from another package. If you choose to do this step, you can skip step 6 below.

6. Display the Degree Works package link on a main menu. Do this if you skipped step 5 above. This option will allow you to display a full page containing the DW link and any additional information you wish to provide.

    a. Go back to WebTailor > Web menus and Procedures.

    b. Select the menu where you want to add Degree Works (for example the main menu bmenu.P_GenMnu or the student main menu bmenu.P_StuMainMnu).

    c. Click “Customize Menu Items”.

    d. Click “Add a New Menu Item” (If you do not see that button yet, then you will need to click “Copy Baseline to Local” first).

    e. Enter the URL: DW_Student.P_SignOn

    f. Enter the link text, description and sequence number according to your preference.

    g. Check the box next to Database Procedure.

7. Configure WebTailor Parameters, as follows:

    a. DWLINKTEXT – The text you want to display as the link to Degree Works.

    b. DWURL – The URL to Degree Works.
    You should enter the Gateway URL + the context path of the Dashboard. This would be the same composite URL that you would see in your browser window if you just type in the Gateway URL and access the Dashboard that way.
    For instance, if your Gateway URL is https://your.server.com:8471. And you have your Dashboard deployed at /Dashboard, your DWURL WebTailor parameter should be set to:
    https://your.server.com:8471/Dashboard/

    c. DWDISPLAYBUTTON – Set to “1” to have a button displayed instead of an automatic redirect.

Menu setup for Faculty role

Setting up a menu item for a Faculty role is similar to the steps above with some differences listed as follows:

1. The sql file `app/sql/dwssbfaculty.sql` contains another package DW_Faculty with a procedure called P_SignOn. The default LOGONUSERCLASS in this package is “ADV”. If you are providing this link to users who are ADVX or REG instead of ADV, then modify the hardcoded value for LOGONUSERCLASS in this file. Insert the package into the Banner database using dbb.

2. The “Page Name” in step 4.a. above will be: DW_Faculty.P_SignOn.

3. Choose a module and associated roles that appropriate for faculty members.

4. Add the menu item for faculty members to a menu that is appropriate for your site. One option is to add it to the bmenu.P_FacStuMnu menu.

Installation Steps: Degree Works server

1. To redirect users to the SSB page when they click “Logout”, configure the following settings in `dwenv.config`:

   ```
   ENABLE_EXTERNAL_LOGOUT=1
   gsCfgExternalLogoutUrl=http://<Self-Service Banner URL>
   ```

   Change the example URL to your Banner Self-Service URL.

2. Enable the link back to SSB in the header frame. Use the Configuration tab in Controller to set `localization.dashboard.header.showPortal` to true, which will display the link back to SSB in `SD_HeaderFrame.jsp`. When the user clicks “Log Out”, the session will be ended and the user will be taken back to the URL configured above.

   You can localize the text of both the “Log Out” and “Back to Self-Service” links. Use Composer to create a localized version of DashboardServletMessages.properties and change the value of `dw.dashboard.header.link.logOut` and `dw.dashboard.header.link.backToSelfService`.

3. The URL to simply go back to SSB without logging out can be configured in the SD2WORKS shpscripts file. Use the Composer application to create a localized version of SD2WORKS and then locate the following code in the file under the `function BackToSelfServiceBanner(sMode)` section:

   ```
   window.location.href = "http://yourserver/somepath/" + sMenuName;

   } // backtoselfservicebanner
   ```

   Change the example URL to your SSB URL.

4. In that BackToSelfServiceBanner function, there is code to also log out of Degree Works before navigating back to SSB. If you would rather let the user remain logged in to Degree Works (until user chooses to Log Out or until their timeout occurs), then you can remove or comment out this code in localized version of SD2WORKS:

   ```
   // Send an asynchronous request to log out of Degree Works before
   navigating back to SSB
   // Localize this if you don't want users to log out of Degree Works
   when navigating.
   $.ajax(
   {
      async: false,
      url: top.sLogoutUrl
   });
   ```

Student ID Pass-along to Degree Works

Overview

An advisor working in SSB must select a student before clicking on Degree Works. When the user does choose Degree Works the student being reviewed is passed to the student context area within Degree Works. As soon as the student appears the Worksheet tab is automatically selected and the student's most recent degree audit is displayed.

Allowing an advisor user coming from SSB to switch to another student in Degree Works would cause much confusion when the user then switched back to SSB since the new student ID is not then pushed back to SSB - they would be surprised to see the old student ID still sitting there in SSB.

To prevent such confusion it is best to take away the ability for these users to switch to another student within Degree Works. To remove this ability you should remove the SDSTUANY and SDFIND keys from the ADV and/or ADVX groups in Controller.

How this works

In the Responsive Dashboard, the student ID needs to be passed in the URL when connecting to Degree Works using this format:

https://<your server name>/<path to Degree Works>?**studentId**=<someid>

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

In the Dashboard, the link from SSB requests the SD2WORKS Degree Works script. When the request is made the student ID from SSB is sent to Degree Works as PORTALSTUID=<someid>. The SD2WORKS script grabs this student ID and passes it along to the SD2STUCON student context area script. Once the SD2STUCON script sees that an ID was passed in it checks to see that the user is not a student and then immediately loads that student's name, degree, etc. If the user does not have the SDFIND or SDSTUANY keys they will not be able to switch to a new student; switching students must occur in SSB.

User Role Pass-along to Degree Works

Overview

An advisor working in SSB may be reviewing data on her advisees or may be looking at her own student records. The advisor would have two Degree Works links – one on the SSB student tab and one on the SSB faculty tab. When the advisor is working on one of her advisees and clicks the Degree Works link she needs to be able to continue to play the role of an advisor when in Degree

Works. Conversely, when she is examining her own student record in SSB and clicks the Degree Works link she needs to play the role of a student when in Degree Works.

The above section on passing the Student ID from SSB to Degree Works is tightly related to this topic and thus the same adjustments to the advisor's keys need to be made.

Please note that the user-class cannot be passed into the Responsive Dashboard. This ability to pass the user-class is only valid for the Dashboard.

How this works

The faculty tab containing the link to Degree Works must pass the LOGONUSERCLASS of ADV (or ADVX) to Degree Works. The student tab must pass the LOGONUSERCLASS of STU to Degree Works.

The user must have been extracted from Banner as an advisor at some point. This advisor user-class is stored in the shp-user-mst in Degree Works as the primary/overall user-class. When the user connects to Degree Works from SSB the LOGONUSERCLASS passed from SSB is stored in the shp-passport-mst table as the dynamic one to use for the current session. The primary user-class in the shp-user-mst is used to prevent a user with a primary user-class of STU from being changed to another other user-class. Similarly, a primary user-class of REG cannot be overwritten with any other user-class.

Banner Student Profile (Banner 9)

For information on configuring single sign-on with Banner Student Profile in Banner 9, please refer to the *Banner Student Self-Service Installation Guide*.

Integration with Responsive Dashboard

The student ID needs to be passed in the URL when connecting to Degree Works using this format:

https://<your server name>/<path to Degree Works>?**studentId**=<someid>

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

Examples:

Link to the Worksheets tab in Responsive Dashboard:

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

Link to the Exceptions tab in Responsive Dashboard:

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

Link to the Plans tab in Responsive Dashboard:

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

Integration with Dashboard

The link from SSB requests the SD2WORKS Degree Works script. When the request is made the student ID from SSB is sent to Degree Works as PORTALSTUID=<someid>. The SD2WORKS script grabs this student ID and passes it along to the SD2STUCON student context area script. Once the SD2STUCON script sees that an ID was passed in it checks to see that the user is not a student and then immediately loads that student's name, degree, etc. If the user does not have the SDFIND or SDSTUANY keys they will not be able to switch to a new student; switching students must occur in SSB.

Link to the main Dashboard frameset:

```
https://{server}/{deployment-path}/dashboard?SERVICE=SCRIPTER&SCRIPT=SD2WORKS
```

Using Degree Works for Prerequisite Checking in Banner

For information on configuring and using Degree Works for prerequisite checking in Banner, see the *Prerequisite Checking Technical Guide*.

SAP - Satisfactory Academic Progress

Overview of SAP

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 towards earning their degrees. SAP determination now has the ability to 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.

As of Banner 8.5.3, a new form SHISAPP and new database tables are used with Degree Works degree audit processing to capture the data used for checking satisfactory academic progress. A new Degree Works processor, BAN62, will extract course information from a student's degree audit, and store it in the Banner database. For more information on Banner's SAP process, see the *Banner Student 8.5.3 Release Guide*.

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. See the BAN62 section in the *Transit User Guide* for more information on running this processor.

Before running BAN62, check the following items:

- The **UCX-CFG020 DAP14 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.
- Add a new freeze status to UCX-AUD032 if you wish to freeze audits which are generated for input into the BAN62 processor. See *Freezing your SAP Audits* in this document, for more information.

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”, then 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.

Freezing your SAP Audits. It is recommended to freeze audits which 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. See the section **UCX-AUD032 Audit Freeze Types** in the *Degree Works Configuration Technical Guide* for more information on adding and configuring entries in UCX-AUD032.

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. See the *MEP (Multiple-Entity Processing) in Banner* section of this document for more information. 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* section in this document, 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 OTL (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 OTL 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 OTL (Over the Limit), Fall-through or Insufficient sections, then a SHRSARJ record will be written with the rejection reason of “***In-Progress: audit was run with ApplyInProgress=N***”. This indicates that the audit should be rerun for the student with one of these options:

- set the **UCX-CFG020 DAP14 Apply In Progress** flag to “Y”
- when running BAN62 in Transit, un-check 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 OTL (Over the Limit), Fall-through, Insufficient or In-Progress sections, then the class will be written to SHRSARJ with the reject reason "***Unknown where class applies in audit***". 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 rejection reason of "***Rejected reason undefined***" 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, check the spool files in the *$ADMIN_HOME/dgwspool* directory. Two files will be created for each run: `ban62XXXXA.act` and `ban62XXXXL.log` (where XXXX is the Job Number from Transit). 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 that 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 Service Request should be opened with the Degree Works Action Line if this situation occurs.

SHRSAPP Table Layout

The following table lists the columns from the Banner table SHRSAPP, and specifies how each column is populated.

Column Name	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 • 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 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.

Column Name	Populated with
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 SHRGRDE_ATTEMPTED_IND = Y. Transfer: SHRTRCE_CREDIT_HOURS if associated SHRGRDE_ATTEMPTED_IND = Y. Null for In-Progress courses.
SHRSAPP_HOURS_PASSED	The credit hours passed for Historic and Transfer courses. Historic: SHRTCKG_CREDIT_HOURS if SHRTCKN_REPEAT_COURSE_IND = “I” or is blank, and the associated SHRGRDE_PASSED_IND = Y. Transfer: SHRTRCE_CREDIT_HOURS if SHRTRCE_REPEAT_COURSE = “I” or is blank, and the associated SHRGRDE_PASSED_IND = Y. Null for In-Progress courses.
SHRSAPP_HOURS_EARNED	The credit hours earned for Historic and Transfer courses. Historic: SHRTCKG_CREDIT_HOURS if SHRTCKN_REPEAT_COURSE_IND = “I” or is blank, and the associated SHRGRDE_COMPLETED_IND = Y. Transfer: SHRTRCE_CREDIT_HOURS if SHRTRCE_REPEAT_COURSE = “I” or is blank, and the associated SHRGRDE_COMPLETED_IND = Y. Null for In-Progress courses.
SHRSAPP_HOURS_GPA	The credit hours used for GPA calculation for Historic and Transfer courses. Historic: SHRTCKG_CREDIT_HOURS if SHRTCKN_REPEAT_COURSE_IND = “I”, or is blank and the associated SHRGRDE_GPA_IND = Y. Transfer: SHRTRCE_CREDIT_HOURS if SHRTRCE_REPEAT_COURSE = “I” or is blank, and the associated SHRGRDE_COMPLETED_IND = Y. Null for In-Progress courses.

Column Name	Populated with
SHRSAPP_QUALITY_POINTS	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. Historic: SHRGRDE_QUALITY_POINTS * SHRTCKG_CREDIT_HOURS if the associated SHRGRDE_GPA_IND = Y and SHRTCKN_REPEAT_COURSE_IND = “I” or is blank. Transfer: SHRGRDE_QUALITY_POINTS * SHRTRCE_CREDIT_HOURS if the associated SHRGRDE_GPA_IND = Y and SHRTRCE_REPEAT_COURSE = “I” or is blank. Null for In-Progress courses.
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	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. Historic courses: SHRTCKN_REPEAT_COURSE_IND. Transfer courses: SHRTRCE_REPEAT_COURSE. Null for In-Progress courses.
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

The following table lists the columns from the Banner table SHRSARJ, and specifies how each column is populated.

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.

Special Topics

To help you use Degree Works effectively, there are a variety of special topics that can warrant discussion and elaboration. These topics are typically generated from customer feedback when it becomes clear that an extended explanation is needed on some specific issue. The topic often references other documents that contain the specifics of configuration. The special topic can take the form of an abbreviated “how to” document.

Applicants in Degree Works

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

Advisors and REG users can see applicants within DGW just as they see other “regular” students. Applicants must log into DGW 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 into DGW.

There is not a universal “GUEST” applicant extract or login* that recruiters or admissions officers can use to log potential students into DGW.

*Unless the school has created a GUEST type user which they can then use to access DGW. This is a topic for another discussion though.

How to extract applicants

1. Use the REFRESH button (will work only if UCX-CFG020/BANNER->Process_Applicants = “Y”),

2. Use the launchjob script in cron.

3. Transit
   Select RAD33 as the processor
   Must use the SQL file or supply a list of IDs.

Configuration Flags

There are a few configuration flags that will 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

When the REFRESH button is pressed 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`.

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`.

Once extracted 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

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:

1. SORLCUR2
2. SORLFOS2
3. SARADAP

Applicant User Class

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.

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

By default, the SRNAPP group has the following accesses:

SDAUDREV

SDLOKAHD

SDSTUME

SDWEB31

SDWHATIF

SDWORKS

SDXML31

SDAUDPDF

Financial Aid in Degree Works

You should setup 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.ROBINST_AIDY_CODE FROM ROBINST b
AIDYEAR:WHERE_4                  WHERE b.ROBINST_STATUS_IND = 'A')

AIDENRSTATUS:AID             ENROLLSTATUS
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
```

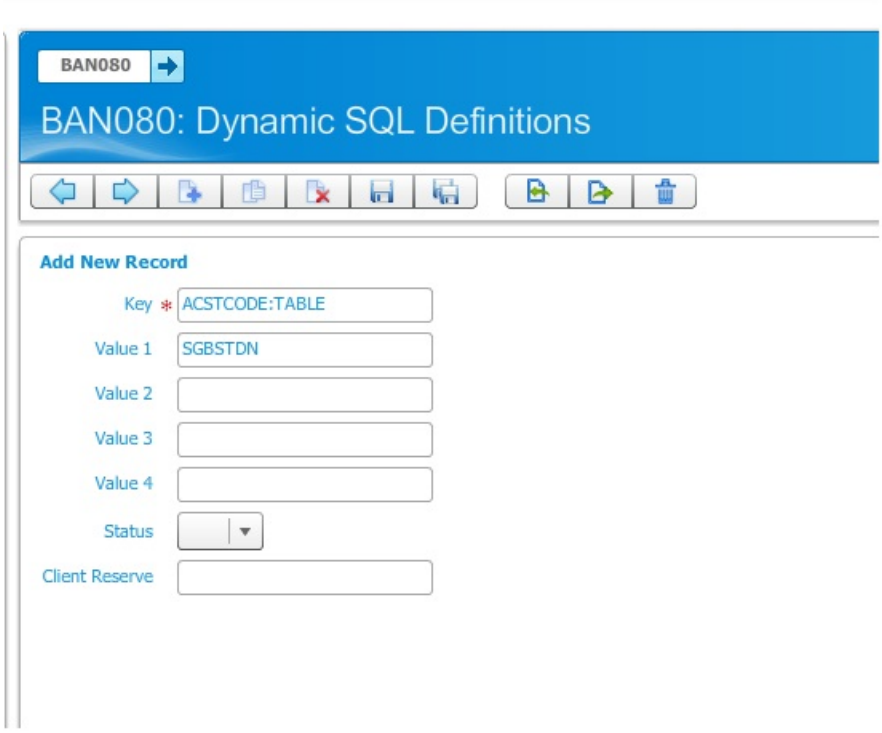
Using Banner Data to create Scribe Custom Data

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.

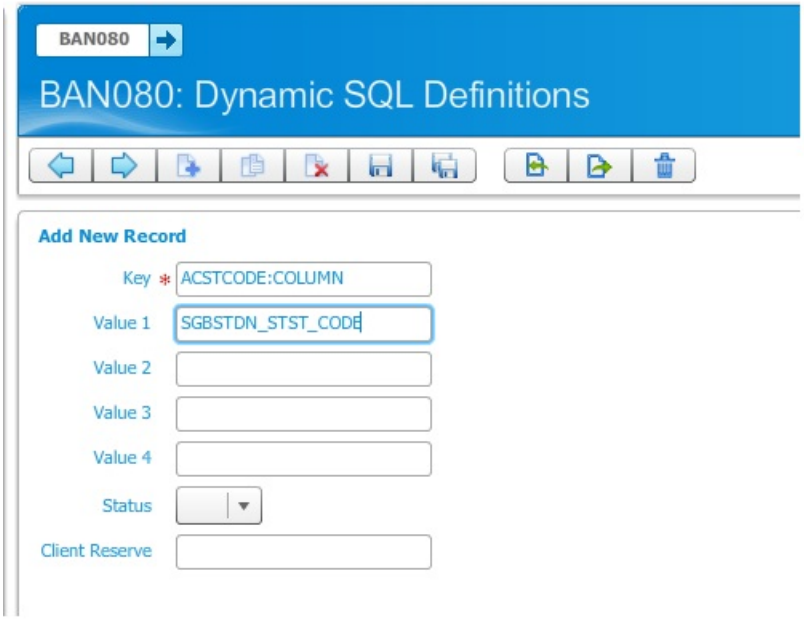
1. Create the variable in the BAN080. In this table you will indicate the column, table, and where statements to retrieve the variable from Banner. The following shows an example of the code set up for the academic standing code. You pick a name for this variable. We will call it ACSTCODE.

   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.



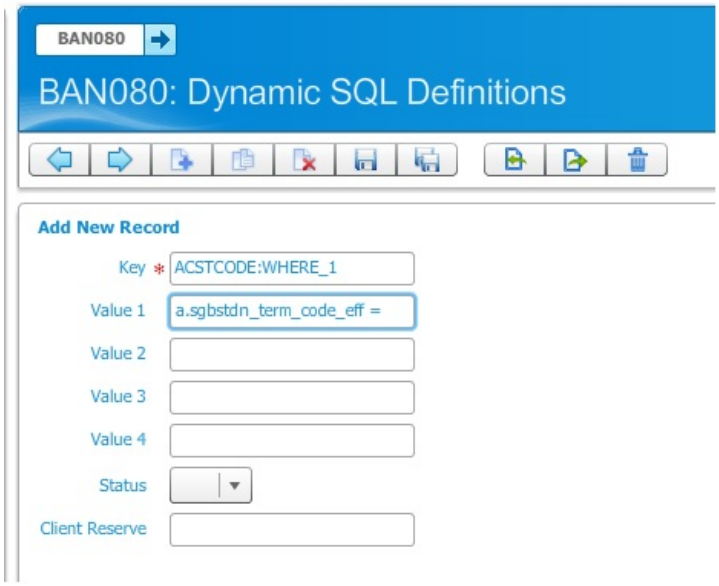
c. Any SQL statements that will be used to find the desired instance of the variable should now be entered into records with keys beginning with WHERE. For multiple where

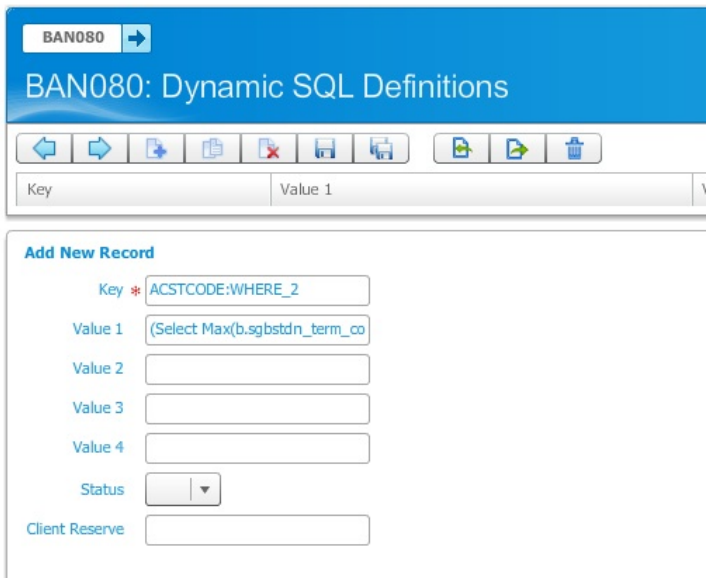
statements, add multiple where records. These records should be named WHERE_1, WHERE_2, etc. In this example, we will find the academic status value for the latest term. The SQL to accomplish this will be:

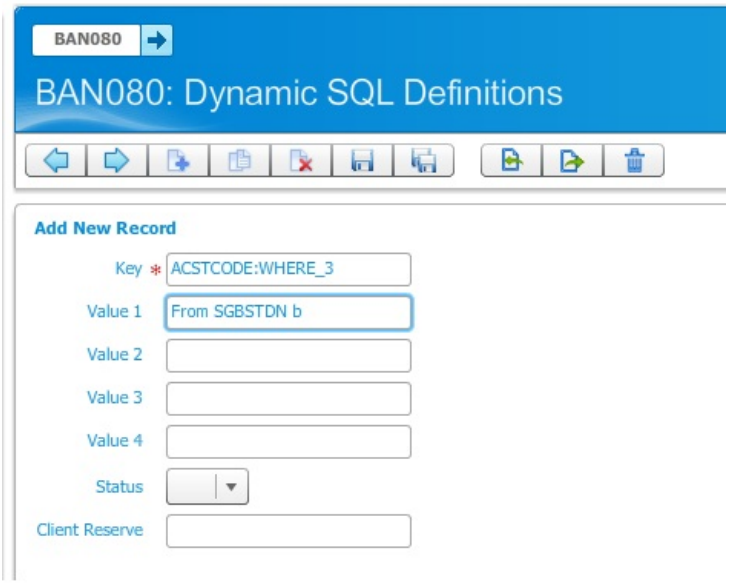
```
Where a.sgbstdn_term_code_eff = (Select Max(b.sgbstdn_term_code_eff)
from
Sgbstdn b where b.sgbstdn_pidm = a.sgbstdn_pidm)
```

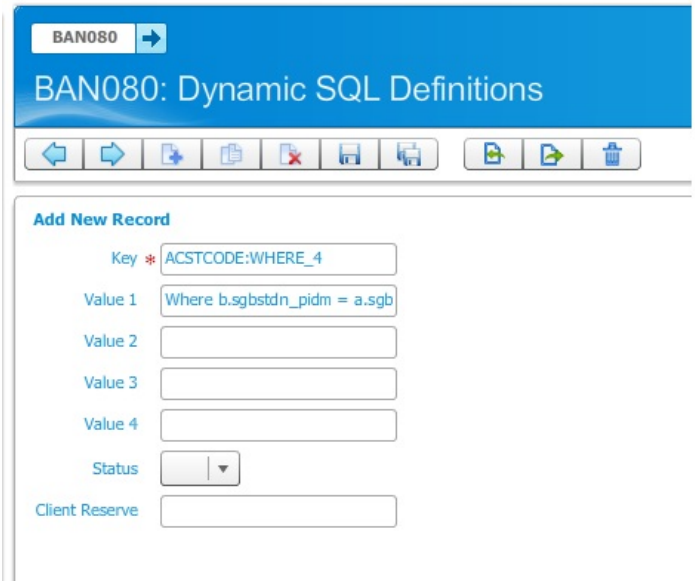
In this example, four records will need to be created. They can be named: ACSTCODE:WHERE_1, ACSTCODE:WHERE_2, ACSTCODE:WHERE_3, and ACSTCODE:WHERE_4.

Screen shots of these are as follows:









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

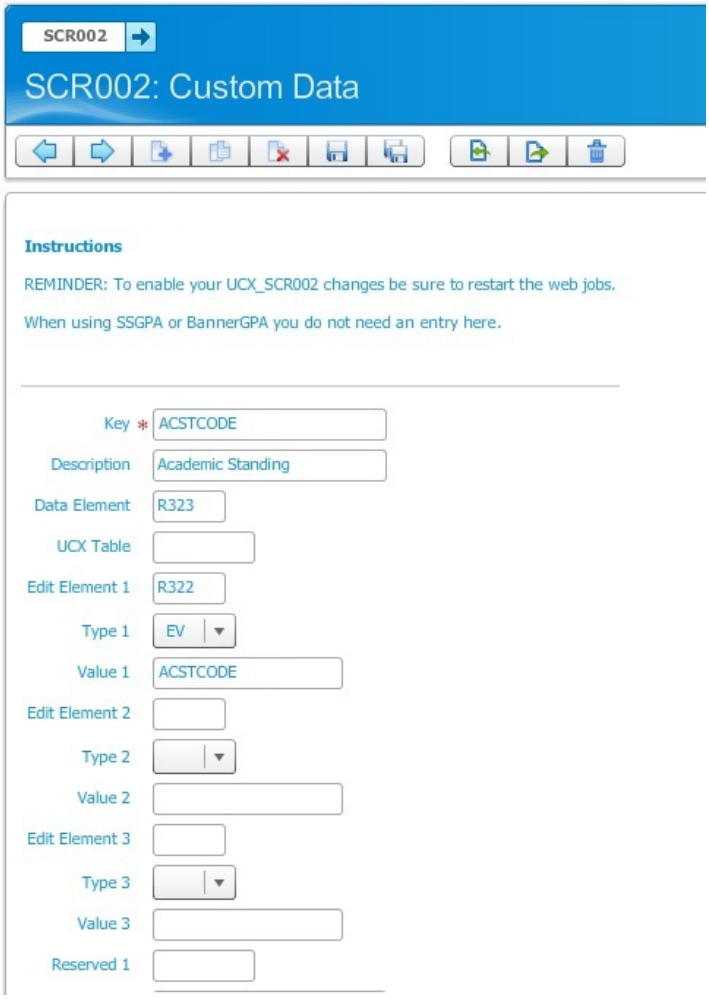
ACSTCODE:WHERE_4	Where b.sgbstdn_pidm = a.sgbstdn_pidm)

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.

   Since 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 since this is coming from BAN080 data.

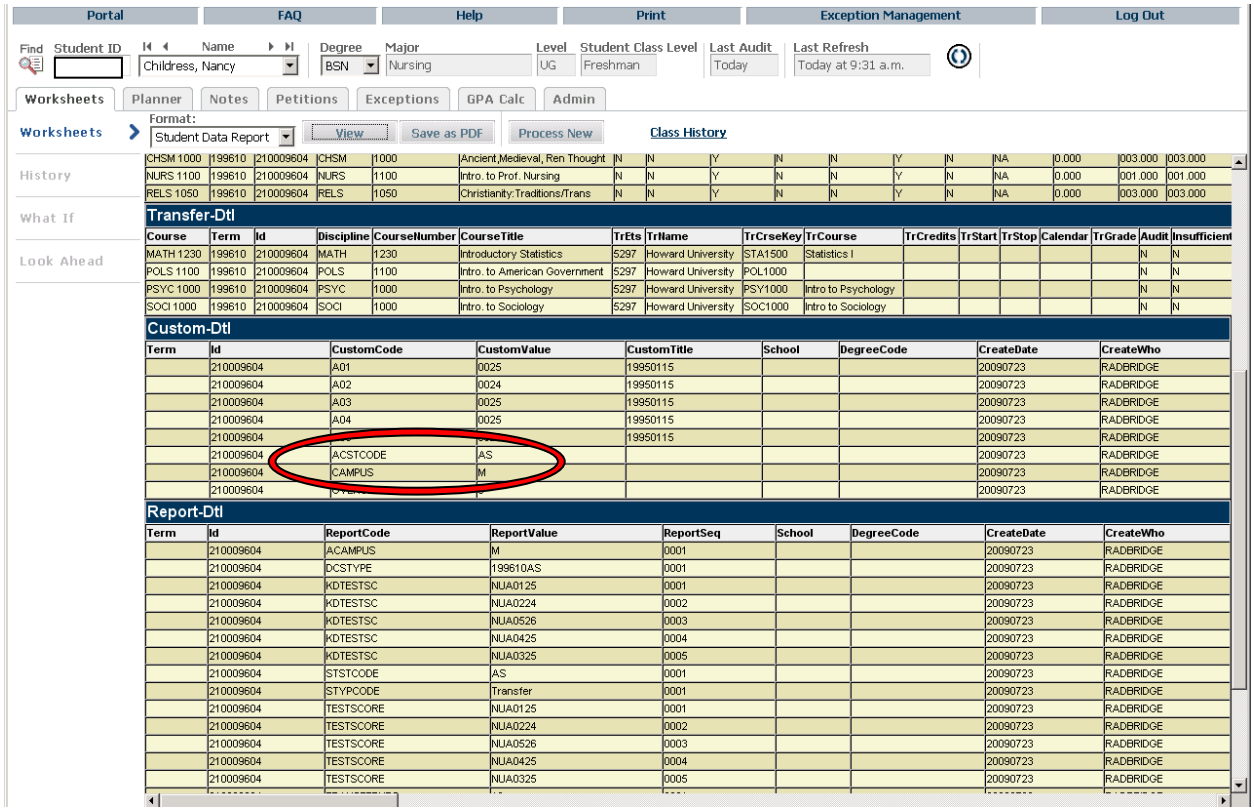
   The Edit Element1 should be the value of the data item. In this case, 'R323' in SYS999 points to the rad_custom_value field, this should be entered into the Edit Element1 field. We will be retrieving all the values for this data item, so the Type value should be set to EV. Finally the value of the SCR002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

The following is the screen shot of the SCR002 record:

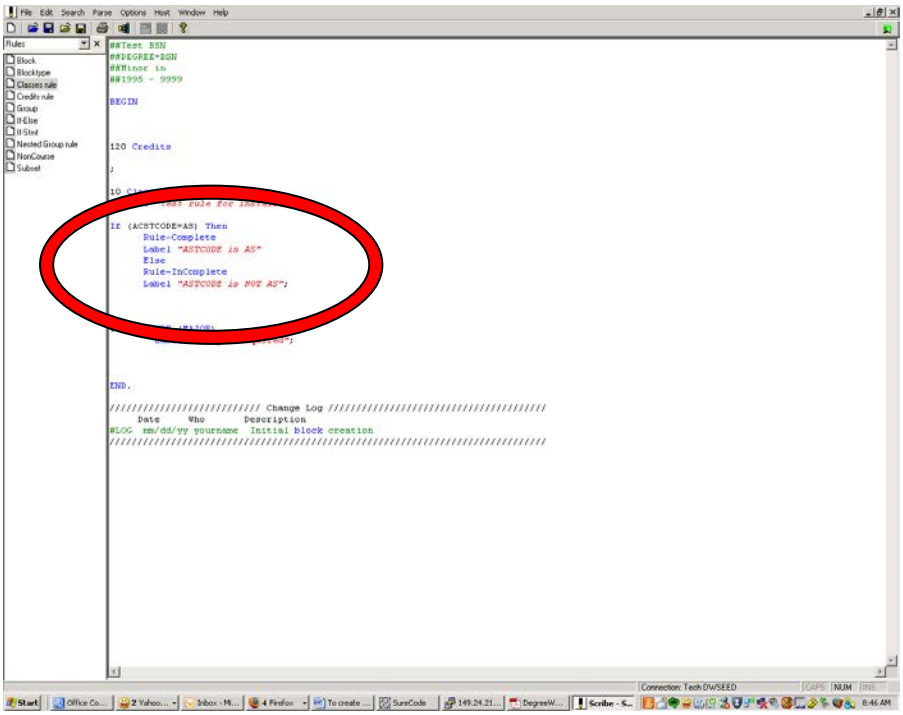


3. After the tables have been set up in Controller for the BAN080 records and the SCR002 record. A webrestart and a UCX12job command should be issued. For students to get this data put into their rad_custom_data table, they will need to be re-bridged from Banner into Degree Works. If the data item does not exist in Banner for the student, the record will not be loaded in the rad_cust_dtl table.

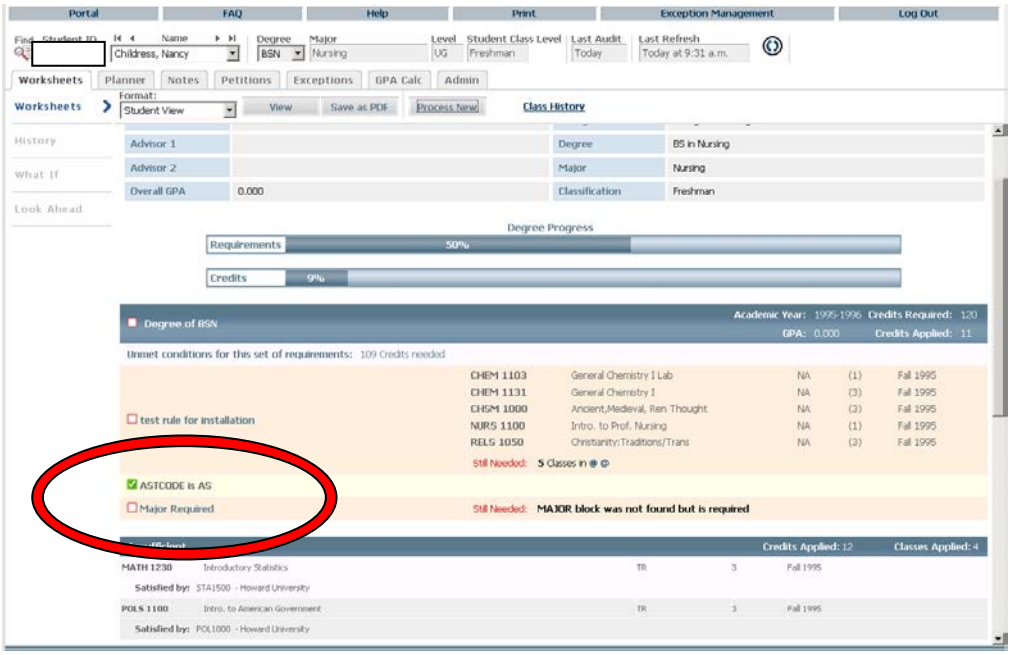
   Once a student has been re-extracted, his student data record will now look similar to the following screenshot:



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. An example using the ACADSTST code is:



5. This rule will then look like the following once an audit is run:

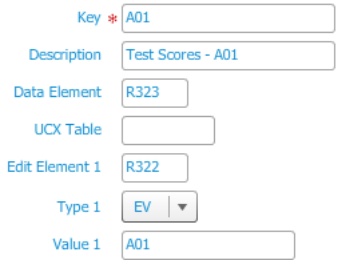


Scribing against Test scores

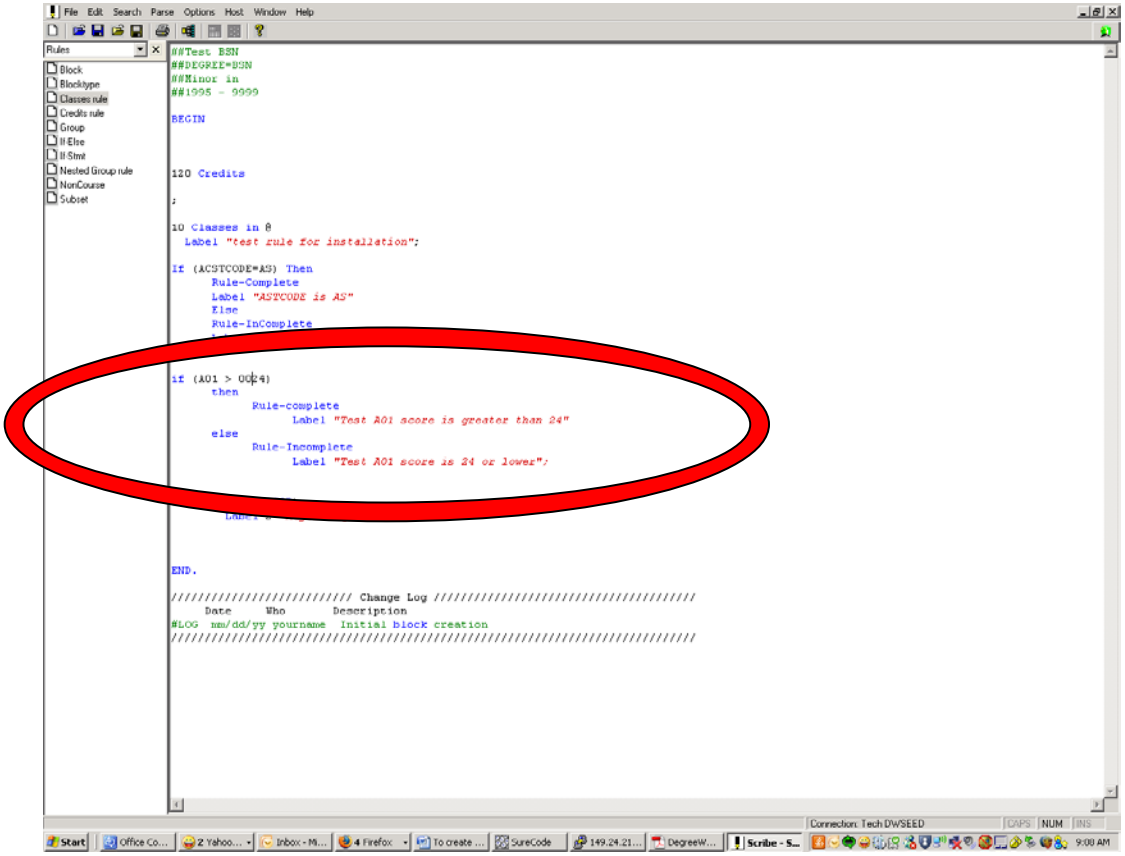
Test scores are brought into the DGW rad_custom_dtl table from the SORLCUR table by default based on your `integration.banner.extract.config` setting. By default, all test scores are retrieved. You can modify `integration.banner.extract.config` to delimit what test scores are brought in by adding WHERE statements into the `integration.banner.extract.config` setting. Since 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:

```
SORTEST_TESC_CODE    SORTEST_TEST_SCORE    SORTEST_TEST_DATE
-----------------    ------------------    -------------------------
A01                    25                  15-JAN-95
A02                    24                  15-JAN-95
A03                    25                  15-JAN-95
A04                    25                  15-JAN-95
A05                    26                  15-JAN-95
```

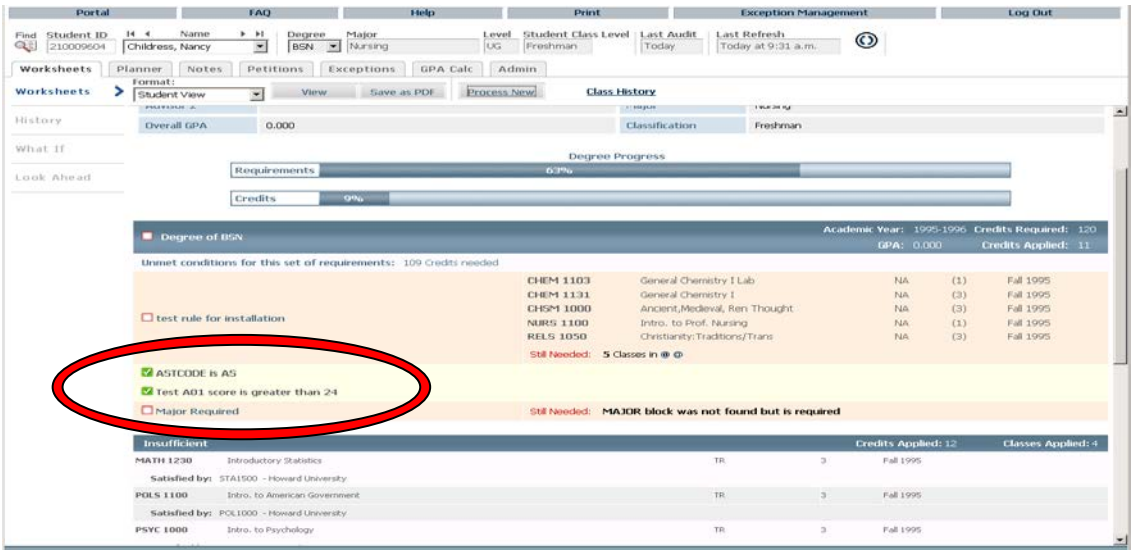
To be able to scribe against one of these, the test code will need to be added to the SCR002 table. For example to scribe against test types of A01, the following SCR002 record needs to be created:



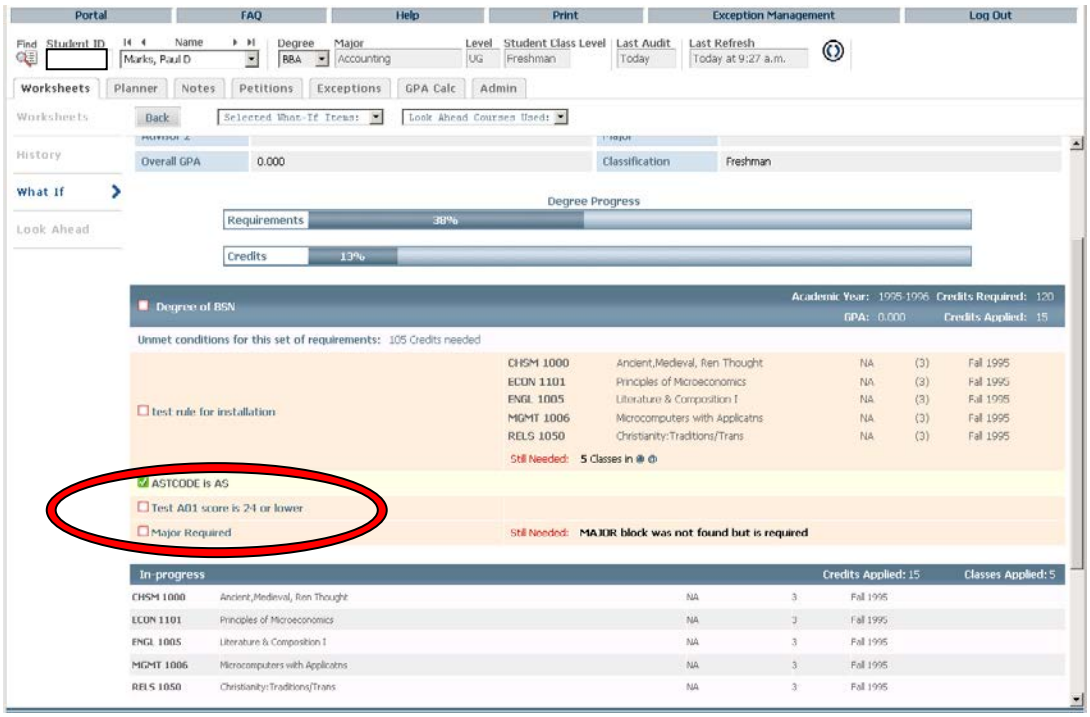
Then in Scribe the following rule can be put into place:



Running the audit with our test student gives the following result:



Running an audit with a student who does not meet this qualification or has not taken this test will get the following result:



Populating SHP_USER_ATTRIB from Banner data

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 wish 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:

Key	Value 1
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	WHERE b.SHRTTRM_PIDM = a.SHRTTRM_PIDM)

2. Add a new entry in BAN080 where the key is the custom code followed by a : 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.