ellucian

Degree Works
Scribe User Guide

Release 4.1.4
August 2014

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

Revision History

Publication Date	Summary
August 2014	New version that supports Degree Works 4.1.4 software.

Contents

Introduction

This guide provides the basic operational instructions for Scribe, the PC Degree Works tool for codifying and storing degree and program requirements. Scribe stores these requirements in units called blocks. When Degree Works produces a student worksheet, its auditor matches the courses on a student's record against the requirements in the Scribe blocks that contain the student's degree requirements.

The heart of the Scribe program is a simple programming language for encoding degree requirements. The two main tools in this language are rules and qualifiers. As with any programming language, Scribe requires an exact syntax and precise articulation of requirements. Scribe is powerful, sophisticated, and adaptable.

Entering degree requirements is labor intensive; however, Scribe is compatible with word processing and spreadsheet programs, so it is possible to copy text and data from these formats into Scribe.

Before a requirement block can be added to the Degree Works database, it must be checked for syntax. In Scribe, syntax is checked a parser utility.

This guide is intended to provide an introduction to Scribe. Similar guides provide introductions to the other Degree Works tools. For more complete descriptions of Degree Works functions, please refer to the technical documentation.

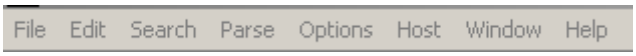
The Scribe processes are listed below as learning objectives:

After reading this guide you will be able to:

- Define basic course rules
- Understand the use of qualifiers
- Parse block requirements
- Assign database tags
- Create and save a block
- Recall and edit an existing block
- Update a set of requirements
- Identify comments, Proxy-Advice, and remarks
- Understand the use of subset rule types
- Understand the Group rule
- Understand If statements and NonCourse rules
- Describe the various block types and multi-block programs
- Recognize Scribe’s Reserved Words
- Recognize the scope of Reserved Words
- Define Reserved Words
- Create and save a block

Menu

The Scribe icon invokes the Degree Requirements Data Entry Interface. The menu choices are as follows:



FILE	EDIT	SEARCH	PARSE
New (Ctrl+N)	Cut (Ctrl+X)	Find (Ctrl+F)	Parse Current
Open (PC) (Ctrl+O)	Copy (Ctrl+C)	Find Next (F3)	Block(Ctrl+B)
Open (Host DB) (Ctrl+H)	Paste (Ctrl+V)	Find Previous (Shift+F3)	View Errors
Save (PC) (Ctrl+S)	Select All (Ctrl+A)	Replace	
Save (Host DB)(Ctrl+T)	Syntax Highlight		
Save As (PC)	(Ctrl+Y)		
Save As (Host DB)	Wordwrap Window		
Delete from Host DB			
Close			
Print... (Ctrl+P)			
Exit			
[file open history]			

OPTIONS	HOST	WINDOW	HELP
Preferences	Logoff	Cascade	SCRIBE Help... (F1)
	Logon	Tile Horizontally	Tech Support
	Refresh Picklists	Tile Vertically	About SCRIBE
		[list of open blocks]	

Overview of Key Terms

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

Every block consists of two sections: the **Header** and the **Body**. The block Header consists of **Header qualifiers** that apply to the block as a whole. Not all Reserved Words can be used in the Header (see Scope of Reserved Words listed at the end of this guide). The block Header section starts with the "BEGIN" command and ends at the first semicolon in the block. The Body section starts after the first semicolon and continues to the "END." command. The Body may consist of course requirements, noncourse requirements, block requirements, **rule qualifiers** that may apply to a particular rule, Proxy-Advice, and remarks.

The following is an example of a simple block: **Note: Reserved Words are not case sensitive. The examples are for readability. Credits could be written as CREDITS, Credits, or credits.**

```
BEGIN

10 Credits
MinGPA 2.0

;

  5 Credits in ENGL 101
    Label "Freshman Composition I";

  5 Credits in MATH @
   MinGrade 2.5
    Label "Elective: Any Class in Mathematics";

END.
```

This block consists of two Header qualifiers ("10 Credits") and ("MinGPA 2.0"). For the block to be completed, at least ten credits must be earned. A cumulative grade point average of 2.0 or higher is required for credits applying to the rules in the Body. Header qualifiers will be discussed in more detail in the "Qualifiers" section of this guide.

The Body of the sample block contains two course requirements. Course requirements will be described in more detail in the "Basic Course Rules" section. Some examples follow:

- Course rules follow a particular syntax. Courses are designated by discipline code and followed by a space, which is followed by the course number. The "@" symbol is a wild card, which means any number is allowed. Wildcards may also be used to specify any discipline (3 Credits in @ = three credits in anything).
- Course rules allow for the use of course labels, which must be enclosed in quotation marks and may be no longer than 50 characters in length.
- Each rule ends with a semicolon.
- Rule qualifiers are placed after the statement of the rule and before the label. In the sample block above, the second course rule has a "MinGrade 2.5" qualifier, which means that a grade of at least 2.5 must be earned for a course to satisfy the rule.

Blocks are stored in the Degree Works database on the system mainframe computer for use in running degree plans. Degree Works refers to the mainframe computer as the **Host**. Blocks can also be stored on the PCs of individual Scribe users. Blocks stored on an individual PC are not available to the Degree Works program, but the PC is a useful place to store sample blocks, test blocks, and partially completed blocks.

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

AWARD (for Financial Aid audits)

COLLEGE

CONC (Concentration)

DEGREE

LIBL (Liberal Learning)

MAJOR

MINOR

OTHER (user defined)

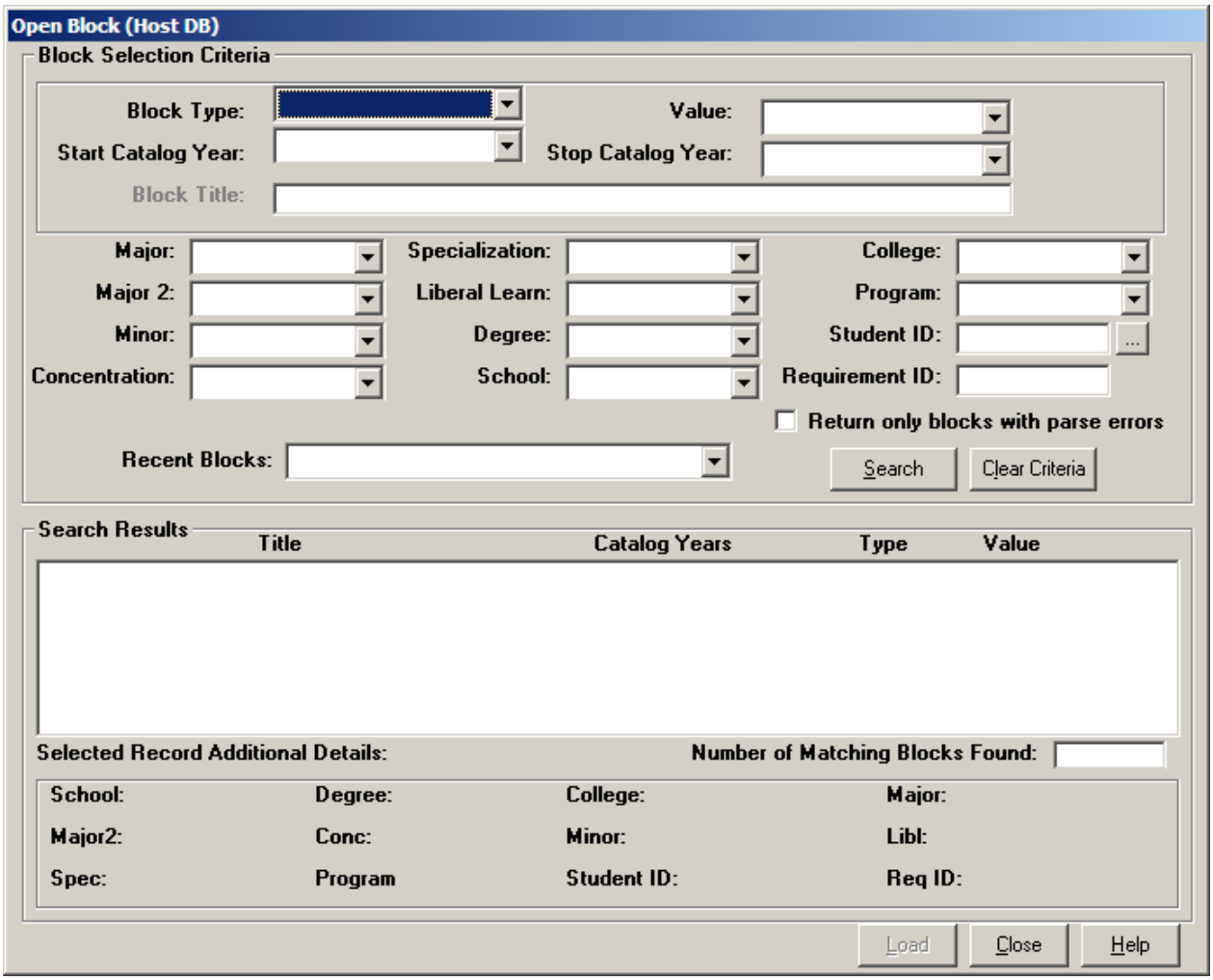
PROGRAM

SCHOOL

SPEC (Specialization)

ID (Identification)

When a block is stored to the host, it is given a set of **database tags**. These tags are the characteristics that Degree Works uses to match blocks to students. The primary tags include Block Type, Value, Start Catalog Year, Stop Catalog Year, and Block Title. The Start and Stop Catalog Year tags determine the range of time over which the requirements in the block are effective. The Block Title is used in worksheet reports. Database tags will be discussed in more detail in the section "Saving Blocks to the Host Database". The followin is an example of the Scribe Block Selection Criteria screen showing the database tags.



Basic Course Rules

There are four different formats for writing a course requirement in the Scribe language:

1) Credits(s)
2) Class(es)
3) Credits and Classes
4) Credits or Classes

The syntax for the different rules is similar. The most common format is Class(es). We will focus on this kind of course statement. Instructions and examples for the other formats can be found elsewhere in this documentation.

Some general comments about the Class(es) rule are as follows:

1) The rule must end in a semicolon.
2) The label must be enclosed in quotation marks and is limited to fifty characters. The label will appear on the student worksheets. It is suggested that the label contain a course title or a description of the requirement, but the label can contain any information the user desires. Leading spaces in a label are not allowed.
3) Carriage returns and spaces can be included in order to make the rule easier to read in Scribe. These have no effect on the appearance of student worksheets, however.

Here are some examples of the Class(es) rule at work.

```
1 Class in SPAN 101 or FRE 101
  LABEL "Intro. Spanish or Intro. French";
```

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

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

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

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

The rule will be satisfied by taking five to ten credits of English at the 100 or 200 level. The colon is used to show a range of credits. An "or" could have been used in place of the comma. The "@" symbol is used as a wild card. For example, 1@ means any course number that begins with the numeral one. If the school has course numbers below 100, such as 010 or 011, these courses will also be included in the 1@ range. In this case, it is best to use a range of 100:199 so that courses numbered below 100 satisfy this rule. The Class(es) and Credit(s) rules will accept courses linked with an "or" or courses linked with an "and", but it is not possible to use both kinds of connectors in a single course rule.

For example:

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

is not allowed. In a case like this it would be necessary to use two Class(es) rules, one for an ENGL 101 requirement and one for an ENGL 102 or ENGL 103 requirement.

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

Qualifiers

Qualifiers are restrictions placed on a requirement or set of requirements. These restrictions must be met for the block to be completed. There are two types of qualifiers: Header qualifiers appear in the Header section of the block and pertain to all rules in the block body. Rule qualifiers appear in the Body section of the block and pertain only to the rule with which they are associated.

The Introduction section of this guide contained an example using qualifiers. More examples are included in this section. For a complete list of qualifiers, including additional examples, review the Rule and Block Header qualifiers in the *Scope of Reserved Words* section of this document. Header qualifiers appear in the block Header, which is the area after the BEGIN command and before the first semicolon. Here are some examples of commonly used Header qualifiers:

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

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

```
MinRes 30 Credits
```

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

```
MaxCredits 3 in PE @
```

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

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

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

```
MinGrade 2.5
```

Any class with a grade less than 2.5 will not satisfy the course requirements in the block. Note that this is a stronger qualification than MinGPA.

Most rule qualifiers appear after the list of courses in a Course rule and before the label. It is required that a single space separate the last course number in the course list from the first qualifier, and that single spaces separate the qualifiers. For ease of reading, however, it is suggested that Scribers use carriage returns to separate the course list from the qualifiers and the qualifiers from each other. The following are some examples of commonly used Rule qualifiers in an easily readable format:

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

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

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

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

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

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

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

Biology 101 satisfies this course rule. Ordinarily, Degree Works uses a default setting of Exclusive (DontShare) for all rule statements. This means that a course can satisfy only one rule. Because of the ShareWith qualifier in this example, however, BIOL 101 can also be applied against another course requirement within this block. For example, if there is another requirement in another part of the block that includes BIOL @ in the course list, then BIOL 101 will be used to satisfy that rule as well.

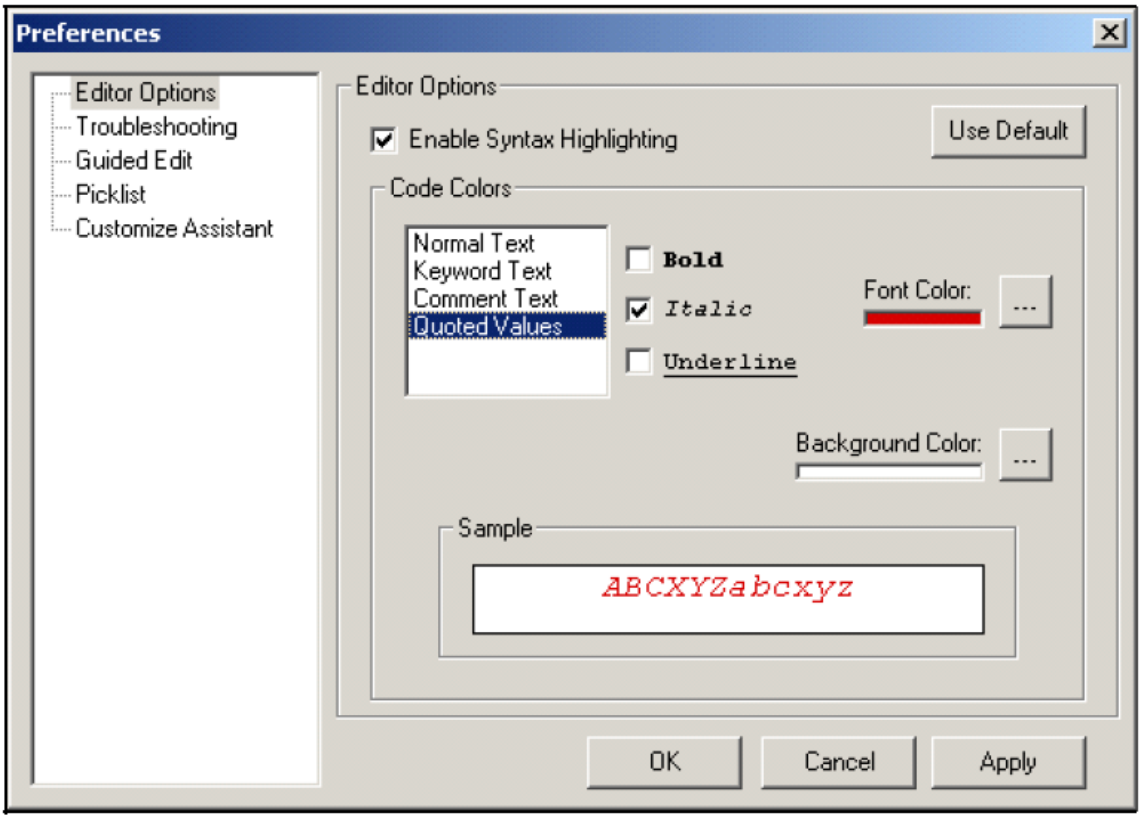
Syntax Highlighting

Syntax Highlighting allows users to display different kinds of Scribe text in different colors and different styles such as **Bold**, *Italic*, or Underline.

Syntax Highlighting is enabled by default. To turn off Syntax Highlighting so that all of your block text is the same color, click the Options on the Scribe menu bar and the select Preferences from the drop-down list; when the Preferences dialog box opens, make sure you are in the Editor Options section. You can deselect Enable Syntax Highlighting by clicking in the box with the checkmark. Click OK to save your changes.

If you would like to change the defaults used by Scribe Highlighting, you can easily do so in the Editor Options area. Within the Code Colors section, Normal Text is defined as the amount of credits or classes in a rule, the institutions course keys, and numbers. Keyword Text refers to Scribe's Reserved Words. Comment Text is the text after a pound sign [#] in a block. Quoted Values would be the text placed within quotes such as label text, Proxy-Advice text, and Remark text.

Once you have made your changes, click OK to save them. If you would like to revert to the default settings, select the syntax area you wish to default to and click the Use Default button in the upper right corner of the dialog box.



Parsing Block Requirements

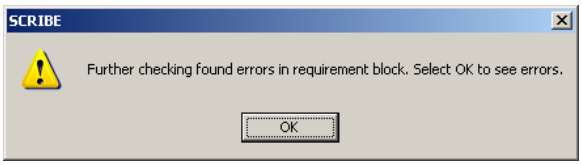
Before a block can be saved to the Host, it must be parsed. The parser utility scans the contents of the block to insure that the block is formatted such that the auditor program can “read” it when producing student worksheets. Included among the block elements that the parser checks are the following:

1) Syntax and Rule structure (e.g. are the rules separated by semicolons? Are the labels no more than 50 characters in length? Are qualifiers placed correctly? Are courses in a course list separated by “or”s or commas, “and”s or plus signs?)
2) Consistency with database information (e.g. are the course disciplines valid?)

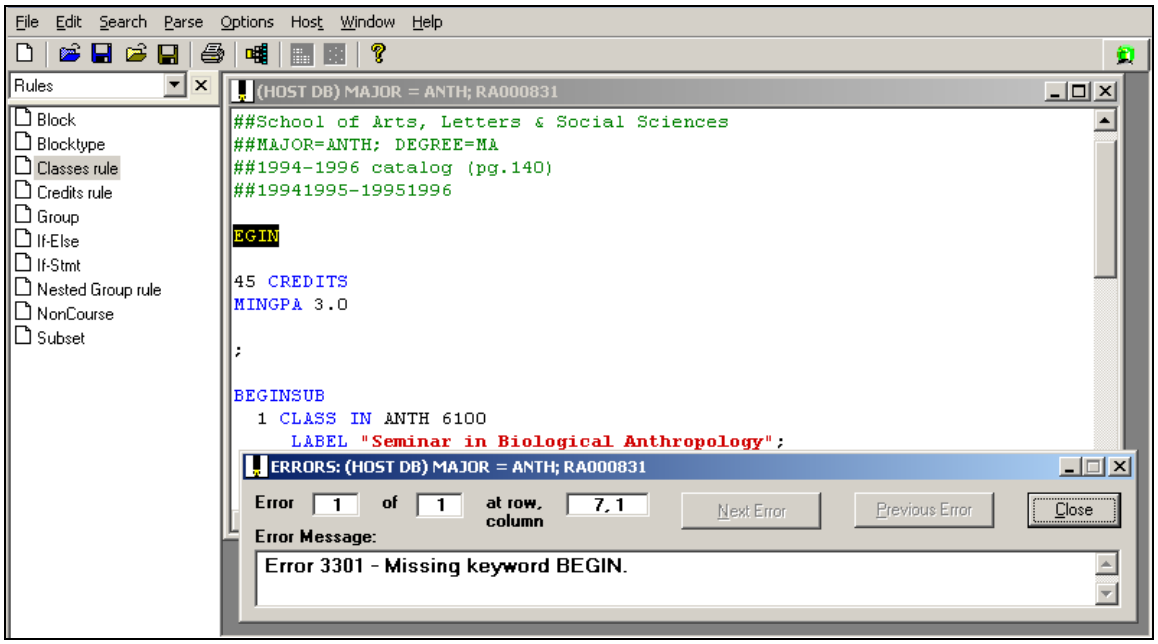
Depending upon how the flags are set in the UCX-CFG020 DAP13 record, the parser can validate discipline codes as well as course numbers. If the appropriate flags are turned on, only valid entries will be allowed in a Scribe block. This approach is recommended.

Clicking the “Parse” icon  will activate the parser. If the block contains no errors, the word “Parsed” will appear in the top bar of the block window. If the parser finds errors, an ERRORS message window will appear.

Clicking the “Save block to Server” icon  will also activate the parser. Before a block can be saved, it must be syntactically correct. If the parser finds errors, an error message box will appear:



Clicking OK will open a new window below the Scribe box with information about the type and location of the error. Error messages will be displayed in sequence, one at a time, in the ERRORS message window as shown below.



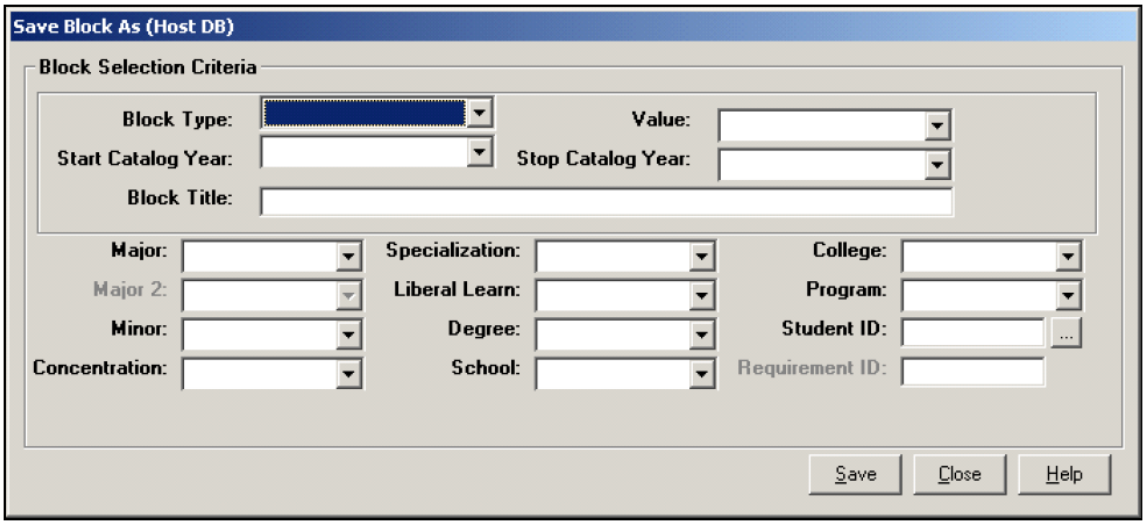
The Error message includes the number of the error (1 of 1), the position of the error (row 7, column 1), and a description of the error (Missing keyword BEGIN). The specified error will also be highlighted in the Scribe window. The Scribe user can use the error message and highlighting to find the error and correct it and can then advance to the next error if there is one.
The parser cannot always determine the exact nature or location of an error. In addition, errors in the block can cause the parser to target error-free parts of the block (these are called "propagation errors"). For these reasons, parsing a block is sometimes as much an art as it is a science. Here are a few parsing tips.

1) If the parser returns a large number of errors, it is often wise to correct the first few errors and then close the Error window and click the Parse icon again. This reduces the number of phantom "propagation errors" that the Scriber must deal with.
2) Remember that the error area highlighted by the parser does not always include the error! If no error is evident, scan the area immediately preceding the highlighted area.
3) It is a good idea to parse as you go. Create several rules and then parse. This will save you time and allow you to correct errors more easily. You will know to look for errors, starting with your most recent Scribe code.

Assigning Database Tags

After a block has been parsed successfully, you need to save it to the Host database to be used in student worksheets. If you attempt to save the block before it is parsed, the parser will automatically check for errors when either save process is selected.

Click the Save Host icon  or select Save As (Host DB) from the File drop-down menu. The Save As action will open the Block Selection Criteria window seen below.



The various fields in the Block Selection Criteria window represent the database tags. The tags are divided into two types: primary tags (which are required) and secondary tags (which may be optional). Primary tags include the following:

Block Type	Block Types are predefined in Scribe and are chosen here from a drop-down list**.**
Value	For most Block Types, the relevant Value is chosen from a drop-down list that has been populated with the institution's various codes. For example, if the Block Type "Degree" was chosen, then the Value will be the code for the degree. Values for blocks with the Block Type "Other" are user defined during the Save process, so these codes are typed into the Value field rather than selected from the drop-down list.
Start Catalog Year	The Start Catalog Year is the earliest catalog date for which the block requirements are effective.
Stop Catalog Year	The Stop Catalog Year is the last catalog date for which the block requirements are effective. If the block requirements are still in effect, then an "infinity value," such as 9999 or 99999999, is chosen in this field.
Block Title	The Block Title will appear in the Heading of the block on student worksheets and is limited to 50 characters.

Notice that the five primary tags are grouped together in a box at the top of the window. Blocks cannot be saved to the Host unless these five fields are populated. Secondary block tags are grouped together below the primary tag box. These tags, while not required to save blocks to the Host, may be required to distinguish blocks from one another. For example, at Ellucian University, the College of Engineering and the School of Nursing both have an OTHER block titled GENED. Even though the primary block tags are the same, the blocks themselves contain different requirements. To distinguish these blocks from each other, the secondary tag COLLEGE may be used to indicate which GENED block is being referenced.

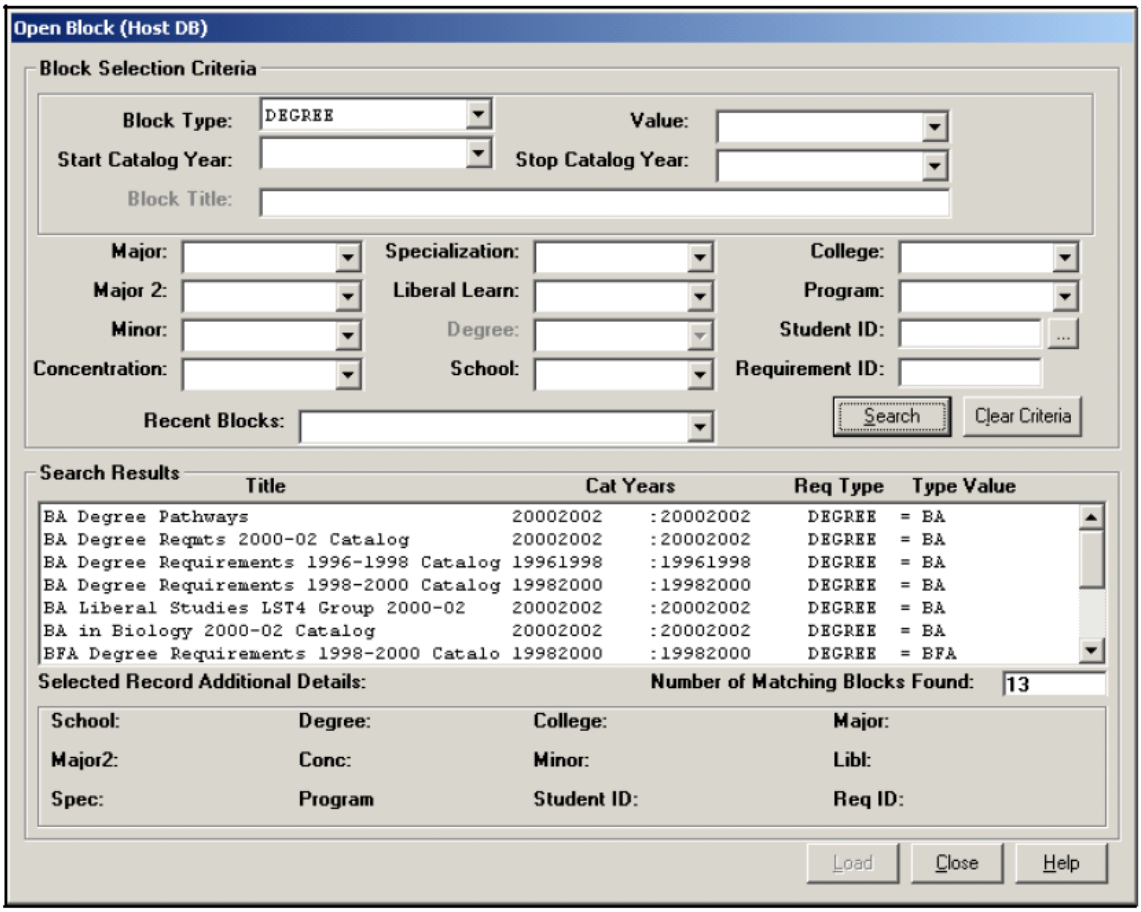
Also notice that the Major 2 secondary tag cannot be used when saving a block for any Block Type except Major. An example requiring a Major 2 secondary tag could be an Art Major that also requires a History Major. You would then place a secondary Major 2 tag of History on one of the Art Major blocks.

Remembering Last Search Criteria

When a user searches for blocks using the Open Host feature, Scribe remembers the search criteria. If a specific block is retrieved from the search results and the user returns to Open Host, the same blocks are displayed from the previous search.

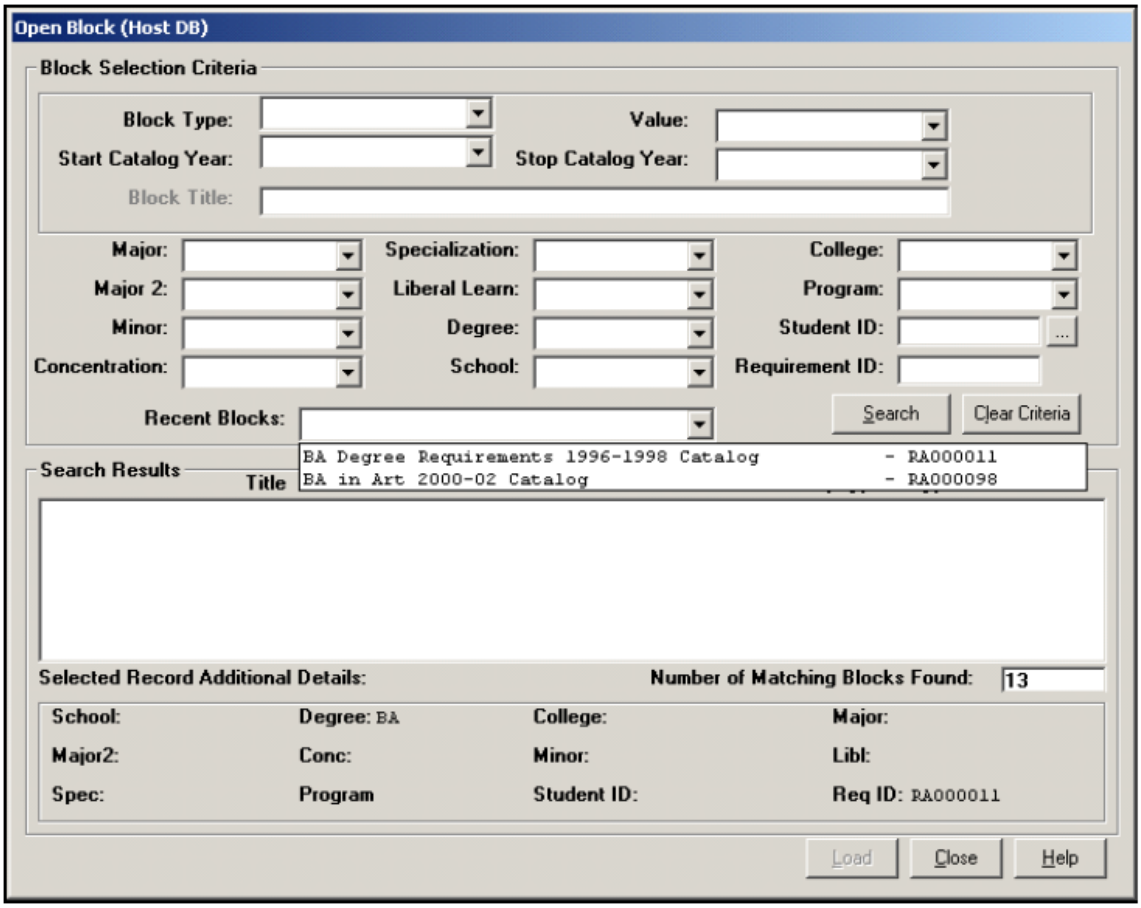
For example, searching by a Block Type of Degree will return all of the institution's Degree blocks. Even after one Degree block is opened, the user can return to Open Host and all the Degree blocks will still be displayed.

To clear the criteria from the last search, click the Clear Criteria button to the right of the Search button. Search criteria are also cleared when a user exits the Scribe session.


Open Block (Host DB)
Block Selection Criteria
Block Type:
DEGREE
Value:
Start Catalog Year:
Stop Catalog Year:
Block Title:
Major:
Specialization:
College:
Major 2:
Liberal Learn:
Program:
Minor:
Degree:
Student ID:
Concentration:
School:
Requirement ID:
Recent Blocks:
Search
Clear Criteria
Search Results
Title
Cat Years
Req Type
Type Value
BA Degree Pathways 20002002 :20002002 DEGREE = BA
BA Degree Reqmts 2000-02 Catalog 20002002 :20002002 DEGREE = BA
BA Degree Requirements 1996-1998 Catalog 19961998 :19961998 DEGREE = BA
BA Degree Requirements 1998-2000 Catalog 19982000 :19982000 DEGREE = BA
BA Liberal Studies LST4 Group 2000-02 20002002 :20002002 DEGREE = BA
BA in Biology 2000-02 Catalog 20002002 :20002002 DEGREE = BA
BFA Degree Requirements 1998-2000 Catalo 19982000 :19982000 DEGREE = BFA
Selected Record Additional Details:
Number of Matching Blocks Found:
13
School:
Degree:
College:
Major:
Major2:
Conc:
Minor:
Libl:
Spec:
Program
Student ID:
Req ID:
Load
Close
Help

Recent Blocks

In the Block Selection Criteria window, you can also review or select from a list of Recent Blocks. Scribe remembers the previous 10 blocks you have accessed. To access your list of recently viewed blocks, simply click the down arrow to the right of the Recent Blocks: field; a drop-down list appears, displaying the last 10 blocks you have accessed.



Creating a Block – An Example

Consider the following program from the Ellucian University Liberal Arts division:

Certificate in Liberal Arts Appreciation (Program Code 1110)

ENGL 266	World Literature: 7th to 18th Century	5 cr.
MATH 124	Calculus and Analytical Geometry I*	5 cr.
	Electives in History, Literature**, Languages, and/or Philosophy***	20 cr.

Student must take 30 credits to complete this certification and must maintain a cumulative 2.0 GPA. Only 10 credits may be transferred from another college towards these requirements.

*	A grade of 2.5 or higher must be earned in this class.
**	Literature classes are all 200-level English classes.
***	Electives must be chosen from at least two disciplines.

The first step in creating this block is to log into Scribe and click the "New Block" icon.

The following Header values are entered after the BEGIN command:

```
30 Credits
MinRes 20 Credits
MinGPA 2.0
```

The first Header qualifier is the credit total; every block should contain a credit total. Ellucian University allows up to 10 credits in transfer classes and requires a cumulative GPA of 2.0 or higher for completion of its programs, hence the second and third qualifiers.

After the first semicolon the three course rules are scribed:

```
;
  1 Class in ENGL 266
    Label "World Literature: 7th to 18th Century";
  1 Class in MATH 124
   MinGrade 2.5
    Label "Calculus and Analytical Geometry I";
  20 Credits in HIST @, ENGL 2@, SPAN @, FREN @, PHIL @
   MinSpread 2
    Label "Electives";
END.
```

After the requirements are entered, the block is parsed and any errors are corrected. Finally, the "Save Host" icon is clicked, and the block is given the following tags:

```
Block Type              Degree
Value                   1110
Start Catalog Year      19981999
Stop Catalog Year       99999999
Block Title             Certificate in Liberal Arts Appreciation
```

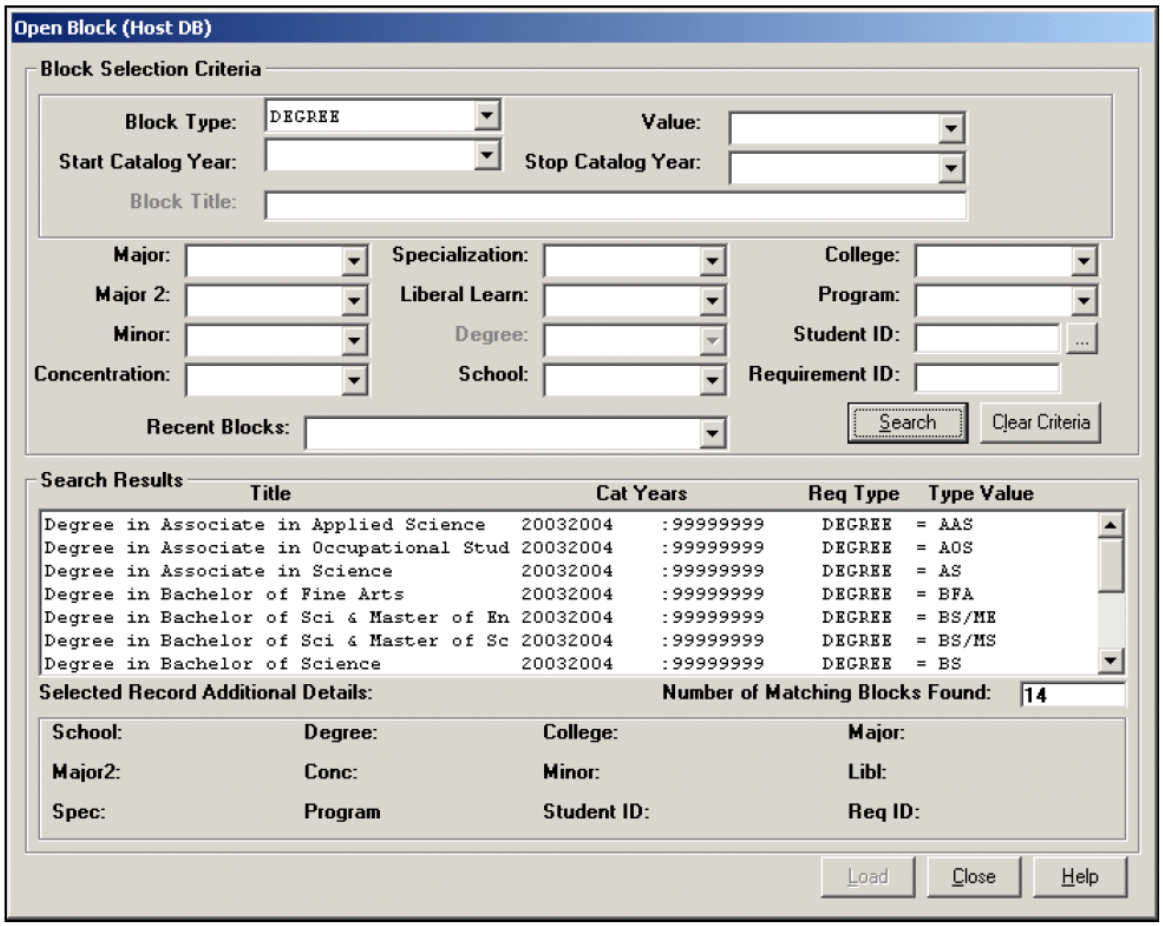
This degree was last modified in the Fall of 1998 and is still available at Ellucian University, hence the given Start and Stop Catalog Years. The catalog year of 99999999 reflects infinity. We do not know when or if these requirements will change, and until they do (and the block is modified), the block will be valid from 19981999 until forever.

Recalling and Editing an Existing Block

After a block is parsed and added to the Host database, it can be recalled using the Open Host icon. 

To recall a block, the Block Type and Value tags for the block can be selected from the drop-down lists in the appropriate fields in the Block Selection Criteria window. The window below shows all blocks in the database that meet the primary selection criteria of Block Type = Degree. After the Search Results are returned, you can open any of the blocks in the window either by double-clicking on it or by clicking on it once and then clicking Load.

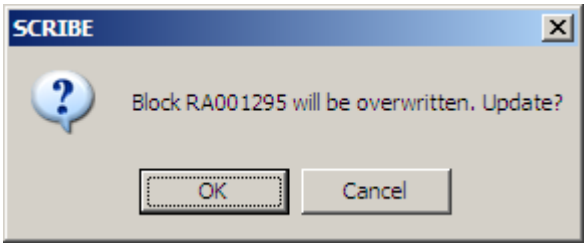
If only one block meets the Selection Criteria entered, then that block will automatically open and no secondary selection window will be seen. Alternatively, you may leave some of the block selection fields empty and click Search and choose the block from the Search Results Drop-down List Box. You can also leave all Selection Criteria fields blank and then click Search, which will produce a list of all the blocks in the database.



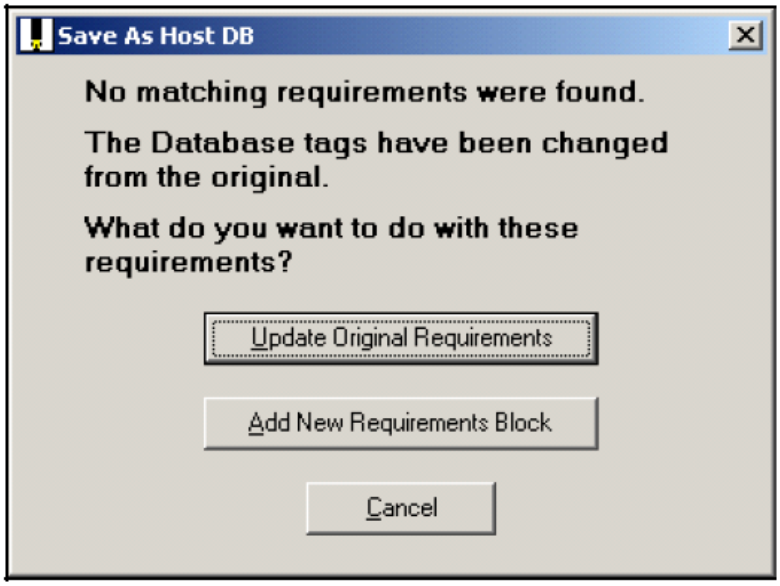
The easiest method of retrieving the block you want, however, is to know the Requirement ID, more commonly known as the “RA number”. The RA number is an eight-character identifier comprised of “RA” and six numbers (for example, RA000001, RA000025, RA000674, etc.). Each block is assigned a unique, sequential RA number, as is it created.

If you know a block's RA number, simply type the RA number into the Requirement ID field in the Open Block widow and then click Search. For example, if a block's RA number is RA000350, then it was the 350th block created in the database. To search for this block, you can enter into the Requirement ID window "RA000350", "000350", or just "350", and the block will be retrieved and opened in Scribe.

You may choose to modify the block after you have recalled it. Any changes made to the block after being recalled will require you to parse the block again prior to saving it back to the Host. To save the block with the modifications, parse the block and then click the "Save Host" icon. This action will overwrite the original requirements with the new modifications if you select OK when the following message is received:



To modify the database tag information for a block (e.g. the Block Title, the Start or Stop Catalog Year, etc.), recall the block using the appropriate Block Selection Criteria. Make your modifications and then click File on the main menu bar and select Save As (Host DB) from the File drop-down menu. This action brings up the Block Selection Criteria screen where you can modify the tags assigned to the block. After making the appropriate changes, click Save. The following window will then appear:



By clicking Update Original Requirements, you will save the original block with the new primary tags and secondary tags that were assigned to the block in the Block Criteria Selection section. The original block will be overwritten. By clicking Add New Requirements Block, you will create a new block with the new tags and you will leave the original block intact. It is important to remember that changing the tags on a block is the same as changing the name of the block file when saving to the Host. You may also save a block to the hard drive of your computer using the Save PC icon. Blocks that are saved to your PC do not have to be parsed, unlike blocks saved to the Host. Blocks saved to the PC also do not require primary or secondary tags. Hence, it is often

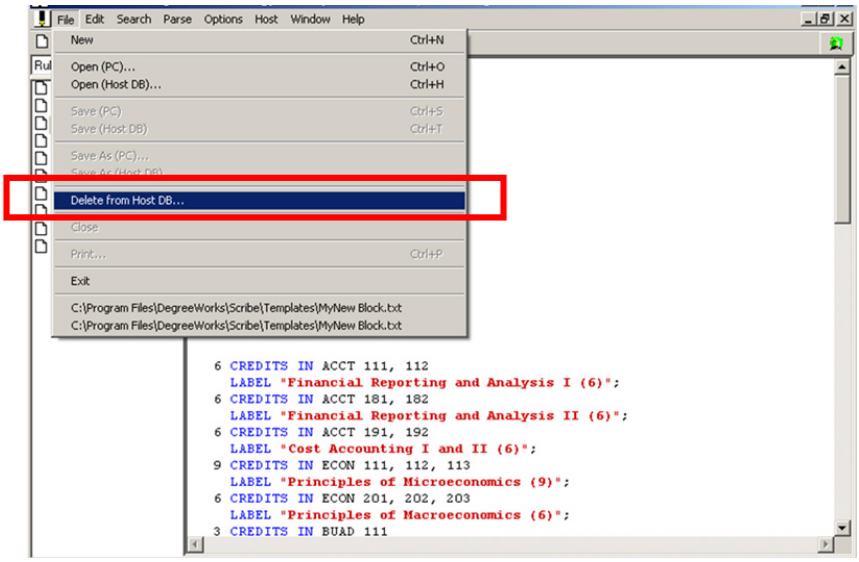
useful to save test blocks, template blocks, and “blocks in progress” to your PC. These blocks can later be recalled with the Open PC icon.

Deleting a Block

Deleting blocks is rare at most institutions, however, there may be a need to clean up your database after some time when old Scribe requirements may no longer be needed. More typically, users sometimes create blocks to test or practice with and then decide that those blocks can be deleted. Below is a detailed process to delete blocks.

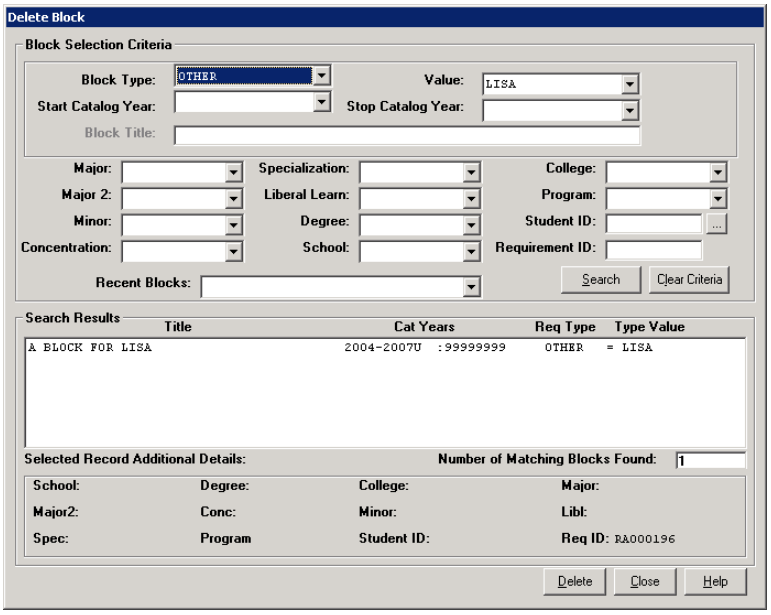
Note: Once a block has been deleted from the database, the only way to recover it is from a back up.

On the main menu bar, click File, and then select **Delete from Host DB** from the drop-down menu.

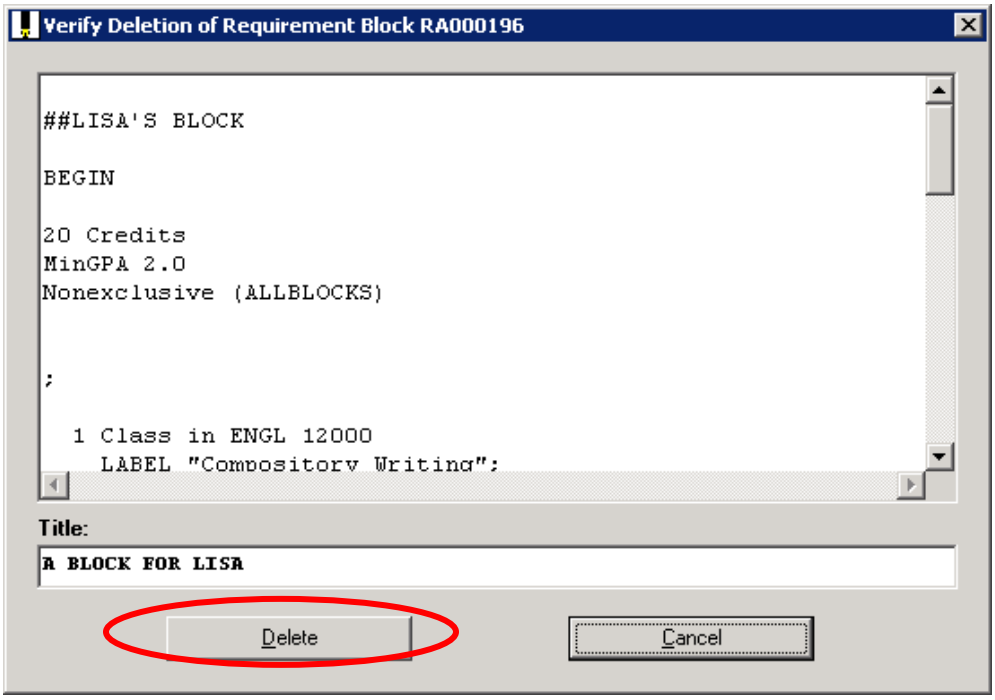


The Delete Block screen will display. Find the block you want to delete by populating the Block Type and Value fields or entering the Requirement ID. For this example, our selection is Block Type: OTHER and Value: LISA. (Lisa was a practice block named after the person who created it. This is an easy naming convention to make sure your blocks are not used unintentionally.)

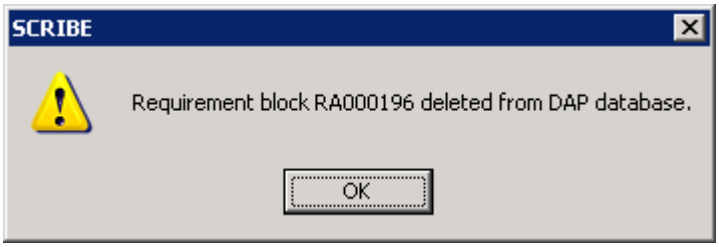
Next, click Search. The Lisa block appears in the Search Results window. Select the block by double-clicking its title.



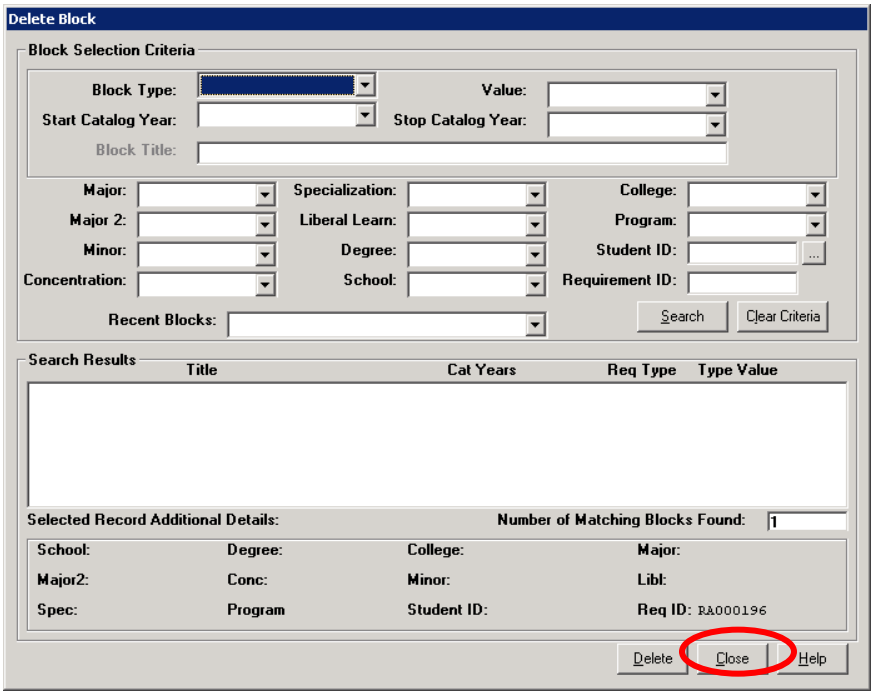
The Verify Deletion of Requirement window appears. This window allows you to see the entire block and its title. You can easily review the contents to ensure you have selected the correct block, and then either Delete the block or Cancel your actions if you have retrieved the incorrect set of requirements. Once you've determined that this is the block you wish to delete, click Delete.



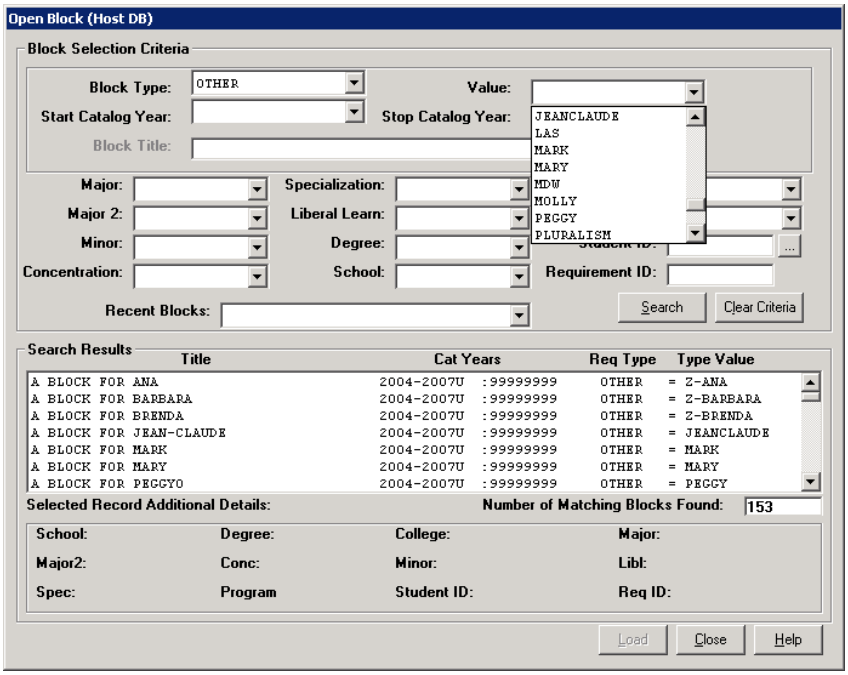
After the block is deleted, the following Scribe message window will appear to inform you that the block you selected has indeed been deleted from the database



When you click OK in the Scribe message window, the Delete Block window will appear again because you are still in the delete mode. If you want to delete more blocks, continue doing so. If you are finished deleting blocks, click Close in the Delete Block window.



To verify that the block has been deleted, click File again, and then select Open (Host DB). The Open Block (Host DB) window will appear. Search for the block you just deleted. In this example, the OTHER: LISA block has disappeared from the pick list of the **Value** field.

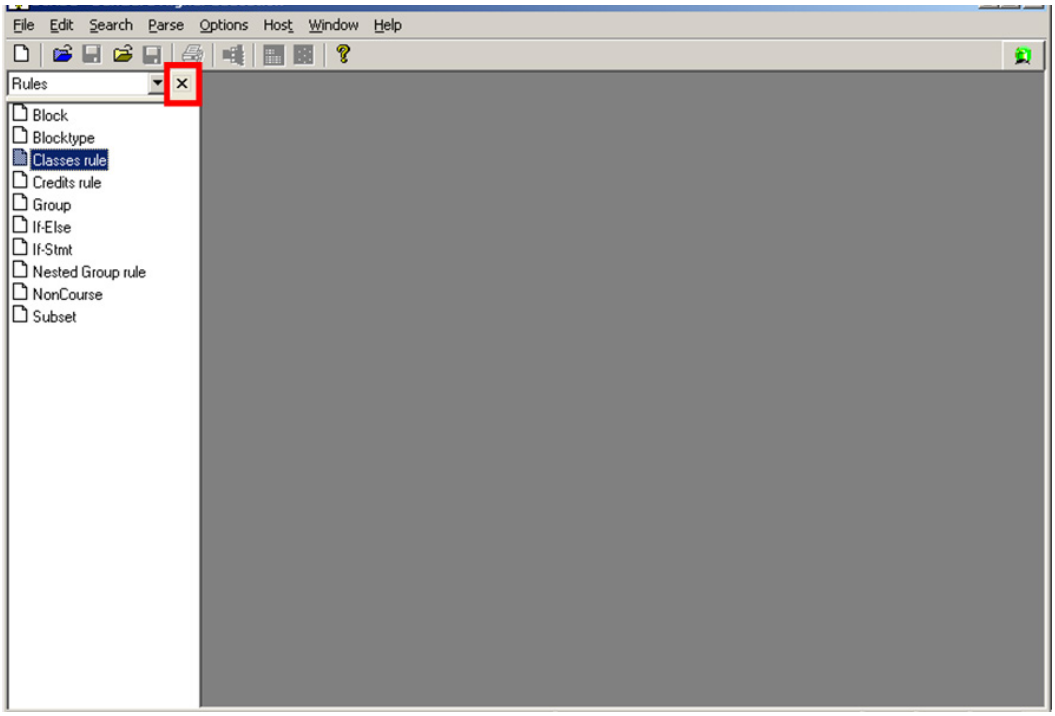


Scribe Assistant

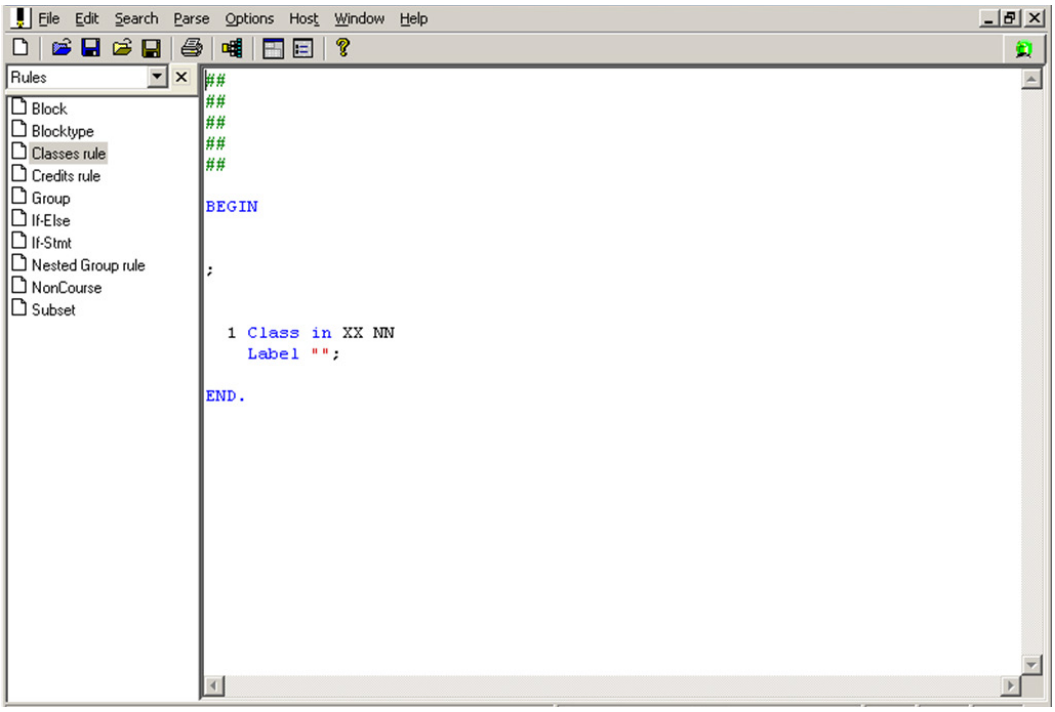
The Scribe Assistant is used to quickly insert predefined templates for Rules, Header qualifiers, and Rule qualifiers. By using the Assistant you can be certain that the commonly-used syntax will be properly formatted. Once the templates have been placed, you then only have to scribe your specific requirements.

You can either double-click or drag-n-drop the Scribe Assistant syntax into a requirement block window. The block will not parse until you fill in your specific course keys, numbers, and credit/class requirements.

When Scribe is open, the Assistant is viewable on the left side of the interface. To hide the Assistant, click the X in the upper right corner of the Assistant bar.



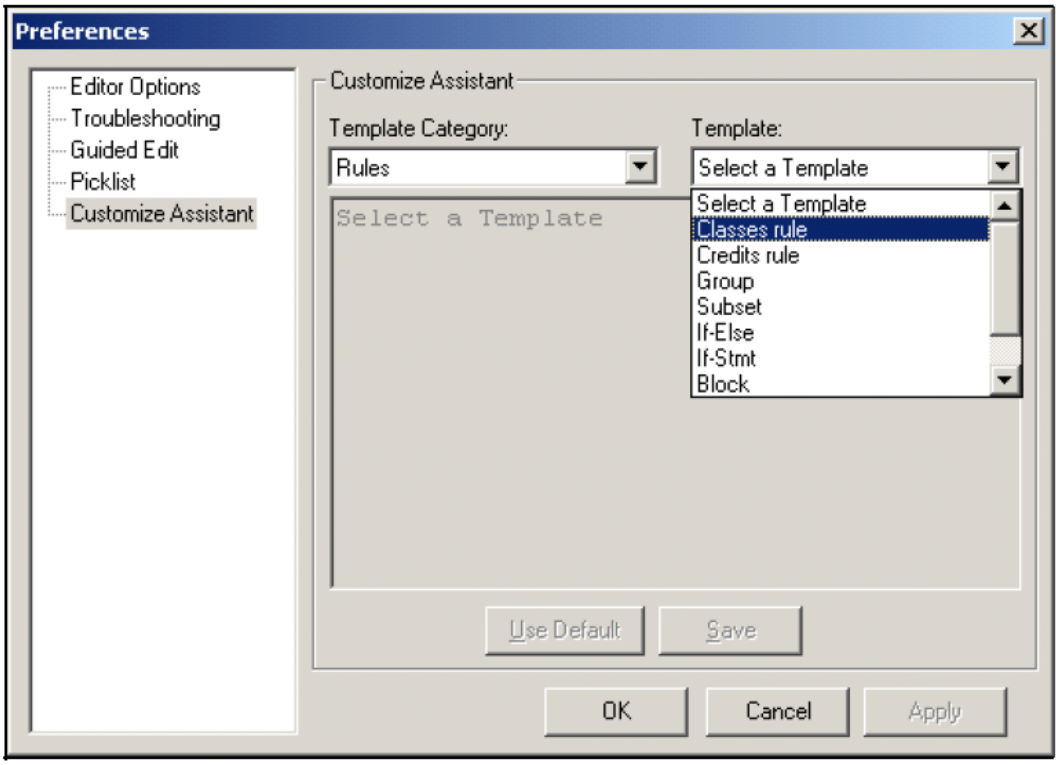
To use the predefined templates, open an existing or new block and place the cursor on the line where you want the template to be inserted. To insert a template, drag-and-drop, or double-click the template. You can choose additional templates by selecting from the drop-down list box at the top of the Assistant. Once the desired template is inserted, modify the scribe so that your institutional requirements are reflected, parse, and then save the block in order to use it in future student worksheets.



You can develop your own custom templates or change any of the predefined templates to meet your specific needs. It is important to note, however, that customized templates are saved to the computer where the changes are made. If you open Scribe on another PC, your customizations will not exist on that computer

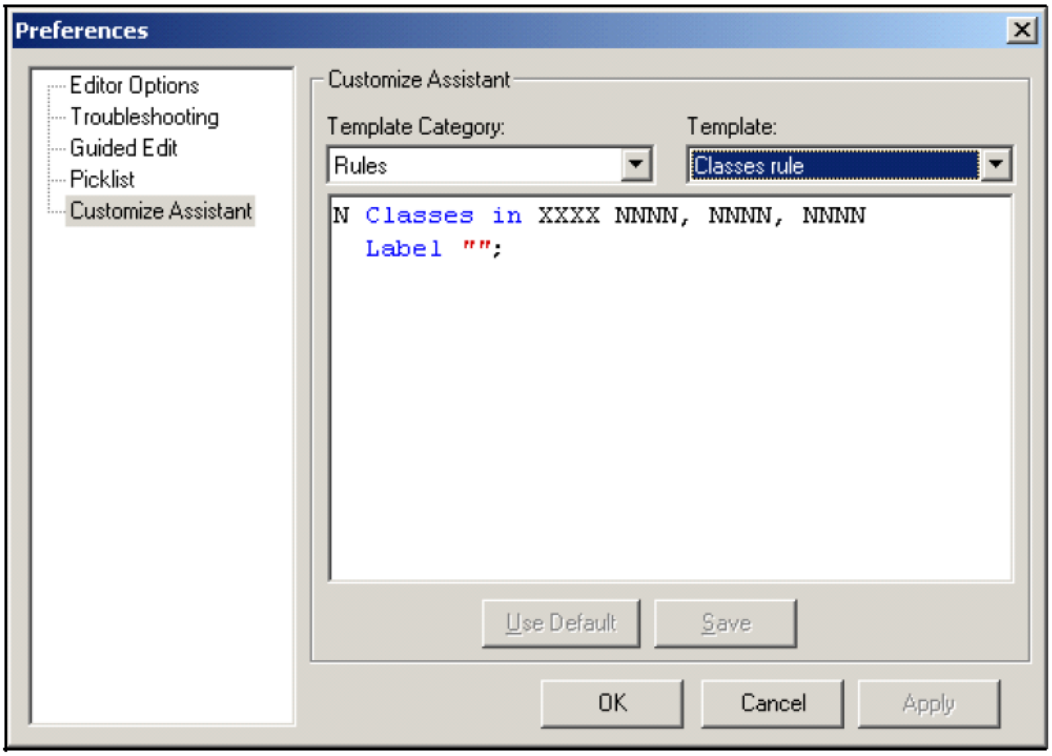
To change a predefined template, click Options on the main menu bar, and then click Preferences from the drop-down menu. The Preferences window then appears. In the Preferences window, click Customize Assistant. Select a Template Category using the drop-down arrow attached to the field (the Template Category selected in our example is Rules). Next, select a Template using the drop-down arrow attached to the Template field (in our example, the Classes rule template is being selected).

Selecting the Classes rule template will bring up the default rule, which is what will be inserted into your requirement block if you use the Scribe Assistant.

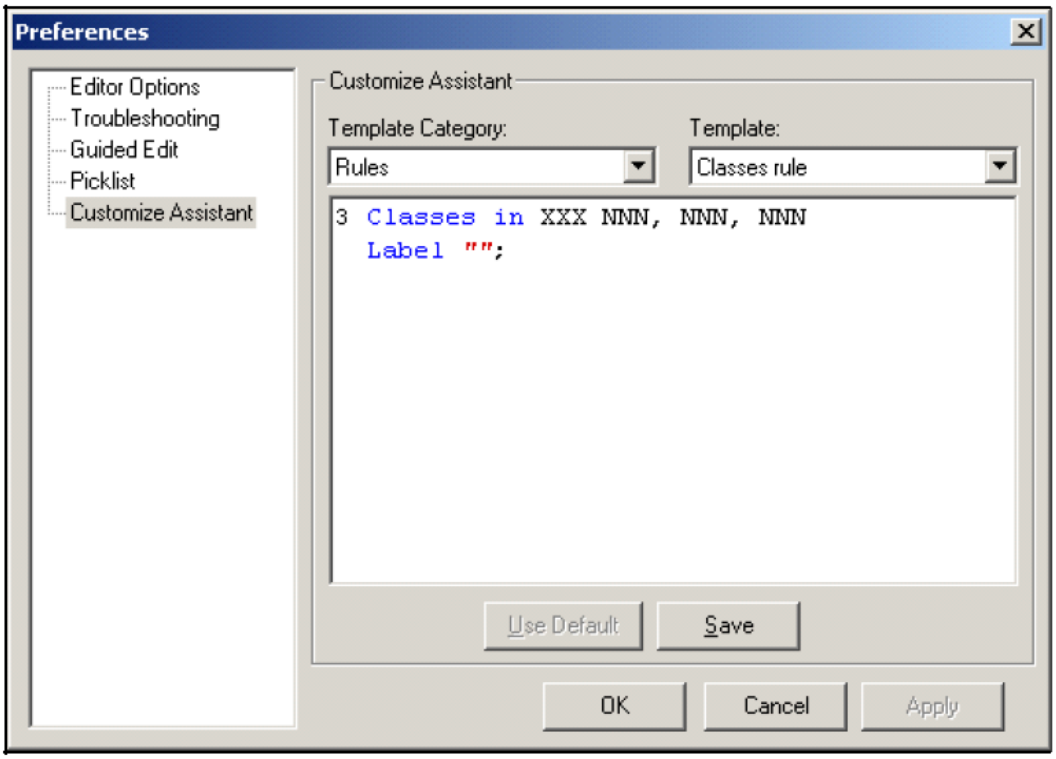


An example of a change to an existing template would be to modify the rule so that instead of four bytes for course key and number (our default) three bytes will appear in the template.

This screen shot reflects the default template with four bytes.



This screenshot reflects the modified template with three bytes.



After changing the template, click OK to retain your changes.

You can always revert back to the original by re-accessing the modified template (using the Template Category and Template fields again) and then clicking Use Default. This action will prompt you that you will lose your personal changes, and then the class rule template will be restored to the original.

To develop your own custom templates, select My Templates from the drop-down list in the Scribe Assistant bar. This option will allow you to create your own rules, qualifiers, or complete block templates, if desired. Once you have created your custom template, click Save to retain your template.

Updating a Set of Requirements – An Example

Suppose that in the Fall of 2007, the requirements for the Certificate in Liberal Arts Appreciation (from our earlier Creating and Saving a Block example) are changed. The courses Math 125 (Calculus and Analytical Geometry II) and MUSIC 201 (Music of the Enlightenment) must be added to requirements list, and the elective requirement must be reduced from 20 credits to 15 credits. The old version of the requirements will still be in effect for students who began the program in the previous year. To accomplish this update, you must do two things in Scribe:

(1) Restrict the catalog year for the original block
(2) Create a new block for the new version of the Certificate

Here is a description of the two steps.

Step 1. Restrict the catalog year range of the original block. When it was saved to the Host, the original block requirements started in Fall 2006 and were given a Start Catalog Year of 20062007 and a Stop Catalog Year of 99999999. Since the new requirements take effect in Fall 2007, the Stop Catalog Year of the original block must be changed to 20062007. Click the Open Host icon to recall the original block. After the block is recalled, use the File drop-down menu to select Save As (Host DB). The Block Selection Criteria window appears. In the Block Selection Criteria window, make the change to the Stop Catalog Year. Click Save and then choose the Update Original Requirements option from the Save As (Host DB) window.

Step 2. Build the new block. The easiest way to do this is to start with the original block since it is still mostly correct. If it is not already recalled, recall the original block and modify it by adding the new course rules. An easy way to add the Math 125 course rule is to copy the Math 124 rule and paste it into the block, change the course number from '124' to '125', and in the label, change the 'I' to 'II'. After the changes are made, parse the block to check for errors (this is a required step). Choose Save As (Host DB) from the File drop-down menu. In the Block Selection Criteria window, change the Start Catalog Year to 20072008 and the Stop Catalog Year to 99999999. Click Save and then choose the Add New Requirements Block option from the Save As (Host DB) window.

You began with one block for the Liberal Arts Certificate; now there are two versions of the Certificate in the Degree Works database. The two versions share the same Block Type, Block Value, and Block Title, but they have different Catalog Year tags.

Before:

(Original Version)	
Value	1110
Start Catalog Year	20062007
Stop Catalog Year	99999999

After:

(Original Version)		(New Version)	
Value	1110	Value	1110
Start Catalog Year	20062007	Start Catalog Year	20072008
Stop Catalog Year	20062007	Stop Catalog Year	99999999

Catalog Year information is stored with Degree information in your Student Information System database. Degree Works uses this information to assign the correct block to a student. Therefore, a student with a Catalog Year of 20062007 will be audited against the original block, while a student with a catalog year of 20072008 will be audited against the new block.

Special Topics

AdviceJump and RemarkJump and RuleTag

AdviceJump

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

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

```
16 Credits in @ (WITH HONR=Y)
    ProxyAdvice "16 Credits of upper-division Honors courses are required."
    RuleTag AdviceJump=”http://myschool.edu/catalog/UpperHonors.html”
   LABEL "Honors Requirement";
```

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

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

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

☐ Honors Requirement	Still Needed: 16 Credits of upper-division honors courses are required
☐ Writing Requirement	Still Needed: Click here to see classes that meet this requirement

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

If your .html files are located somewhere other than where your Degree Works web server files are located, your RuleTag values need to have the full path or relative path to the html page being referenced. RuleTag can be used to associate any other piece of information with your rules, and the RuleTag code specified can then be trapped for in the stylesheet to give special meaning to certain rules. Using AdviceJump with RuleTag here is merely one example of the power of RuleTag. For more information on RuleTag, see the RuleTag definition in the *Reserved Words Definition* section of this guide.

RemarkJump

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

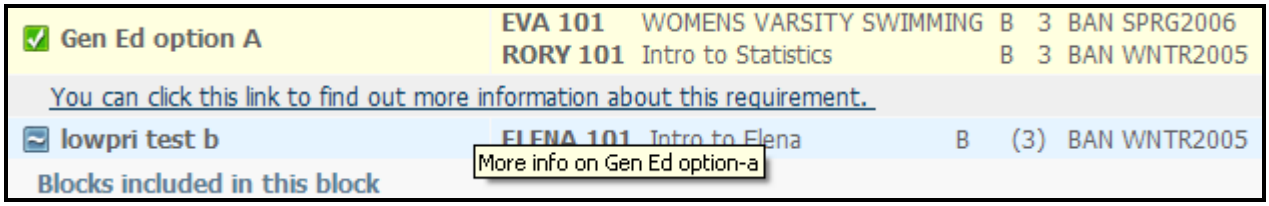
```
2 Classes in EVA 101 , RORY @, ELENA @
 RuleTag RemarkJump="http://some.place.edu/ontheinternet/anywhereisfine/"
  RuleTag RemarkJump="support/getmemoreinfo.html"
  RuleTag RemarkHint="More info on Gen Ed option-a"
  Label "Gen Ed option A";
Remark "You can click this link to find out more information";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

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

You can optionally specify a **RemarkHint** RuleTag to give your user a small pop-up hint when the mouse is placed over the remark. Again, you may specify one or more RemarkHint RuleTags. To add this same functionality to the header remarks you make use of the HeaderTag instead:

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

The **AdviceJump** can only be used while the rule is not complete. Once the rule is complete the advice and thus the link goes away. With **RemarkJump**, however, the link is always available since the Remark text appears on the worksheet regardless of whether the rule is complete or not.



Block Security

You can setup security to restrict which users are allowed to save certain types of blocks. Typically, you want your registrar's office users to have access to all blocks. By default, the users assigned to the REG user-class are given the ANYBLOCK key. This key allows these users to save any block type.

You may want some users to only be able to save blocks of a particular type. These users should not get the ANYBLOCK key and should only be given the keys for the blocks they are allowed to save. For example, a user who should only be allowed to save minor blocks should be given the MINOR key. This user will be allowed to view any block in Scribe but will only be able to save changes to MINOR blocks.

By default, those users in the ATHL user-class are given the ATHLETE key allowing them to save changes to the ATHLETE blocks. By default, those users in the AID user-class are given the AWARD key allowing them to save changes to the AWARD blocks.

You can use Shepentry or SHPCFG to manage your keys.

See the *Security* section in the *Technical Guide* for a complete list of Scribe keys.

Block Types and Multi-Block Programs

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

DEGREE blocks usually specify degree requirements, such as total credits, residency, and GPA. DEGREE blocks usually call in MAJOR blocks and OTHER blocks.

CONC, LIBL, MAJOR, MINOR, SPEC, PROGRAM, SCHOOL, and ID blocks can be used to best fit your institution's requirements.

OTHER blocks are generally used for General Education requirements and Core requirements that are common among several blocks.

AWARD blocks are generally used for Financial Aid requirements. These blocks are called in automatically based on the AWARD records found on the database.

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

```
BEGIN

90 Credits
MinGPA 2.0
;

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

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

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

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

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

END.
```

Notice that the main (Degree) block contains some of the requirements for the degree but not all. The other requirements are scribed in the "DISTRIB" and "ELECT" blocks, which have been saved to the host as separate blocks.

These blocks are called from the main AA Degree block and, as such, are included as part of the AA degree requirements. Also note that there is a Header qualifier of minGPA 2.0 in the main (Degree) block. This qualifier also applies to requirements in the DISTRIB and ELECT blocks even though those blocks may or may not contain the minGPA qualifier.

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

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

Comments, Proxy-Advice, and Remarks

Comments

Any line in a Scribe block that begins with the symbols "#" or "!" is a Scribe comment. Scribe comments are ignored by the parser and the auditor. These comments are for scribers' eyes only. By convention, many scribers include one or more comment lines at the beginning of a block, which list the database tags for the block. Another use of Scribe comments is as internal documentation. The person who scribes the block knows why the qualifiers and rules were entered in a particular way, but this logic is not always evident to a person reading the block for the first time. Comments such as, "#This qualifier enforces the maximum three credits in PE rule," or, "#At least one class must be a laboratory class," make the block more readable to others.

Proxy-Advice

Proxy-Advice was designed to remove the default advice on worksheets (what the auditor places based on Scribe code) and replace it with something more "friendly" and college-related, if desired. An example could be a thesis requirement using a NonCourse. The advice the auditor gives to students and staff is as follows: *Still Needed: 1 NonCourse (Thesis).* This advice, however, may not use terminology that is common to a particular campus. It would make more sense to a student for the auditor to provide the following advice instead: *Still Needed: Your Completed Thesis Statement.* With **Proxy-Advice**, we can accomplish this modification. Here is an example of how to write **Proxy-Advice**.

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

If you need to display a large amount of advice, **Proxy-Advice** can be used more than once. Within the quotes, you can include up to 65 bytes of text. If your desired advice exceeds this amount, simply scribe "**Proxy-Advice"** again on the next line of the block and continue scribing your advice. The text from each line will concatenate.

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

The <APPLIED> and <NEEDED> can only be used on course rules and on header "Min" qualifiers – both are replaced by the number of credits or classes still applied and needed. Some other examples of using Proxy-Advice:

```
BEGIN
  40 Credits
    ProxyAdvice "40 credits are required in this major - you still need
<NEEDED> credits more."
  LastRes 10 Credits
    ProxyAdvice "The last 10 credits in this major must be taken here."
  MinGPA 2.5
```

```
    ProxyAdvice "You need a 2.5 GPA in this major – your current major GPA is
<APPLIED>."
;
If (THESIS != PASSED) then
  RuleIncomplete
    ProxyAdvice "You need to submit a thesis in order to graduate.";
1 NonCourse (RECITAL >= 4)
  ProxyAdvice "You need to get a score of 4 or more on your recital."
END.
```

Remarks

Worksheet reports include course information and advice for completing rules for a degree. Remarks allow for additional narrative descriptions in some worksheet formats.

Note: Not all worksheet formats show remarks.

A line containing a remark starts with the reserved word **Remark**. The remark is enclosed in quotation marks, and the line ends in a semicolon. Each remark line can contain a maximum of 70 characters enclosed in the quotation marks. Successive remark lines will be concatenated in the worksheets. Remarks are not entered in the block Header. A remark entered immediately after the first semicolon of the block will appear in the Header section of the worksheets. Here is an example of how to scribe remarks:

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

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

Group Rule

The **Group** rule allows the scriber to scribe more complicated degree requirements that cannot be handled with simple course lists. For example, suppose that a degree requires a three-course sequence in either Spanish or Chinese.

Review the following course rule:

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

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

The following is another example of the course rule:

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

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

The Group rule is the easiest solution:

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

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

More examples of Groups can be found in the documentation, and in the blocks prepared for your school by Ellucian.

Label Tags

You should always use a **label tag** on a requirement label. The text in the tag uniquely identifies the rule and allows users to change the label text when needed without causing exceptions to become unhooked. For more information on unhooked exceptions, see the *Technical Guide*.

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

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

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

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

The first example above shows where to place a label tag in reference to the label text of "English 230 or Literature 205". In the first example, the student has a choice of 5 credits in either ENGL 230 or LIT 205. The label tag of "literature" is placed after the reserved word LABEL and before the open quotes of the label text.

Original Requirement

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

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

Changed Requirement

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

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

Qualifier Tags

You should always use a **tag** on a qualifier. The text in the tag uniquely identifies the qualifier and allows users to change the qualifier when needed without causing exceptions to become unhooked. For more information on unhooked exceptions, see the *Technical Guide*.

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

```
Begin
  45 Credits
  MinCredits 12 in ENGL 2@, LIT @
    Tag=lit
    ProxyAdvice "You need at least 12 credits in literature"
  MinCredits 6 in ENGL 24@, 25@

    Tag=writing
    ProxyAdvice "You need at least 6 credits in writing"
  MaxCredits 8 in ENGL 3@, 4@
    Except ENGL 343, 344

;
```

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

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

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

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

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

You should also place tags on your rule qualifiers:

```
  15 Credits in HIST 2@, 3@
   MinCredits 5 in HIST 2@ Tag=HIST2
   MinCredits 3 in HIST 3@ Tag=HIST3
```

```
Label HIST "History Requirement";
```

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

Change of Catalog Year

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

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

The label-tag can be an alias like CHEM-ELECTIVE, similar to the following:

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

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

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

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

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

Max Header Qualifiers

When the auditor attempts to enforce the maximum number of credits specified on a header qualifier, one of the steps includes a check for marginal credits. This is when the auditor finds that the removal of a class from the block will bring the credits applied to the qualifier below the maximum specified. The class is considered “marginal” because it is bordering the maximum: keeping the class exceeds the maximum while removal of the class brings us below the maximum.

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

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

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

Max Zero

Often you want to completely exclude certain classes from the audit. To do this you can specify zero credits or classes in a Max qualifier in the degree block. Classes matching the qualifier will be placed in the over-the-limit section and will not count. For example, to exclude all remedial classes below 100-level you can do this:

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

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

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

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

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

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

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

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

This actually applies to any class that was targeted to go to the insufficient section – not just

failed classes.

Warning:

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

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

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

However, this will work fine since the IF-statement has been resolved already: here the WHO failed and passes classes will end up in over-the-limit:

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

Subset Rule Types

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

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

In this example we only had to use the MinGrade qualifier once instead of twice (once on each rule). The subset qualifier option can be very helpful when there are many rules that have to be qualified and many that do not. Although the BeginSub label follows the course rule in the block layout, the label will be placed above the course rules in the worksheet layout. The individual rules that make up the subset will be indented underneath the subset label on the worksheet. As a matter of style and to make worksheets easier to interpret, Ellucian suggests that BeginSub labels be written in upper-case letters. Labels for rules contained within subsets should be written in mixed case to distinguish them from BeginSub label.

More examples of subsets can be found in the documentation, and in material presented to your school by Ellucian.

Course List Order

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

Ordering for Readability

When users view the rule advice on the worksheet it is best if the courses are listed in order by discipline and course number. This is especially important when the rule contains a long list of courses. You want a user to quickly and easily see that a particular course is or is not in the list. Take the rule below as an example. Since the courses are listed in sorted order it is very easy for the user to see that GEOG 2023 is not in this list simply by looking at the section of GEOG courses.

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

Ordering for Maintainability

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

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

Ordering for Performance

When the courses are in sorted order the auditor can save time when applying classes to rules and qualfieirs. The software stops comparing when it knows it won't find any more matches. In

the example above, if the class being applied is GEOG 2023 the auditor will now stop when it sees GEOG 2300 because 2300 is greater-than 2023. If the auditor was trying to apply CHEM 1104 it would stop when it encountered ENGG 2000 because ENGG is greater than CHEM. Generally speaking the auditor will now attempt half as many matches when the rules are sorted – which is a great time savings.

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

You may have special cases where you don't want to list the courses in order but it is best if most of your lists are ordered for the reasons mentioned above.

Complex Scribing

Sometimes you need to scribe complex requirements. Here are some examples of such requirements:

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

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

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

# In this GenEd block, CHEM students can only share12 credits with their minor
# All other students can share unlimited credits with the minor
If (Major = CHEM) then
  ShareWith 12 Credits (Minor)
Else
  ShareWith (Minor)

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

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

# Waive the humanity elective if the student has 2 majors or 2 minors
```

```
If (NumberOfMajors >= 2 or NumberOfMinors >= 1) THEN
  RuleComplete
   Label "Gened elective waived because of double-major or additional minor"
Else
 10 Credits in ENGL @, WRIT @, HUM  @
    Label "Humanity elective";


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

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

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

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

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

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

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

Additional Scribe Blocks

REQUISITE

REQUISITE blocks are Scribe blocks that specify and check for course pre-requisites or co-requisites. For information on scribing for prerequisite checking in the Student Educational Planner and/or Banner, see the *Degree Works Prerequisite Checking Technical Guide*.

Reserved Words

Reserved Word List

The Scribe language consists of Reserved Words that have a special meaning for the degree advisory process. These keywords were chosen to be descriptive of their tasks. Although shown here in upper case, Reserved Words are not case sensitive. Letters within square brackets are optional, but if you use the optional letters listed, use all of them. The number sign (#) and exclamation point (!) are used in the Scribe language to begin free-text comments that are never printed on the output as part of the requirements.

1STCONC	**END.**	**MINSPREAD**
1STMAJOR	**ENDSUB**	**MINTERM**
1STMINOR	**EXCEPT**	**NOCOUNT**
1STPROGRAM	**EXCLUSIVE**	**NONCOURSE[S]**
1STCOLLEGE	**FROM**	**NONEXCLUSIVE**
1STSPEC	**GROUP[S]**	**NOTGPA**
1STLIBL	**HEADERTAG**	**NUMBEROFCONCENTRATIONS**
(2nd, 3rd, etc. also for each)	**HIDE**	**NUMBEROFMAJORS**
	HIDE-RULE	**NUMBEROFMINORS**
	HIGH-PRIORITY	**NUMCONCS**
	IF	**NUMMAJORS**
AlwaysShowInAdvice	**IN**	**NUMMINORS**
ACCEPT	**INCLUDE-BLOCKS-WITH**	**OPTIONAL**
ALLBLOCKS	**INCLUDING**	**OR or ","**
ALLOW	**LABEL**	**OTHER**
AND or "+"	**LASTRES**	**PROGRAM**

AT	**LIBL**	**PROXY-ADVICE**
BEGIN	**LOW-PRIORITY**	**PSEUDO**
BEGINSUB	**LOWEST-PRIORITY**	**REMARK**
BLOCK[S]	**MAJOR**	**RULE-COMPLETE**
BLOCKTYPE[S]	**MAXCLASS[ES]**	**RULE-INCOMPLETE**
CheckElectiveCreditsAllowed	**MAXCREDIT[S]**	**RULETAG**
	MAXPASSFAIL	**SAMEDISC**
CLASS[ES]	**MAXPERDISC**	**SCHOOL**
COLLEGE	**MAXSPREAD**	**SHAREWITH**
COLLEGE-nn	**MAXTERM**	**SPEC**
CONC	**MAXTRANSFER**	**SPMAXCREDIT[S]**
COPYRULESFROM	**MINAREA[S]**	**SPMAXTERM**
COURSE[S]	**MINCLASS[ES]**	**STANDALONEBLOCK**
CREDIT[S]	**MINCREDIT[S]**	**TAG**
DECIDE	**MINGPA**	**THEN**
DEGREE	**MINGRADE**	**THISBLOCK**
DISPLAY	**MINOR**	**UNDER**
DONTSHARE	**MINPERDISC**	**WITH**
ELSE	**MINRES**	

Reserved words used with the Financial Aid Audit are:
(See the *Financial Aid Audit* section in the *Technical Guide* for more information on how these are used.)

CompletedTermCount	**DegreeCreditsRequired**	**ResidenceCreditsEarned**
ResidenceCompletedTermCount	**LastCompletedTermType**	**TotalCreditsAttempted**
CreditsAttemptedThisTerm	**Previous**	**TotalCreditsEarned**
CreditsEarnedThisTerm	**Current**	

CreditsAttemptedThisAidYear

CreditsEarnedThisAidYear

Reserved words used with the Athletic Eligibility Audit:
(See the *Athletic Eligibility Audit* section in the *Technical Guide* for more information on how these are used.)

CompletedTermCount	**DegreeCreditsRequired**	**ResidenceCreditsEarned**
ResidenceCompletedTermCount	**LastCompletedTermType**	**TotalCreditsAttempted**
CreditsAppliedTowardsDegree	**PreviousAcademicYearEarnedCredits**	**TotalCreditsEarned**
PreviousTermEarnedCredits	**PreviousFullYearEarnedCredits**	**Previous2TermsEarnedCredits**
PreviousTermEarnedCredits-Fall	**FirstYearEarnedCredits**	**Previous3TermsEarnedCredits**

Scope of Reserved Words

Reserved words that can only be used in the block Header section:
BEGIN, LASTRES, MAXCLASS[ES], MAXCREDIT[S], MINGPA, MINRES, OPTIONAL, SPMAXCREDIT[S], SPMAXTERM, STANDALONEBLOCK, CHECKELECTIVECREDITSALLOWED, PSEUDO, TAG

Reserved words that can only be used in the block Body section:
ACCEPT, ALLOW, BEGINSUB, BLOCK[S], BLOCKTYPE[S], COLLEGE-nn, COPYRULESFROM, ELSE, ENDSUB, GROUP[S], HIDE, IF, INCLUDING, INCLUDING-BLOCKS-WITH, MAXSPREAD, MINAREAS, MINSPREAD, NOCOUNT, NONCOURSE[S], NOTGPA, RULE-COMPLETE, THEN

Reserved words that can be used in both the block Header and the block Body sections:
ALLBLOCKS, AND, AT, CLASS[ES], COLLEGE, CONC, CREDIT[S], DECIDE, DEGREE, DONTSHARE, ELSE, EXCEPT, EXCLUSIVE, FROM, HIGH-PRIORITY, IF, IN, LIBL, LOW-PRIORITY, LOWEST-PRIORITY, MAJOR, MAXPASSFAIL, MAXPERDISC, MAXTERM, MAXTRANSFER, MINCLASS[ES], MINCREDIT[S] , MINGRADE, MINOR, MINPERDISC, MINTERM, NONEXCLUSIVE, OR, OTHER, PROGRAM, PROXY-ADVICE, SAMEDISC, SCHOOL, SHAREWITH, SPEC, THISBLOCK, WITH

Block qualifiers that can be repeated in the block Header section:
DONTSHARE, EXCLUSIVE, HIGHPRIORITY, LOWPRIORITY, MAXCLASS[ES], MAXCREDIT[S], MAXPERDISC, MAXTERM, MAXTRANSFER, MINCLASS[ES], MINCREDIT[S], MINPERDISC, MINTERM, NONEXCLUSIVE, SHAREWITH, SPMAXCREDIT[S], SPMAXTERM

Block qualifiers that can occur only once in the block Header section:
LASTRES, MAXPASSFAIL, MINGPA, MINGRADE, MINRES, OPTIONAL, SAMEDISC, STANDALONEBLOCK, CHECKELECTIVECREDITSALLOWED,

Rule qualifiers that can be repeated in a rule:
ELSE, HIDE, HIGHPRIORITY, IF, LOWPRIORITY, MAXPERDISC, MINCLASS[ES], MINCREDIT[S] , MINPERDISC, PROXY-ADVICE, SHAREWITH, THEN

Rule qualifiers that can only occur once in a rule:
DECIDE, EXCLUSIVE, LOWEST-PRIORITY, MAXPASSFAIL, MAXSPREAD, MAXTERM, MINAREAS, MINGRADE, MINSPREAD, MINTERM, NOTGPA, SAMEDISC
Reserved words that are stand alone (are not part of a block Header or rule):
END, REMARK

Reserved words that specify courses in a course list in a block Header:
AT, WITH

Reserved words that specify courses in a course list in a rule:
AT, EXCEPT, HIDE, INCLUDING, NOCOUNT, WITH

Reserved Words that are used as part of the WITH qualifier:
DWAGE, DWTITLE, DWTERM, DWLOCATION, DWGRADE, DWGRADELETTER, DWGRADETYPE, DWCREDITS, DWRESIDENT, DWTRANSFER, DWTRANSFERSCHOOL, DWTRANSFERCOURSE, DWCREDITTYPE, DWPASSFAIL, DWSECTION, DWINPROGRESS, DWTERM, DWTERMTYPE

Reserved Words that are only allowed in an If statement in an Aid audit (usually in an AWARD block):
LastCompletedTermType, TotalCreditsEarned, ResidenceCreditsEarned, TotalCreditsAttempted, CreditsAttemptedThisTerm, CreditsEarnedThisTerm, DegreeCreditsRequired, CreditsAttemptedThisAidYear, CreditsEarnedThisAidYear

Reserved Word Definitions

1stConc, 2ndConc, … 9thConc

1stMajor, 2ndMajor, … 9thMajor

1stMinor, 2ndMinor, … 9thMinor

1stProgram, 2ndProgram, … 9thProgram

1stCollege, 2ndCollege, … 9thCollege

1stLibl, 2ndLibl, … 9thLibl

1stSpec, 2ndSpec, … 9thSpec

Used in an if-statement; checks the student's academic data.

```
If (1stMajor = BIO) THEN
 15 Credits in BIO 3@
   Label "Biology elective for Biology majors"
Else If (1stMajor = CHEM) THEN
 12 Credits in BIO 3@
   Label "Biology elective for Chemistry majors"
Else
 10 Credits in BIO 3@
    Label "Biology elective for non-Bio/Chem  majors";
```

Notes:

1.	**nthMAJOR/MINOR/CONC/PROGRAM/COLLEGE/LIBL/SPEC** does not have to be noted in UCX-SCR002.
2.	The st, nd, rd, and th are mostly ignored and can go with any number – but must be present. That is, you can say 1ndMajor, 2rdMinor or 3stConc – the parser does not really care as long as you have something there. You can actually use 2xxMajor if you like.
3.	The values found on the rad_goaldata_dtl determine the order of the majors, minors, and concentrations – and not the order of the blocks found in the audit.

ACCEPT (rule)

Same as **Allow**.

ALLBLOCKS (header)

Indicates the scope for the **NonExclusive** (**Share**, **ShareWith**) qualifier. This scope signifies that all blocks are to be considered for applying the specified number of credits or classes non-exclusively.

```
BEGIN

36 Credits
Share 10 Credits (AllBlocks)

;

BEGIN

36 Credits
ShareWith (AllBlocks)

;
```

Notes:

1.	**AllBlocks** is used within a block header, only with **NonExclusive, Share, or ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can also be used to satisfy requirements in any one other block.

ALLBLOCKS (rule)

Indicates the scope for the **NonExclusive** (**Share**, **ShareWith**) qualifier. This scope signifies that all blocks are to be considered for applying the specified number of credits or classes non-exclusively.

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

Notes:

1.	**AllBlocks** is used within a rule, only with **NonExclusive, Share, or ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the rule can also be used to satisfy requirements in any one other block.

ALLOW (rule)

Indicates to the Auditor that a fewer number of credits are allowed to satisfy the course rule. You may use **Allow** (same as **Accept**) to deal with transfer classes that are not the same number of credits at your school. It is suggested that you use **Allow** because **Accept** sounds too much like **Except,** which is another Degree Works feature.

```
Audit Advice will display as "5 Credits in HIST 134" even though the student
may take the class for 4 credits at a transfer institution.

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

Notes:

1.	**Allow** is used within a course rule statement to indicate what range of credits the Auditor Engine will actually use to determine whether or not the rule has been satisfied.
2	The audit advice always hides the allowable, lower-credit value telling the student that a higher number is required. This is useful when requiring a certain number of credits but allowing a lower number of credits if the class was taken at a transfer institution or if the number of credits for the class has changed over catalog years.

AlwaysShowInAdvice (rule)

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

In this example, ACCT 105 will always show in the advice on the worksheet as long as more classes are needed; that is, as long as the rule is not yet complete. This example represents an institution that would allow ACCT 105 to apply to the rule but yet still show in the advice since the class may be repeated for credit.

```
3 Classes in ACCT 103, 105 AlwaysShowInAdvice, 112
  Label "Accounting Requirement;
```

When using **WITH** the AlwaysShow keyword comes first. You can also use the lengthy AlwaysShowInAdvice or the shorter AlwaysShow.

```
6 Credits in RORY 104 AlwaysShow (WITH DWAge < 5)
  Label "Rory Requirement";
```

Notes:

1.	The **AlwaysShowInAdvice** keyword is only allowed on courses specified in a course rule.
2.	The keyword follows the course number and precedes the WITH qualifier.
3.	The keyword is allowed with a course rule specified with a plus(+) list of classes but it would be a rare situation in which it would make sense to do this. If you do add this keyword to a course on a plus list the course will continue to show as being needed even after the student has taken the class.
4.	Using this keyword along with the HideFromAdvice keyword on any particular course is not advised. You can scribe both, but the Hide operator takes precedence.

AND (header)

Boolean operator; connector in list of courses or a link between **Classes** and **Credits**.

```
BEGIN
36 Credits and 12 Classes;

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

Notes:

1.	**And** can connect **Credits[s]** and **Class[es].**
2.	The plus sign (+) is not allowed in place of **And** in the block header. The plus cannot be used as a connector between **Credit [s]** and **Class[es].**

AND (rule)

Boolean operator; connector in lists of courses or data conditions in an **If** rule, or a link between **Classes** and **Credits**.

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

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

IF ((MAJOR=HIST) and (MINOR=EDU)) THEN
 1 Block (OTHER=EDUH)
  Label "Education Requirements";
```

Notes:

1.	The semicolon at the end of a rule statement implies **and** as the connector between rules.
2.	**And** can connect data conditions in an **If** statement.
3.	**And** can connect **Credits**[s] and **Class**[es] in a course statement.
4.	**And** or plus sign (+) can be used between courses in a list in a course rule. The plus in a course list is equivalent to **and**. The plus cannot be used in place of **and** in an **If** condition or as a connector between **Credit**[s] **and Class**[es].
5.	**And** cannot be intermingled with **or** in a course list. The connectors in a list of courses must all be either **and**/plus or **or**/comma. For example: "BIO 100, CHE 100 + PHY 100" is invalid.
6.	If **and** is used in a course list, then the number of courses listed must match the number of **Classes** specified. For example, "4 **Classes in** BIO100 + CHE100 + PHY100" is invalid. This syntax check is not done if any of the courses contain a wildcard or range. For example, "4 **Classes in** BIO100:199 + CHE1@ + PHY@" is valid.

AT (header)

Precedes a location code in a course list.

```
##maximum of 3 courses in MUS100 at St. Mary's

BEGIN

Maxclasses 3 in MUS100 at SM

;
```

Notes:

1.	Some colleges offer courses at multiple locations and, depending on the location where the course was taken, the course may or may not fulfill a requirement. The **at** keyword has been added to the DAP language to handle this situation. For example, "MATH100, 200 **at** SM", will count MATH 200 only if taken at location SM. If **at** is not specified, then no location is checked for the course.
2.	Location codes must be valid in the institution's code list (UCX-STU576).
3.	Only one location can be listed for a course. If multiple locations are needed, then repeat the course, for example, "ENG 100 **at** SM, ENG 100 **at** XX". If no location is given for a course, then the Auditor Engine does not check location when matching a student's course to the requirement.
4.	Only the last course before **at** is qualified by the location. For example, "ENG 101, 110 **at** SM" qualifies only ENG 110 as being a St. Mary's course. If both ENG 101 and ENG 110 are taught at St. Mary's, then the rule would be "ENG 101 **at** SM, ENG 110 **at** SM".
5.	For ranges of course numbers, the location applies to all courses in the range. For example, "ART 100:102 **at** SM" is the same as "ART 100 **at** SM, ART 101 **at** SM, ART 102 **at** SM".
6.	**At** is only allowed after a course. See **Class**[**es**] for a definition and examples of courses.
7.	Only use **at** after a course if your institution assigns location to courses and the student system can supply the location to the Auditor Engine.

AT (rule)

Precedes a location code in a course list.

```
##ENG 101 at St. Mary's

1 Class in  ENG 100,101 at SM
  Label "English";

1 Class in  ENG 101 at SM, ENG 110 at XX
  Label "English";
```

Notes:

1.	Some colleges offer courses at multiple locations and, depending on the location where the course was taken, the course may or may not fulfill a requirement. The **at** keyword has been added to the DAP language to handle this situation. For example, "MATH100, 200 **at** SM", will count MATH 200 only if taken at location SM. If **at** is not specified, then no location is checked for the course.
2.	Location codes must be valid in the institution's code list (UCX-STU576).
3.	Only one location can be listed for a course. If multiple locations are needed, then repeat the course, for example, "ENG 100 **at** SM, ENG 100 **at** XX". If no location is given for a course, then the Auditor Engine does not check location when matching a student's course to the requirement.
4.	Only the last course before **at** is qualified by the location. For example, "ENG 101, 110 **at** SM" qualifies only ENG 110 as being a St. Mary's course. If both ENG 101 and ENG 110 are taught at St. Mary's, then the rule would be "ENG 101 **at** SM, ENG 110 **at** SM".
5.	For ranges of course numbers, the location applies to all courses in the range. For example, "ART 100:102 **at** SM" is the same as "ART 100 **at** SM, ART 101 **at** SM, ART 102 **at** SM".
6.	**At** is only allowed after a course. See **Class[es]** for a definition and examples of courses.
7.	Only use **at** after a course if your institution assigns location to courses and the student system can supply the location to the Auditor Engine.

BEGIN (header)

Begins a block of requirements.

```
##Bachelor of Arts degree

BEGIN

120 Credits
MinGPA 2.0

;
```

Notes:

1.	The first keyword of each block must be **BEGIN**. Only blank lines and comments beginning with "**#**" or "!" are allowed prior to **BEGIN**.
2.	For each **BEGIN** there is an **END**. at the end of the requirements block.

BEGINSUB (rule)

Begins a subset of rules that should be treated as one rule.

```
## Of the 9 credits in PSY 1@ and SOC 1@, 6 credits can also apply to the
major.

BeginSub
  6 Credits in PSY 1@
    Label PSYCH100 "Psychology 100 Level";
  If (Attribute = SPCL) then
   3 Credits in SOC 1@, 2@
     Label SOC100200 "Sociology 100 or 200 Level"
  else
   3 Credits in SOC 1@
     Label SOC100 "Sociology 100 Level";
EndSub
 Nonexclusive 6 Credits (MAJOR)
  Label "100 LEVEL PSY AND SOC";
```

Notes:

1.	For each **BeginSub** there must be an **EndSub**. A "subset" is the list of rules between **BeginSub** and **EndSub**.
2.	Following **BeginSub** is one or more rule statements, each ending in a semicolon. Following the last rule in the subset is **EndSub**. Following **EndSub** is an optional list of rule qualifiers that will be applied by the Auditor Engine to the entire subset of rules as if the subset was one rule.
3.	**BeginSub** and **EndSub** create a subset of rules against which rule qualifiers can be applied.
4.	**BeginSub** is allowed only at the beginning of a rule or within an **If** statement. **BeginSub** cannot be used within a **Group**.
5.	Any rule other than one beginning within **BeginSub** can be part of a subset.
6.	**BeginSub** and **EndSub** cannot be nested. "Nested" means two **BeginSub** keywords in the same rule.
7.	Allowable rule qualifiers: **Exclusive, High-Priority, Low-Priority, Lowest-Priority, MaxPerDisc, MaxPassFail, MaxTransfer, MaxSpread, MinGrade, MinPerDisc, MinSpread, NonExclusive, NotGPA, Proxy- Advice, SameDisc, MinClass, MinCredit.**
8.	**Remarks** are not allowed within a subset.

BLOCK[s] (rule)

Pulls another requirements block into the current block.

```
1 Block (OTHER=PHYSED)
  Label "Phys Ed and Health";

If (ROTC=YES) Then
  1 Block (OTHER=ROTC)
  Label "ROTC Requirement";
```

Notes:

1.	**Block**[**s**] must be preceded by the number one and followed by left parenthesis, followed by a valid block type (**OTHER** for custom blocks), followed by an equals sign, followed by a valid code for that block type, followed by right parenthesis.
2.	If the block is type **OTHER**, then the code following the equals sign must be in the custom block list (constructed from all requirement blocks with block type **OTHER** in the DAP database); otherwise, the code must be valid in the institution's list of codes for the block type. For example, "1 **Block** (**OTHER**=GENED)" requires that a requirements block with block type **OTHER** and block type value GENED must exist in the database. Another example is "1 **Block** (**MAJOR**=ART)", which requires that ART be valid in the institution's code list for **MAJOR** and that a requirements block with block type **MAJOR** and block type value ART must exist in the database. However, this ART block will only be pulled into the audit if the student has a major of ART on her curriculum record. Unlike with OTHER blocks, you cannot pull in a MAJOR, MINOR, etc block into the audit without that goal type being on the student's record. For this reason it is rare that the Block rule would use any type other than OTHER.
3.	The **Block**[**s**] rule indicates a link to the referenced block. The referenced block must already exist or the Parser Engine will give an error.
4.	**Block**[**s**] is like a macro to the Auditor Engine. When the student's requirements are assembled from various blocks, the referenced block is inserted into the current block at this point. The **Block** rule saves a scriber from retyping course lists in multiple places. The Auditor Engine substitutes the block definition for the referenced block whenever the **Block** statement is used.
5.	**Block**[**s**] cannot be used with **ID** because it is assumed that linking the local block to a custom block for an ID is unnecessary. A custom block for an ID may link in another block but no other block should link in an **ID** block.
6.	When **Block**[**s**] is used as a stand-alone rule, rule qualifiers other than **NotGPA** are not allowed. For example, "1 **Block** (**OTHER**=LANG) **NotGPA**" is valid but "1 **Block** (**OTHER**=LANG) **MinGrade** 3.0" is invalid. If **MinGrade** is needed, it should be entered as part of the referenced block, the LANG block in this example.
7.	A semicolon is not used to end the block rule when **Block**[**s**] is used within **Group**. When used within a **Group** rule, the **Block** rule is enclosed in parentheses.
8.	Allowable rule qualifiers: **NotGPA**.

BLOCKTYPE[S] (rule)

Precedes a required block type.

```
1 BlockType (CONC)
  Label "Concentration required";

1 BlockType (MAJOR)
  Label "Major required";

1 BlockType (MINOR)
  Label "Minor required";
```

Notes:

1.	**BlockTypes**[**s**] must follow an integer that indicates the number of block types listed in parentheses that must be satisfied for this rule to be considered complete by the Auditor Engine.
2.	**BlockTypes**[**s**] must be followed by one of these block types, **COLLEGE, CONC, LIBL, DEGREE**, **MAJOR**, **MINOR**, **PROGRAM**, **SCHOOL**, or **SPEC**, enclosed in parentheses.
3.	To specify that more than one block type is required, enter a **BlockType** rule for each block type. For example, if both a minor and a concentration are required, the rules are: "1 **BlockType** (**MINOR**)" "1 **BlockType** (**CONC**)".
4.	**ID** and **OTHER** are not valid for **BlockType** because they require specification of a particular block.
5.	The **BlockType** rule indicates that a student must have a block of a certain type (e.g. **MAJOR**), but the value for that type does not matter (any major, not a specific major).
6.	**BlockType** [**s**] constitutes a stand-alone rule. If the student's data does not cause a block of that type to be evaluated by the Auditor Engine, or the block is evaluated but is incomplete, then the student does not pass the audit.
7.	A semicolon is not used to end the block type rule when **BlockType** [**s**] is used within **Groups**. When used within a **Group** rule, the **BlockType** rule is enclosed in parentheses.
8.	Allowable rule qualifiers: none.
9.	When the audit includes multiple majors and multiple concentrations, for example, the auditor attempts to locate the concentration that is associated with the major that is requiring the concentration – via the Blocktype(CONC) requirement. The auditor first checks to see if either the major or the conc block has a secondary tag associating it with the other block. If this check fails to find a connection then the auditor will see if the concentration is attached to the major via the attach fields on the CONC rad_goalData_dtl record.

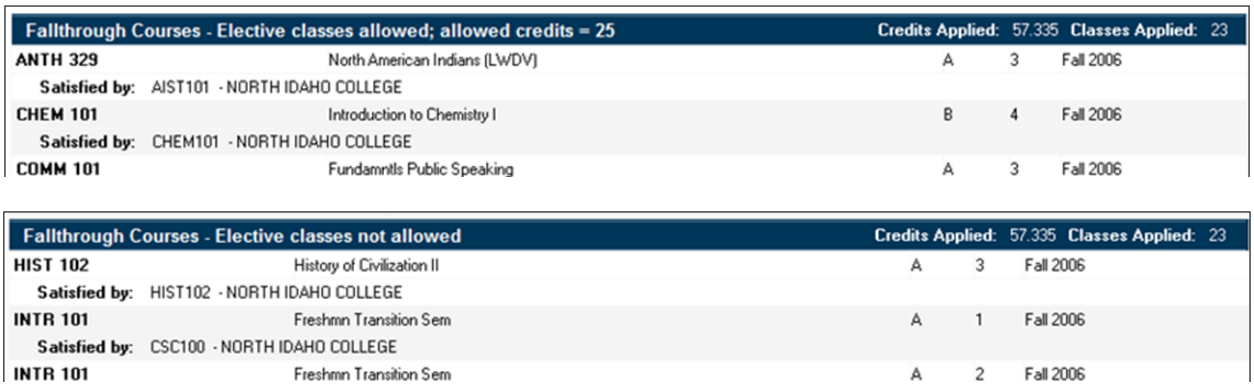
CHECKELECTIVECREDITSALLOWED (header)

Used in the starting block to tell the auditor to calculate how many elective credits the student is allowed to take based on the credits required in each of the other blocks.

```
BEGIN
CheckElectiveCreditsAllowed
;
```

Notes:

1.	Only used in the starting block (usually DEGREE)
2.	Each block must have a **NN Credits** qualifier – (use Pseudo if block does not have a real credit limit).
3.	Elective credits allowed (ECA) = Degree credits minus sum of block credits plus waived credits plus shared credits. Example: 120 Degree credits – 40 major credits – 30 minor credits – 30 gened credits + 3 waived credits + 9 shared credits = 32 ECA credits
4.	Classes with a WA (waived) status are looked up on the rad-course-mst to get the amount of waived credits; these waived credits are added to the ECA total.
5.	For each additional block where a class applies we add those amount of credits to the ECA total.
6.	As the student progresses towards her degree their ECA may go up and down depending on how requirements share courses and what classes a student takes.
7.	The fall-through list is examined using the ECA credit value. If there are more credits in fall-through than allowed the classes are processed and marked as ECA-OVERFLOW if they exceed the ECA limit; all other fall-through classes are marked at ECA-OK. Completed classes may also be marked as ECA-OVERFLOW - not just the in-progress classes.
8.	When the ECA limit is exceeded the auditor first removes (marks as OVERFLOW) classes starting at those in the most future term (in case there are preregistered classes) and then moves backwards in time term by term. The auditor will also make as efficient use of credits as possible while still preferring to keep older in-progress classes over more future in-progress classes. However, regardless of credits a completed class will be kept over an in-progress class.
9.	The ECA overflow credits are extracted into the CPA dap-result-dtl FALLCRCL record as rad-value4 and rad-number4.
10.	The ECA-OK and ECA-OVERFLOW values are extracted onto the CPA dap-result-dtl table for FALLCLASSAPPLIED record as rad-value4.
11.	This qualifier is created as a normal Qualifier xml node with the attributes and values for Overall credits allowed, Blocksum credits, Waived credits, Shared credits and Overflow credits.

12.	The Fallthrough/Class xml nodes are created with a Code attribute containing the ECA-OK or ECA-OVERFLOW values.
13.	The standard worksheets do not show any of these values except for the Diagnostics Report. If you want to make use or display any of these ECA values in your worksheet you must localize the xsl files. A sample of what this might look like follows:  **Fallthrough Courses - Elective classes allowed; allowed credits = 25** — Credits Applied: 57.335 Classes Applied: 23 ANTH 329 North American Indians (LWDV) A 3 Fall 2006 Satisfied by: AIST101 - NORTH IDAHO COLLEGE CHEM 101 Introduction to Chemistry I B 4 Fall 2006 Satisfied by: CHEM101 - NORTH IDAHO COLLEGE COMM 101 Fundamntls Public Speaking A 3 Fall 2006 **Fallthrough Courses - Elective classes not allowed** — Credits Applied: 57.335 Classes Applied: 23 HIST 102 History of Civilization II A 3 Fall 2006 Satisfied by: HIST102 - NORTH IDAHO COLLEGE INTR 101 Freshmn Transition Sem A 1 Fall 2006 Satisfied by: CSC100 - NORTH IDAHO COLLEGE INTR 101 Freshmn Transition Sem A 2 Fall 2006
14.	The Calculate Elective Credits Allowed flag in UCX-CFG020 DAP14 is trumped if this qualifier is in place. That is, if you are using this qualifier the CFG020 flag is ignored. The calculation performed when the CFG020 flag is enabled is different from the results of this calculation. When the CFG020 flag is used the auditor checks to see which blocks are required vs optional and makes appropriate calculations. This special check is not performed when this qualifier is used instead of the flag. Also review the UCX-CFG020 DAP14 documentation.

CLASS[ES] (header)

Indicates how many courses are required.

```
BEGIN
20 Classes
;

BEGIN
40:50 Classes and 120 Credits
;
```

Notes:

1.	**Class[es]** must NOT be followed by a course list when used in a block header. (If used within another qualifier, then a course list may follow, depending on the qualifier.)
2.	**Class[es]** must follow an integer that indicates the number of courses required. The number of courses can be a range, specified as "integer colon integer". If a range is used, then the lower bound (left side) must be less than or equal to the upper bound (right side). If the integer after the colon is omitted, then the Parser will give an error. For example, "2: **Classes**" is invalid. A range is used to indicate that the number of courses that satisfy the requirement varies between a lower and upper bound. The requirement will be met if the number of courses taken by the student is greater than or equal to the lower bound. For example, "2:4 **Classes**" is satisfied if the student takes 2, 3, 4, or more courses. Only the lower number in the range will display in the audit.
3.	The Auditor treats the number of classes specified in the header as a minimum that must be satisfied. If the student takes at least the minimum number of courses, the qualifier will be met.
4.	When specifying both a number of courses and a number of credits, connect **Class[es]**] to **Credits**[**s**] with **and** or **or**. For example, "3 **Classes and 6 Credits**" means both conditions must be satisfied. But, "3 **Classes or 6 Credits**" means either condition can be satisfied to meet the requirement.

CLASS[ES] (rule)

Indicates how many courses are required.

```
3 Classes and 9 Credits in MATH @
  Label "Math Requirement";

3 Classes in PE @
  Label "PE Requirement";

2 Classes From BIO @, PHY @
  Label "Science Requirement";

1 Class in MUS 101
  Label "Introduction to Music";
```

Notes:

1.	**Class[es]** must be followed by a course list when used in a course rule. **Course** is described below.
2.	**Class[es]**] must follow an integer that indicates the number of courses required. The number of courses can be a range, specified as "integer colon integer". If a range is used, then the lower bound (left side) must be less than or equal to the upper bound (right side). If the integer after the colon is omitted, then the Parser will give an error. For example, "2: **Classes**" is invalid. A range is used to indicate that the number of courses that satisfy the requirement varies between a lower and upper bound. The requirement will be met if the number of courses taken by the student is greater than or equal to the lower bound and less than or equal to the upper bound. For example, "2:4 **Classes**" is satisfied if the student takes 2, 3, or 4, courses. The upper bound of the range of classes indicates that no more than the upper bound number of courses will be used to satisfy the requirement. It is a strict cap. If the range is 2:4 then no more than 4 courses will be used to fill the requirement. Only the lower number in the range will display in the audit.
3.	If a range is not used, then the exact number of courses specified by the integer preceding **Class[es]** must be taken to satisfy this requirement. For example, "2 **Classes**" specifies that exactly 2 courses must be taken to satisfy the requirement.
4.	When specifying both a number of courses and a number of credits, connect **Class[es]** to **Credits[s]** with **and** or **or**. For example, "3 **Classes and 6 Credits**" means both conditions must be satisfied. However, both numbers are a minimum. It is possible 6 1-credit classes could be used to satisfy this requirement. As long as at least 3 classes and at least 6 credits apply the requirement will be complete. Another example: **1 Class and 3 Credits**. If you really want a single 3-credit class to apply then you should scribe it as **1 Class in MATH @ (With DWCredits=3)**. **Note:** "3 **Classes or** 6 **Credits**" means either condition can be satisfied to meet the requirement.
5.	Within a course rule, it is possible to use **Class[es]** before the course list and **Credit[s]** as part of rule qualifiers. Likewise, it is possible to use **Credit[s]** before the course list and **Class[es]** as part of rule qualifiers. For example, "12.0 **Credits in** BIO **@ Share** 3 **Classes**" is valid. "4 **Classes in** MATH **@ Share** 9 **Credits"** is also valid.
6.	**Class[es]** may optionally be followed by **in** or **from** in a course rule.
7.	A semicolon is not used to end the course rule when **Class[es]** is used within a **Group** or within an **If** rule. When used within a **Group** rule, the course rule is enclosed in parentheses.

8.	Allowable rule qualifiers: DontShare, Exclusive**,** Hide, Hide-Rule**,** High-Priority, Low-Priority, Lowest-Priority, MaxPassFail, MaxPerDisc, MaxSpread, MaxTerm, MaxTransfer, MinAreas, MinGrade, MinPerDisc, MinSpread, MinTerm, NonExclusive, NotGPA, Proxy-Advice, SameDisc, Share, ShareWith, Then, With.
9.	The wildcard (@) can be used as part of the discipline or course number. The wildcard signifies one or more of any alphanumeric characters. When a wildcard is used at the end of a discipline or the beginning of a course number, it is best to separate discipline from course number with a space. For example: Assuming the maximum discipline length is 4 then “BIO@L” will parse as discipline of “BIO@” and course number of “L”, but “BIO @L” will parse as discipline of “BIO” and course number of “@L”.
10.	A range of course numbers is indicated by separating two course numbers with a colon. The course numbers cannot contain any letters or wildcards. The lower bound (left side) must be less than or equal to the upper bound (right side). For example, “MATH 100:199” is valid; “BIO 100:199L” is invalid, and “SOC 3@:499” is also invalid.
11.	The discipline is followed by one course number, many course numbers, or the wildcard. The list of courses is separated by either commas (**or**) or plus signs (**and**). Examples: “MATH100, BIO 115, 120, 120L, PE 100:199, HE @”; “CHE 1@ + 1@L + PHY 100:115”; “FRE 100:299 OR SPA 100:299 OR GER 100:299”; “GRK 100 AND 125 AND LAT 100 AND 115”.
12.	When listing multiple courses for the same discipline, the discipline is optionally repeated before the course number. For example, “MUS 100, 115” is equivalent to “MUS 100, MUS 115”. If course numbers in a multiple course listing with the same discipline start with an alpha character, the discipline must be repeated before each course number that starts with an alpha character. For example, “MUS P100, MUS P115, MUS P200”.
13.	Courses in a list must be connected by a logical **and** or a logical **or**. Do not intermingle **and** and **or** in a course list. A logical **or** is represented by the literal **or** or a comma. A logical **and** is represented by the literal **and** or a plus sign (+). A logical **and** can only be used in the first course list that appears in a course rule, not with a course list associated with **Including**, **Except**, rule qualifiers, or block qualifiers.
14.	A course cannot be repeated within a list of courses. For example, “MATH101, 101” is invalid, as is “MATH101, MATH101”.

15.	Courses are validated in the student system using a subroutine called GETCOURSE. Only courses that do not contain a wildcard or course number range are validated. The GETCOURSE subroutine looks up the course key, composed of discipline and course number, in the Degree Works database (RAD COURSE-MST for stand-alone).
16.	A new flag has been added to UCX-CFG020 DAP13 to control whether the wildcard matches zero or more characters or one or more characters. When the flag is set to 1 (matching the normal behavior), ENGL 123 would not match the requirement; when the flag is set to 0 (stating that the wildcard matches 0 or more characters), ENGL 123 would match the requirement "**1 Class in ENGL 123@**". Regardless of the flag's setting, classes such as ENGL 123A, 123B, etc. would also match.

COLLEGE (header)

Indicates a college block; precedes a college code; designates an organizational unit within a university, such as the College of Engineering.

```
BEGIN

36 Credits
Nonexclusive 10 Credits (COLLEGE)

;

BEGIN

36 Credits
ShareWith (COLLEGE)

;
```

Notes:

1.	**COLLEGE** is used within a block header, only with **Nonexclusive, Share, or ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can also be used to satisfy requirements in the **COLLEGE** block (if the **COLLEGE** block exists for the student).

COLLEGE (rule)

Indicates a college block; precedes a college code; designates an organizational unit within a university, such as the College of Engineering.

```
1 BlockType (COLLEGE)
  Label "College block is required";

If (COLLEGE=ENGR) Then
  MATH 400
  Label "Math 400";

6 Credits in BUS 1@
 ShareWith (COLLEGE)
  Label "Business";
```

Notes:

1.	**COLLEGE** is followed by a college code, such as ENGR for Engineering or AS for Arts and Sciences, except in a block type rule.
2.	The college code must be valid in the institution's code list.
3.	**COLLEGE** can be used as part of a condition following **If**.
4.	**COLLEGE** can be used with **BlockType**[s], **If**, or **NonExclusive**, **Share**, or **ShareWith**.

COLLEGE-nn (rule)

Indicates a college associated with a major or minor; precedes a college code; designates an organizational unit within a university, such as the College of Engineering.

```
If (COLLEGE-1=ENGR) Then
  1 Class in MATH 400
    Label "Math 400";

If (COLLEGE-1=chem or COLLEGE-2=CHEM) Then
 3 Classes in CHEM 32@
   Label "Organic Chemistry";
```

Notes:

1.	**COLLEGE**-nn is followed by a college code, such as ENGR for Engineering or AS for Arts and Sciences.
2.	The college code must be valid in the institution's code list.
3.	**COLLEGE**-nn can only be used as part of a condition following **If**.
4.	The "nn" in **COLLEGE**-nn can be 1-99. Currently, **COLLEGE**-1 refers to the student's first college goal. **COLLEGE**-2 refers to the student's second college goal. **COLLEGE**-3 and **COLLEGE**-4 follow logically.

CONC (header)

Indicates a concentration block type (**CONC**) against which **NonExclusive** is applied.

```
BEGIN

36 Credits
Share 9 Credits (CONC)

;

BEGIN

36 Credits
ShareWith(CONC)

;
```

Notes:

1.	**CONC** is used within a block header, only with **Nonexclusive, Share,** or **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can also be used to satisfy requirements in the **CONC** block (if the **CONC** block exists for the student).

CONC (rule)

Indicates a concentration block type (**CONC**), which, in some cases, may precede a concentration code.

```
1 BlockType (CONC)
Label "Concentration block is required";

If (CONC=AME) Then
 1 Class in HIST 300
  Label "History 300";

6 Credits in BUS 1@
 Nonexclusive (CONC)
  Label "Business";
```

Notes:

1.	**CONC** is followed by a concentration code, such as AME for American History, except in a block type rule.
2.	The concentration code must be valid in the institution's code list.
3.	**CONC** can be used as part of a condition following **If**.
4.	**CONC** can be used with **BlockType**[**s**], **If**, **Nonexclusive**, **Share**, or **ShareWith**.

CopyRulesFrom (rule)

A special command used to copy rules from another block. This is best used when you have a requirement that is used by many different blocks and is changed often - this cuts down on block maintenance.

Example - you may have two blocks like this:

```
BEGIN  # RA000456
;
1 Class in HIST 300
  Label "History 300";

CopyRulesFrom (RA000123);

6 Credits in BUS 1@
  Label "Business";
END.


BEGIN  # RA000123
;
14 Credits in ARCH 356, AGEC 383, 451, ANTH @ (With ATTRIBUTE = UPDV)
    Label "Humanities requirement";

 1 Classes in COMM 101, ENGL 207, ENGL 208, ENGL 209, ENGL 313, ENGL 316
    Label "Writing";
END.
```

You must first modify block RA000123 and then parse and save – and then you can parse and save block RA000456 (and all other blocks that refer to this block) – resulting in the following – the CopyRulesFrom will be replace with all of the rules from RA000123 – though you will not actually see this replacement in Scribe – this replacement is visible only within the audit.

```
BEGIN  # RA000456
;
1 Class in HIST 300
  Label "History 300";

14 Credits in ARCH 356, AGEC 383, 451, ANTH @ (With ATTRIBUTE = UPDV)
    Label "Humanities requirement";

 1 Classes in COMM 101, ENGL 207, ENGL 208, ENGL 209, ENGL 313, ENGL 316
    Label "Writing";

6 Credits in BUS 1@
  Label "Business";
END.
```

When using an IF-statement use a subset or BeginIf/Endif

```
BEGIN  # RA000233
;
1 Class in HIST 300
  Label "History 300";

If (Major = PSY) then
  BeginIf
```

```
  CopyRulesFrom (RA000355)
  EndIf

6 Credits in BUS 1@
  Label "Business";
END.
```

Notes:

1.	Copy-Rules-From can be used instead of CopyRulesFrom.
2.	Header qualifiers in the other block are not copied – they are ignored.
3.	The referenced block must be parsed/saved first – and then the calling blocks must be parsed/saved after that. It is best to parse/save the referenced block and then run DAP16 in Transit to be sure all of the blocks that use CopyRulesFrom are reparsed.
4.	You cannot use CopyFromRules in a group, but you can use it in a subset.
5.	You can use CopyRulesFrom in an IF-statement but if the block you are copying has more than one rule then you need to enclose the CopyRulesFrom within a subset or within BeginIf/EndIf. Always enclose your CopyRulesFrom in a subset or BeginIf/EndIf to be safe.

COURSE[S] (rule)

Same as **Class**[**es**].

CREDIT[S] (header)

Units toward a degree; formatted as nnn.nnn.

```
BEGIN
36 Credits
;

BEGIN
120:132 Credits or 36:45 Classes
;

BEGIN
128.0 Credits
;

BEGIN
36 Classes or 120 Credits
;

BEGIN
38 Credits Pseudo
;
```

Notes:

1.	When used in a block header, **Credits** should follow a real number but should not be followed by a course list.
2.	**Credits** [**s**] must follow a real number that indicates the number of credits, for example, 3, 3., 3.0, 3.05, 3.125. The number of credits can be a range, specified as "real number colon real number". If a range is used then the lower bound (left side) must be less than or equal to the upper bound (right side). If the number after the colon is omitted, then the Parser will give an error. For example, "3: **Credits**" is invalid. A range is used to indicate that the number of credits that satisfy the requirement varies between a lower and upper bound. The requirement will be met if the number of credits taken by the student is greater than or equal to the lower bound. For example, "2:4 **Credits**" is satisfied if the student takes 2, 3, 4, or more credits. Only the lower number in the range will display in the audit.
3.	If a range is not used, then the number of credits specified by the real number preceding **Credits**[**s**] must be taken to satisfy the requirement. For example, "2 **Credits**" specifies that 2 or more credits must be taken to satisfy the requirement.
4.	When specifying both a number of courses and a number of credits, connect **Class**[**es**] to **Credits**[**s**] with **and** or **or**. For example, "3 **Classes and** 6 **Credits**" means both conditions must be satisfied. But, "3 **Classes or** 6 **Credits**" means either condition can be satisfied to meet the requirement.
5.	See **Class**[**es**] for a definition and examples of courses.
6.	Pseudo is used in a block that does not have a strict credit limit. It is used in conjunction with CheckElectiveCreditsAllowed in the degree block. When Pseudo is specified the credits qualifier will always be satisfied but the auditor is told how many credits this block is worth when calculating the elective credits allowed.

CREDIT[S] (rule)

Units toward a degree; formatted as nnn.nnn.

```
3 Credits in ART@
  Label "Art requirement";

6:9 Credits from BIO @, PHY @
  Label "General Science";

1.125 Credits PE @
  Label "Physical education";

6 Credits and 2 Classes in MATH 200:299
  Label "Intermediate Math";
```

Notes:

1.	**Credit[s]** must be followed by at least one course when used in a course rule.
2.	**Credit[s]** must follow a real number that indicates the number of credits, for example, 3, 3., 3.0, 3.05, 3.125. The number of credits can be a range, specified as real number colon real number. If a range is used then the lower bound (left side) must be less than or equal to the upper bound (right side). If the number after the colon is omitted then the Parser will give an error. For example, "3: **Credits**" is invalid. A range is used to indicate that the number of credits that satisfy the requirement varies between a lower and upper bound. The requirement will be met if the number of credits taken by the student is greater than or equal to the lower bound and less than or equal to the upper bound. For example, "2:4 **Credits**" is satisfied if the student takes 2, 3, or 4, credits. Only the lower number in the range will display in the audit. Different from **Class**[es], the upper bound of the range of credits is not a strict cap. The upper bound can be marginally exceeded if a course has too many credits to fall on the upper bound. For example, if the range is "6:9 **Credits",** then 4 courses of 2 credits will fill the requirement, as will 2 courses of 5 credits. However, if 4 courses of 3 credits are taken, only 3 of the courses will be used to satisfy the rule.
3.	If a range is not used, then the number of credits specified by the real number preceding **Credit[s]** must be taken to satisfy the requirement. For example, "2 **Credits**" specifies that exactly 2 credits (or marginally over 2 credits) must be taken to satisfy the requirement. One course for 2 or more credits, or two 1-credit courses, can satisfy the rule.
4.	When specifying both a number of courses and a number of credits, connect **Class[es]** to **Credit[s]** with **and** or **or**. For example, "3 **Classes and** 6 **Credits**" means both conditions must be satisfied. But, "3 **Classes or** 6 **Credits**" means either condition can be satisfied to meet the requirement.
5.	Within a course rule, it is possible to use **Class[es]** before the course list and **Credit[s]** as part of rule qualifiers. Likewise, it is possible to use **Credit[s]** before the course list and **Class[es]** as part of rule qualifiers. For example, "12.0 **Credits in** BIO **@ Nonexclusive** 3 **Classes** (**THISBLOCK**)" is valid. "4 **Classes in** MATH **@ Nonexclusive** 9 **Credits** (**THISBLOCK**)" is also valid.
6.	**Credit[s]** may optionally be followed by **in** or **from** in a course rule.
7.	See **Class**[es] for a definition and examples of courses.
8.	A semicolon is not used to end the course rule when **Credit[s]** is used within **Group**. When used within a **Group** rule, the course rule is enclosed in parentheses.
9.	Allowable rule qualifiers: DontShare, Exclusive, Hide**,** Hide-Rule, High-Priority, Low-

	Priority, Lowest-Priority, MaxPassFail, MaxPerDisc, MaxSpread, MaxTerm, MaxTransfer, MinAreas, MinGrade, MinPerDisc, MinSpread, MinTerm, NonExclusive, NotGPA, Proxy-Advice, SameDisc, Share, ShareWith, Then, With.

DECIDE (header)

Indicates to the Auditor how to decide which classes should be removed and which should be kept when the maximum has been exceeded on a block qualifier.

```
BEGIN
MaxCredits 10 (DECIDE=NEWEST) MATH @, CHEM @
;

BEGIN
MaxPerDisc 2 Classes (DECIDE=HTRMHNUM) in (ENGL, HIST)
;

BEGIN
MaxClasses 1 (DECIDE=LOWTERM) in MATH 112, 133, 134
;

BEGIN
MaxTransfer 1 Class (DECIDE=HIGHNUM) from (UR, CO, OL)
;

BEGIN
MaxPassFail 4 Credits (DECIDE=HIGHGRADE)
;
```

Notes:

1.	**DECIDE** is used within a block header only with **Max** or **SpMax** qualifiers to indicate how the Auditor Engine should decide which classes to keep when the maximum number of classes or credits has been exceeded.
2.	The **DECIDE** code specified resides in UCX-SCR045. UCX-SCR045 is much like the UCX-CFG020 **TIEBREAK** record in that you have up to nine ways to tell the Auditor Engine how to distinguish one class as being more valuable than another.

DECIDE (rule)

Indicates to the Auditor how to decide which classes should be removed and which should be kept when the maximum has been exceeded on a course rule.

```
10 Credits (DECIDE=NEWEST) in ARTH 200:300
  Label "Art History";

2 Classes (DECIDE=BESTGRADE) in PHIL 22@, 334
  Label "Philosophy";

1 CLASS (DECIDE=ORDER) in BUS 156, 159, 168, ECON 211, 215, 218
  Label "Business or Economics";

3 Credits (DECIDE=ORDER) in MATH 123, 112, 145
  Label "Math";
```

Here, if the student takes all three classes, the Auditor places a precedence based on the order in which the classes are listed in the rule. In the above example, MATH 123 will be kept on the rule and MATH 112 and 145 will be placed elsewhere. **DECIDE=ORDER** tells the Auditor that the order of the courses in the course list within the rule is important. The Auditor then applies this logic in determining which courses to use to satisfy the rule. **ORDER** does not have to exist in UCX-SCR045.

Notes:

1.	**DECIDE** is used within a course rule statement to indicate how the Auditor Engine should decide which classes to keep when the maximum number of classes or credits has been exceeded.
2.	The **DECIDE** code specified resides in UCX-SCR045. UCX-SCR045 is much like the UCX-CFG020 **TIEBREAK** record in that you have up to nine ways to tell the Auditor Engine how to distinguish one class as being more valuable than another.
3.	The **DECIDE** keyword is only allowed on course statements; it is not allowed on groups, subsets, blocks, or blocktypes.
4.	**ORDER** is a special reserved code within the **DECIDE** statement. **ORDER** does not have to exist in UCX-SCR045. It indicates that the Auditor should decide which classes to remove and keep based on the order of the classes on the rule. The first classes listed have a higher precedence over those at the end. A class appearing earlier in the rule will be kept on the rule over one appearing later in the rule.

DEGREE (header)

Indicates a degree block type (**DEGREE**) against which **NonExclusive** is applied.

```
BEGIN
36 Credits
Share 10 Credits (DEGREE, THISBLOCK);

BEGIN
36 Credits
ShareWith (DEGREE);
```

Notes:

1.	**DEGREE** is used within a block header, only with **Nonexclusive, Share,** or **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can also be used to satisfy requirements in the **DEGREE** block (if the **DEGREE** block exists for the student).

DEGREE (rule)

Indicates a degree block type (**DEGREE**), which, in some cases, may precede a degree code.

```
1 Block (DEGREE=BA)
  Label "Bachelor of Arts";

1 BlockType (DEGREE)
  Label "Degree block is required";

If (DEGREE=BA) Then
 1 Class in PHIL 100
   Label "Philosophy 100"
ELSE
1 Class in MATH 100
  LABEL "Math 100";

6 Credits in BUS 1@
 ShareWith(DEGREE)
   LABEL "Business";
```

Notes:

1.	**DEGREE** is followed by a degree code, such as BA for Bachelor of Arts, except in a block type rule.
2.	The degree code must be valid in the institution's code list.
3.	**DEGREE** can be used as part of a condition following **If**.
4.	**DEGREE** can be used with **BlockType**[s], **If**, or **Nonexclusive, Share,** or **ShareWith**.

DISPLAY (header)

DISPLAY can be used to specify the advice text to show in place of the normal advice Degree Works would display for any Min header qualifier. The text for DISPLAY will appear all of the time. The text will not disappear when the block header qualifier has been completed. DISPLAY can be used for the entire text of the header qualifier, or can be used with PROXY-ADVICE. When used with PROXY-ADVICE, only the DISPLAY portion of the text will remain when the header qualifier is complete. DISPLAY can be used on the following block header qualifiers: MinGPA, Minres, LastRes, MinCredits, MinClasses, MinPerDisc, MinTerm, Under, Credits/Classes.

```
MinGPA 2.5 in MATH @, BIOL @
  Display "You must have a 2.5 GPA in your math and biology classes."
  Display "Your GPA in these classes is currently <APPLIED>."

MinRes 30 Credits
  Display "You have taken <APPLIED> credits here at this school."

  Proxy-Advice "You have not taken 30 credits at State College."

MinCredits 10 in PE
  DISPLAY "You have taken <APPLIED> PE credits so far."
```

Notes:

1.	**Display** is followed by up to 65 bytes of text enclosed in quotes.
2.	**Display** can be repeated as many times as needed; the text is appended together.
3.	The text will always be displayed, even when the header qualifier is complete
4.	**Display** must come before ProxyAdvice, not after it.

DONTSHARE (header)

Indicates that sharing cannot exist between this block and those specified; equivalent of **Exclusive**.

```
# DontShare can be used in conjunction to specify the specific blocks that
are
# excluded from the ShareWith
# Note: this is equivalent to saying: ShareWith (Major, Major<>CHEM)
ShareWith (Major)
DontShare (Major=CHEM)

# DontShare can be used by itself in the header to prevent certain blocks
# from sharing with it. In this case we are preventing the minor from
sharing
# with this block – in case the minor has ShareWith specified.
DontShare (Minor)

# This block does not allow sharing from any block whatsoever
DontShare (AllBlocks)

# You can black-list certain blocks from sharing with this block
DontShare (Minor=ART, Minor=HIST, Major=LANG)

# Don't share with the student's first major
DontShare (MAJOR=1st)

# Don't share with major that is associated with this conc block
DontShare (MAJOR=associated)
# Don't share with concentration that is associated with this major block
DontShare (CONC=associated)
```

Notes:

1.	**DontShare** indicates the blocks with which sharing cannot occur.
2.	**DontShare** can be used in the header along with **ShareWith**. The **DontShare** has precedence. That is, if the **ShareWith** says to share with the Major and the **DontShare** says not to share then the block will not share.
3.	**DontShare** can be used in the header without **ShareWith** to block sharing specified in other blocks.
4.	**DontShare** (Thisblock) is not supported.
5.	**DontShare** (Major**<>**CHEM) **–** do not use not-equal-to in **DontShare.**
6.	You cannot specify a number of classes or credits on DontShare in the header.
7.	**1st, 2nd, 3rd, 4th, etc** See the notes on this option under SHAREWITH (header).
8.	**ASSOCIATED** See the notes on this option under SHAREWITH (header).

DONTSHARE (rule)

Indicates that credits or courses cannot fulfill multiple requirements; equivalent of **Exclusive**.

```
6 Credits in INS1 @
  DontShare
 Label "INS1@";

3 Classes in HIST @
  DontShare 1 Class
 Label "History";
```

Notes:

1.	**DontShare** indicates that the credits or classes applied by the Auditor Engine towards satisfying this rule cannot be used to satisfy requirements in other blocks and in other rules in this block. A course applied to a **DontShare** rule is applied only to that rule and to no other rule.
2.	**DontShare** cannot be used in the same rule as **ShareWith**.
3.	Unlike **ShareWith**, there is no need to specify to which block the classes or credits apply—they apply to the current block only.
4.	**DontShare** is needed as part of the rule only if the block has been declared **ShareWith**. **DontShare** is not needed on a rule if the **ShareWith** block qualifier is not used. By default, a requirements block is exclusive.

5.	The number of classes or credits to be treated as **DontShare** optionally follows the keyword **DontShare**. To treat some classes/credits as **DontShare** and the rest as **ShareWith**, use **ShareWith** as a block qualifier and use **DontShare** with the number of classes or credits as a rule qualifier. For example:
	`BEGIN`
	
	`  36 Credits`
	
	`;`
	
	`  3 Classes in ENG 115, SOC 120, PSY 110`
	`     DontShare 6 Credits`
	`    Label "English, Sociology, and Psychology";`
	
	`END.`
	
	`----------`
	`BEGIN`
	
	`  36 Credits`
	`  ShareWith (ALLBLOCKS)`
	
	`;`
	
	`  3 Classes in ENG 115, SOC 120, PSY 110`
	`     DontShare 3 CREDITS`
	`    Label "Humanities Requirements";`
	
	`END.`

ELSE (rule)

Branch of a conditional rule.

```
If (MAJOR=HIST AND CONC=EHST) Then
  6 Credits in SPA @, FRE @, GER @, GRK @
  Label "Foreign Language"
Else
  3 Credits in LANG 100
  Label "Intro to Languages";

If (MINOR=EDU) THEN
  If (MAJOR=BIO OR MAJOR=CHE) THEN
    1 Block (OTHER=EDUS)
     Label "Education for Science Majors"
  Else
    1 Block (OTHER=EDU)
     Label "Education Requirements"
Else
  BeginSub
    3 Credits in EDU 300
      Label "Education 300";
    3 Credits in PSY 320
      Label "Psychology 320";
  EndSub
    MaxPassFail 0 Credits
     Label "Education Minor Requirements;

IF (MINOR=EDU) Then
  If (CONC=AMER) Then
    1 Block (OTHER=EDUA)
     Label "American Standards in Education"
  Else                      ##else for (CONC=AMER)
    If (CONC=ASIA) Then
      1 Block (OTHER=EDUI)
       Label "Asian Education Concentration"
    Else                    ##else for (CONC=ASIA)
      If (CONC=EURO) Then
        1 Block (OTHER=EDUE)
       Label "European Education System"
      Else                  ##else for (CONC=EURO)
       1 CLASS IN EDU 115
         Label "Education 115";
```

Notes:

1.	**Else** is optionally followed by one valid rule. If no rule appears after **Else** then **Else** must be followed by either another **Else** or a semicolon.
2.	If **Else** is not stated or no rule appears after **Else**, then the else rule is empty.
3.	**Else** must be paired with a preceding **If** and **Then**.
4.	The rule preceding **Else** should not end in a semicolon.
5.	**Else** may be repeated in a rule statement, but there must be a corresponding **If** for each **Else**.
6.	The rule following **Else** can be a single rule or multiple rules bracketed by BeginSub and EndSub.

END

End of a block of requirements; block terminator.

```
BEGIN
MinGPA 2.0

;

END.
```

Notes:

1.	The last keyword of each block must be "**END.**"
2.	**END.** follows the last rule statement or remark. Any characters after **END.** are discarded. That means anything after END is treated like a comment, without using the #. LOG and TODO can be used after the END to allow notes which can be printed with the appropriate Transit report. (do not preface either LOG or TODO with # or the report will not identify them)
3.	If the block contains zero rule statements then **END**. will follow the block header or the last remark.
4.	For each **END.** there is a **BEGIN**.
5.	**END.** is part of neither a rule or a block header. It stands alone.

ENDSUB (rule)

Ends a subset of rules that should be treated as one rule.

```
BeginSub
  6 Credits in PSY 1@
   Label "Psychology Intro";
  3 Credits in SOC 1@
   Label "Sociology Intro";
EndSub
  Share 6 Credits (MAJOR)
Label "PSYCHOLOGY AND SOCIOLOGY REQUIREMENTS";
```

Notes:

1.	For each **EndSub** there must be a preceding **BeginSub**. A subset is the list of rules between **BeginSub** and **EndSub**.
2.	Following **EndSub** is an optional list of rule qualifiers that will be applied by the Auditor Engine to the entire subset of rules as if the subset were one rule.
3.	**BeginSub** and **EndSub** create a subset of rules against which rule qualifiers can be applied.
4.	**EndSub** is allowed only as part of a subset of rules or within an **If** statement. **EndSub** cannot be used within a **Group** rule.
5.	**BeginSub** and **EndSub** cannot be nested. "Nested" means two **EndSub** keywords in the same rule.
6.	Any rule other than one beginning within **BeginSub** can be part of a subset.
7.	Allowable rule qualifiers: DontShare, Exclusive, Hide, High-Priority, Low-Priority, Lowest-Priority, MaxPerDisc, MaxPassFail, MaxSpread, MaxTransfer, MinGrade, MinPerDisc, MinSpread, NonExclusive, NotGPA, Proxy-Advice, SameDisc, Share, ShareWith, MinClass, MinCredit.

EXCEPT (rule)

Indicates courses that should not fill a requirement or be considered as part of a qualifier.

```
MinClasses 10 in CHEM 1@
  Except CHEM 120,121

MaxCredits 0 in ABC @
  Except ABC 2@ (With Attribute=SPCL)

6 Credits in ART 113:129
  Except ART 120,121
   Label "Art Studies";
```

Notes:

1.	**Except** is allowed in a course rule and any header qualifier that allows a course list. **Except** is preceded by a course list.
2.	**Except** is followed by a list of courses that will not fill the requirement but will match the preceding course list.
3.	**Except** is appropriate only when the preceding course list contains wildcards or ranges of courses.
4.	Use a comma or **or** to separate courses in the course list following **Except**. The plus sign or **and** is not allowed. All courses following **Except** will not be applied to the rule or qualifier if taken by the student.

EXCLUSIVE (rule)

Same as **DontShare**.

FROM (header)

Precedes a list of courses, disciplines, or transfer codes.

```
BEGIN

MaxClasses 3 from MUS@
;
```

Notes:

1.	**From** is optional. It is used only for readability.
2.	**From** is equivalent to **in**. For example, "3 **Credits in** ART@" is equivalent to "3 **Credits from** ART@".
3.	**From** follows one of these keywords: MaxClass[es], MaxCredit[s], MaxPerDisc, MaxTerm, MaxTransfer, MinClass[es], MinCredit[s], MinPerDisc, MinTerm.

FROM (rule)

Precedes a list of courses, groups, disciplines, or transfer codes.

```
3 Classes from INS@
  Label "International Studies";
```

Notes:

1.	**From** is optional. It is used only for readability.
2.	**From** is equivalent to **in**. For example, "3 **Credits in** ART@" is equivalent to "3 **Credits from** ART@".
3.	**From** follows one of these keywords: Class[es], Credit[s], Group[s], MaxPerDisc, MaxTerm, MinPerDisc.
4.	**From** can be repeated in a rule statement but only if each **from** follows a distinct Class[es], Credit[s], Group[s], MaxPerDisc, MaxTransfer, or MinPerDisc.

GROUP[S] (rule)

Precedes a list of requirement choices, of which a specified number of rules must be satisfied.

```
1 Group in
 (1 Class in ENGL 101
   Label "Remedial English") OR
 (1 NonCourse (ENGTEST)
   Label "English Entrance Exam")
Label "ENTRY-LEVEL ENGLISH";

1 Group in
 (6 Credits in FRE @, GER @, SPA @
   Label "Group A") OR
 (1 Group in
   (1 Class in INS 142
     Label "INS Course") OR
   (4 Credits in CCS @
     Label "CCS Course")
   Label "Group B")
Label "TRANSCULTURAL STUDIES";
```

Notes:

1.	**Groups** must be followed by a list of one or more rules. The list of rules following the **Group**[s] keyword is referred to as the group_list. Each rule in the group_list is a group_item. Each group_item is enclosed in parentheses and does not end with a semicolon. a) Each rule in the **Group** list is one of the following types of rules: **Class**, **Block**, **BlockType**, **Group**, or **NonCourse**. A group_item cannot be an **If** rule or a subset rule. b) Each rule in the group_list is enclosed in parentheses. c) Each rule in the group_list is connected to the next rule by **or** or a comma. d) Each rule in the group_list can include qualifiers (like **MinGrade**) that apply only to that rule.
2.	**Group** [s] is used when the requirement may be filled in any of a number of different ways.
3.	**Group** [s] must follow an integer that indicates the number of rules in the group_list that must be satisfied. This integer cannot be more than the total number of rules in the group_list.
4.	**Groups**[s] may optionally be followed by **in** or **from**.
5.	A **Group** statement can be nested within another **Group** statement.
6.	Qualifiers that must be applied to all rules in the group_list must occur after the last ")" and before the semicolon at the end of the **Group** statement. Qualifiers that apply only to a specific rule in the group_list must appear inside the parentheses for that group _item rule.
7.	A semicolon is not used to end the **Group** rule when a **Group**[s] is nested within another

	Group. When used within another **Group** rule, the nested **Group** rule is enclosed in parentheses.
8.	Allowable rule qualifiers: DontShare, Exclusive, Hide, Hide-Rule, High-Priority, Low-Priority, Lowest-Priority, MaxPassFail, MaxPerDisc, MaxSpread, MaxTransfer, MinGrade, MinPerDisc, MinSpread, NonExclusive, NotGPA, Proxy-Advice, SameDisc, Share, ShareWith. MinClass, MinCredit.
9.	**Do not** attempt to mix course rules with Block rules in a group. **Do not** do this: 1 Group in (3 Credits in @ (With Attribute=ABCD) Label ABCD “Requirement”) or (1 Block (Other=ABCD) Label ABCDBLK “Block”) Label “Label”; Although this will parse, the auditor will not handle this well. If the ABCD course can be used on some other rules in the audit there is a good change it will be used there instead of here and the auditor will not “go back” and pull in the OTHER=ABCD block it thought it did not need.

HEADERTAG

A special qualifier that you can place in the block header to give it special meaning when the audit worksheet is being displayed. Any HeaderTag name-value pair that you add to your header will be available for use within the xsl stylesheets, to show the block in a special way—hide it, use a different color, etc. Both the name and the value following **HeaderTag** are limited to 50 bytes each. The value must be enclosed in double-quotes if it contains non-alphanumeric characters.

```
Begin
    30 Credits
    HeaderTag Critical=Yes
;

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

Notes:
1. Allowable within the header only.
2. HeaderTag names of RemarkJump and RemarkHint have special meaning and are used by the standard Degree Works worksheets. See the *AdviceJump*, *RemarkJump*, and *RuleTag* section for more information.

HIDE (rule)

Allows certain courses to satisfy a requirement while hiding this fact from the audit advice. You may also use **Hide-From-Advice** if you prefer. Either format works.

In this example, ECON 112 will never show as advice on the audit report but can be used to help satisfy the requirement. This example represents an institution that would allow ECON 112 to be used if the student takes it, however, because of departmental preferences, the institution does not want to "advise" students to take the ECON course over the ACCT courses.

```
3 Classes in ACCT 103, 105, {HIDE ECON 112}
  Label "Accounting Requirements;
```

The **HIDE** and **WITH** operators can be used in conjunction with each other as an alternative to the course equivalence table. For example, if MATH 104 was changed to MATH 184 in the term 1981, then the course rule would be scribed as follows:

```
5 Credits in MATH 184, {HIDE MATH 104 (WITH TERM<"1981")}
  Label "Math Requirement";
```

In this example, the rule will be satisfied by either MATH 184 taken in any term or by MATH 104 taken before the 1981 term. Furthermore, the course advice for this rule is "5 Credits in Math 184" (MATH 104 is not mentioned in the audit advice).
You may also use **HIDE** within the WITH qualifier:

```
5 Credits in HIST 184 (WITH Hide DWResident = Y)
  Label "History Requirement";
```

This can be especially useful if you have a long list of classes and have placed WITH on each. The advice may be hard to read with each of the WITH information appearing and using HIDE can help with readability.

Notes:

1.	The **HIDE** keyword is only allowed on courses specified in a class rule.
2.	When you use {**HIDE**} inside an **AREA** type rule with [ ], you need to place the **HIDE** as the first course in the rule, inside the starting square bracket "[{. An example would be as follows: `[{Hide PSY 154}, ANT 222, HIST 101]` The curly brace "}" cannot be followed immediately by the square bracket "]".
3.	The **HIDE** keyword is not allowed with a course rule specified with a plus(+) list of classes. It makes no sense to require all classes but to hide some of them from the audit advice.
4.	One or more classes can be listed within the **HIDE** braces, and multiple occurrences of **HIDE** are allowed within the same course rule.
5.	The trick to knowing where the place the commas when using **HIDE** is to be sure that everything within and including the braces can be removed leaving a valid, parseable course rule.

HIDE-RULE (rule)

Hides a rule and the rule advice on audit reports. Typically, this rule qualifier is used to hide one or more options within a **Group** rule or stand-alone **Block** and **BlockType** rules in the **DEGREE** or starting block, although the qualifier can be used on any rule type.

```
BEGIN
60 Credits
;
1 Block (OTHER=GE)
   Hide-Rule
  Label "General Education Requirements";
1 BlockType (MAJOR)
   Hide-Rule
LABEL "Major Requirements";

END.
```

In the examples shown above, the **Hide-Rule** qualifier will hide the label and advice for the **Block** and **BlockType** rules. These labels will not appear on audit reports. Since the label and rule advice for **Block** and **BlockType** rules is not very useful to students, these are good choices for the use of the **Hide-Rule** qualifier.

The **Hide-Rule** qualifier can also be used to hide one or more options within a **Group** rule from the advice. This is useful when you want to allow an option to be completed by a student without having to process an exception but not give the student advice that the option is available.

```
1 Group in
    (2 Classes in MATH 101, 103, 105
       Label "Math Option I - College Algebra") OR
    (2 Classes in MATH 121, 122, 123, 126
       Label "Math Option II - Calculus Sequence") OR
    (2 Classes in BUSI 201, 221, 225
          Hide-Rule
       LABEL "Math Option III - Business Math")
 LABEL "MATH REQUIREMENT";
```

In the example above, the third option (Business Math) will be hidden from the advice. If the student takes one of the Business Math courses listed, the advice for that rule will then appear on the audit and the class taken will be applied to the rule. The advice for the remaining courses left to be taken to complete the rule will also be shown. The advice for the other rules will disappear if a student selects the Business Math option.

Notes:

1.	The **Hide-Rule** keyword can be used on any rule.
2.	Once a class is taken towards a rule that is hidden, the rule will appear on the audit report and the class(es) taken towards the rule will appear in the rule advice.
3.	It is recommended that you NOT use the **Hide-Rule** qualifier on rules where rule advice is necessary on audit reports. Doing so will suppress the advice for the rule until a student takes a class that applies towards the rule.

HIGH-PRIORITY (rule and block)

Tell the Auditor that a certain rule or block should be satisfied before others. High-**Priority** is used mostly when classes fit multiple rules and you want to specify a preference.
In this example, ACCT 103 and 105 will be marked preferred to satisfy this rule over the other rules against which they might be applied:

```
5 Credits in ACCT 103, 105
   High-Priority
  Label "Accounting";
```

In this example, ACCT 101-110 classes will be marked as being preferred to satisfy this rule over the other rules against which they might be applied. Because **High-Priority** appears twice on the rule, this rule has preference over a rule with only one **High-Priority** qualifier:

```
5 Classes in ACCT 101:110
   High-Priority
   High-Priority
  Label "Accounting";
```

In this example, the rules in the block will be tagged as being the preferred location to place the classes that apply. Placing **High-Priority** on a Major block is a good example of how this option might be used:

```
BEGIN
  30 Credits
  High-Priority
;
```

Notes:

1.	**High-Priority** can be repeated on a block or rule to give a higher preference over those rules or blocks with only one **High-Priority** qualifier listed.
2.	**High-Priority** can be placed on a group or subset or directly on a course rule.
3.	Each **High-Priority** increases the match level of a class on a rule by 5 points.
4.	**High-Priority** can be used at the block level and simultaneously on any rules within the block.
5.	**High-Priority** may be combined with **Low-Priority** as needed; each cancels out the other, however.
6.	The hyphen in **High-Priority** is optional.

Special Note: Placing **High-Priority** after the last course listed on a rule applies the qualifier to all courses listed on a rule:

```
10 Credits in HIST 102, 106, 112
    High-Priority
Label "History requirement";
```

Placing **High-Priority** before a course, however, applies the qualifier to just that course:

```
10 Credits in HIST 102, 106, High-Priority 112
Label "History requirement";

10 Credits in High-Priority HIST 102, 106, 112
Label "History requirement";

10 Credits in HIST 102, {HIDE High-Priority 106,} 112
Label "History requirement";

10 Credits in BIOL @, PHYS @, High-Priority CHEM @
Label "Science requirement";
```

IF (header)

Begins a conditional header qualifier. **If** statements may be used in the block header, and any header qualifier may be used within the **If** statement.

BeginIf and **EndIf** are not required if you have one qualifier. However, it is recommended that **Begin**-**End** be used for consistency of scribing. For example:

```
BEGIN
LastRes 30 Credits

If (MAJOR=HIST) THEN
    BeginIf
      MaxCredits 15 in CHEM @
    EndIf

Else
    BeginElse
      MaxCredits 10 in CHEM @
    MinGPA 2.5
EndElse
;
```

Begin-**End** is required if you have more than one qualifier:

```
BEGIN
LastRes 30 Credits
If (MAJOR=HIST) Then
  BeginIf
      MaxCredits 15 in MATH @
      MaxCredits 15 in CHEM @
  EndIf
Else
  BeginElse
      MaxCredits 10 in MATH @
      MaxCredits 10 in CHEM @
  EndElse
MinGPA 2.5
;
```

You can use **Else If** as needed:

```
If (MAJOR=HIST) Then
  MaxCredits 15 in MATH @
Else If (MAJOR=ACCT) Then
  MaxCredits 10 in MATH @
Else If (MAJOR=ENGL) Then
  MaxCredits 12 in MATH @
Else
  MaxCredits 20 in MATH @
```

You may use a **wildcard** as needed – but you must use quotes in the value when doing so:

```
If (MAJOR = "BIO@") Then
  MaxCredits 45 in BIO @
Else
  MaxCredits 10 in BIO @
```

IF (rule)

Begins a conditional rule.

```
# You can use Else If and Else as needed
If (1stMajor = BIO) THEN
 15 Credits in BIO 3@
    Label "Biology elective for Biology majors"
Else If (1stMajor = CHEM) THEN
 15 Credits in CHEM 3@
    Label "Biology elective for Chemistry majors"
Else
 10 Credits in BIO 3@
    Label "Biology elective for non-Biology majors";

# You can have an IF without an Else if you like
If (CONC=AHST OR CONC=EHST) Then
  2 Classes in POL 100:199
    Label "Political Science";

# You can use a subset to list multiple rules
If (Attribute = HONR) Then
  BeginSub
  5 Classes in HIST 100:199
    Label "History 100 level classes"
  2 Classes in HIST 200:299
    Label "History 200 level classes"
  EndSub
    Label "History for honors students";

# You can also use BeginIf/EndIf to list multiple rules; this may be
preferred over
# using a subset if you don't want that extra subset label
# You can also use BeginElse/EndElse as needed
# Using BeginIf/EndIf or BeginElse/EndElse you can add remarks to particular
rules.
If (Attribute = HONR) Then
  BeginIf
  5 Classes in HIST 100:199
    Label HIST100A "Honors: History 100 level classes";
  2 Classes in HIST 200:299
    Label HIST200A "Honors: History 200 level classes";
  Remark "As an honors student you need to take 5 classes"
  Remark " at the 100 level and 2 at the 200 level"
  EndIf
else
  BeginElse
  3 Classes in HIST 100:199
    Label HIST100B "History 100 level classes";
  1 Classes in HIST 200:299
    Label HIST200B "History 200 level class";
  EndElse

# You can scribe against the studen't GPA from the student system
If (SSGPA > 2.0) Then  # Student System GPA; Banner sites may prefer to use
BannerGPA
  RuleComplete
    Label "2.0 GPA requirement met"
Else #insufficient GPA
```

```
  RuleIncomplete
    ProxyAdvice "Your GPA is below the 2.0 required"
    Label "2.0 GPA requirement not met";

# You can check for the number of majors or minors
If (NumberOfMajors >= 2 or NumberOfMinors >= 1) THEN
  RuleComplete
    Label "Gened elective waived because of double-major or additional
minor"
Else
 10 Credits in ENGL @, WRIT @, HUM  @
    Label "Humanity elective";


# You may use a wildcard but you must use quotes around the value
If (MAJOR = "BIO@") Then

  BeginIf

  18 Credits 45 in BIO 2@

    Label BIO "Biology 200-level"
  EndIf
```

You can also use **If** statements to tell if a class was taken, passed, failed, etc. and control what rules and labels appear.

The complete list of options, shown as examples is as follows:

```
If (MATH 101 (WITH DWG > "B") was TAKEN or MATH 321 was PASSED) Then
If (MATH 101 was FOUND) Then # historic or in-progress; passed or failed
If (MATH 101 was TAKEN) Then # historic (or not in-progress); passed or
failed
If (MATH 101:104 was Passed) Then # at least one class in range 101 to 104 -
passed grade
If (MATH 1@ was Failed) Then # any 100 level math class that the student
failed
If (MATH 101 isnt INPROGRESS) Then
If (MATH 101 wasnt TRANSFERRED) Then
If (MATH 101 was PASSED and MATHTEST > 61) Then
```

The keywords "**is**" and "**was**" are interchangeable as are "**isnt**" and "**wasnt**".

Notes:

1.	**If** is followed by a conditional expression, which is followed by **Then**. A conditional expression is everything between **If** and **Then**, and it is enclosed in parentheses. A conditional expression consists of one or more data conditions.
2.	A data condition tests student data for a certain value. Each data condition is of the form student_data ROP value, where: Student_data is one of the words from the custom_data list (UCX-SCR002) or one of the following standard data fields: COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, PROGRAM, SCHOOL, SPEC, OTHER. ROP is a relational operator: = equal <> not equal > greater than < less than >= greater than or equal to <= less than or equal to Only comparative conditions are allowed. Arithmetic is not allowed, i.e. addition, subtraction, multiplication, and division are not supported. Value is a data value, usually a code from the institution's code list.

3.	You may check to see if the student has met your student system's GPA requirement using an **If** statement. As with MAJOR, MINOR, COLLEGE, etc., there exists a standard keyword that examines the GPA stored in Degree Works that came from the students' records in your system. SSGPA or SISGPA (student system GPA) is the keyword that may be used to perform this check (Banner sites may use SSGPA, or SISGPA, or BannerGPA—they behave identically). Again, like MAJOR, MINOR, etc., SSGPA and BannerGPA do not need to be set up in UCX-SCR002. There is a new Scribe template called If-GPA in the Additional Templates section for your convenience.
	
	The following is an example of how to use SSGPA in an **If** statement:
	
	`If (SSGPA >= 2.0) then  # Banner sites may use BannerGPA`
	`Rule-complete`
	`Label "You meet the minimum GPA requirement."`
	`Else`
	`Rule-incomplete`
	`Label "Your GPA is below 2.0; please see an advisor.";`

4.	If multiple data conditions are used, then they must be joined by **and** or **or**. Plus and comma are not allowed to connect conditions.
5.	The order of precedence for evaluating the **If** expression is: (…), **and**, **or**. Expressions in parentheses are evaluated first. Then expressions connected by **and** are evaluated. Finally, expressions connected by **or** are evaluated. Evaluation is performed from left to right. For example, "**If** (**MAJOR**=HIST **or MAJOR**=POL **and MINOR**=EDU)" is true if a student is either a History major OR a Political Science major with an Education minor. The precedence changes if parentheses are used as follows: "**If** ((M**A**JOR=HIST **or MAJOR**=POL) **and MINOR**=EDU)". This expression is true if a student has an Education minor AND either a History or Political Science major.
6.	The rule following **Then** is used when the condition following **If** is true. Otherwise the rule following **Else** is used.
7.	The rule following **Then** or **Else** can be a single rule or multiple rules bracketed by **BeginSub** and **EndSub**.
8.	**If** may optionally be followed by **Else**. If **Else** does not exist and the **If** condition is false, then no rule is used.
9.	**If** may be repeated (nested) in a rule. For each **If**, there must be a corresponding **Then**.

10.	See also **Else, Then**.
11.	Rule qualifiers cannot be used on the **If** rule but can be used on other rules within an **If** rule
12.	Allowable rule qualifiers: none on the **If** statement, but qualifiers are allowed on rules within an **If** statement.
13.	When using relation operator <> in a list of several data conditions, you must connect conditions with **and**, otherwise the conditions will not be met.

14.	You may want to change your rules or qualifiers based on the type of audit that is being run or the action that is being taken.
	
	`If (AuditAction <> "WHATIF") then`
	`  MaxClasses 0 in @ (With Attribute = "XYZ")`
	
	Valid values for **AuditAction** are *NORMAL, WHATIF, LOOKAHEAD, EXCEPTIONS, PLANNER, TRANSFER*
	Valid values for **AuditType** are *ACADEMIC, FINANCIALAID, ATHLETIC, REQUISITE*

15.	If you are using an IF-statement with a course specified you cannot use the MinGrade or MinGPA qualifiers in the header within such an IF-statement. The reason for this is that the auditor needs to know whether there is a MinGrade/MinGPA qualifier in the header before it applies the classes – and the IF-statement is only evaluated after all classes have been applied. Although the parser allows you to do this the auditor will not give you expected results.
16.	**Semicolons** – where do you put them? Historically semicolons were required on all rules and scribing IF-statements was tricky. They are now optional on all rules and remarks. If you scribe a semicolon – that is fine. If you don't scribe a semicolon – that is fine too. Bottom line – don't worry about them. (Though a semicolon is still required to end the header qualifiers.)
17.	Using **BeginIf**/**EndIf** and **BeginElse**/**EndElse** allows you to scribe multiple rules and allows you to add remarks specific to the IF or the ELSE part.
18.	**BeginIf** and **BeginElse** are interchangeable. You can use **BeginIf** in the Else and you can use **BeginElse** in the IF part – the parser is not that picky. **EndIf** and **EndElse** are also interchangeable. When you have an **Else If** you can use either the BeginIf/EndIf or the BeginElse/EndElse – your choice.

19.	When **OTHER** is specified in the IF expression, the auditor checks to see if that specific OTHER block is in the list of blocks that was sent to it. This is best used for particular OTHER blocks like those created by the planner – as this example shows:
	If (**OTHER = PLAN**) then
	  1 block (Other=PLAN)
	  Label "Planned requirements"
	Else
	  1 block (Other =GENED)
	  Label "Gened requireiments"
	You only want to require the PLAN block if one exists for this student.

IN (rule)

Precedes a list of courses, groups, disciplines, or transfer codes.

```
3 Classes in INS@
  Label "International Studies";

1 Group in
  (1 Class in MATH 115,116
    LABEL "Group A") OR
  (1 Class in MATH 125, MATH 126
     LABEL "Group B")
LABEL "FRESHMAN MATH";
```

Notes:

1.	**In** is optional. It is used only for readability.
2.	**In** is equivalent to **from**. For example, "3 **Credits in** ART@" is equivalent to "3 **Credits from** ART@".
3.	**In** follows one of these keywords: Class[es], Credit[s], Group[s], MaxPerDisc, MaxTransfer, MinPerDisc.
4.	**In** can be repeated in a rule statement but only if each **in** follows a distinct Class[es], Credit[S], Group[s], MaxPerDisc, MaxTransfer, or MinPerDisc.

INCLUDE-BLOCKS-WITH (rule)

Specifies the blocks that should be included in this block's header qualifier calculations.

```
Include-Blocks-With (CONC="XY@")
  Label "Included Requirements";

Include-Blocks-With (COLLEGE=ART and SCHOOL=UG)
  Label "Included Requirements";
```

Notes:

1.	Use of wildcards within the code requires quotes.
2.	**And** (or the plus) must separate multiple type-code combinations.
3.	The type can be in upper, lower, or mixed case.
4.	The code is up-shifted only if the DAP14 flag to up-shift all codes is turned on.
5.	The rule is as complete as the average of the blocks that are included based on the type and code specified.
6.	If no blocks of the specified type are found, the rule becomes 100 percent complete.
7.	No rule qualifiers are allowed.

INCLUDING (rule)

Indicates mandated courses from course list.

```
6 Credits in ART 113:129
    Including ART 120
  Label "Art Studies";

3 Classes in MATH@, PHIL@
    Including PHIL 110, MATH 115
  Label "Logic";
```

Notes:

1.	**Including** is allowed only within a course rule. It is preceded by a course list.
2.	**Including** is followed by a list of courses that must be taken to fulfill the requirement. The list that follows **Including** must be a subset of the list that precedes **Including**.
3.	**Including** is appropriate when the preceding course list contains wildcards or ranges of courses.
4.	Use a comma or **or** to separate courses in the course list that follow **Including**. The plus sign or **and** is also allowed. The Parser does not care how the courses are separated following the **Including**. Because the courses are part of an include list, they will always be audited as though the **and** operator was used. Therefore, using **and** (or the plus) has the same effect as using **or** (or a comma) in that all courses on the list are required.

LABEL (header)

A free-text comment used to identify the header qualifier.

```
MinCredits 6 in FRE@, GER@, SPA@
  Label FORLANG "Foreign Language";

MinGPA 2.5 in ENGL @, LIT @
    ProxyAdvice "You need a GPA 2.5 but currently have a <APPLIED> GPA";
    Label engllitgpa "Literature/English GPA 2.5 requirement";

If (MINOR=EDU) Then
  MinClasses 3 in TEACH @
    Label STUTEACH "Student Teaching"
Else
  MinClasses 2 in TEACH @
    Label APPREC "Student Teaching";

You can place the label text on a line by itself if it is very long:
  MinRes 30 Credits
    Label minres
  "This is a very long label which is exactly 78 chars long; on a line by
  itself"
  ;
```

Label with Tag – See Special Topics and the Notes section below for more information about Label Tags

Notes:

1.	**Label** is followed by a free-text comment enclosed in quotes. The text can be 0 to 78 characters long but cannot include quotation marks. The text can be empty, i.e. **Label** " ". Seven-eight was chosen as the maximum length of a **Label** since each line is a maximum of 80 characters and we need to allow for the two double-quotes. You are allowed to put the Label keyword and label-tag on one line and the quotes and text on a separate line – this is how you can get a full 78 character label.
2.	**Label** text appears on worksheets along with an appropriate checkbox graphic. The label is optional for header qualifiers – but if found, it appears similar to a rule.
3.	Labels are only allowed on these header qualifiers: Classes, Credits, MinClasses, MinCredits, LastRest, MinRes, MinGPA, MinPerDisc, MinTerm, Under. That is, any qualifier specifying a minimum limit.
5.	You may use a tag on the label to uniquely identify your labels and qualifiers to which they are associated. The tag uniquely identifies the qualifier, which allows users to change the label text as much as needed and not cause exceptions to become unhooked. There is a maximum of 20 characters for the tag, and it is not case sensitive. If a label exists on a qualifier along with a label-tag you do not have to add a qualifier tag to the rule to help ensure that the qualifier exceptions remain in place; the label-tag will suffice.

LABEL (rule)

A free-text comment used to identify the requirement.

```
6 Credits in FRE@, GER@, SPA@
  Label FORLANG "Foreign Language";

1 Group In
 (6 Credits in FRE @, SPA @, GER @
    Label GRPA "Group A: Modern Languages") OR
 (6:9 Credits in LAT @, GRK @
    Label GRPB "Group B: Ancient Languages")
Label "LANGUAGE REQUIREMENTS";

If (MINOR=EDU) Then
  1 Block (OTHER=TEACH)
    Label STUTEACH "Student Teaching"
Else
  1 Class in EDU 100:120
    Label APPREC "Teacher Appreciation";
```

You can place the label text on a line by itself if it is very long:

```
3 Classes in ART 2@
  Label ART2
"This is a very long label which is exactly 78 chars long; on a line by
itself"
;
```

Label with Tag – See Special Topics and the Notes section below for more information about Label Tags

```
5 Credits in ENGL 23@, LIT @
    Label literature "Literature requirement";
```

Notes:

1.	**Label** is followed by a free-text comment enclosed in quotes. The text can be 0 to 78 characters long but cannot include quotation marks. The text can be empty, i.e. **Label** " ". 78 was chosen as the maximum length of a **Label** since each line is a maximum of 80 characters and we need to allow for the two double-quotes. You are allowed to put the Label keyword and label-tag on one line and the quotes and text on a separate line with the semicolon on the next line if needed – this is how you can get a full 78 character label.
2.	**Label** free-text is used on Degree Works reports that do not have room for the full remarks. The label should be a brief description for the advisor of the requirement.
3.	**Label** must follow the rule qualifiers and precede a right parenthesis ending a rule in a **Group** or precede **Else** in an **If** rule or precede a semicolon ending a rule. One **Label** is allowed per rule statement. Each rule in a **Group** statement can have a label, as can each rule in a subset.
4.	**Label** is optional or required based on UCX-CFG020 DAP13 Require Labels flag. If the label is set to **N**, then the audit output will not display a description of the requirement. Labels in a block should be unique to help with exception processing in audits.
5.	You can use a tag on the label to uniquely identify your labels. The tag uniquely identifies the rule, which allows users to change the label text as much as needed and not cause exceptions to become unhooked There is a maximum of 20 characters for the tag, and it is not case sensitive.

LASTRES (header)

Credits or classes that must be taken in residence as the last credits/classes.

```
LastRes 12 Credits  # the last 12 credits must be taken in residence

LastRes 4 Classes # the last 4 classes must be taken in residence

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

LastRes 15 of 30 Credits in @ (With DWResident=Y or Attribute=SE)
  Except PE @
             # SE courses are also considered in residence
```

Notes:

1.	**LastRes** is followed by a real number and the **Credits** keyword, or an integer and the C**lasses** keyword. The number indicates how many credits or classes preceding the conferring of the degree must be earned in residence.
2.	**LastRes** can only be used if all of a student's courses and transfer credits have been linked to a term.
3.	**LastRes** can also be followed by a real number and the **Credits** keyword, or an integer and the **Classes** keyword, followed by the connector **of**, followed by a real number and the **Credits** keyword greater in value than the first number, or an integer and the **Classes** keyword greater in value than the first integer. The first number indicates how many credits or classes out of the second number of last credits or classes taken by the student must be earned in residence.
4.	Only one **LastRes** qualifier is allowed in the block header.
5.	LastRes allows a list of courses to specify what courses should be considered "in residence". Without a course list those courses that were not transferred in are considered in residence. When a course list is present only those courses that match the course list will be considered in residence. In doing this, special transfer classes with certain attributes are lumped into the residency bucket and count towards the LastRes qualifier.
6.	When LastRes is on a block other than the starting block only those courses in the block's scope are counted towards the LastRes credits or classes. For example, if LastRes is on the Major block only the transfer and resident courses in the Major block (and any blocks it references) are examined – those in the Gen Ed block or fall-through, for example, are ignored.

LIBL (header)

Indicates a liberal learning-type block (LIBL) against which **NonExclusive** is applied.

```
BEGIN
36 Credits
ShareWith 9 Credits (LIBL);

BEGIN
36 Credits
ShareWith (LIBL, PROGRAM);
```

Notes:

1.	LIBL is used within a block header, only with **Share**, **ShareWith**, or **NonExclusive**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of the block can also be used to satisfy requirements in the LIBL block (if the LIBL block exists for the student).

LIBL (rule)

Indicates a liberal learning requirement block, which, in some cases, may precede a liberal learning code.

```
1 Block (LIBL=LIFE)
  Label "Liberal Learning";

1 BlockType (LIBL)
  Label "Liberal Learning required";
```

Notes:

1.	LIBL is followed by a single liberal learning code, such as LIFE for Life Studies, except in a **BlockType** rule.
2.	The liberal learning code must be valid in the institution's code list.
3.	LIBL can be used as part of a condition following **If**.
4.	LIBL can be used with Block[s], BlockTypes[s], If, NonExclusive, Share, or ShareWith.

LOW-PRIORITY (rule and block)

Tells the Auditor that a certain rule or block should be considered less important than others. **Low-Priority** is used mostly when classes fit multiple rules and you want to specify a preference.

In this example, ACCT 103 and 105 will be marked as having a lower preference in this rule than in other rules against which they might be applied:

```
5 Credits in ACCT 103, 105
Low-Priority
Label "Accounting";
```

In this example, ACCT 101-110 classes will be marked as having a lower preference in this rule than in other rules against which they might be applied. Because **Low-Priority** appears twice on the rule, this rule has a lower preference than a rule with only one **Low-Priority** qualifier:

```
5 Classes in ACCT 101:110
    Low-Priority
    Low-Priority
  Label "Accounting Courses";
```

In this example, the rules in this block will be tagged as being a less preferred location to place the classes that apply. Placing **Low-Priority** on a Minor block is a good example of how this option might be used:

```
BEGIN
30 Credits
Low-Priority
;
```

Notes:

1.	**Low-Priority** can be repeated on a block or rule to give a lower preference than is given to the rules or blocks with only one **Low-Priority** qualifier listed.
2.	**Low-Priority** can be placed on a group or subset or directly on a course rule.
3.	Each **Low-Priority** decreases the match level of a class on a rule by 5 points.
4.	**Low-Priority** can be used at the block level and simultaneously on any rules within the block.
5.	**Low-Priority** may be combined with **High-Priority** as needed; each cancels out the other, however.
6.	The hyphen in **Low-Priority** is optional.

Special Note: Placing **Low-Priority** after the last course listed on a rule applies the qualifier to all courses listed on a rule:

```
10 Credits in HIST 102, 106, 112
 Low-Priority
  Label "History requirement";
```

Placing **Low-Priority** before a course applies the qualifier to just that course:

```
10 Credits in HIST 102, 106, Low-Priority 112
Label "History requirement";

10 Credits in Low-Priority HIST 102, 106, 112
Label "History requirement";

10 Credits in HIST 102, {HIDE Low-Priority 106,} 112
Label "History requirement";
```

```
10 Credits in BIOL @, PHYS @, Low-Priority CHEM @
Label "Science requirement";
```

LOWEST-PRIORITY (rule qualifier)

Tells the Auditor that a certain rule should be considered less important than others. **Lowest-Priority** is used mostly when classes fit multiple rules and you want to specify a preference. **Lowest-Priority** is stronger than Low-Priority because the classes on rules with **Lowest-Priority** are removed early on in the audit allowing the other rules where these classes apply to use these classes. However, during the redemption phase these classes are reapplied to these **Lowest-Priority** rules if the classes are in fall-through or if they can be shared (via NonExclusive/ShareWith) with existing fits for these classes.

In this example, ACCT 103 and 105 will be removed from the Accounting Elective rule early on in the audit process. This should allow them to be used on the Intro to Accounting rule. However, if ACCT 103 was taken for 5 credits ACCT 105 should end up in fall-through allowing redemption to reapply it to the elective rule.

```
5 Credits in ACCT 103, 105
   Label "Intro to Accounting";

5 Credits in ACCT 1@
   Lowest-Priority
   Label "Accounting Elective";
```

It is best to use **Lowest-Priority** on elective rules in the major or in your general education blocks. Lowest-Priority may also be useful on certain group rules. The rules on which this qualifier is placed are basically the rules that should be considered last by the auditor. Use it wisely.

Notes:

1.	**Lowest-Priority** can be placed or rule to give a lower preference than is given to other rules.
2.	**Lowest-Priority** can be placed on a group or subset or directly on a course rule.
3.	Each **Lowest-Priority** sets the match level to -9876 – a very low value.
4.	**Lowest-Priority** cannot be used at the block level.
5.	The hyphen in **Lowest-Priority** is optional.
6.	Unlike with Low-Priority, Lowest-Priority cannot be placed on a specific course on a rule.

MAJOR (header)

Indicates a major block type (MAJOR) against which **NonExclusive** is applied.

```
BEGIN
36 Credits
ShareWith 9 Credits (MAJOR);
```

Notes:

1.	**MAJOR** is used within a block header, only with **NonExclusive**, **Share**, or **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of the block can also be used to satisfy requirements in the MAJOR block (if the MAJOR block exists for the student).

MAJOR (rule)

Indicates a major requirement block, which, in some cases, may precede a major code.

```
1 Block (MAJOR=CHE)
  Label "Chemistry Major requirements";

1 BlockType (MAJOR)
  Label "Major required";

If (MAJOR=BIO) THEN
 1 Class in CHE 300
   Label "CHE 300";

6 Credits in BUS 1@
    ShareWith(MAJOR)
  Label "Business";
```

Notes:

1.	**MAJOR** is followed by a major code, such as CHE for Chemistry, except in a **BlockType** rule.
2.	The major code must be valid in the institution's code list.
3.	**MAJOR** can be used as part of a condition following **If**.
4.	**MAJOR** must be used with Blocks[s], BlockTypes[s], If, ShareWith, or NonExclusive.

MAXCLASS[ES] (header)

Indicates the maximum number of courses that can be applied to a requirement.

```
MaxClasses 8 in MUS 1@

MaxClasses 3 from PSY 100:199
   Except PSY 108, 117

##maximum of 8 courses in MUS @ or 8 courses in ART @, or both summed
MaxClasses 8 in MUS @, ART @

##maximum of 8 courses in MUS @ and 8 courses in ART @
MaxClasses 8 in MUS @
MaxClasses 8 in ART @

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

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

Notes:

1.	**MaxClass[es]** is followed by an integer that indicates the maximum number of courses that can be applied to the rule. The integer is optionally followed by **in** or from, which must be followed by a course list.
2.	The connector in the course list following **MaxClass**[es] can be comma or **or**. The plus sign or **and** is not allowed. The logical **or** connector is not exclusive. If multiple courses are listed, then the sum of all courses that satisfy the list is compared against the integer following **MaxClass**[es].
3.	**MaxClass**[es] applies to all courses in the succeeding course list.
4.	The **MaxClass**[es] qualifier is applied by the Auditor Engine against all the courses used to fill the block. The number of courses in **MaxClass**[es] is a cap that cannot be exceeded.
5.	The course list associated with **MaxClass**[es] allows **Except** but not **Including**.

MAXCREDIT[S] (header)

Indicates the maximum number of credits that can be applied to a requirement.

```
MaxCredits 24 in MUS 1@
   Except MUS 109, 112

##maximum of 8 credits in MUS @ or 8 credits in ART @, or both summed
MaxCredits 8 in MUS @, ART @

##maximum of 8 credits in MUS @ and 8 credits in ART @
MaxCredits 8 in MUS @
MaxCredits 8 in ART @

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

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

Notes:

1.	**MaxCredit**[s] is followed by a real number that indicates the maximum number of credits that can be applied to the rule. The number is optionally followed by **in** or from, which must be followed by a course list.
2.	The connector in the course list following **MaxCredit**[s] can be a comma or **or**. The plus sign or **and** is not allowed. The logical **or** connector is not exclusive. If multiple courses are listed, then the sum of all credits that satisfy the list is compared against the number following **MaxCredit**[s]
3.	**MaxCredit**[s] applies to all courses in the succeeding course list.
4.	The **MaxCredit**[s] qualifier is applied by the Auditor Engine against all the courses used to fill the block. The number of credits in **MaxCredit**[s] is a strict upper bound. For example, "**MaxCredit**[s] 2" cannot be exceeded even if the only course taken was for 3 credits.
5.	The course list associated with **MaxCredit**[s] allows **Except** but not **Including**.

MAXPASSFAIL (header)

Indicates the maximum number of credits/classes that can be taken pass-fail.

```
BEGIN
36 Credits
MaxPassFail 12 Credits
;

BEGIN
12 Classes
MaxPassFail 4 Classes
;
```

Notes:

1.	**MaxPassFail** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The number indicates the maximum number of courses or credits that can be taken with pass-fail grading. This number is a strict cap.
2.	**MaxPassFail** applies to all courses used to fulfill the block requirements.

MAXPASSFAIL (rule)

Indicates the maximum number of credits/classes that can be taken pass-fail.

```
6 Credits in BIO @, CHE @
MaxPassFail 3 Credits
Label "Science";

3 Classes in ENG 300:499
MaxPassFail 0 Classes
Label "English";
```

Notes:

1.	**MaxPassFail** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The number indicates the maximum number of courses or credits that can be taken with pass-fail grading.
2.	**MaxPassFail** applies to all courses in the preceding course list.

MAXPERDISC (header)

Indicates the maximum number of credits/classes in each discipline listed that can be applied to a requirement.

```
BEGIN
36 Credits

##maximum of 8 credits per discipline in BIO or CHE or PHYS, or all three
summed
MaxPerDisc 8 Credits (BIO, CHE, PHYS)
;

BEGIN
12 Classes
MaxPerDisc 4 Classes (SOC)
;

BEGIN

##maximum of 8 credits in BIO and 8 credits in CHE

MaxPerDisc 8 Credits (BIO)
MaxPerDisc 8 Credits (CHE)
;
```

Notes:

1.	**MaxPerDisc** indicates the maximum number of credits or classes per discipline represented in the discipline list that will be applied to a requirement. This number is a strict cap.
2.	**MaxPerDisc** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The integer or number indicates the maximum number of courses or credits that must be taken in each discipline, followed optionally by **in** or **from** followed by left parenthesis, followed by a discipline code, optionally followed by additional discipline codes separated by a comma or **or**, followed by right parenthesis.
3.	The list of disciplines following **MaxPerDisc** indicates the disciplines that will be evaluated. Each discipline code must be valid in the institution's code list.
4.	If the configuration setting for maximum discipline length (UCX-CFG020 DAP13 Disc_length) is zero, then **MaxPerDisc** is invalid.

MAXPERDISC (rule)

Indicates the maximum number of credits/classes in each discipline listed that will be applied to a requirement.

```
12 Credits in BIO100:299, CHE100:299, PHY100:299
  MaxPerDisc 4 Credits (BIO)
   Label "Science";

##maximum of 8 credits per discipline in BIO or CHE, or both summed
16.0 Credits in BIO@, CHE@
  MaxPerDisc 8.0 Credits in (BIO, CHE)
   Label "Biology and Chemistry Series";

##maximum of 2 classes in ART and 2 classes in MUS
10 Classes in ART@, MUS@, THE@
  MaxPerDisc 2 Classes (ART)
  MaxPerDisc 2 Classes (MUS)
   Label "Arts Requirements";
```

Notes:

1.	**MaxPerDisc** indicates the maximum number of credits or classes per discipline represented in the discipline list that will be applied to a requirement. The disciplines listed must be present in the course list that precedes **MaxPerDisc**.
2.	**MaxPerDisc** follows a course list.
3.	**MaxPerDisc** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The integer or number indicates the maximum number of courses or credits that must be taken in each discipline, followed optionally by **in** or **from**, followed by left parenthesis, followed by a discipline code, optionally followed by additional discipline codes separated by a comma or **or**, followed by right parenthesis.
4.	The list of disciplines following **MaxPerDisc** indicates the disciplines that will be evaluated. Each discipline code must be valid in the institution's code list.
5.	If the configuration setting for maximum discipline length (UCX-CFG020 DAP13 Disc_length) is zero, then **MaxPerDisc** is invalid.

MAXSPREAD (rule)

Indicates the maximum number of disciplines from course list in which courses can be taken.

```
##2 courses spread across no more than 2 of the 3 disciplines listed
4 Classes in BIO @, CHE @, PHY @
    MaxSpread 2
  Label "Science";
```

Notes:

1.	**MaxSpread** is followed by an integer that indicates the maximum number of disciplines from the course list that can be represented in the courses used to satisfy the requirement. If there are 3 disciplines listed and **MaxSpread** is 2, then the courses that satisfy the requirement can be from no more than 2 of the 3 disciplines.
2.	**MaxSpread** applies to all courses in the preceding course list.
3.	**MaxSpread** can be used only within a course rule or subset, not with Block, BlockType, Group, If, or NonCourse.
4.	If the configuration setting for maximum discipline length (UCX-CFG020 DAP13 Disc_length) is zero, then **MaxSpread** is invalid.

MAXTERM (header)

Indicates the maximum number of credits/classes that will be applied to a requirement each term.

```
MaxTerm 1 Class SOC @
   Except SOC 115

##maximum of 6 credits in courses listed, sum of ME100:115 and PE100:115
MaxTerm 6 Credits in ME100:115, PE100:115
```

Notes:

1.	**MaxTerm** is followed by either an integer followed by **Class**[es] or a real number followed by **Credits**[s]. The number indicates the maximum number of courses or credits that can be taken per term. This number is a strict cap.
2.	The syntax for **MaxTerm** varies depending on its scope. If used in the block header, then **MaxTerm** is followed by the number of classes or credits, optionally followed by **in** or **from**, followed by a course list in which comma/**or** is allowed but plus/**and** is not.
3.	**MaxTerm** can only be used if all of a student's courses and transfer credits have been linked to a term.
4.	**MaxTerm** applies to all courses that match the succeeding course list and were used to fulfill any rule or rules in the block.
5.	The course list associated with **MaxTerm** allows **Except** but not **Including**.

MAXTERM (rule)

Indicates the maximum number of credits/classes that will be applied to a requirement each term.

```
##maximum of 3 credits of MUS 299 per term
12 Credits in MUS 299
    MaxTerm 3 Credits
   Label "Music Practice";
```

Notes:

1.	**MaxTerm** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The number indicates the maximum number of classes or credits that can be taken per term.
2.	The syntax for **MaxTerm** varies depending on its scope. If used in a rule statement, then **MaxTerm** must follow a course list and be followed by the number of classes or credits.
3.	**MaxTerm** can only be used if all of a student's courses and transfer credits have been linked to a term.
4.	**MaxTerm** applies to all courses in the preceding course list.
5.	**MaxTerm** can be used only with Course, Group, or Subset rules.

MAXTRANSFER (header)

Indicates the maximum number of transfer credits/classes that will be applied to a requirement.

```
MaxTransfer 36 Credits (AP, CLEP)

MaxTransfer 3 Classes

MaxTransfer 5 Classes (ALEV)
MaxTransfer 36 Credits (AP, CLEP)

# Another way to scribe a transfer limit is to use MaxClasses or MaxCredits
MaxClasses 15 in PE @ (With DWTransfer=Y)
```

Notes:

1.	**MaxTransfer** is followed by either a real number and the **Credit[**s] keyword or an integer and the **Class**[es] keyword, indicating the maximum number of credits or classes that can be transferred from another college or university. This number is a strict cap.
2.	Following the class/credits statement is an optional **in** or **from**, followed by a list of transfer types in parentheses. The list of transfer types is required when **in** or **from** is used but is optional otherwise.
3.	The comma or **or** must be used as a separator between transfer types in the list. The plus sign or **and** is not allowed.
4.	Each transfer type must be valid in the institution's code list. If no transfer types are listed, then all transfer types are assumed by the Auditor Engine to be acceptable.
5.	When used in a block header, **MaxTransfer** applies to all courses used to fulfill the block requirements.

MAXTRANSFER (rule)

Maximum number of transfer credits or classes.

```
##maximum of 3.5 transfer credits in MUS 1@
12 Credits in MUS 1@
MaxTransfer 3.5 Credits
Label "Music";
```

Notes:

1.	**MaxTransfer** is followed by either a real number and the Credit[s] keyword or an integer and the Class[es] keyword, indicating the maximum number of credits or classes that can be transferred from another college or university.
2.	Following the class/credits statement is an optional "In" or "From" followed by a list of transfer types in parentheses. The list of transfer types is required when in or From is used but is optional otherwise.
3.	The comma or 'Or' must be used as a separator between transfer types in the list. The plus sign or 'And' is not allowed.
4.	Each transfer type must be valid in the institution's code list. If no transfer types are listed, then all transfer types are assumed by the Auditor Engine to be acceptable.
5.	**MaxTransfer** applies to all courses in the preceding course list.

MINAREA[S] (rule)

Indicates the minimum number of areas from a course list to which courses will be applied. The **MinAreas** qualifier is useful for grouping courses within a course list across multiple disciplines. This qualifier provides greater flexibility then the **MinSpread** qualifier used for grouping classes by discipline code. The **MinAreas** qualifier can only be used within a course rule. The grouped areas are defined by the Scribe user and can contain classes from multiple disciplines. The Parser does some checking, but it is up to you to set up the rule correctly.

```
12 Credits in [PSY 130, 135, BIO 156,]
              [BIO 145, 177, CHE 205 ]
    MinAreas 2
  Label "Science";

10:15 Credits in
   [MATH 103, STAT 123, 124,]
   [CHEM 103, 104, 105,]
   [BIO  103, 104, 105 ]
   MinAreas 2
Label "Science";
```

The areas are defined using square brackets. It is usually easiest to write your list of courses first without the brackets and then add the brackets later. Note the placement of the commas. The **MinAreas** qualifier does NOT support the use of course ranges (e.g., MATH 101:238). Also note that the last course in the list area must have a space before the right bracket.

Notes:

1.	**MinAreas** is followed by an integer that indicates the minimum number of areas from the list that can be represented in the courses used to satisfy the requirement. If there are 3 areas defined and **MinAreas** is 2, then the courses that satisfy the requirement must be from at least 2 of the 3 areas.
2.	**MinAreas** applies to all courses in the preceding course list.
3.	**MinAreas** can be used only with a course rule, not with Blocks, BlockType, Group, If, NonCourse, or BeginSub/EndSub.
4.	**MinAreas** is used in place of **MinPerDisc** whenever the areas cover more than one discipline. **MinAreas** works very much like **MinPerDisc** except that the brackets define the areas for the former while the discipline defines the areas for the latter.
5.	Square brackets are used to define the areas. As the example above shows, the comma after the course is placed inside of the ending bracket.
6.	When you use {**HIDE**} inside an area-type rule with [ ], you need to place the **HIDE** as the first course in the rule, inside the starting square bracket "[{". An example would be as follows: `[{HIDE PSY 154}, ANT 222, HIST 101 ]` The curly brace "}" cannot be followed immediately by the square bracket "]".

MINCLASS[ES] (header)

Indicates the minimum number of courses that must be earned to satisfy the requirement.

```
MinClasses 8 in PE @

MinClasses 3 from PSY @
  Except PSY 112
  Tag=MINPSY
  ProxyAdvice "You have taken <APPLIED> PSY classes but "
  ProxyAdvice "still need <NEEDED> more. (PSY 112 cannot count)"
  Label "3 PSY classes"

##minimum of 8 courses in MUS @ or 8 courses in ART @, or both summed
MinClasses 8 in MUS @, ART @

##minimum of 8 courses in MUS @ and 8 courses in ART @
MinClasses 8 in MUS @
MinClasses 8 in ART @
```

Notes:

1.	**MinClass[es]** is followed by an integer that indicates the minimum number of courses that can be applied to the rule. The integer is optionally followed by **in** or **from**, which must be followed by a course list.
2.	The connector in the course list following **MinClass[es]** must be comma or **or**. The plus sign or **and** is not allowed. The logical **or** connector is not exclusive. If multiple courses are listed, then the sum of all courses that satisfy the list is compared against the integer following **MinClass[es]**
3.	**MinClass[es]** applies to all courses in the succeeding course list.
4.	The **MinClass[es]** qualifier is applied by the Auditor Engine against all the courses used to fill the block.
5.	The course list associated with **MinClass[es]** allows **Except** but not **Including**.

MINCLASS[ES] (rule)

Minimum number of courses applied to the rule.

```
12 Classes in BUS 3@, ACCT 3@
    MinClasses 1 in BUS 321, ACCT 306 # one of the classes has to be BUS 321
or ACCT 306
    MinClasses 1 in BUS 327, ACCT 312 # one of the classes has to be BUS 327
or ACCT 312
    Label "Accounting electives";

# Note we don't have repeat the WITH operator on the course in the
MinClasses qualifier as
# the class can only be placed on MinClasses if it fits the normal part of
the rule
5 Classes in HIST 2@  (With Attribute=WRIT), ENGL 300:319
    MinClasses 2 in HIST 206, ENGL 311, 316 # two of the classes must be
from this list
    Label "Writing intense option";
```

Notes:

1.	**MinClass[es]** is followed by an integer that indicates the minimum number of courses that can be applied to the rule. The integer is optionally followed by "In" or "From" which must be followed by a course list.
2.	The connector in the course list following **MinClass[es]** must be comma or 'Or'. The plus sign or 'AND' is not allowed. The logical Or connector is not exclusive. If multiple courses are listed then the sum of all courses that satisfy that list is compared against the integer following **MinClass[es]**
3.	**MinClass[es]** applies to all courses in the succeeding course list.
4.	The **MinClass[es]** qualifier is applied by the Auditor Engine against all the courses used to fill the rule.
5.	Can be used on individual rules within a Group or Subset also. May not be used as a qualifier for the entire Group or Subset.

MINCREDIT[S] (header)

Indicates the minimum number of credits that must be earned to satisfy a requirement.

```
MinCredits 8 in ART @, MUS @
  Except ART 112
  Tag=MINARTMUS
  ProxyAdvice "You have taken <APPLIED> ART and MUS credits but "
  ProxyAdvice "still need <NEEDED> more. (ART 112 cannot count)"
  Label "8 credits in art and music"

##minimum of 8 credits in MUS @ or 8 credits in ART @, or both summed
MinCredits 8 in MUS @, ART @

##minimum of 8 credits in MUS @ and 8 credits in ART @
MinCredits 8 in MUS @
MinCredits 8 in ART @
```

Notes:

1.	**MinCredit[s]** is followed by a real number that indicates the minimum number of credits that can be applied to the rule. The number is optionally followed by **in** or **from**, which must be followed by a course list.
2.	The connector in the course list following **MinCredit[s]** must be comma or **or**. The plus sign or **and** is not allowed. The logical **or** connector is not exclusive. If multiple courses are listed, then the sum of all credits that satisfy the list is compared against the number following **MinCredit[s]**.
3.	**MinCredit[s]** applies to all courses in the succeeding course list.
4.	The **MinCredit[s]** qualifier is applied by the Auditor Engine against all the courses used to fill the block. The number of credits in **MinCredit[s]** is a lower bound that must be met or exceeded.
5.	The course list associated with **MinCredit[s]** allows **Except** but not **Including**.

MINCREDIT[S] (rule)

Minimum number of credits applied to the rule.

```
30 Credits in BUS 3@, ACCT 3@
    MinCredits 3 in BUS 321, ACCT 306 # one of the classes has to be BUS 321
or ACCT 306
    Label "Accounting electives";

# Note we don't have repeat the WITH operator on the course in the
MinClasses qualifier as
# the class can only be placed on MinClasses if it fits the normal part of
the rule
15 Credits in HIST 2@(With Attribute=WRIT), ENGL 300:319
    MinCredits 6 in HIST 206, ENGL 311, 316 # two of the classes must be
from this list
    Label "Writing intense option";
```

Notes:

1.	**MinCredit[s]** is followed by a real number that indicates the minimum number of credits that can be applied to the rule. The number is optionally followed by "In" or "From" which must be followed by a course list.
2.	The connector in the course list following **MinCredit[s]** must be comma or 'Or'. The plus sign or 'AND' is not allowed. The logical OR connector is not exclusive. If multiple courses are listed then the sum of all credits that satisfy that list is compared against the number following **MinCredit[s]**
3.	**MinCredit[s]** applies to all courses in the succeeding course list.
4.	The **MinCredit[s]** qualifier is applied by the Auditor Engine against all the courses used to fill the rule. The number of credits in **MinCredit[s]** is a lower bound which must be met or exceeded.
5.	Can be used on individual rules within a Group or Subset also. May not be used as a qualifier for the entire Group or Subset.

MINGPA (header)

Indicates the minimum Grade Point Average for the requirements block; formatted as nnn.nnn.

```
MinGPA 2.0

# You may also include a course list when needed
MinGPA 3.0 in CHEM @, PHYS @, BIOL @
  Except CHEM 109, BIOL 243, 244


# You may also specify a scope – referencing another block – though normally
# this is done in an AWARD block
MinGPA 2.0 in (MAJOR)
```

Notes:

1.	**MinGPA** is followed by a real number whose maximum value is determined by a configuration setting, not to exceed 999.999.
2.	**MinGPA** indicates the minimum GPA for all courses used in the block. A GPA is calculated for the block by summing the grade points for all courses applied to the block, summing the graded credits earned for all courses applied to the block, and dividing the total grade points by the total graded credits earned.

MINGPA (rule)

Indicates the minimum Grade Point Average for the classes applied to the rule.

```
6 Credits in HIST 2@
  MinGPA 2.5
  Label hist2 "History requirement";

BeginSub
    6 Credits in HIST 2@
      Label hist1 "History requirement";
    1 Class in HIST 1916
      Label hist2 "History of Ireland";
EndSub
  MinGPA 2.5
  Label hist "History requirement";
```

Notes:

1.	**MinGPA** is followed by a real number whose maximum value is determined by a configuration setting, not to exceed 999.999.
2.	**MinGPA** indicates the minimum GPA for all courses used in the rule. A GPA is calculated for the rule by summing the grade points for all courses applied to the rule, summing the graded credits earned for all courses applied to the rule, and dividing the total grade points by the total graded credits earned.

MINGRADE (header)

Indicates the minimum numeric grade that must be achieved for each course applied to the block; formatted as nnn.nnn.

```
BEGIN
36 Credits
MinGrade 1.000
;
```

Notes:

1.	Grade equals the numeric equivalent of the letter grade, for example, A=4.0, B=3.0, C=2.0, D=1.0, and F=0.0.
2.	**MinGrade** can be used to indicate the minimum passing grade that must be achieved for a requirement to be satisfied.
3.	**MinGrade** is followed by a real number whose maximum value is determined by a configuration setting, not to exceed 999.999.
4.	**MinGrade** indicates the minimum grade for all courses used in the block.
5.	**MinGrade** is needed as part of the rule statement only if the block has not declared **MinGrade** or if the rule has a different **MinGrade** than the block.

MINGRADE (rule)

Indicates the minimum numeric grade that must be achieved for each course applied to the block; formatted as nnn.nnn.

```
6 Credits in BUS 5@
   MinGrade 2.0
  Label "Graduate Business";
```

Notes:

1.	Grade equals the numeric equivalent of the letter grade, for example, A=4.0, B=3.0, C=2.0, D=1.0, and F=0.0.
2.	**MinGrade** can be used to indicate the minimum passing grade that must be achieved for a requirement to be satisfied.
3.	**MinGrade** is followed by a real number whose maximum value is determined by a configuration setting, not to exceed 999.999.
4.	**MinGrade** indicates the minimum grade for all courses used in the rule.
5.	**MinGrade** is needed as part of the rule statement only if the block has not declared **MinGrade** or if the rule has a different **MinGrade** than the block.

MINOR (header)

Indicates a minor requirement block (MINOR) against which **NonExclusive** is applied.

```
BEGIN
36 Credits
Share 9 CREDITS (MINOR);

BEGIN
36 Credits
ShareWith (PROGRAM, MINOR);
```

Notes:

1.	**MINOR** is used within a block header, only with **NonExclusive**, **Share**, or **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can also be used to satisfy requirements in the MINOR block (if the MINOR block exists for the student).

MINOR (rule)

Indicates a minor requirement block (MINOR), which, in some cases, may precede a minor code.

```
1 Block (MINOR=ENG)
  Label "English Minor requirements";

1 BlockType (MINOR)
  Label "Minor required";

If (MINOR=ENG) Then
  1 Class in ENG300
    Label "ENG 300";

6 Credits in BUS 1@
  ShareWith (MINOR)
   LABEL "Business";
```

Notes:

1.	**MINOR** is followed by a minor code, such as ENG for English, except in a BlockType rule.
2.	The minor code must be valid in the institution's code list.
3.	**MINOR** can be used as part of a condition following **If**.
4.	**MINOR** can be used with Block[s], BlockType[s], If, NonExclusive, Share, or ShareWith.

MINPERDISC (header)

Indicates the minimum number of credits/classes in each discipline listed that must be earned in order to satisfy the requirement.

```
##minimum of 6 credits per discipline in BIO or CHE, or both summed
MinPerDisc 6 Credits (BIO, CHE)

MinPerDisc 4 Classes (PSY)

##minimum of 6 credits BIO and 6 credits CHE
MinPerDisc 6 Credits (BIO)
MinPerDisc 6 Credits (CHE)
```

Notes:

1.	**MinPerDisc** indicates the minimum number of credits or classes per discipline represented in the discipline list.
2.	**MinPerDisc** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The integer or number indicates the minimum number of classes or credits that must be taken in each discipline, followed optionally by **in** or **from**, followed by left parenthesis, followed by a discipline code, optionally followed by additional discipline codes separated by a comma or **or**, followed by right parenthesis.
3.	The list of disciplines following **MinPerDisc** indicates the disciplines that will be evaluated. Each discipline code must be valid in the institution's code list.
4.	If the configuration setting for maximum discipline length (UCX-CFG020 DAP13 Disc_length) is zero, then **MinPerDisc** is invalid.

MINPERDISC (rule)

Indicates the minimum number of credits/classes in each discipline listed that must be earned in order to satisfy the requirement.

```
12 Credits in BIO 100:299, CHE 100:299, PHY 100:299
   MinPerDisc 3 CREDITS (BIO)
   LABEL "Science";

## minimum of 8 credits per discipline in BIO or CHE, or both summed
16.0 Credits in BIO @, CHE @, PHY @
    MinPerDisc 8.0 Credits in (BIO, CHE)
    Label "Science";

## minimum of 4 classes in ART and 4 classes in MUS
12 Classes in ART @, MUS @, THE @
    MinPerDisc 4 Classes in (ART)
    MinPerDisc 4 Classes (MUS)
    Label "The Arts";

BeginSub
  1:2 Classes in LAWJ 2@
    Label "200-level LAWJ classes";
  3:4 Classes in LAWJ 3@
    Label "300-level LAWJ classes";
EndSub
  MinPerDisc 14 Credits (LAWJ)
  Label LAWJ "LAWJ requirement";
```

Notes:

1.	**MinPerDisc** indicates the minimum number of credits or classes per discipline represented in the discipline list. The disciplines listed must be present in the course list that precedes **MinPerDisc**.
2.	**MinPerDisc** follows a course list on a course rule and is allowed on a group and subset.
3.	**MinPerDisc** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The integer or number indicates the minimum number of classes or credits that must be taken in each discipline, followed optionally by **in** or **from**, followed by left parenthesis, followed by a discipline code, optionally followed by additional discipline codes separated by a comma or **or**, followed by right parenthesis.
4.	The list of disciplines following **MinPerDisc** indicates the disciplines that will be evaluated. Each discipline code must be valid in the institution's code list.
5.	If the configuration setting for maximum discipline length (UCX-CFG020 DAP13 Disc_length) is zero, then **MinPerDisc** is invalid.

MINRES (header)

Indicates the minimum number of credits/classes that must be earned in residence.

```
BEGIN
120 Credits
MinRes 36 Credits
;

BEGIN
12 Classes
MinRes 12 Classes
;
```

Notes:

1.	**MinRes** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The number indicates how many classes or credits must be earned in residence.
2.	When used in a block header, **MinRes** applies to all courses used to fulfill the block requirements.

MINSPREAD (rule)

Indicates the minimum number of disciplines from the course list in which courses must be earned to satisfy the requirement.

```
12 Credits in PSY @, BIO @, CHE @
MinSpread 2
Label "Science";

##3 courses spread across at least 2 of the 3 disciplines listed
3 Classes in MATH @, HIST @, PHIL @
MinSpread 2
Label "Freshman Spectrum";
```

Notes:

1.	**MinSpread** is followed by an integer that indicates the minimum number of disciplines from the list that can be represented in the courses used to satisfy the requirement. If there are 3 disciplines listed and **MinSpread** is 2, then the courses that satisfy the requirement must be from at least 2 of the 3 disciplines.
2.	**MinSpread** applies to all courses in the preceding course list.
3.	**MinSpread** can be used only with a course rule or subset, not with Block, BlockType, Group, If, or NonCourse.
4.	If the configuration setting for maximum discipline length (UCX-CFG020 DAP13 Disc_length) is zero, then **MinSpread** is invalid.

MINTERM (header)

Indicates the minimum number of credits/classes that must be taken per term.

```
MinTerm 1 Credit PE @

MinTerm 1 Class MUS 100:199
  Except MUS 106

## minimum of 3 credits per term in ME100:115 or 3 credits in PE100:115, or
both summed
MinTerm 3 Credits in ME100:115, PE100:115

## minimum of 3 credits per term in ME100:115 and 3 credits in PE100:115
MinTerm 3 Credits in ME100:115
MinTerm 3 Credits in PE100:115

## minimum of 12 credits per term
MinTerm 12 Credits
```

Note: This last option only works in the starting block or an AWARD block. When no list is specified, the Auditor looks at credits attempted (not earned) in the entire audit.

Notes:

1.	**MinTerm** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The number indicates the minimum number of classes or credits that must be taken per term.
2.	The syntax for **MinTerm** varies depending on its scope. When used in the block header, **MinTerm** is followed by the number of classes or credits, optionally followed by **in** or **from**, followed by a course list in which comma/**or** is allowed but plus/**and** is not allowed.
3.	**MinTerm** can only be used if all of a student's courses and transfer credits have been linked to a term.
4.	**MinTerm** applies to all courses that match the succeeding course list and were used to fulfill any rule or rules in the block.
5.	The course list associated with **MinTerm** allows **Except** but not **Including**

MINTERM (rule)

Indicates the minimum number of credits/classes that must be taken per term.

```
8 Classes in MUS 199
   MinTerm 1 CLASS
  Label "Music Practice";

##minimum of 3 credits of Music courses per term
12 Credits in MUS @
    MinTerm 3 Credits
   Label "Music";
```

Notes:

1.	**MinTerm** is followed by either an integer followed by **Class**[es] or a real number followed by **Credit**[s]. The number indicates the minimum number of classes or credits that must be taken per term.
2.	The syntax for **MinTerm** varies depending on its scope. When used in a rule statement, **MinTerm** must follow a course list and be followed by the number of classes or credits.
3.	**MinTerm** can only be used if all of a student's courses and transfer credits have been linked to a term.
4.	**MinTerm** applies to all courses in the preceding course list.
5.	**MinTerm** can be used only with a course rule, not with Block, BlockType, Group, If, NonCourse, or BeginSub/EndSub.

NOCOUNT (rule)

Allows courses to satisfy specific requirements without affecting the total credit count or GPA calculation. (The type of requirements that were the catalyst for the **NoCount** option involved ROTC classes that can satisfy the PE requirement but do not count toward the degree.)

```
5 Classes in PE @, ROTC @ NoCount
  Label "PE/ROTC";

2 Classes in MIL 100:299 NoCount, HIST 100:299
  Label "History and Military";
```

Notes:

1.	The **NoCount** keyword is only allowed on courses specified in a class rule. The **NoCount** keyword is not allowed with a course rule specified with credits or with classes and/or credits. The class rule must be expressed in terms of **Classes**.
2.	**NoCount** can be listed only once per course. If multiple locations are used, then list **NoCount** after each appropriate location, for example, "ENG100 **at** SM **NoCount**, ENG100 **at** XX **NoCount**".
3.	Only the last course before **NoCount** is qualified by **NoCount**. For example, "ENG 101, 110 **NoCount**" qualifies only ENG 110 as **NoCount**.
4.	For ranges of course numbers, **NoCount** applies to all courses in the range. For example, "ART 100:102 NoCount" is the same as "ART 100 **NoCount**, ART 101 **NoCount**, ART 102 **NoCount**".
5.	Only use **NoCount** after a course if you do not want the Auditor Engine to count the course in the number of credits and GPA. The **NoCount** keyword will mask courses applied to the requirement by using a credit value of 0. Rule qualifiers on a rule with the **NoCount** keyword will count classes applied to the requirement as a class but will treat these courses as 0 credit courses.

6.	The **NoCount** courses are invisible to all block qualifiers. They are not counted in a block's class or credit counts. For example:
	
	`BEGIN`
	`MinCredits 10 in @ 100:200`
	`MinClasses 3 in @ 100:200`
	
	`;`
	
	`10 classes in GEF @, ABC 100:200 NoCount, XYZ @`
	`MinPerDisc 3 Credits(ABC, XYZ)`
	`MinPerDisc 5 Classes(GEF, ABC);`
	
	`END.`
	
	If a student takes 8 ABC courses for 3 credits each, the Auditor Engine will apply them to the above rule (assuming they do not apply anywhere else). Let us assume that the student does not take any GEF or XYZ courses. The 8 ABC courses do satisfy the **MinPerDisc** 5 **Classes** rule qualifier but do not satisfy the **MinPerDisc** 3 **Credits** rule qualifier. The ABC courses also do not satisfy the **MinCredits** and **MinClasses** block qualifiers. The Auditor Engine marks the rule as 80 percent complete since 8 of its 10 classes were satisfied.

7.	**NoCount** courses do not count in a rule's GPA calculation, a block's GPA calculation, or in the overall GPA calculation.
8.	A course that is applied to a count rule has higher priority over a **NoCount** location.
9.	A course can only be applied to one **NoCount** rule regardless of any sharing rule qualifiers.
10.	Any **NoCount** courses that are not needed at the rule level are applied to other rules, if

	needed, or go to fall-through.
11.	Using **NoCount** and **NonExclusive**, **Share**, or **ShareWith** together could yield unpredictable results.

NONCOURSE[S] (rule)

Indicates a required non-course activity, such as a thesis, recital, or exam.

```
1 NonCourse (LIFE, COMM)
  Label "Life Issues or Community Service";

1 NonCourse (CHAPEL > 50)
  ProxyAdvice "You need to attend chapel 50 times or more"
  Label "Chapel Attendance 50 times or more";

##2 from Life Issues or College Service or Community Service
2 NonCourses (LIFE, COLL, COMM)
  Label "Human Services";

##2 recitals with a DONE status are required.
2 NonCourses (RECITAL = DONE)
  Label "Recital";

1 Group in (1 NonCourse (RECITAL)
              Label "Recital") OR
           (1 NonCourse (THESIS)
              Label "Thesis")
  Label "GRADUATION REQUIREMENTS";
```

Notes:

1.	**NonCourse**[s] is preceded by an integer and followed by the code for a non-course requirement or a list of non-course requirements connected by a comma or **or**. The list of noncourses must be enclosed in parentheses. The integer specifies how many of the noncourses listed are required. The number of noncourse codes listed in parentheses is not checked against the integer specified. The integer can be less than, equal to, or greater than the number of noncourse codes listed.
2.	Multiple noncourses can be listed if separated by comma or **or**. Plus or **and** is not allowed in a list of noncourses. To specify that multiple, different noncourses are required, enter a **NonCourse** rule for each noncourse. For example, if both LIFE and SERV are required, enter: "**NonCourse** (LIFE) **NonCourse** (SERV)".
3.	The code following **NonCourse**[s] must appear in the institution's noncourse_code list (UCX–SCR003), where it is mapped to a field in the database.
4.	**NonCourse**[s] includes noncourse values. The routine that passes data from the student database also passes along this value. You can choose to ignore the value or you can use equals, not-equals, greater-than or less-than to check the value. For example, 1 NonCourse (Thesis > 67) – the student not only must have the THESIS noncourse but also must have a value greater than 67.
5.	A semicolon is not used to end the **NonCourse** rule when **NonCourse**[s] is used within Group. When used within a **Group** rule, the **NonCourse** rule is enclosed in parentheses.
6.	Allowable rule qualifiers: none.

NONEXCLUSIVE (header)

Same as **Share** and **ShareWith**.

NONEXCLUSIVE (rule)

Same as **Share** and **ShareWith**.

NOTGPA (rule)

Indicates courses that count in neither the block GPA nor the overall GPA.

```
If (ROTC=YES) Then
  6 Classes in MIL @
   NotGPA
 Label "ROTC";

6 Credits in INS1 @ NotGPA
  Label "International Studies";
```

Notes:

1.	**NotGPA** indicates that the credits applied towards satisfying the rule should not be counted in the block GPA.
2.	The Auditor Engine will calculate one GPA per requirements block for a student. The **NotGPA** keyword indicates that the courses used to satisfy a particular rule should count toward neither the block GPA nor the overall GPA.
3.	DAP will not calculate the cumulative GPA for a student. The student system calculates the cumulative GPA, and DAP then receives the cumulative GPA from the student system.
4.	It is possible to construct special requirements blocks that serve only to calculate a GPA for a group of courses. For example, a science GPA is desired but there is no requirements block for science. Build a requirements block for science (OTHER=SCIENCE), put ShareWith in the block header, and create the rules that list the courses to be used in the science GPA. Then, in the blocks for specific science majors (e.g., MAJOR=BIO, MAJOR=CHEM, MAJOR=PHYS), add the following rule "1 **Block** (**OTHER**=SCIENCE)".
5.	If a course is on one rule with **NotGPA** and on another rule without **NotGPA**, then the course will be counted in the block's GPA.

NumberOfConcentrations

NumberOfMajors,

NumberOfMinors,

Used in an if-statement to find out how many majors, minors or concentrations are included in the audit.

```
If (NumberOfMajors >= 2 or NumberOfMinors >= 1) THEN
  RuleComplete
   Label "Gened elective waived because of double-major or additional minor"
Else
 10 Credits in ENGL @, WRIT @, HUM  @
    Label "Humanity elective";

##Communication majors require a minor unless they double major
  If (MAJOR=COMM and NumberOfMajors = 1) Then
   1 BlockType (MINOR)
      Label 7 "Minor Requirements";
```

Notes:

1.	**NumConcs** may be used in place of **NumberOfConcentrations** **NumMajors** may be used in place of **NumberOfMajors** **NumMinors** may be used in place of **NumberOfMinors**
2.	The Auditor examines the count of blocks of the specified type that were pulled into the audit, it does not count the number of values on the student's academic record – though normally the counts should match.
3.	None of these names need to be setup in UCX-SCR002

OPTIONAL (header)

Indicates a block that does not have to be passed to graduate.

```
BEGIN
24 Credits
Optional

;

Remark "Honors Program - Optional For Students";
```

Notes:

1.	**Optional** indicates that the block should be evaluated but that the degree can still be awarded if the requirements are not met.
2.	**Optional** applies to the entire block, not just a specific rule or qualifier.

OR (header)

Boolean operator; connector in list of courses, disciplines, or transfer types.

```
BEGIN
36 Credits or 12 Classes
;

BEGIN
40:45 Classes or 120:135 Credits
;

BEGIN
MaxClasses 3 from PE @ or HE @
;


BEGIN
MaxTransfer 30 Credits from (AP, CLEP)
;
```

Notes:

1.	**Or** can connect **Credit**[s] and **Class**[es].
2.	The comma in a list is equivalent to **or** but is not always interchangeable with **or**. The comma cannot be used as a connector between **Credit**[s] and **Class**[es]. **Or** can be used in place of a comma anywhere.
3.	**Or** or a comma can be used between courses in a list associated with MaxClass[es], MaxCredit[s], MaxTerm, MinClass[es], MinCredit[s], or MinTerm; between disciplines in a list associated with MaxPerDisc, MinPerDisc, SameDisc; and between transfer types in a list associated with MaxTransfer.

OR (rule)

Boolean operator; connector in list of courses, disciplines, transfer types, or **If** conditions.

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

6 Credits or 3 Classes in MAT @
  Label "Math";

If (MAJOR=PE or MAJOR=HE) Then
  3 Classes in EDU 100, EDU 115, EDU 125
    Label "Education";


##equivalent to INS101 or INS102
1 Class in INS 101, 102
  Label "International Studies";

1 Group From
  (6 Credits in FRE @, SPA @, GER @
      Label "French/Spanish/German") OR
  (9 Credits in GRK @, LAT @
      Label "Greek/Latin")
  Label "LANGUAGE";

4 Classes in BIO@, CHE@
 MinPerDisc 1 Class (BIO, CHE)
Label "Biology or Chemistry";

1 Block (OTHER=LIFE, OTHER=COMM)
  Label "Communication";

1 BlockType (SPEC, CONC)
  Label "Specialization";

1 NonCourse (RECITAL, GALLERY)
   Label "Graduation Requirements";
```

Notes:

1.	**Or** can connect conditions in an **If** statement.
2.	**Or** can connect rules listed within a **Group** rule.
3.	**Or** can connect **Credits**[s] and **Class**[es].
4.	The comma in a list is equivalent to **or** but is not always interchangeable with **or**. The comma cannot be used in place of **or** in an **If** statement or as a connector between **Credits**[s] and **Class**[es]. **Or** can be used in place of a comma anywhere.
5.	**Or** cannot be intermingled with **and** in a course list. The connectors in a list of courses must all be either **and**/plus or **or**/comma. For example, "**BIO 100, CHE 100 + PHY 100**" is invalid.
6.	**OR** or a comma can be used between courses in a list associated with **Class**[es], **Credit**[s], **Except**, or **Including**; between disciplines in a list associated with **MaxPerDisc**, **MinPerDisc**, **SameDisc**; between transfer types in a list associated with **MaxTransfer**; between noncourses in a list associated with **NonCourse**[s]; between blocks in a list associated with **Block**[s]; between block types in a list associated with **BlockType**[s]; between groups in a list associated with **Group**[s].

OTHER (header)

Identifies a block as a custom block.

```
BEGIN
36 Credits
ShareWith (OTHER=GENED)
;
```

Notes:

1.	**OTHER** is always followed by a single custom_block code. This code can be up to 12 bytes long and describes the requirements contained in the block.
2.	The custom_block code must already be in the DAP database as a block type value associated with block type OTHER. In other words, the linked block (the one specified after BLOCK) must be created before the requirements block is parsed.
3.	**OTHER** is used within a block header, only with **NonExclusive**, **Share**, or **ShareWith**, to indicate that the credits or classes applied towards satisfying the requirements in the block can be used to satisfy requirements in the OTHER block if the OTHER block exists.

OTHER (rule)

Identifies a block as a custom block.

```
1 Block (OTHER=GENELEC)
  Label "General Electives";

6 Credits in ENGL1 @
 ShareWith (OTHER=GENED)
  Label "English";
```

Notes:

1.	**OTHER** is always followed by an equal sign and a single custom_block code. This code can be up to 12 bytes long and describes the requirements contained in the block.
2.	The custom_block code must be in the DAP database as a block type value associated with block type OTHER.
3.	When used in a rule statement, **OTHER** must immediately follow **Block**[s], **NonExclusive**, **Share**, or **ShareWith**.
4.	**OTHER** indicates which block of type OTHER must be evaluated.

PROGRAM (header)

Indicates a program type of block (PROGRAM) against which **NonExclusive** is applied.

```
BEGIN
36 Credits
Share 9 Credits (PROGRAM)
;

BEGIN
36 Credits
ShareWith (PROGRAM, CONC)
;
```

Notes:

1.	**PROGRAM** is used within a block header, only with **NonExclusive**, **Share**, or **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can also be used to satisfy requirements in the PROGRAM block (if the PROGRAM block exists for the student).

PROGRAM (rule)

Indicates a program requirement block (PROGRAM), which, in some cases, may precede a program code.

```
1 Block (PROGRAM=HONORS)
  Label "Honors Program";
```

Notes:

1.	**PROGRAM** is followed by a program code, such as HONORS, except in a **BlockType** rule.
2.	The program code must be valid in the institution's code list.
3.	**PROGRAM** can be used as part of a condition following **If**.
4.	**PROGRAM** can be used with **Block**[s], **BlockType**[s], **If**, or **NonExclusive**, **Share** or **ShareWith**.

PROXY-ADVICE (header)

Specifies the advice text to show in place of the normal advice Degree Works would display for the block. The proxy-advice text will appear as long as the block header qualifier is not complete; as soon as the block header qualifier has been completed, the proxy-advice text will be suppressed. Proxy-advice can be used on the following block header qualifiers: Classes, Credits, LasRes, MinGPA, MinPerDisc, MinRes, MinCredits.

```
BEGIN
 130 Credits
   Proxy-Advice "130 credits are required to graduate. "
   Proxy-Advice "You still need <NEEDED> credits."

 MinGPA 2.0
   Proxy-Advice "Your GPA is <APPLIED>; a GPA of 2.0 is required."
;
```

Notes:

1.	**Proxy-Advice** is followed by up to 65 bytes of text enclosed in quotes.
2.	**Proxy-Advice** can be repeated as many times as needed; the text is appended together.
3.	Advice text appears as long as header qualifier is not complete.
4.	The hyphen in **Proxy-Advice** is optional.

PROXY-ADVICE (rule)

Specifies the advice text to show in place of the normal advice Degree Works that would display for the rule. The proxy-advice text will appear as long as the rule is not complete; as soon as the rule has been completed the proxy-advice text will be suppressed. **Proxy-Advice** should be used on rules containing long lists of classes or complex group rules.

```
3 Credits in MATH 104, 109, 115, 119, 135, 156, 178, 198, 148, 199
    Proxy-Advice "3 Credits in some math class not related to business"
    Proxy-Advice " or computers is required. You have taken <APPLIED>
credits"
    Proxy-Advice " and need <NEEDED> more."

  Label "Math requirement";

1 Group in
   (9 Credits in SPAN 1@ Label "Spanish") OR
   (9 Credits in FREN 1@ Label "French") OR
   (9 Credits in IRSH 1@ Label "Irish")
     Proxy-Advice "9 Language Credits are required"
  Label "LANGUAGE REQUIREMENT";
```

Notes:

1.	**Proxy-Advice** is followed by up to 65 bytes of text enclosed in quotes.
2.	**Proxy-Advice** can be repeated as many times as needed; the text is appended together.
3.	Advice text appears as long as rule is not complete.
4.	The hyphen in **Proxy-Advice** is optional.
5.	**Proxy-Advice** may be used in any rule that gives advice.
6.	<APPLIED> and <NEEDED> can only be used on course rules; i.e., they cannot be used on group, subset, blocktype rules etc. The value will be credits if it is a credits rule and the value will

	be classes if it is a classes rule. The value will also be credits if the rule is specified as credits and/or classes.

PSEUDO

See **Credits** (header)

REMARK

Begins comments to be kept by Parser Engine as part of requirements.

```
Remark "BSC103 is recommended for freshmen.";

Remark "Proficiency in English demonstrated to the satisfaction of "
Remark "the English Proficiency Council (a score of at least 44 on "
Remark "the Test of Standard Written English, or a grade of C or "
Remark "better in ENG 111).";
```

Notes:

1.	**Remark** is used after a rule statement or at the start of the body section of the block. If used for a rule, the **Remark** follows the semicolon that ends the previous rule statement. It can be on the same line after the semicolon or on a new line. Putting the remark on a new line allows room for longer remark text.
2.	**Remark** can occur multiple times; it's use is unlimited.
3.	**Remark** is followed by a free-text comment enclosed in quotes. The text can be 0 to 72 characters long and cannot include quotation marks. The text may be blank, i.e. **Remark** " ";. Remarks are limited to 72 bytes to ensure they will display properly on worksheet output. Remark text which is exactly 72 characters long cannot fit on the 80-byte line with the "REMARK" keyword, the trailing semicolon and surrounding quotes. If you happen to have a remark that is exactly 72 bytes long you can either split the remark between two lines or simply insert a carriage-return after the "REMARK" keyword to place the quoted value on a new line.
4.	A semicolon may optionally follow the closing quote.
5.	The free-text comment is used as description on DAP output. It should be a description phrased for the advisor, similar to the college catalog.
6.	Remarks are stored and output in the order entered. They are not stored as part of the block or rule. They relate to the requirement rules only by proximity to the rule and free-text description of rules in the block. In this way, Degree Works handles comments that relate to multiple rule statements.
7.	**Remark** statements are outside the scope of rule. They stand alone.
8.	Remarks are not allowed within a subset or within a Group's list of rules.

RULE-COMPLETE (rule)

A dummy rule that is always 100 percent complete and has no requirements. Although it can be used anywhere a course rule can be used, its main purpose is to be used within an **If** statement.

```
Rule-Complete
 Label "This requirement is complete";

If (ROTC=YES) Then
  Rule-Complete
   Label "PE requirement satisfied because of ROTC"
Else
 15 Credits in PE @
    Label "PE Requirement";

If (SATV >= 0670) THEN
  Rule-Complete
   Label "Writing Requirement Waived: SAT Score"
Else
 4 Credits in ENGL 101
   Label "Freshman Composition";
```

Notes:

1. Allowable rule qualifiers: None.

RULE-INCOMPLETE (rule)

A dummy rule that is always 0 percent complete and has no requirements. Although it can be used anywhere a course rule can be used, its main purpose is to be used within an **If** statement.

```
If (ENGLTEST < 4) THEN
   Rule-Incomplete
     ProxyAdvice "Need to retake the English test"
     Label "English Requirement";
ELSE
   Rule-Complete
     Label "English test covers English Requirement";
```

Notes:

1. Allowable rule qualifiers: None.

RULETAG

A special qualifier you can place on any rule to give it special meaning when the audit worksheet is being displayed. Any RuleTag name-value pair you add to your rule will be available for use within the xsl stylesheets, so show the rule in a special way—hide it, use a different color, etc. Both the name and the value following **RuleTag** are limited to 50 bytes each. The value must be enclosed in double-quotes if it contains non-alphanumeric characters.

```
5 Credits in @ (WITH ATTRIBUTE=WRIT)
    ProxyAdvice "Click here to see classes that meet this requirement"
    RuleTag AdviceJump="writing.html"
  Label abc "Writing Requirement";


2 Classes in ART 101, ARTH @, GRAPH @

    RuleTag RemarkJump="http://some.place.edu/ontheinternet/anywhereisfine/"

    RuleTag RemarkJump="support/getmemoreinfo.html"

    RuleTag RemarkHint="More info on Gen Ed option-a"

    Label 123 "Gen Ed option A";

Remark "You can click this link to find out more information";

Remark "about this requirement.";


5 Credits in BIOL 2@, PHYS 250:270
    RuleTag Category="Special Science Classes"
  Label "Science classes";
```

Notes:

1.	Allowable with any rule.
2.	RuleTag names of AdviceJump, RemarkJump and RemarkHint have special meaning and are used by the standard Degree Works worksheets. See the *AdviceJump* and *RemarkJump* and *RuleTag* sections for more information.

SAMEDISC (header)

Also-Known-As Discipline; used to equivalence disciplines in a rule.

```
##FRT100 and FRN100 count as the FRE discipline.
##SPT100 counts as the SPA discipline

BEGIN
36 Credits
MaxPerDisc 2 Classes (FRE, SPA)
SameDisc (FRT=FRE, FRN=FRE, SPT=SPA)
;

##If a student takes 5 courses in HE and 3 in PE,
##then 8 courses count toward MaxPerDisc discipline of PE.

BEGIN
120 Credits
MaxPerDisc 8 Credits (PE)
SameDisc (HE=PE)
;
```

Notes:

1.	**SameDisc** sets an equivalence between two disciplines. The equivalence is used only when evaluating **MaxPerDisc** and **MinPerDisc**, not when matching courses taken by the student to required courses. Typically, **SameDisc** is used to equate obsolete disciplines with current disciplines.
2.	**SameDisc** is followed by a left parenthesis, followed by a valid discipline code from the institution's code list, followed by an equals sign, followed by a second discipline from the institution's code list, optionally followed by additional disc=disc combinations separated by a comma, followed by a right parenthesis.
3.	The second discipline must appear in the discipline list associated with **MaxPerDisc** or **MinPerDisc**.
4.	If the configuration setting for maximum discipline length (UCX-CFG020 DAP13 Disc_length) is zero, then **SameDisc** is invalid.

SAMEDISC (rule)

Also-Known-As Discipline; used to equivalence disciplines in a rule.

```
##Counts FRT and SPT as the ENG discipline.  A student who took FRT100
##and SPT100 has not satisfied the MinSpread requirement for 2 disciplines
##because FRT and SPT both count as the ENG discipline.

11 Credits in ENG @, FRT @, GER @, PHL @,
              REL @, SPA @, SPT @
    MinSpread 2
    SameDisc(FRT=ENG,SPT=ENG)
  Label "General Studies";
```

Notes:

1.	**SameDisc** sets an equivalence between two disciplines. The equivalence is used when evaluating **MaxPerDisc**, **MaxSpread**, **MinPerDisc**, and **MinSpread**, not when matching courses taken by the student to required courses. Typically, **SameDisc** is used to equate

	obsolete disciplines with current disciplines.
2.	**SameDisc** is followed by a left parenthesis, followed by a valid discipline code from the institution's code list, followed by an equals sign, followed by a second discipline from the institution's code list, optionally followed by additional disc=disc combinations separated by a comma, followed by a right parenthesis.
3.	The second discipline must appear in the course list associated with **MaxSpread**/**MinSpread** or the discipline list associated with **MaxPerDisc**/**MinPerDisc**.
4.	If the configuration setting for maximum discipline length (UCX-CFG020 DAP13 Disc_length) is zero, then **SameDisc** is invalid.

SCHOOL (header)

Indicates a school type of block (SCHOOL) against which **NonExclusive** is applied.

```
BEGIN
36 Credits
Share 9 Credits (SCHOOL)
;

BEGIN
36 Credits
Share (SCHOOL)
;
```

Notes:

1.	**SCHOOL** is used within a block header, only with **NonExclusive**, **Share**, or **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can also be used to satisfy requirements in the SCHOOL block (if the SCHOOL block exists for the student).

SCHOOL (rule)

Indicates a school requirement block (SCHOOL), which, in some cases, may precede a school code.

```
1 Block (SCHOOL=GR)
  Label "Graduate Studies";

1 BlockType (SCHOOL)
  Label "School requirements";

If (SCHOOL=PA) Then
 1 Class in PHYS 100
   Label "Physics";

6 Credits in BUS 1@
   ShareWith (SCHOOL)
  Label "Business";
```

Notes:

1.	**SCHOOL** is followed by a school code, such as GR for Graduate School, except in a **BlockType** rule.
2.	The school code must be valid in the institution's code list.
3.	**SCHOOL** can be used as part of a condition following **If**.

4.	**SCHOOL** can be used with **Block**[s], **BlockTypes**[s], **If**, or **NonExclusive**, **Share**, or **ShareWith**.

SHAREWITH (header)

Indicates that credits/classes can fulfill multiple requirements.

```
## 10 Credits from this block can also be applied to the minor block or the
## major block
Share 10 CREDITS (MINOR, MAJOR) # one of these blocks – but not both

## all credits from this block can also be applied to any other block;
## however, a class ## that is applied here can only be applied to one other
## block – not to all of the blocks
ShareWith (ALLBLOCKS)

## all credits from this block can be applied to the general education block
## and to multiple rules within this block
ShareWith(OTHER=GENED, THISBLOCK)

##all credits from this block can also be applied to any English or
## Literature Major block
ShareWith (MAJOR=ENGL, MAJOR=LIT) # one of these blocks – but not both

##all credits from can also be applied to both English and Literature Major
## block
ShareWith (MAJOR=ENGL) # share with this block
ShareWith (MAJOR=LIT)  # and also share with this block

## all credits from this block can also be applied to any Undergraduate
## College block
ShareWith (COLLEGE=U@)

## all credits from this block can shared with any major except CHEM
ShareWith (MAJOR, MAJOR<>CHEM) # share with any major except for CHEM

## all credits from this block can shared with any block except HIST major
## block
ShareWith (AllBlocks, MAJOR<>HIST) # share with any block except for HIST
major

# Share with the student's first major
ShareWith (MAJOR=1st)

# Share with major that is associated with this conc block
ShareWith (MAJOR=associated)
# Share with concentration that is associated with this major block
ShareWith (CONC=associated)
# Share with any concentration except the one that is associated with this
## major block
ShareWith (CONC, CONC<>associated)
```

Notes:

1.	**Share** or **ShareWith** indicates that the credits or classes applied by the Auditor Engine towards satisfying the requirements in this block can also be used to satisfy requirements in other blocks. Any single class will share with only one of the blocks listed – not all of the blocks to which they may apply. If you have "Share 5 Classes (Major, Minor)" it is possible to have 3 classes shared with the Major and 2 with the Minor or the auditor might share 1 class with the Major and 4 with the Minor – but any single class will only be shared with one of those blocks – not both. If you do want to share with both the Major and Minor you can simply use two separate qualifiers: Share 5 Classes (Major) Share 5 Classes (Minor) However, there is no guarantee that the same 5 classes will be shared between all three blocks. It is possible 5 classes will be shared with the Major and a different 5 classes will be shared with the Minor.
2.	**Share** or **ShareWith** is needed as part of the rule only if the block has not been declared **Share** or **ShareWith**.
3.	**Share** or **ShareWith** is optionally followed by either a real number and **Credit**[s] or an integer and **Class**[es] to indicate how many credits or classes the Auditor Engine can apply to this block as well as to other blocks. A scope or scope list is required to indicate the blocks in which the courses can be applied non-exclusively.
4.	When specified, the number of credits or classes must be less than or equal to the number of credits or classes specified before **Share** or **ShareWith**. Subtract the number following **Share** or **ShareWith** from the number before **Share** or **ShareWith** to get the number of credits or classes that will be Exclusive. If the number of credits or classes is not specified, then all credits and classes in the block or rule can be applied here and in subsequent rules and blocks.
5.	**Share** or **ShareWith** is followed by a block type (COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, OTHER, PROGRAM, SCHOOL, SPEC) or scope (ALLBLOCKS, THISBLOCK) to indicate which blocks can share the credits or classes. Any of the keywords used as block type can be used here, except **ID**. If **OTHER** is used, then a custom_block code must be specified, e.g., "**Share** 6 Credits (OTHER=GENED)". A value can also be used with block types other than OTHER, e.g., "**ShareWith** (MAJOR=ENGL, MAJOR=LIT)". Values also support wildcards (@).

6.	To treat some classes/credits as exclusive and the rest as nonexclusive, use **Share** or **ShareWith** as a block qualifier and use **DontShare** with the number of classes or credits as a rule qualifier.
	Examples:
	`BEGIN ##!exclusive block with nonexclusive rule`
	`36 Credits`
	`;`
	`3 Classes in ENG 115, SOC 120, PSY 110`
	`Share 6 Credits (MAJOR, MINOR, OTHER=GENED)`
	`Label "Humanities Course Requirement";`
	`END.`
	
	`BEGIN ##nonexclusive block with exclusive rule`
	`36 Credits`
	`ShareWith (MAJOR, MINOR)`
	`;`
	
	`3 Classes in ENG 115, SOC 120, PSY 110`
	`DontShare 3 CREDITS;`
	`END.`

7.	Do not use **Share** or **ShareWith** as a block qualifier and use **Share** or **ShareWith** with the number of classes or credits as a rule qualifier.
8.	Using **NoCount** and **Share** or **ShareWith** together could yield unpredictable results. Using **SpMaxCredits** and **Share** or **ShareWith** together could also yield unpredictable results.

9.	Also refer to the details about the *StandAloneBlock*. It allows sharing in a more global way and may better suit your business requirement.
10	The not-equals (<>) block specification has precedence in that it overrides any other sharing specified on the qualifier.
11.	**1st, 2nd, 3rd, 4th, etc** You can use an ordinal value to specify that sharing should only occur with the 1st, 2nd, 3rd, etc block of a certain type. For example, ShareWith (Major=1st) will only allow sharing with the first major. Conversely, ShareWith(Major, Major<>1st) will allow sharing with all majors except for the first one. The auditor determines which is the 1st, 2nd, etc block of the specific type based on the order of the blocks that were passed to the auditor – and this is related to the sequence number on the rad_goalData_dtl. Caveat: if the student has multiple majors and multiple concentrations within each major specifying ShareWith(Conc=1st) may not result in the concentration you intended. The ordinal specification is not relative to the block doing the sharing. That is, the auditor does not find the 1st concentration for that given major; the auditor simply finds the first concentration in its list of concentrations. If this is an issue for you then you might consider using ASSOCIATED instead.
12.	**ASSOCIATED**: Instead of specifying a specific block you can use ASSOCIATED to let the auditor figure out which specific block is associated with the parent block. ShareWith(Major=**associated**) can be used to find the major that is associated with this concentration block (or any other type of block but usually it is majors and concentrations that are associated). The reverse can also be used: ShareWith(Conc=**associated**) would be placed in the major block. The auditor will attempt to find the associated block in two ways: First, the auditor examines the secondary tags of both the major and the conc blocks to see if one is tied to the other. Second, if the secondary tags don't lead to an association then the auditor examines the ATTACH values on the rad_goalData_dtl records. If the auditor finds a CONC record that has an ATTACH code for the parent major block with the ShareWith qualifier then the auditor has found an association. **Note:** If two concentrations are associated with the same major and the major has ShareWith (conc=**associated**) then the major will only share with the first concentration. However, if the concentration blocks instead have ShareWith(major=**associated**) then both concentrations will share with the major. As mentioned previously, associations can go from any block type to any other block type but most often they are setup between the concentration and major. You can always use this as a way to prevent sharing with the associated block: ShareWith (Conc, Conc<>**associated**). This will allow sharing with any conc block except for the associated one. When you think ASSOCIATED you can also think ATTACHED because of the field names on the rad_goalData_dtl.

SHAREWITH (rule)

Indicates that credits/classes can fulfill multiple requirements.

```
##6 credits from this rule can also be applied to the MAJOR block and
##3 of those credits can also be applied to the MINOR block
6 Credits in INS1 @
   Share 6 Credits (MAJOR)
   Share 3 Credits (MINOR)
  Label "International Studies";

##courses from this rule can also be applied to the MINOR and/or CONC block
3 Classes in HIST @
   Share 2 Classes (MINOR, CONC)
  Label "History";

##all courses from this rule can also be applied to other rules in this
block
12 Credits in ENG 100:199, PHI 100:199
    ShareWith (THISBLOCK)
   Label "English and Philosophy";

##courses from this rule can also be applied to English and/or Literature
MAJOR blocks
3 Classes in HIST @
   Share 2 Classes (MAJOR=ENGL, MAJOR=LIT)
  Label "History";

##courses from this rule can also be applied to rules in any Undergraduate
COLLEGE block
12 Credits in ENG 100:199, PHI 100:199
    ShareWith (COLLEGE=U@)
   Label "English and Philosophy";

## courses from this rule can also be applied any major except for LIT
3 Classes in HIST @
   Share 2 Classes (MAJOR, MAJOR<>LIT)
  Label "History";

# Share with the student's first major
3 Classes in HIST @
   ShareWith (MAJOR=1st)
  Label "History";

# Share with major that is associated with this conc block
3 Classes in HIST @
ShareWith (MAJOR=associated)
  Label "History";

# Share with concentration that is associated with this major block
3 Classes in HIST @
ShareWith (CONC=associated)
  Label "History";

# Share with any concentration except the one that is associated with this
major block
3 Classes in HIST @
ShareWith (CONC, CONC<>associated)
  Label "History";
```

Notes:

1.	**Share** or **ShareWith** indicates that the credits or classes applied towards satisfying this rule can be used to satisfy requirements in other blocks or in subsequent rules in this block.
2.	**Share** or **ShareWith** is needed as part of the rule statement only if the block has not been declared **Nonexclusive**.
3.	**Share** or **ShareWith** must follow a course list.
4.	**Share** or **ShareWith** is optionally followed by either a real number and **Credit**[s] or an integer and **Class**[es] to indicate how many credits or classes the Auditor Engine can apply to this rule as well as to subsequent rules.
5.	When specified, the number of credits or classes must be less than or equal to the number of credits or classes specified before **Share** or **ShareWith**. Subtract the number following **Share** or **ShareWith** from the number before **Share** or **ShareWith** to get the number of credits or classes that will be exclusive. If the number of credits or classes is not specified, then all credits and classes in the block or rule can be applied here and in subsequent rules and blocks.
6.	**Share** or **ShareWith** is followed by a block type (COLLEGE, CONC, DEGREE, LIBL, MAJOR, MINOR, OTHER, PROGRAM, SCHOOL, SPEC) or scope (ALLBLOCKS, THISBLOCK) to indicate which blocks can share the credits or classes. Any of the keywords used as block type can be used here, except **ID**. If **OTHER** is used, then a custom_block code must be specified, e.g., "**Share** 6 Credits (OTHER=GENED)". A value can also be used with block types other than OTHER, e.g., "**ShareWith** (MAJOR=ENGL, MAJOR=LIT)". Values also support wildcards (@).
7.	**DontShare** cannot be used in the same rule as **Share** or **ShareWith**.
8.	Disallow duplicates of block types if **Share** or **ShareWith** is repeated in a rule. For example: "**Share** 6 **Credits** (MAJOR)" and "**Share** 12 **Credits** (MAJOR)" on the same rule is not allowed.

9.	To treat some classes/credits as exclusive and the rest as nonexclusive, use **Share** or **ShareWith** as a block qualifier and use **DontShare** with the number of classes or credits as a rule qualifier. Examples:
	`BEGIN ##exclusive block with nonexclusive rule`
	`36 Credits`
	`;`
	`3 Classes in ENG 115, SOC 120, PSY 110`
	`  Share 6 Credits (PROGRAM)`
	`  Label "Humanities Requirement";`
	`END.`
	
	`BEGIN  ##!nonexclusive block with exclusive rule`
	`36 Credits`
	`ShareWith (ALLBLOCKS)`
	`;`
	`3 Classes in ENG 115, SOC 120, PSY 110`
	`  DontShare 3 Credits`
	`  Label "Humanities Requirements";`
	`END.`

10.	Do not use **Share** or **ShareWith** as a block qualifier and use **Share** or **ShareWith** with the number of classes or credits as a rule qualifier.
11.	Using **NoCount** and **Share** or **ShareWith** together could yield unpredictable results.
12.	Do not use **Share** or **ShareWith** with **SpMaxCredits** or **SpMaxTerm**.
13.	**1st, 2nd, 3rd, 4th, etc** See the notes on this option under SHAREWITH (header)
14.	**ASSOCIATED** See the notes on this option under SHAREWITH (header)

SPEC (header)

Indicates a specialization type of block (SPEC) against which **ShareWith** is applied.

```
BEGIN
36 Credits
Share 9 Credits (SPEC, CONC)
;

BEGIN
36 Credits
ShareWith (SPEC)
;
```

Notes:

1.	**SPEC** is used within a block header, only with **NonExclusive**, **Share**, **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can also be used to satisfy requirements in the SPEC block (if the SPEC block exists for the student).

SPEC (rule)

Indicates a specialization requirement block (SPEC), which, in some cases, may precede a specialization code.

```
1 Block (SPEC=BIOM)
  Label "Biomechanics";

1 BlockType (SPEC)
  Label "Specialization required";
```

Notes:

1.	**SPEC** is followed by a specialization code, such as BIOM for Biomechanics, except in a **BlockType** rule.
2.	The specialization code must be valid in the institution's code list.
3.	**SPEC** can be used as part of a condition following **If**.
4.	**SPEC** can be used with **Block**[s], **BlockType**[s], **If**, **NonExclusive**, **Share**, or **ShareWith**.

SPMAXCREDIT[S] (header)

Indicates a maximum number of credits after which the excess credits should be split across blocks.

```
BEGIN
120 Credits
SpMaxCredits 24 in MUS 1@
;

BEGIN
12 Classes
SpMaxCredits 9 from PSY 100:199
;

##maximum of 8 credits in MUS @ or 8 credits in ART @, or both summed
BEGIN
SpMaxCredits 8 in MUS @, ART @;

##maximum of 8 credits in MUS @ and 8 credits in ART @
BEGIN
SpMaxCredits 8 in MUS @
SpMaxCredits 8 in ART @
;
```

Notes:

1.	**SpMaxCredit**[s] is followed by a real number that indicates the maximum number of credits that can be applied to the rule. The number is optionally followed by **in** or from, which must be followed by a course list.
2.	The connector in the course list following **SpMaxCredit**[s] can be a comma or **or**. The plus sign or **and** is not allowed. The logical **or** connector is not exclusive. If multiple courses are listed, then the sum of all credits that satisfy that list is compared against the number following **SpMaxCredit**[s]. Credits in excess of that number may cause courses to be split—some credits applied to this block and some applied to other blocks. If the block is the starting block, then the excess credits go to Over-The-Limit.
3.	**SpMaxCredit**[s] applies to all courses in the succeeding course list.

4.	The **SpMaxCredit**[s] qualifier is applied by the Auditor Engine against all the courses used to fill the block. The number of credits in **SpMaxCredit**[s] is a strict upper bound. For example, "**SpMaxCredit**[s] 2" cannot be exceeded even if the only course taken was for 3 credits. The excess credit will be applied to other blocks or to Over-The-Limit if this is the starting block.
5.	The course list associated with **SpMaxCredit** [s] allows **at** but not **Except** or **Including**.

SPMAXTERM (header)

Indicates a maximum number of credits per term, after which the excess credits should be split across blocks.

```
##maximum of 1 credit of Band per term, apply excess to other blocks
BEGIN
12 Credits
SpMaxTerm 1 Credit BAND @
;

##maximum of 6 credits in courses listed, sum of ME100:115 and PE100:115
BEGIN
SpMaxTerm 6 Credits in ME 100:115, PE 100:115
;
```

Notes:

1.	**SpMaxTerm** is followed by a real number followed by **Credit**[s]. The number indicates the maximum number of credits that can be taken per term. This number is a strict cap. If a course exceeds the specified number of credits, then the excess credits from that course can be applied to other blocks. If **SpMaxTerm** is in the starting block then the excess credits are applied to Over-The-Limit.
2.	**SpMaxTerm** is followed by the number of credits, optionally followed by **in** or **from**, followed by a course list in which comma/**or** is allowed but plus/**and** is not allowed.
3.	**SpMaxTerm** can only be used if all of a student's courses and transfer credits have been linked to a term.
4.	**SpMaxTerm** applies to all courses that match the succeeding course list and were used to fulfill any rule or rules in the block.
5.	The course list associated with **SpMaxTerm** allows **at** but not **Except** or **Including**.

STANDALONEBLOCK (header)

Denotes that classes in this block are applied without regard to classes applied in the other blocks of the audit.

```
BEGIN
StandAloneBlock
;
10 Credits in ENGL 101, 102, 105, 110, 121
Label "English Courses";

END.
```

Notes:

1.	The Auditor will place classes in this block and will ignore this block when determining how or if any sharing is done among the other blocks.
2.	Example: Major1 can share with Core and Major2 can share with Core. Major1 and Major2 cannot share in some colleges but can share in other colleges. Using **StandAloneBlock** in the general education block lets the major blocks deal with nonexclusive pointing to each other.

TAG (header)

A free-text value used to identify the header qualifier to help prevent unhooked exceptions.

```
MinCredits 12 in ENGL 2@, LIT @
  Tag=lit
  ProxyAdvice "You need at least 12 credits in literature"
MinCredits 6 in ENGL 24@, 25@
  Tag=writing
  ProxyAdvice "You need at least 6 credits in writing"
```

See the *Label Tags* section under *Special Topics*, and the notes below for more information about qualifier Tags

Notes:

1.	**Tag** is followed by a 20-character value.
2.	You can now use a tag on the qualifier to uniquely identify your qualifiers. The tag uniquely identifies the qualifier, which is essential if there is more than one of these types of qualifiers in the block header. This allows users to change the qualifier as much as needed and not cause exceptions to become unhooked.
3.	A Tag is allowed on any header qualifier except LowPriority, HighPriority, StandAloneBlock, CheckElectiveCreditsAllowed,
4.	The Tag needs to be placed before the ProxyAdvice that you may have on the qualifier.

TermCount

ResidenceTermCount

Used in an if-statement; checks the student's academic data.

```
If (TermCount > 2) THEN
 MinGPA 2.0

If (CompletedTermCount > 2) THEN
 MinGPA 2.0

If (ResidenceCompletedTermCount > 2) THEN
 MinGPA 2.0
```

Notes:

1.	Does not have to be noted in UCX-SCR002.
2.	TermCount and CompletedTermCount are synonyms. ResidenceTermCount and ResidenceCompletedTermCount are synonyms.
3.	The keyword checks how many terms the student has completed. TermCount counts the total residence and transfer terms while ResidenceTermCount only counts the terms completed at your school.
4.	The keyword checks how many terms the student has completed. TermCount counts the total residence and transfer terms while ResidenceTermCount only counts the terms completed at your school.

THEN (rule)

Part of a conditional rule to be executed when **If** condition is met.

```
If (MAJOR=HIST and CONC=EHST) Then
  6 Credits in GER @, GRK @, FRE @, SPA @,
    Label "Foreign Language"
Else
  3 Credits in LANG 100
    Label "Intro to Languages";

If (CONC=AHST OR CONC=EHST) Then
  2 Classes in POL 100:199
    Label "Political Science";

If (MINOR=EDU) Then
  If (MAJOR=BIO OR MAJOR=CHE) Then
    1 Block (OTHER=EDUS)
  Else
    1 Block (OTHER=EDU)
Else
  BeginSub
    3 Credits in EDU 300
       Label "Education";
    3 Credits in PSY 320
      Label "Psychology";
  EndSub
MaxPassFail 0 Credits
```

```
Label "Psychology Education Majors";
```

Notes:

1.	**Then** is followed by one of the seven types of rules. The rule must be valid according to its syntax. If there is an **Else** after the rule, then do not end the **Then** rule in a semicolon.
2.	The rule following **Then** is used only if the student fits the condition preceding **Then**.
3.	**Then** must be paired with a preceding **If**.
4.	**Then** is used only within an **If** statement.
5.	**Then** may be repeated in an **If** statement but there must be a corresponding **If** for each **Then**.

THISBLOCK (header)

Indicates the scope for the **NonExclusive**, **Share**, or **ShareWith** qualifier. This scope signifies that all rules within this block are to be considered for applying the specified number of credits or classes non-exclusively.

```
BEGIN
36 Credits
Share 10 Credits (THISBLOCK)
;

BEGIN
36 Credits
ShareWith (THISBLOCK)
;
```

Notes:

1.	**THISBLOCK** is used within a block header, only with **NonExclusive**, **Share**, or **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying the requirements of this block can be used to satisfy all rules within this block.

THISBLOCK (rule)

Indicates the scope for the **NonExclusive**, **Share**, or **ShareWith** qualifier. This scope signifies that all rules within this block are to be considered for applying the specified number of credits or classes non-exclusively.

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

Notes:

1.	**THISBLOCK** is used within a rule, only with **NonExclusive**, **Share**, or **ShareWith**, to indicate that the credits or classes applied by the Auditor Engine towards satisfying this rule can also be used to satisfy other rules within this block.

UNDER (header)

Specifies a limit on the number of credits or classes than can be used for the particular set of coursework. Unlike **MaxClasses**/**MaxCredits** the **Under** qualifier does not remove classes from the block or audit—it simply checks to see that the limit was not exceeded. The qualifier will be satisfied as long as the number specified is not exceeded. **Under** can be used in any block but is most often used in an **AWARD** block in a Financial Aid audit.

```
Under 10 Classes in ART @, MUS @
   ProxyAdvice "You have exceeded the 10 class limit."
   Label "Only 10 art and music classes are allowed."

Under 30 Credits in @ (With Attribute=DEV)
   Except PE @, MUS @
   ProxyAdvice "You have exceeded the 30 credit limit."
   Label "Only 30 developmental credits are allowed."
```

Notes:

1.	**Under** is followed by a real number and the **Credits** keyword, or an integer and the **Classes** keyword.

WITH (header)

Additional “custom” class specifiers in a course list. The **WITH** keyword is particularly useful when course content is determined by the course key and additional data, such as section.

```
##maximum of 3 courses in MUS100 with a grade at or below a D
MaxClasses 3 in MUS100 (WITH DWGradeNumber <= "1.0");

##maximum of 3 credits in Humanities with User-Def3=AB@ or XYZ
##MYCODE is equated to RIM CLASS-HIST-DTL RIM-USER-DEF3 in UCX-SCR044
MaxCredits 3 in HUM @ (WITH MYCODE="AB@", XYZ), ENGL 323;

##In this example only those HIST 229 classes taken before term 1999A
##can be used in the MAXCREDITS calculation
MaxCredits 15 in HIST 213, 214, 224, {HIDE HIST 229 (WITH DWTERM < 1999A),}
HIST 235;

##In this example only the ACCT 4700 will have residency enforced
MinCredits 12 in ACCT 31000, 3200, 3300, 4350, 4600, 4700 (WITH
DWResident=Y)

# Look up ADMITTERM on the student's custom-dtl; the "STUDENT-" prefix
indicates
# that this is a variable name - not an actual value
MaxClasses 0 in @(WITH DWTerm < STUDENT-AdmitTerm)
```

Notes:

1.	**WITH** offers additional specification of a particular course by qualifying the preceding course with “custom” data from the student system’s class. The “custom” data is defined in UCX-SCR044 and can reference any piece of data in the class record.
2.	The token following **WITH** must either be defined in UCX-SCR044, WITH Custom Class Data, or be a DW named field. After the UCX-SCR044 custom data, an operator must exist, followed by a valid value for that custom data item. The value can contain a wildcard (@) to signify zero or more occurrences of any character. If the UCX table in UCX-SCR044 record is filled in, then the value must be valid in that UCX table unless the value contains a wildcard (@).
3.	The entire **WITH** expression must be enclosed in parentheses, including **WITH**.
4.	Quotation marks are required around the code to the right of the equals sign if a non-alphanumeric character is used.
5.	If a list of values following **WITH** are needed, then separate the values with a comma, for example, ENG100 (**WITH DWSection**=01, 02). Doing this indicates that the class must contain one of the values in the list.
6.	Only the last course before **WITH** is qualified by the **WITH** custom data. For example: “ENG 101, 110 (**WITH MYCODE**=“XX”)” qualifies only ENG 110 as requiring **MYCODE**=“XX”.
7.	For ranges of course numbers, **WITH** applies to all courses in the range. For example, “ART 100:102 (**WITH DWSection**=“G@”)” is the same as “ART 100 (**WITH DWSection**=“G@”), ART 101 (**WITH DWSection**=“G@”), ART 102 (**WITH DWSection**=“G@”)”.
8.	**WITH** is only allowed after a course. See **Class**[es] for a definition and examples of courses.
9.	**WITH** is optional and should be used only if additional class specification is needed.
10.	The following DW named fields can be used instead of using UCX-SCR044: **DWAge**, **DWCredits**, **DWCreditType**, **DWCourseNumber, DWDiscipline, DWGradeNumber**, **DWGradeLetter**, **DWGradeType**, **DWLocation**, **DWPassFail**, **DWResident**, **DWSchool**, **DWSection**, **DWTerm**, **DWTitle**, **DWTransfer**, **DWTransferCourse**, **DWTransferSchool**, **DWInprogress**, **DWPreregistered**, **DWTermType**, **DWPassed**

11.	DWResident will have a value of **Y** if the class is taken in residence and a value of **N** if the class is a transfer class. DWTransfer will have a value of **Y** if the class is a transfer class and a value of **N** if the class was taken in residence.
12.	A wildcard may be needed if the length of the value specified is not the same as length of the field being referenced: (**WITH DWGradeLetter**="B@"). Wildcards are not needed at end of title fields.
13.	The **WITH** operator supports the relational operators: - > (greater than), < (less than), >= (greater than or equal to), <= (less than or equal to), <> (not equal to), and = (equal to).
14.	You can string multiple **WITH** codes together. For example: "1 class in ENGL 1@ (**WITH** DWTerm=1822 **and** DWSection=XY, AB **and** DWResident=Y) Label "English class".
15.	You may not mix **and** with **or**, however. Doing this will cause a parser error. For example, "1 class in ENGL 1@ (**WITH** DWTerm=1822 **or** DWSection=XY, AB **and** DWResident=Y) Label "English class"" is invalid.
16.	**DWTerm** can be used with **Current** and **Previous** (**WITH** DWTerm=**Current**). This option is most often used in an AWARD block when performing a Financial Aid audit. **Current** is defined as the student's active term, while **Previous** is defined as the term previous to the active term for which the student took classes. **DWTerm** can also be used with **REGTERM** for prereq checking (With DWTerm = **REGTERM**). Scribing DWTerm = **REGTERM** will ensure that the course is a coreq; scribing DWTerm < **REGTERM** will ensure that the course is a true prereq and not a coreq.
17.	**DWInprogress** is only "Y" when the class's term value matches the active-term on the rad_student_mst and when the Inprogress flag on the rad_class_dtl is "Y".
18.	**DWPreregistered** is only "Y" when the class's term value is greater than the active-term on the rad_student_mst and when the Inprogress flag on the rad_class_dtl is "Y".
19.	**DWTransferCourse** uses whatever is bridged to the rad_transfer_dtl.rad_tr_crse_key field. If "MATH101" is in this field then you need to use "WITH DWTransferCourse="MATH101". If "MATH 101" (with a space) is in this field then you need to use "WITH DWTransferCourse="MATH 101" – be sure to use whatever spaces are in the actual value bridged to this field. The Banner extract pulls over the transfer course without any space between the discipline and number so Banner schools should follow the first example here.
20.	When using **DWTransferSchool**, you can either use the school's full name, for example, University of Learning, or you can use the school's ETS/FICE ID code. As school names can change, for example from 'College' to 'University', and because of inconsistencies in how users have enter the school names into your ERP (for example, Univ. vs U. vs University) you should consider using the school's ID code instead of the school name in your WITH qualifiers. Both of these are supported: WITH DWTransferSchool="University of Learning" and WITH DWTransferSchool="123456". If you use the school ID it is best to put a comment in your scribe block noting the school's name.
21.	You can place a **STUDENT**- prefix in front of the WITH value on the right of the operator to tell the auditor to lookup the specified name on the rad-custom-dtl and use the value found. Example, WITH DWTerm > STUDENT-AdmitTerm - the auditor would lookup ADMITTERM on the rad-custom-dtl and use the value associated with that record. Note – AUDITTERM must be defined in UCX-SCR002 also. Currently STUDENT- only works with DWTerm.

22.	**DWAge** looks at the age of the class in years by looking at the class's term value and subtracting it from the current/active term for the student. A term value of 200510 is converted to a decimal value of 2005.10 and then subtracted from the active term – which might be 201220 – which is converted to a decimal value of 2012.20. Subtracting the two gives 2012.20 – 2005.10 = 7.1 years old. If the rule specifies DWAge < 10 then this age of 7.1 years will be within the age limit and would apply to the rule. Three different formats of terms are supported with lengths 6, 5 and 4: 6 characters: 200510 – the first four characters are the year with the last two characters being the term within the year; 5 characters: 20051 – the first four characters are the year with the last character being the term within the year; 4 characters: 1051 - the first three characters are the year with the last character being the term within the year (here the leading "1" means 21st century with terms prior to the year 2000 having a leading "0" instead – ex: 0991). If your terms have some other format you may not be able to use DWAge. Planned classes have a term value of PLANNED normally. For the DWAge calculation these planned classes are given an age of zero months/years.
23.	**DWGrade**, **DWGradeNumber**, and **DWGradeNum** are all synonyms. However, when you use DWGradeNum or DWGradeNumber you must specify a numeric grade. For example, DWGradeNumber > 2.5. When you use DWGrade you can specify a numeric grade or a letter grade. However, if a letter grade is specified the grade will be looked up on UCX-STU385 and will be translated to a numeric grade. For example, DWGrade > C will result in meaning DWGrade > 2.0. You may have the need to use DWGradeLetter (or DWLetterGrade) for special cases where multiple letter grades have the same value. This usually happens with grades of Incomplete, Withdrawn, etc where they all have a grade of 0.0. In a header qualifier you may want to do something like this: MaxClasses 0 in @ (With DWGradeLetter = I). When the letter grade is specified like this no lookup on UCX-STU385 is performed to get the numeric equivalent of the letter. For this reason you should not use less-than or greater-than with DWGradeLetter.
24.	**DWDiscipline** and **DWCourseNumber** can be used for special situations such as when you want to use a wildcard but want to exclude a special discipline. For example: MaxCredits 5 in @ (With DWPassfail=Y and **DWDiscipline**<>ANTH). Here you are putting a max on the number of passfail credits but are allowing an unlimited number of ANTH passfail credits; that is, the ANTH classes are being excluded. You will not be using DWDiscipline and DWCourseNumber often, but they can come in handy in special situations. **Note:** When an equivalence is in place, it is the new discipline and number that is being used and not the original values. For example, if SOC 123 was renamed to ANTH 145, it will be ANTH and 145 that will be used in DWDiscipline and DWCourseNumber and not SOC or 123.

WITH (rule)

Additional “custom” class specifiers in a course list. The **WITH** keyword is particularly useful when course content is determined by the course key and additional data, such as section.

```
## ENG 101 with section 01
1 Class in ENG 100, 101 (WITH DWSection=01); # WITH only applies to 101 –
not to 100

1 Class in ENG 101 (WITH MYCODE=ABC),
           ENG 110 (WITH MYCODE=XYZ)
  Label “English”;

3 Credits in MATH 250 (WITH DWGradeNumber >= “2.0”), 255
  Label “Math”;


15 Credits in HIST 213(WITH DWTerm < 200920 and DWGrade > C)
  Label “History”;

1 Class in @ (With Attribute=HONR or Attribute=SPEC)
  Label “Honor class required”;

# Hide the WITH information from the worksheet advice
5 Credits in HIST 184 (WITH Hide DWResident = Y)
  Label “History Requirement”;


# Look up ADMITTERM on the student’s custom-dtl; the “STUDENT-“ prefix
indicates
# that this is a variable name – not an actual value
3 Credits in CHEM 244 (WITH DWTerm > STUDENT-AdmitTerm)
  Label “Chemistry Requirement”;
```

Notes:

1.	See special notes above in **WITH (header)**.

Creating and Saving a Block: An Exercise

Create a scribe block for the following certificate program. Use subsets to gather both program core and elective requirements. (Hint: the writing skills requirement will require a **Group** rule; the independent study restriction on the electives can be handled with a header qualifier.)

Certificate in Fine Arts

Writing Skills (5-7 Credits)
Students may take either English (ENGL) 101 or English (ENGL) 100 and Business PCs (BUSE) 100

Program Core (15 Credits)

ART 212	History of Art-Ancient	5 credits
MUS 111	Music Theory I	5 credits
MATH 107	Math: A Practical Art	5 credits

Electives (12 Credits)
Students must take a minimum of 4 credits worth of independent study (299) as part of their electives.

Electives must be distributed as follows:

4 credits in Art
4 credits in Music
4 credits in Math

One Solution to the Exercise

Here is one solution to the exercise. There is more than one way to scribe these requirements.

```
BEGIN

32:34 Credits
MinCredits 4 in MATH 299, ART 299, MUS 299 #Independent Study Requirement

;

1 Group in
   (1 Class in ENGL 101
       Label "Freshman Composition") OR
   (2 Classes in ENGL 100 and BUSE 100
       Label "Business English and Word Processing")
 LABEL "WRITING SKILLS";

BeginSub
   1 Class in ART 212
      Label "History of Art - Ancient";
   1 Class in MUS 111
      Label "Music Theory I";
   1 Class in MATH 107
      Label "Math - A Practical Art";
EndSub
  LABEL "PROGRAM CORE";

BeginSub
   4 Credits in ART @
      Label "Art Elective";
   4 Credits in MUS @
      Label "Music Elective";
   4 Credits in MATH @
      Label "Math Elective";
EndSub
  LABEL "ELECTIVES";

REMARK "Electives must include at least 4 credits in independent "
REMARK "study classes - ART 299, MUS 299 and/or MATH 299.";

END.
```