Degree Works
Scribe
Administration Guide

Release 5.0.3.1
March 2020

Notices

Contents

Introduction

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

This Administration Guide provides instructions to setup, configure and debug the Scribe application.

Initial Set-up and Configuration

This section describes the steps to configure and use Scribe for your site.

Shepherd Settings

Most configurations to manage the behavior of Scribe can be maintained using the **Configuration** tab of Controller. Review all entries with the 'scribe' prefix and ensure that they are configured correctly for your environment. See the *Degree Works Configuration Technical Guide* for more information. To make any setting specific to just Scribe, use the spec value "scribe".

The following settings should be configured before you deploy :

```
classicConnector.dap08.port
classicConnector.serverNameOrIp
scribe.*
```

Deploying the Scribe User Interface

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

Prerequisites

Java

Java version 8 or above is the only third-party dependency required to run this application. Oracle Java and OpenJDK are supported.

SSL

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

Environment Settings

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

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

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

Required Environment Variables

- **DW_DATASOURCE_URL**: A JDBC connection string for the Degree Works database. For example: `jdbc:oracle:thin:@{SERVER}:{PORT}/{SID}` where `{SERVER}` is the address, `{PORT}` is the port of the listener, and `{SID}` is the SID of the database. This is the same database and schema used by the classic Degree Works software.
- **DW_DATASOURCE_USERNAME**: The Degree Works database username.
- **DW_DATASOURCE_PASSWORD**: The Degree Works database password. This can be encoded. To get the encoded value, use the `showdbpasswords --password` command in the classic environment. It will display the encoded value to place here (e.g. `ENC(skdfjldjs)`).
- **SERVER_PORT**: The desired port for Scribe. For example, "443". If using 443, then your links or bookmarks to the application don't need to specify a port in the URL because it is the standard port browsers will assume for URLs that begin with https://.
- **SERVER_CONTEXT_PATH**: An optional context path. If this is blank, the application will be deployed at the root URL like https://Scribe.myserver.tld. If a value is specified like /my/context/path/, then the application would be accessible at a URL like https://Scribe.myserver.tld/my/context/path/.
- **SERVER_SSL_KEY_STORE**: The path to a keystore file you created to contain your SSL certificate.

- **SERVER_SSL_KEY_STORE_PASSWORD**: The password for your keystore file.
- **SERVER_SSL_KEYSTORETYPE**: The type of your keystore. JKS is the default.
- **SERVER_SSL_KEY_ALIAS**: The alias of the key you created.

Optional Environment Variables

- **JAVA_OPTIONS:** The memory allocation for the application can be defined using this variable, and is recommended. The value defines the initial heap size and max heap size and is in the following format: `-Xms1536m -Xmx1536m`. The heap size values should be adjusted as appropriate for your server.
- **SERVER_TOMCAT_REDIRECT_CONTEXT_ROOT:** Embedded Tomcat for Spring Boot has a default behavior to redirect a request without a trailing slash to one with a trailing slash. However, that redirect happens without evaluating forward headers like X-Forwarded-Proto. So, when SSL offloading is in place, the redirect happens to http instead of https. This can be an issue especially for load balanced environments. To disable the default behavior, set this variable to “false”.
- **DEBUG**: A boolean "true" or "false" which will enable or disable debug logging. The log file is by default named Scribe.log. But, the location and name of that file can be customized by setting the LOGGING_PATH and LOGGING_FILE environment variables.
- **DEPLOY_LOCATION**: the location of the ScribeUI.jar file. This location should be set then referenced in the “java -jar” command that starts the application.
- **LOGGING_FILE**: the name of the file including the path location. A date is appended to this name to produce the actual file name (see LOG_DATE_PATTERN below). If this is specified, LOG_PATH will not be used.
- **LOG_PATH**: the location of the log file. The default PATH location is the Java temporary java folder from the Java property *java.io.tmpdir*. This is usually `/tmp`. The file name will consist of this value plus “scribe.log” plus a date (see LOG_DATE_PATTERN below).
- **LOG_DATE_PATTERN**: the date pattern to use as the suffix for the file. This is used to ensure that the file does not get too big. It will create a new file when the date pattern would normally produce a different value. The date patterns follow the Java SimpleDateFormat. See https://docs.oracle.com/javase/8/docs/api/java/text/SimpleDateFormat.html for details. The default is yyyy-MM-dd (e.g. 2020-01-31). This will create a new file each day. For a monthly cycle, you may use yyyy-MM (e.g. 2020-01), and for a weekly cycle use yyyy-ww (e.g 2018-01).

Database Pool Configuration for Microservice Applications

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

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

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

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

```
export DW_DATAPOOL_MAX_ACTIVE=100
```

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

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

Deployment

Once the prerequisites and configuration are in place, start up the application using a "java -jar" command like this one:

```
        java $JAVA_OPTIONS -jar ScribeUI.jar
```

UCX Tables

Several existing UCX tables are used for Block Values and Tags. These should be reviewed to ensure they contain the necessary entries and the descriptions are clear and accurate. After updating one of these UCX tables, refresh the page in the browser to load the changes – it is not necessary to restart the application.

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

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

Granting Users Access to Scribe

Several standard Shepherd groups have been delivered to help you organize the Shepherd keys needed for Scribe users. Add these groups on an individual user basis using Controller or add the groups to a userclass in SHPCFG. See the *Security* section in the *Degree Works Technical Guide* for more information.

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

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

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

Scribe Database Table Structure

Brief descriptions of the tables used in Scribe are provided in the sub-sections that follow. For additional information, see the *Scribe* section in the *Advanced Reporting Technical Guide*.

DAP_REQ_BLOCK

Scribe stores all requirement block data in the dap_req_block. The dap_req_block table contains the block details such as title, type and tags as well as the block text in a CLOB field. (CLOB is Character Large Object – a database field that can store a lot of text.)

Managing Scribe Blocks

To transfer blocks from one environment to another, use the *dapblockunload* and *dapblockload* scripts. In order to load blocks into the new table you must specify the name of the tar file (created by *dapblockunload*) as the first parameter. See the *Transfer blocks between two different environments* topic in the *Maintaining Degree Works* section of the *Technical Guide* for more information.

To bulk insert blocks from files, use the *dapblockinsert* script. See the *Special Scripts* section of the *Technical Guide* for more information.

Scribe Navigation

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

Navigation Menu

Access the navigation menu by clicking the *scribe navigation menu* button on the left of the navigation bar. From the menu you can navigate to other pages in the application. **Welcome**, **New Block**, **Open Block (Search)**, **Open Local Block** and **Debug** (if the user has access to this functionality) are options on all pages.

Online Help

The online help for Scribe can be accessed by clicking on the *help* button on the right of the navigation bar. This on-line document is searchable, and the user can print the pages.

Logout

To log out of Scribe, click the **Sign Out** button on the right of the navigation bar.

Notification Center

The *Notification Center* displays success and error messages for Scribe operations. After taking an action such as searching for, parsing, or saving a block, the *Notification Center* will display the required message.

Searching for Blocks

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

To open a block by the Requirement ID, the user should enter the numeric value of the Requirement ID in the text box. The field will automatically left zero-fill. That is, if you type in "1234" you will see "001234" appear. The "RA" prefix will default and is applicable to most blocks. To search for a block created in the student planner, select "RB" from the drop down list. Clicking **Open** will take the user to the *Edit* page and if a valid Requirement ID was provided, the block will display in the editor. The user can also right-click on the button and use the browser options to open the block in the editor in a new tab or window. If an invalid Requirement ID is provided, the user will be shown an error message in the Notification Center.

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

A start and/or stop catalog year can also be provided when searching. Note that if a *Requisite Block Value* is selected, the period labels will be **Start Term** and **Stop Term** instead of **Start Catalog Year** and **Stop Catalog Year**.

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

The search may be limited to only those blocks with parsing errors by checking the **Only Blocks With Errors** checkbox. If the block has parse_status = "NO", it will be included in the search results.

To perform the search by criteria and open the *Search Results* page, click **Search for Blocks**. Clicking **Clear** will reset the selected criteria back to the initial values.

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

The *Secondary Tags* that display on the *Search* page can be configured. To remove a secondary tag from the display, set the corresponding `scribe.secondaryTag.show*` setting to “false”.

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

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

Search Results

The results of a block search by criteria will be returned in a grid on the Search Results page. The results are initially sorted by Block Title with a secondary sort on Start Catalog Year/Term. If no matching blocks are found based on the selection criteria provided, the user will be shown an error message in the *Notification Center*.

The total number of matching **Results** will be displayed on the top of the grid, and the number of results shown per page can be configured in `scribe.query.pageSize.blockSearchResults`. This is delivered initially as “20”, but can be adjusted as necessary. Increasing this value can impact performance. Depending on the total number of matching blocks, the user can navigate through the pages of results using the pagination bar.

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

Selecting a block in the results grid and clicking **Open Selected Block** will take the user to the *Edit* page. The user can also double-click on the row of the selected block to open it in the editor, or right-click on the **Title** and use the browser options to open the block in the editor in a new tab or window. The user can also Ctrl+click or Shift+click on the **Title** or on **Open Selected Block** to open the block in a new tab or window, respectively. The user can also right-click on **Open Selected Block** and choose the appropriate options.

A user can delete a block from the results grid by selecting the block and clicking **Delete Selected Block**. The user will be prompted to confirm the delete before the block is deleted.

Clicking **Back to Criteria** will take the user back to the *Search* page.

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

```
scribe.query.pageSize.blockSearchResults
scribe.secondaryTag.showCollege
scribe.secondaryTag.showConcentration
scribe.secondaryTag.showDegree
scribe.secondaryTag.showLiberalLearning
scribe.secondaryTag.showMajor1
scribe.secondaryTag.showMajor2
scribe.secondaryTag.showMinor
scribe.secondaryTag.showProgram
scribe.secondaryTag.showSchool
scribe.secondaryTag.showSpecialization
scribe.secondaryTag.showStudentId
```

Creating a New Block

A new requirement block can be created by clicking on **New Block** on the *Welcome* page or on the navigation menu, or by clicking the **New** button on the *Edit* page. A new block template will be opened in the editor. The user may also Ctrl+click or Shift+click on either of these buttons to open the new block in a new tab or window. The user may also right-click on either button and choose the appropriate options. This new block template can be localized in the `scribe.template.newBlock` setting in Controller. (You will want to press Ctrl+A followed by Ctrl-C in the Value field in Controller to copy the text and then paste it into a text editor. After editing the text and copying it, you can then do Ctrl-A and Ctrl-V to paste the text back into the Value field replacing what was there.

The user can define the *Block Details* by clicking on the **Edit** button on the **Details** tab. See the *Block Details* section in this document for additional information.

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

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

```
scribe.template.newBlock
```

Editing a Block

Block requirements and details can be modified on the *Edit* page. The user can update an existing block, create a new block, export the block to a file on the user's computer, print the block, and parse the block to check for syntax errors.

The block's **Title** will display with the **Requirement ID** above the editor as well as in the browser tab. If the block is new and has not yet been saved to the database, the title will display as "Untitled".

To edit requirement text, click on the **Editor** tab. To edit or view block information such as *Block Type* and *Value* or the *Parse Status*, click on the **Details** tab.

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

The **Details** tab shows the *Block Details* (the editable *Primary* and *Secondary Tags* for the block) and the *Access Details* (display-only create, modify and parse information). The user can define the *Block Details* by clicking on the **Edit** button on the Details tab. See the *Block Details* section in this document for additional information.

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

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

Ctrl-U	change to upper case – either the current word or the selected text
Ctrl-Up-arrow	scroll line up
Ctrl-V	paste from clipboard
Ctrl-X	cut selected text
Ctrl-Z	undo last change
Triple-click	Triple-click on a line to highlight the entire line allowing you to copy the text (using Ctrl-C or right-click plus Copy)
F11	Maximize browser window. Press F11 again to reset.
Windows-right-arrow	Puts the current window on the right half of the screen. Use in conjunction with Windows-left-arrow to get two windows lined up next to each other.
Windows-left-arrow	Puts the current window on the left half of the screen. Use in conjunction with Windows-right-arrow to get two windows lined up next to each other.

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

Parse

A block can be checked for parse errors before saving by clicking the **Parse** button. If no errors are found, a success message will display in the Notification Center. If errors are found, an error message will display in the Notification Center and the errors will be highlighted in the editor. An arrow will display at the beginning of each line that has a parse error. Hovering over this arrow will display a detailed error message. Additionally, the specific text in the line with the error will be underlined. All parse errors must be corrected before a block can be successfully saved.

There is no keyboard shortcut to execute the Parse button but you can navigate to the button without using the mouse. When the focus is in the editor press **Esc** and then press **Tab** twice – that puts focus on the **Parse** button. You may then press **Enter** to execute the **Parse**. After the parse status is returned you can then press Tab four times to get back to where you were in the editor and can continue editing the text.

Review the following UCX table to ensure that it is configured correctly for your Scribe environment:

- UCX-CFG020 DAP13 – all flags

Find

To find and replace text in a block, click on the **Find & Replace** button. The *Find/Replace* toolbar will be displayed.

Enter a string of text in the **Find Text** field and press the Enter key or click the down or up arrow to find the first instance of that string. The string search is case-insensitive – all matching instances of that string will be returned.

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

Open Local Block

A plain text file of requirements on the user’s computer can be opened in Scribe for editing and can then be saved to the database. Select **Open Local Block** from the navigation menu, click **Choose File** and navigate to the file you want to open. Click **Open** to load the file in the editor.

Block Details

Block details can be accessed from several points in Scribe: by clicking on the **Block Details** edit button on the **Details** tab of the *Edit* page, clicking **Save** for a new block, or by clicking **Save As**.

The **Title**, **Block Type**, **Block Value**, and **Start/Stop Catalog Year** are required fields. When a **Block Type** is selected, the **Block Value** drop down list will load with the values from the UCX validation table for that block type, unless the block type is *Other*, *Requisite*, or *Student ID*. For Other and Requisite Block Types, when the user starts typing in the field, a drop down list of matching blocks will be displayed and the user can select one of these values or can enter a new value. The `scribe.upshift.codes` setting controls if what the user has typed in the **Block Value** field should be upshifted. For a Student ID Bock Type, the **Block Value** is a text field and the user should enter a valid student ID.

A Start and/or Stop Catalog Year can also be provided. If a Requisite Block Value is selected, the period labels will be **Start** and **Stop Term** instead of **Catalog Year**.

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

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

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

Saving a Block

After creating or editing a block, the user can either update the existing block, add a new block to the database, or can save the block to a file on their computer.

When updating an existing block or saving a new block, before the block will be written to the database, it will be parsed for syntax and rule structure and checked to ensure it does not overlap catalog years or terms with other blocks. If any errors are encountered, these will be shown in the *Notification Center* and the block will not be saved. The errors need to be resolved before the block can be saved to the database.

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

Save

To save changes made to an existing block, users can click on the **Save** button. The user may be prompted to confirm that they wish to update the existing block if the `scribe.save.enable.confirmation` setting is 'true'. When this setting is 'false', no confirmation message will be displayed. If enabled, clicking **Yes** will save the block and clicking **Cancel** will take the user back to the editor.

When saving a new block, the block details edit window will be displayed after clicking **Save**. The user should review and fill in the block details and then click **Save.**

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

```
scribe.save.enable.confirmation
```

Save As

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

Save Local File

Requirement text can be saved as a file on the user's computer for archival purposes or to edit and add to the database at a later time. Click **Save Local File** – depending on your browser configurations, you will either be prompted for a location where the file is to be saved or the file will be saved in the default location configured in the browser. The requirement text for the block is saved in the form of a text (.txt) file. The default filename is a combination of the block type, value, requirement ID, and title. The filename will be ‘untitled’ if the block details have not yet been assigned.

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

Deleting a Block

A requirement block can be deleted by either clicking on **Delete Block** on the navigation menu when on the *Edit* page or by clicking the **Delete Selected Block** button on the *Search Results* page. A delete confirmation message will display – click **OK** to delete the block or **Cancel** to return to the page.

Printing a Block

To print a block’s requirement text, select **Print Block** from the navigation menu. The browser's print window will open, allowing the user to select the desired printer, preview the output, change the headers and footers, etc. The print output will show the block's Requirement ID and Title, and then all of the requirement text. Long lines of text will wrap in the printed output.

Localizing Scribe

Application Text, Labels, and Messages

All the text displayed in Scribe can be localized for a site by either modifying UCX tables or a properties file in the application. The *UCX Tables* section of this document describes which UCX tables are used in Scribe. Modifying code descriptions in these tables will alter the contents of drop down lists and the values shown in the *Search Results* grid.

For information on how to modify application properties, please see the “Localizing an Application” section of the *Composer Administrative Guide*.

Editor Colors

The colors appearing for the text in the Scribe editor is controlled by the *theme-scribe.js* file located next to the *WEB-INF* directory in the *Scribe.war* file. This is not the usual css file used to localize colors in Degree Works applications and special caution must be taken in changing the editor colors. Because of the complexity of this file and as the changes will affect all Scribe users, modifying the editor colors is not recommended.

Debugging Scribe

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

Access to Debugging

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

Enabling Debug

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

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

Accessing Debug Files

Debug will appear in the Scribe log file on the application server. This is typically `$CATALINA_HOME/logs` for Apache Tomcat or `$MIDDLEWARE_HOME/user_projects/domains/<domainname>/servers/<servername>/logs` for WebLogic.

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

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

The `packdebug` script can be used to collect the log files generated for a user session's Debug Tag. For more information on using this script, please refer to the *Degree Works Technical Guide*.