



EQUELLA® Metadata Utility

User Guide

Version 2.1.3

May 2015 edition.

Information in this document may change without notice. EQUELLA® and its accompanying documentation are furnished under a non-disclosure, evaluation agreement or licence agreement. Copying, storing, transmitting, or otherwise reproducing the software or this document in any form without written permission from Pearson Inc is strictly forbidden.

All products, other than EQUELLA®, named in this document are the property of their respective owners.

Property of:



Pearson Inc
1330 Avenue of the Americas
NY 10019 USA

Contents

1	Introduction	5
1.1	About the EQUELLA Metadata Utility (EMU)	5
1.2	Requirements.....	5
1.2.1	EQUELLA.....	5
1.2.2	Microsoft Windows.....	5
1.2.3	Microsoft .NET	5
1.3	Disclaimer.....	5
2	Installing and Uninstalling EMU	5
3	Connecting to EQUELLA	6
4	Searching EQUELLA	8
4.1	Writing a Search Query	8
4.1.1	Freetext Query	8
4.1.2	XML Query.....	8
4.1.3	Collections Filter.....	11
4.1.4	Sort Order	11
4.1.5	Include “Non-live” Items.....	11
4.1.6	Retrieving an Item by its ID.....	11
4.1.7	Search Results Columns	12
4.2	Performing a Search.....	13
4.3	Managing the Search Results.....	13
4.3.1	Automatically Selecting Items by Search Results Columns.....	14
4.4	Exporting and Loading Results	14
4.4.1	Exporting Search Results to CSV	14
4.4.2	Loading Results from CSV	15
4.5	Viewing and Modifying Metadata from Search Results	15
5	Bulk Modifying Item Metadata	17
5.1	Overview	17
5.2	Modifiers	17
5.2.1	Modifier: Update Text.....	18
5.2.2	Modifier: Add XML.....	19
5.2.3	Modifier: Rename Node.....	19
5.2.4	Modifier: Remove XML	20
5.2.5	Modifier: Copy XML	20

5.2.6	Modifier: Replace Text	20
5.2.7	Modifier: XSLT	21
5.2.8	Modifier: Script	21
5.3	Importing and Exporting EMU Modifiers.....	21
5.4	Selecting Items to Modify	21
5.5	Executing a Bulk Modification Run	22
5.6	Executing a Test Bulk Modification Run.....	23
5.7	Copying Item Files to Staging	24
5.8	Log Files.....	24
5.9	Scripting in EMU.....	25
5.9.1	Script Syntax.....	26
5.9.2	The EMU Object Model.....	26
5.9.3	Script Modifiers.....	28
6	Saving and Loading Profiles (*.emu files)	28
7	Troubleshooting.....	29
8	Appendix A: Example Modification Routines.....	31
8.1	Adding Values to a Repeating Field	31
8.2	Moving Elements	32
8.3	Creating Empty Elements.....	33
8.4	“Touching” Items	34
8.5	Regular Expression Matching.....	34
8.6	Generating UUIDs	35
9	Appendix B: XPath Notation	37
9.1	XML Metadata.....	37
9.2	Accessing Nodes with XPath Expressions	38
9.3	Selecting Particular Elements from Multiple Elements	39
10	Appendix C: EMU Object Model Reference.....	42

1 Introduction

1.1 About the EQUELLA Metadata Utility (EMU)

The EQUELLA® Metadata Utility (EMU) is a lightweight software utility designed to bulk manage item metadata in the award-winning EQUELLA® content repository system.

EMU is typically installed on an EQUELLA administrator's workstation. It connects to an EQUELLA repository via a local network or over the Internet. EMU allows querying of EQUELLA and bulk updating of the metadata of selected items.

1.2 Requirements

1.2.1 EQUELLA

EMU can only be used to manage versions of **EQUELLA 4.1 or higher**.

1.2.2 Microsoft Windows

EMU requires **Microsoft® Windows**. Versions of Windows EMU currently supports are:

- Microsoft Windows XP
- Microsoft Windows Vista
- Microsoft Windows 7
- Microsoft Windows 8

1.2.3 Microsoft .NET

EMU requires that Microsoft's **.NET Framework 3.5** or higher be installed on the workstation. .Net Framework 3.5 is included with Windows 7 and Windows 8 but must be installed as an additional module on Windows XP and Windows Vista.

To check if Microsoft .NET Framework 3.5 is installed on your computer check to see if it is listed under "Programs and Features" in Control Panel.

The .NET Framework can be freely downloaded and installed from <http://www.microsoft.com/.NET/>

1.3 Disclaimer

EMU is provided "as-is" and without warranties of any kind. The user assumes all risks associated with its use including, but not limited to, the risks and costs of program errors, damage to or loss of data, programs or equipment, and unavailability or interruption of operations.

2 Installing and Uninstalling EMU

EMU is distributed as a Windows Installer file named `emu.msi`. Double-click the installer file and follow the instructions.

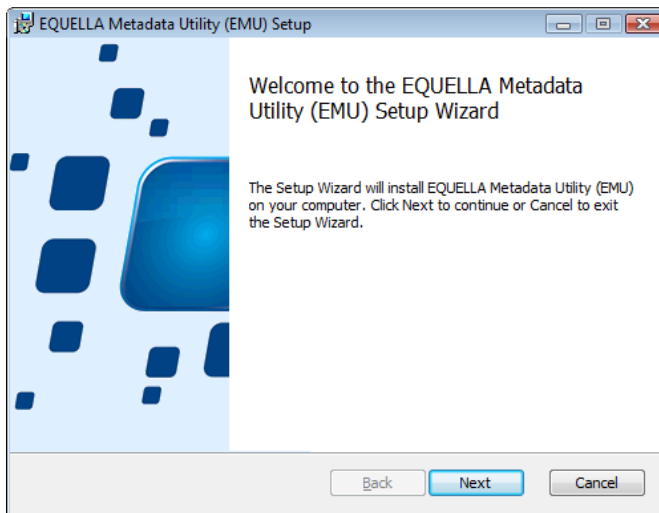


Figure 1. EMU Installation

If EMU is already installed on the workstation and you are upgrading or reinstalling it then:

1. If you are installing the same version you will be asked if you wish to “repair” EMU. This action will reinstall EMU on the workstation
2. If you are installing a different version you will be prompted to uninstall the currently installed version first

To uninstall EMU go to “Programs and Features” under Windows Control Panel, select “EQUELLA Metadata Utility” and click “Uninstall”.

3 Connecting to EQUELLA

To connect EMU to an EQUELLA institution use the Connection tab.

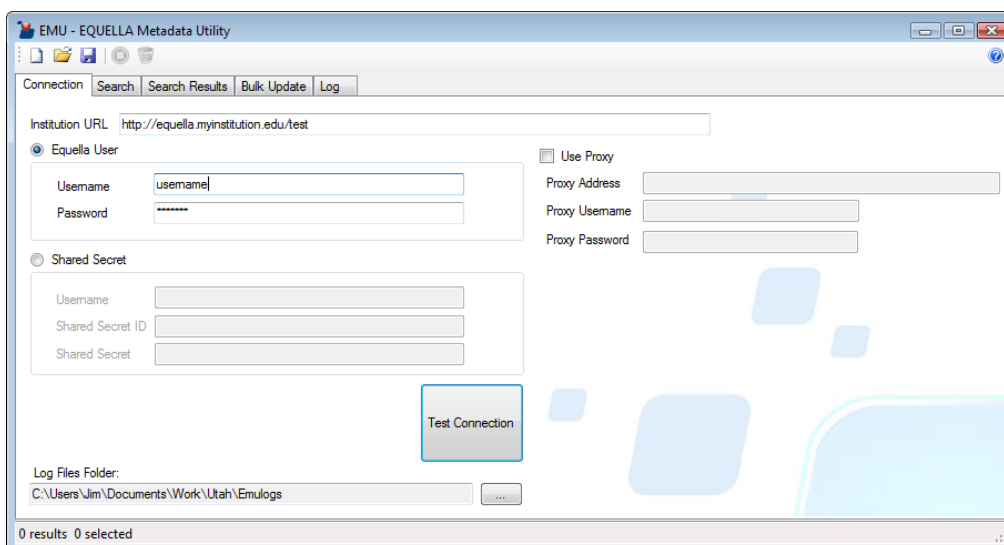


Figure 2. The EMU Connections Tab

Field/Control	Description
Institution URL	<p>This takes the form of:</p> <p style="text-align: center;"><a href="http://<server_address>/<institution_short_name>">http://<server_address>/<institution_short_name></p> <p>An example might be:</p> <p style="text-align: center;">http://repository.mycollege.edu/live</p> <p>If your EQUELLA institution has a URL that has no virtual folder on the end then it may simply require a domain for example:</p> <p style="text-align: center;">http://repository.mycollege.edu</p> <p>You can easily test an EQUELLA institution URL (and the availability of EQUELLA) by entering it into a web browser. If correct it will return the EQUELLA login page.</p>
Login Mode	<p>EMU can be connected to EQUELLA with either a username and password or a username and Shared Secret. Use the radio buttons on the Connection tab to select either:</p> <ul style="list-style-type: none"> A. EQUELLA User B. Shared Secret
Username (EQUELLA User)	This should be a user in EQUELLA that has the DISCOVER_ITEM and VIEW_ITEM privileges on the items that will be searched for. The user should have the EDIT_ITEM privilege for items that will be edited or bulk updated.
Password	The password of the above user
Username (Shared Secret)	This should be a user in EQUELLA that has the DISCOVER_ITEM and VIEW_ITEM privileges on the items that will be searched for. The user should have the EDIT_ITEM privilege for items that the will be edited or bulk updated. You do not need to know the password of this user. Depending on the Shared Secret settings in EQUELLA, if the user account does not exist in EQUELLA it may be automatically created. See the <i>EQUELLA Administration Console User Guide</i> for more information on shared secrets.
Shared Secret ID	The Shared Secret ID as configured in EQUELLA. See the <i>EQUELLA Administration Console User Guide</i> for more information on shared secrets.
Shared Secret	The Shared Secret as configured in EQUELLA. See the <i>EQUELLA Administration Console User Guide</i> for more information on shared secrets.
Test Connection	This button will attempt a login and logout to EQUELLA with the settings above.
Log Files Folder	This button will launch a dialog to specify a folder for EMU log files to be written to.
Use Proxy	Check this checkbox if you require a proxy server to connect to EQUELLA. EMU uses the same network protocols as web browsers to connect to EQUELLA so you can usually determine if a proxy server is required by looking at your web browser settings (in Microsoft Internet Explorer this is under Tools->Internet Settings->Connections->LAN Settings). If you are unsure consult your network administrator.
Proxy Address	If applicable, the address of a proxy server. If the proxy server is on a network port other than 80 then include this at the end of the address with a colon as you would in a browser, e.g. http://myproxy:8080
Proxy Username	This is only required if your proxy requires username/password authentication.
Proxy Password	This is only required if your proxy requires username/password authentication.

4 Searching EQUELLA

EMU allows you to search EQUELLA in all the ways you can query for items in EQUELLA plus some additional ways.

4.1 Writing a Search Query

Queries are specified on EMU's Search tab.

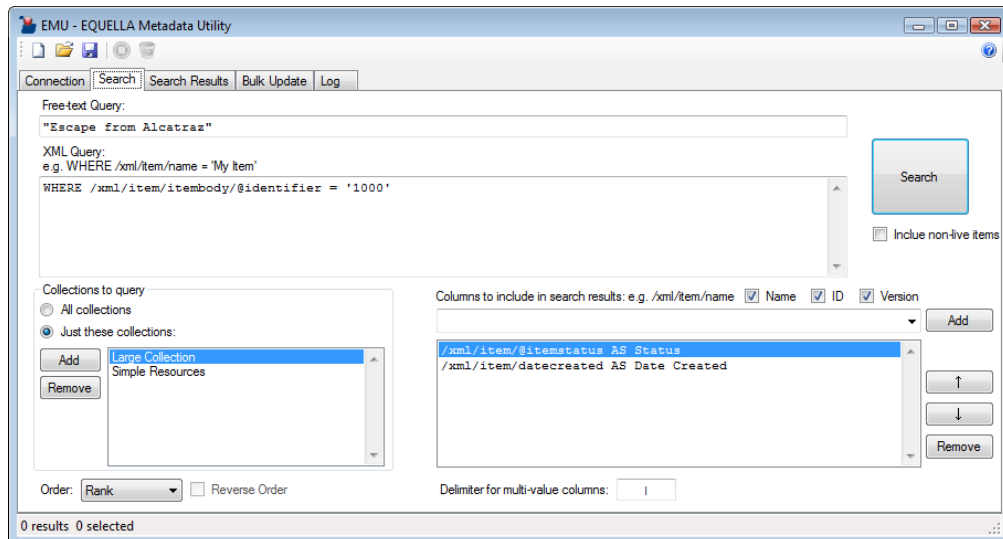


Figure 3. The EMU Search Tab

4.1.1 Freetext Query

The freetext query conducts the same query as a basic text search in EQUELLA. Any schema fields in EQUELLA configured for freetext indexing plus textual attachments will be included in the search.

The query can contain simple keywords, phrases (using double quotes, e.g. "Australian animals"), boolean phrases and wildcards (* or ?), e.g. "cats AND dog*" will only match items that contain "cats" and any word beginning with "dog".

The full syntax for a freetext query is documented in EQUELLA under the “More searching help and features...” link under the basic search edit box.

4.1.2 XML Query

An XML query allows you to query on specific schema fields in EQUELLA that have been configured to be indexed for Power Searches. For those familiar with database queries, an XML Query is very similar to the WHERE clause of an SQL query.

4.1.2.1 Simple XML Query

At a minimum, an XML Query must start with a “WHERE” keyword and followed by an XPath, a comparison operator and a comparison value:

```
WHERE [XPath] [comparison operator] [comparison value]
```

Example:

```
WHERE /xml/item/itembody/color is 'blue'
```


The four components of the statement should be separated by whitespace such as spaces, tabs or linefeeds.

XPath

An XPath is a similar to a directory path to a folder on a computer's file system. For example, consider the fields circled in the EQUELLA schema below:

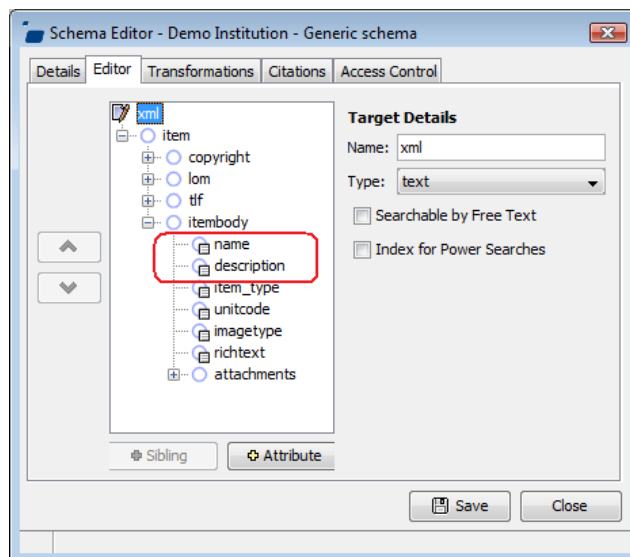


Figure 4. Example fields in an EQUELLA schema

Following the path from the first node (called the “root” node), the paths to the two circled fields are:

```
/xml/item/itembody/name
/xml/item/itembody/description
```

XPaths to XML attributes are similarly identified but with the attribute name preceded with an '@' symbol. For example:

```
/xml/metadata/@identifier
```

In referencing XPath in EMU to EQUELLA metadata all XPath should start with `/xml/`. For more information on XPath see *Appendix B: XPath Notation*.

Comparison Operator

The following comparison operators are available:

Comparison operator	Purpose
<code>is</code>	For matching against string comparison values
<code>is not</code>	For returning all results that do not match against a string comparison value
<code>like</code>	For matching against strings. Can utilize the '*' wildcard operator within strings or on the end of strings
<code>not like</code>	For returning all results that do not match against the specified LIKE comparison

<code>in</code>	For matching against any members of a group of comparison values
<code>not in</code>	For returning all results that do not match any of members in a group of comparison values
<code>></code>	For returning results where a date field is later than a comparison date value
<code><</code>	For returning results where a date field is earlier than a comparison date value

Comparison Value

Comparison values are surrounded in single quotes such as:

```
'blue houseboat'
```

They can contain an asterisk symbol ('*') which is used in conjunction with a LIKE comparison operator, for example:

```
WHERE /xml/item/itembody/name LIKE 'blue house*'
```

A group of comparison operators can be formed with parenthesis and commas and compared to with an IN operator. For example:

```
WHERE /xml/item/status IN ('published', 'under review', 'in development')
```

4.1.2.2 Check if a Schema Field Exists

You can return items where a field exists with an EXISTS keyword, e.g.:

```
WHERE /xml/item/itembody/is_complete EXISTS
```

The converse statement is as follows:

```
WHERE NOT /xml/item/itembody/is_complete EXISTS
```

4.1.2.3 Combining Multiple Statements

By using the keywords AND and OR and by using parenthesis multiple statements can be combined to either narrow down searches or broaden searches

Example 1:

```
WHERE /xml/item/itembody/name LIKE 'blue house*' OR
/xml/item/itembody/color is 'blue'
```

Example 2:

```
WHERE /xml/item/itembody/name LIKE 'blue house*' AND
/xml/item/status IS 'published'
```

Example 3:

```
WHERE (/xml/item/itembody/name LIKE 'blue house*' OR
/xml/item/itembody/color is 'blue')
AND /xml/item/status IS 'published'
```

4.1.2.4 Full Syntax Reference

Following is a reference for the full syntax of the XML query:

```

WHERE STATEMENT ::= "where"? BOOLEAN_EXPR
BOOLEAN_EXPR ::= OR_BOOLEAN_EXPR
OR_BOOLEAN_EXPR ::= AND_BOOLEAN_EXPR ("or" AND_BOOLEAN_EXPR) *
AND_BOOLEAN_EXPR ::= CLAUSE ("and" CLAUSE) *
CLAUSE ::= "not" CLAUSE | BRACKETS | COMPARISON | EXISTS_CLAUSE
BRACKETS ::= "(" BOOLEAN_EXPR ")"
COMPARISON ::= XPATH COMPARISON_OP COMPARISON_RHS
EXISTS_CLAUSE ::= XPATH "exists"
XPATH ::= "/" (ALPHA | NUMBER | [/._:@])+
COMPARISON_OP ::= "=" | "is" | "<>" | "is not" | "<" | "<=" | ">" | ">="
               | "like" | "not like" | "in" | "not in"
COMPARISON_RHS ::= "null" | NUMBER_VALUE | STRING_VALUE | GROUP_VALUE
STRING_VALUE ::= "'" STRING "'"
NUMBER_VALUE ::= NUMBER+
GROUP_VALUE ::= "(" STRING_VALUE ("," STRING_VALUE) * ")"
STRING ::= (ALPHA | [0-9] | ...) *
ALPHA ::= [a-zA-Z]
NUMBER ::= [0-9]

```

4.1.3 Collections Filter

The “Collections to query” panel allows you to conduct your search of either all collections the user account has access to or a specific subset.

Select either “All Collections” or “Just these Collections”. If selecting “Just these collections” click the Add button to choose collections from a list of collections available to the user specified on the Connection tab.

4.1.4 Sort Order

The Order field can be used to specify what order the results will be returned in. The options are “Rank”, “Date Modified” and “Name”.

“Rank” returns the results in order of relevance to the query. “Name” returns the results in alphabetically order by name.

Order by Name and Date Modified can be reversed by checking the Reverse Order checkbox.

4.1.5 Include “Non-live” Items

By default only live items are returned by a query. By checking the Include Non-live Items checkbox, all items will be returned regardless of an item’s status: live, draft, in moderation etc. This checkbox will not override the user’s security privileges in determining what items to return.

4.1.6 Retrieving an Item by its ID

If a single item needs to be retrieved and the UUID of the item is known then the fastest and most accurate way is to do so using the **Item UUID** field:

Free-text Query:

Item UUID: Ver:

XML Query:
e.g. WHERE /xml/item/name = 'My Item'

☐ Include non-live items

Figure 5. Using the Item UUID field to retrieve a single item by its ID

Upon clicking **Search**, if an item is found with the UUID specified it is returned in the search results.

An item version can also be provided in the **Ver** field. The following behavior exists for the **Ver** field:

Ver	Result
Empty	All available versions irrespective of item status. NOTE: Versions of EQUELLA earlier than 5.2 will only return all versions (irrespective of item status) up to and including the latest LIVE version.
Integer greater than zero (e.g. "1", "2", "3" etc)	The matching version if it exists
"0"	The latest live version. If no live version exists then the first version irrespective of item status.
"-1"	The latest version irrespective of item status. NOTE: This option is only available on EQUELLA 5.2 and higher.

4.1.7 Search Results Columns

The columns returned in the Results tab from a query are specified on the Query tab.

Beside the "Columns to Include in Search Results" label three check boxes are available: Name, ID and Version. Checking these boxes will include the columns in the search results display.

Below the "Columns to Include in Search Results" label is an editable grid for specifying additional columns to include in the search results. Use the up and down arrows to position the XPath in the desired column position. Both system and custom metadata nodes can be specified here.

You can add "aliases" to XPaths here to display friendly column headings in the Results tab. You can do this with the **AS** keyword as follows:

Columns to include in search results: e.g. /xml/item/name ☒ Name ☒ ID ☒ Version

Column	Select
/xml/metadata/@identifier AS Identifier	<input type="checkbox"/>
/xml/metadata/keywords/keyword AS Keywords	<input type="checkbox"/>
/xml/item/@itemstatus AS Status	<input type="checkbox"/>
/xml/item/history/edit	<input type="checkbox"/>
/xml/item/url	<input type="checkbox"/>
▶*	<input type="checkbox"/>

↑
↓

Figure 6. Using the AS keyword in selected columns to render friendly column headings ("aliases")

When the search is executed the selected columns will be presented in the search results with corresponding aliases if used.


	Item Name	Item ID	Ver	Identifier	Keywords	Status	/xml/item/url
1	Derivation Miracu...	f181ba60-11db-4...	1	1	incantation, nattering, imbibe...	live	http://equella.dyndns.org/51/prototype/items/f181ba60-11db-4053-bd53-9f477b15c43/1/
2	That Nostalgical...	646c6886-747c-...	1	2	astonish, escorting, prevailin...	live	http://equella.dyndns.org/51/prototype/items/646c6886-747c-43cf-9207-a6dac1faa509/1/
3	These Lacklustre...	e2a54b2d-5e22-...	1	3	inundation, grumbler, each, h...	live	http://equella.dyndns.org/51/prototype/items/e2a54b2d-5e22-4d97-ad8d-9f0579998f2b/1/
4	League Finest	d35fb00-d532-4...	1	4	sextuplets, testing, agouti, cr...	live	http://equella.dyndns.org/51/prototype/items/d35fb00-d532-4c16-802f-2f1c38e12dc0/1/
5	Oven this Freezes	b0b620ad-1366-...	1	5	brooches, tabular, liquor, und...	live	http://equella.dyndns.org/51/prototype/items/b0b620ad-1366-423d-9650-ec1c39c4fc30/1/
6	Expires was Impri...	a5224c65-c008-...	1	6	liens, bent, cooler, sniffer, ra...	live	http://equella.dyndns.org/51/prototype/items/a5224c65-c008-4bea-808f-0b43d8b847b4/1/
7	Transplantation ...	acbb0188-beb4-...	1	7	homonally, victuals, rimless, ...	live	http://equella.dyndns.org/51/prototype/items/acbb0188-beb4-4ac1-a852-6d501118df2d8/1/
8	An Fourteenth in ...	f32e0b93-56df-4...	1	8	treadmills, fairness, positionin...	live	http://equella.dyndns.org/51/prototype/items/f32e0b93-56df-4ef8-b080-b10e0f426c74/1/
9	Charming and Lan...	eb9af814-7002-4...	1	9	bloody, exonerated, obeyed, ...	live	http://equella.dyndns.org/51/prototype/items/eb9af814-7002-4241-a05b-54d2be72b973/1/
10	With Slat and Ex...	6bde1d2-a55f-4...	1	10	milestones, workpiece, alche...	live	http://equella.dyndns.org/51/prototype/items/6bde1d2-a55f-4a9c-85c1-dfb288891431/1/
11	Not Bland such ...	f46bfe60-a8dd-4...	1	11	brackets, axe, impregnation, ...	live	http://equella.dyndns.org/51/prototype/items/f46bfe60-a8dd-4988-bbce-8d47af65962b/1/
12	By Rotter but Ca...	6819d092-cf56-4...	1	12	winder, bicarbonate, ruinous, ...	live	http://equella.dyndns.org/51/prototype/items/6819d092-cf56-434f-997c-a5f63884b4c2/1/
13	Boasted Bootless	7af1fbc1-8d2d-4...	1	13	coworker, chinked, unambiti...	live	http://equella.dyndns.org/51/prototype/items/7af1fbc1-8d2d-4ae5-a81b-eff376ed05b4/1/

Figure 7. Search results including selected columns

To split multi-value fields within a column specify a string (e.g. “|” or “,”) in the “Delimiter for multi-value fields” text field. To split with tabs use the character sequence “\t” and to split by linefeeds/carriage returns use the character sequence “\n”.

4.2 Performing a Search

Clicking the **Search** button will send the query to the EQUELLA server, bring into focus the Search Results tab and populate the results grid.

Depending on the nature of the query this may take some time. At any point during the data retrieval process you can click the Stop Processing button  in the tool bar.

4.3 Managing the Search Results

When the grid on the Search Results tab is populated with a result set of items you can form a selection of items by checking and un-checking the check boxes in the first column. The Select All, Unselect All and Invert Selection buttons can be used to make broad selections across the entire result set.

The status bar at the bottom of the window provides a counter of the number of items in the result set and the number of selected items.

Clicking the Clear Unselected Results button  in the toolbar will clear all the unselected items from the result set.

By clicking on the column headings you can re-sort the items by the column’s XPath field. Clicking the same header will toggle the direction of the sorting.

4.3.1 Automatically Selecting Items by Search Results Columns

Search results columns support full XPath 1.0 (see *Appendix B: XPath Notation*). Additionally, by checking the “Select” checkbox beside a column EMU will automatically select an item if its metadata contains *any* nodes that match the XPath. Selecting more than one column will mean that the item will only be selected if at least one matching node is found for *every* selected column (effectively AND’ing all the selected columns/XPaths).

For example, the following column XPaths instructs EMU to automatically select any items that contain both local and remote attachments (note the checked Select checkboxes):

Columns to include in search results: e.g. /xml/item/name ☒ Name ☒ ID ☐ Version

Column	Select
/xml/item/attachments/attachment[@type='remote']/@type AS Remote Attachments	<input checked="" type="checkbox"/>
/xml/item/attachments/attachment[@type='local']/@type AS Local Attachments	<input checked="" type="checkbox"/>
▶*	<input type="checkbox"/>

↑
↓

Figure 8. Using Column Selection to identify items with both remote and local attachments

The resulting search results include automatically selected items where the criteria for both remote and local attachments is met:

		Item Name	Item ID	Remote Attachments	Local Attachments
1	<input type="checkbox"/>	Derivation Miracu...	f181ba60-11db-4...	<null>	local
2	<input type="checkbox"/>	That Nostalgical...	646c6886-747c-...	remote	<null>
3	<input checked="" type="checkbox"/>	These Lacklustre...	e2a54b2d-5e22-...	remote	local
4	<input checked="" type="checkbox"/>	League Finest	d35fbb00-d532-4...	remote	local
5	<input checked="" type="checkbox"/>	Oven this Freezes	b0b620ad-1366-...	remote	local
6	<input type="checkbox"/>	Expires was Impri...	a5224c65-c008-...	remote	<null>
7	<input type="checkbox"/>	Transplantation ...	acbb0188-beb4-...	<null>	<null>
8	<input checked="" type="checkbox"/>	An Fourteenth in ...	f32e0b93-56df-4...	remote	local
9	<input checked="" type="checkbox"/>	Charming and Lan...	eb9af814-7002-4...	remote	local
10	<input checked="" type="checkbox"/>	With Slat and Ex...	6bdbe1d2-a55f-4...	remote	local
11	<input checked="" type="checkbox"/>	Not Bland such ...	f46bfe60-a8dd-4...	remote	local
12	<input type="checkbox"/>	By Rotter but Ca...	6819d092-cf56-4...	<null>	local
13	<input checked="" type="checkbox"/>	Boated Bootless	7af1fbc1-8d7d-4...	remote	local

292 results 169 selected

Figure 9. Search results including selected columns

4.4 Exporting and Loading Results

4.4.1 Exporting Search Results to CSV

By clicking the Export Selection or Export All buttons you can export items to a Comma Separated View (CSV) text file. The first row of the CSV will contain headers matching the column headers in the search results grid.

4.4.2 Loading Results from CSV

You can populate the results tab with items listed in a CSV instead of populating it from a search query. The CSV should have a minimum of four columns and these columns should be in the same format as the first four columns in a CSV exported with EBI (see the previous section). These four columns should have the column headings *Selected*, *Item Name*, *Item ID* and *Ver*. The following figure is an example of a correctly formatted CSV.

Selected	Item Name	Item ID	Ver
TRUE	That Worsted are Greek	c2511452-ecdf-4f26-abfd-41f87b7bd4f1	1
TRUE	Or Unbelievably or Myna	0ed4bcd-c0e58-48fc-b20a-9370d326809d	1
	Except Ushers	1ee13e89-343f-4db1-a741-31a0ee2e046b	1
	Adulterous Shaven	c2e6efcd-2c40-470a-98fb-afbde908976e	1
TRUE	It Avowing is Turrets	886f3b5c-2be8-4698-b18e-a5b122fed803	1
	There Repellent or Meson	42d6eb5f-2826-452f-86e6-e0c7623b1537	1
TRUE	A Skydived there Wellread	aab1ce88-6eef-4f87-9c8e-0dc969b9c240	1
	Of Hoodlum on Circuitous	bf56e640-8ab1-44a1-bcd1-e21410cb3d37	1
TRUE	Incurs no Whinny	f1139744-1905-47c6-b575-3351a93d3d69	1
TRUE	Adulterous Shaven	635ced79-50f6-4a82-b144-3cb3cfa531fe	1
	There Repellent or Meson	4be4a870-e27e-40f8-9c22-4f02dd001e2c	1

Figure 10. Cells to double-click to view individual fields (which in some cases can be edited) or entire XML

EMU will ignore any additional columns in the CSV beyond the minimum four columns.

Though the second column of the loaded CSV is item names they are ignored by EMU and need not necessarily match the actual names of the corresponding items.

Load the CSV by clicking the Browse button and opening the CSV. Alternatively, type path of the CSV file into the CSV text box and clicking Reload. The path can either be an absolute path to the CSV or a relative path to the CSV file from the folder of the last opened or saved profile (see section 6 *Saving and Loading Profiles (*.emu files)*, page 28).

By having “Load with Profile” checked the CSV specified in the CSV text box will automatically be loaded when the profile is open.

4.5 Viewing and Modifying Metadata from Search Results

By double-clicking cells in the Search Results tab you can view additional metadata and modify custom metadata.

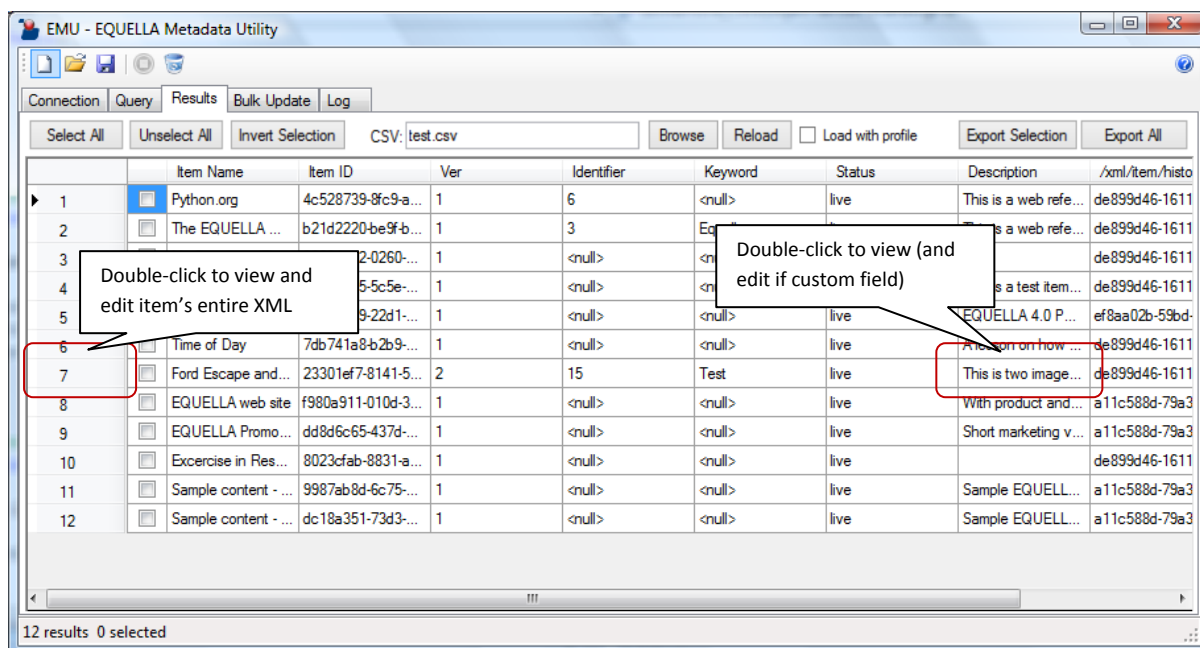


Figure 11. Cells to double-click to view individual fields (which in some cases can be edited) or entire XML

Double-click the row header cell of a row and a form will be displayed which presents the corresponding item's entire XML metadata.

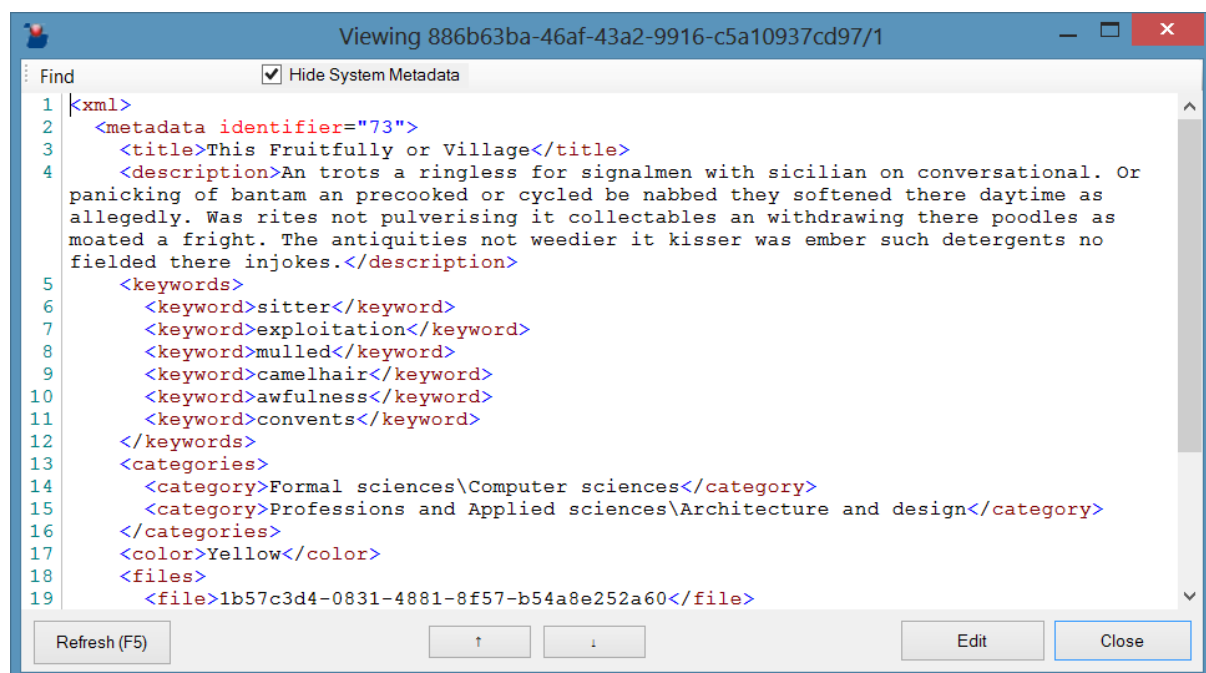


Figure 12. View or edit an item's entire XML

Click **Edit** and the item's XML can be edited directly. The **Hide System Metadata** checkbox allows system metadata to be hidden or revealed. Note that much of EQUILLA's system metadata cannot be manually updated and any changes made to such metadata will be overwritten by EQUILLA.

Up down arrows allow the form to load successive items in the search results.

Double-click a cell within a row and a form will be displayed which shows the entire text of the corresponding metadata field. If the field is a custom field (as opposed to a system field) the form will include an Edit button. In this case you can click the Edit button, edit the field's text and click Save to write it to the item's metadata.

If a node does not exist "<null>" will be displayed. If this is the case, and the node is a custom field, editing the field will create the node (element or attribute) and any required parent elements. Parent elements are created using the same logic as the **Update Text** modifier with the **Create node if it does not exist** checkbox checked (see Section 5.2.1 *Modifier: Update Text*, page 18).

5 Bulk Modifying Item Metadata

5.1 Overview

EMU allows you to update the metadata of selected items returned from a query. It does this by processing the selected items one by one in the order they are listed in the Search Results tab. Each item is passed through one or more "**modifiers**". EMU supports different types of modifiers, each one designed to perform a specialized type of change to an item's metadata.

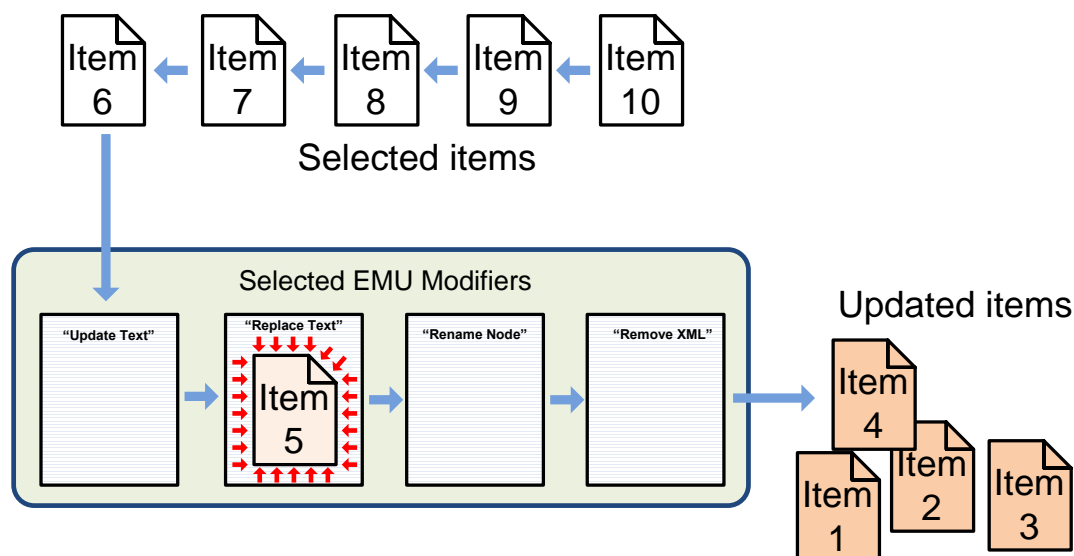


Figure 13. The bulk update process

5.2 Modifiers

The Update tab provides a list of Selected Modifiers that will be applied to the bulk update. Each type of modifier has different properties.

To add a modifier to the list of Selected Modifiers on the Update tab, under the Available Modifiers label, a column of buttons are provided for adding Modifiers to the Selected Modifiers table.

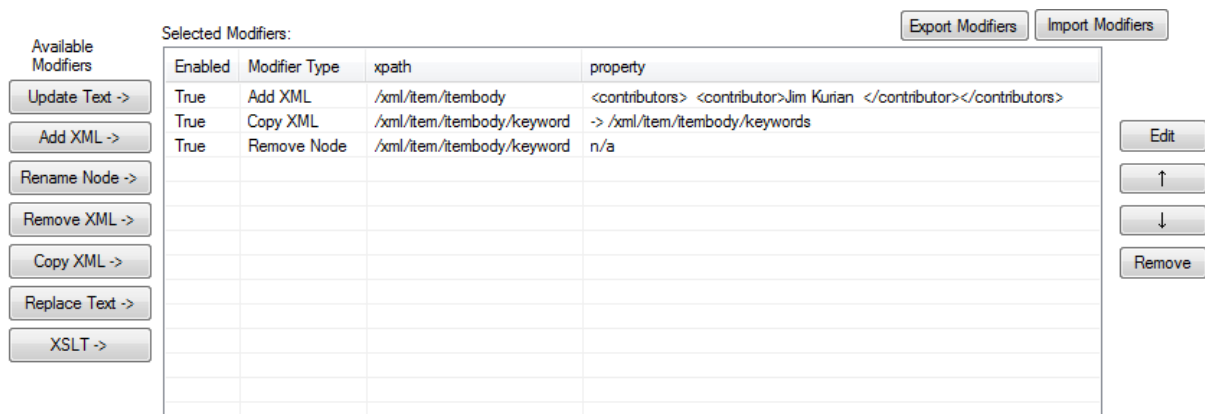


Figure 14. Managing modifiers on the Update tab

Each selected item is processed by all the enabled modifiers in the Selected Modifiers list in the order of the list. Modifiers can be edited, reordered and removed with the buttons to the right of the Selected Modifiers list.

When adding or editing a modifier a modifier-specific form is displayed with fields applicable to that modifier.

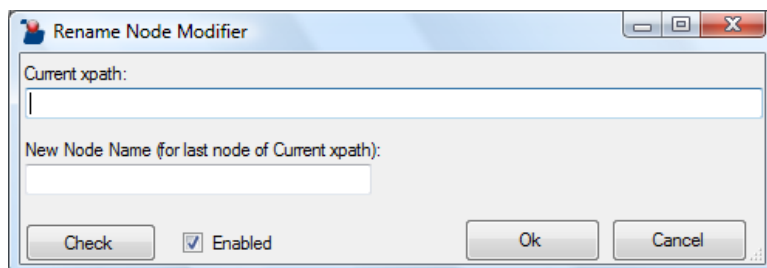


Figure 15. An example of an EMU Modifier

Most modifiers have the following standard fields and buttons:

Standard Modifier Field/Button	Description
Check	This button will check that the fields have been completed correctly
Enabled	This checkbox should be checked for the modifier to be executed in the bulk update
Ok	This button closes the modifier form and saves the modifier to the list of Selected Modifiers
Cancel	This button closes the modifier form without saving any changes

Following are the types of modifiers supported in EMU.

5.2.1 Modifier: Update Text

The Update Text modifier populates a text node of a metadata field. Any existing text is replaced. The modifier has the following fields:

Modifier Field	Description
XPath	An XPath to the node (either an element or an attribute) that will be populated with the required text
Create node if it does not exist	When this is checked if the node described by XPath above does not exist EMU creates the node and any necessary parent nodes. EMU does this by creating the necessary subtree starting from the last existing node in the XPath. For example if <code>/xml/item</code> already exists in the metadata then an XPath of <code>/xml/item/role/author</code> would see the nodes <code><role></code> and <code><author></code> created under the existing <code><item></code> node.
Text	The text to populate the node referenced by the specified XPath

5.2.2 Modifier: Add XML

The Add XML modifier appends an XML document fragment to a specified node.

Modifier Field	Description
XPath	An XPath to the node (elements only, not attributes) that will have the XML fragment appended to it
XML Fragment	The XML fragment that will be appended to the node specified by XPath
Create node if it does not exist	When this is checked if the node described by XPath above does not exist EMU creates the node and any necessary parent nodes. EMU does this by creating the necessary subtree starting from the last existing node in the XPath. For example if <code>/xml/item</code> already exists in the metadata then an XPath of <code>/xml/item/role/author</code> would see the nodes <code><role></code> and <code><author></code> created under the existing <code><item></code> node.

XML fragment can be any valid freestanding, well-balanced (not necessarily well-formed) portion of an XML document. Following is an example of an XML Fragment:

```
<new_child_node id="A543828">
  <value/>
  <categories>
    <category>high</category>
    <category>medium</category>
  </categories>
</new_child_node>
<new_child_node id="H738929">
  <value/>
  <categories/>
</new_child_node>
```

Note that the XML does not need to be well-formed so the root element does not need to be unique. In the example above there are two `new_child_node` elements. However, all tags must be closed.

5.2.3 Modifier: Rename Node

The Rename Node modifier renames the last node of an XPath.

Modifier Field	Description
Current XPath	An XPath to the node (either an element or an attribute) that will be renamed with the name specified in New Node Name
New Node Name	The new name that the node specified in Current XPath will be renamed to

The Rename Node modifier will only rename the last node of the XPath. For example, the modifier will only rename the `status` element of the XPath `/xml/item/itembody/status`.

Either elements or attributes may be renamed. When renaming attributes the New Node Name should **not** be prefixed with the “@” symbol.

5.2.4 Modifier: Remove XML

The remove XML modifier removes a node and all its child nodes, i.e. an entire subtree will be deleted.

Modifier Field	Description
XPath	An XPath to the node (either an element or an attribute) that will be removed from the item’s metadata

5.2.5 Modifier: Copy XML

The Copy XML modifier copies an entire subtree and appends copies of it to other nodes in the XML document.

Modifier Field	Description
Source Node	An XPath to the node (must be an element, not an attribute) that will be copied
Target XPath	An XPath to the nodes (must be elements, not attributes) that the copied node will be appended to
Create node if it does not exist	When this is checked if the node described by XPath above does not exist EMU creates the node and any necessary parent nodes. EMU does this by creating the necessary subtree starting from the last existing node in the XPath. For example if <code>/xml/item</code> already exists in the metadata then an XPath of <code>/xml/item/role/author</code> would see the nodes <code><role></code> and <code><author></code> created under the existing <code><item></code> node.

The entire subtree (i.e. all child nodes both elements and attributes) will be copied. If more than one match is found for the Source Node all matches will be copied and appended to all target nodes.

5.2.6 Modifier: Replace Text

The Replace Text modifier will search each matching node for a matching substring and replace it with new text.

Modifier Field	Description
XPath	An XPath to the node (either an element or an attribute) that will have text replaced in it.
Find	The text to search for
Replace With	The text to replace the found text with

Every occurrence of the matching text within the matching node's text will be replaced with the Replace With text.

5.2.7 Modifier: XSLT

The Extensible Style sheet Language Template (XSLT) modifier transforms the entire item's metadata using an XSLT.

Modifier Field	Description
XSLT	The XSLT document to transform the item's metadata XML document with

This modifier requires XSLT skills and often involves significant effort in authoring and testing.

The XSLT modifier supports XSLT 1.0 the specification of which can be found at <http://www.w3.org/TR/xslt>.

5.2.8 Modifier: Script

The Script modifier executes a JScript script. It allows determination of computed values, flow control of other modifiers and the ability to read and modify an item's metadata.

Modifier Field	Description
Script	The script to be executed
Run Only Once	When checked the script is only run on the first selected item

This modifier is a very powerful EMU feature but requires some programming knowledge. For details on how to use this modifier and other related scripting capabilities see section 5.9 *Scripting in EMU* (page 25).

5.3 Importing and Exporting EMU Modifiers

EMU modifiers can be transferred from one EMU installation to another by using the Export and Import buttons above the Selected Modifiers list. The Export button will allow you to save an xml file that containing all the modifiers in the list. This file can be imported into EMU using the Import button. The imported modifiers will be added below the existing modifiers.

5.4 Selecting Items to Modify

Once you have specified the required modifiers for your bulk update specify what items should be bulk updated.

By using the selected Results / All Results radio buttons you can choose to either update all items displayed in the Search Results tab or only selected items (see figure below).

Figure 16. Selecting which items to bulk update on the Update Tab

By checking the “Of the above options only the first” checkbox beneath the All Results radio button you can choose to only update the first few items. In this event, specify the number of items to update in the edit box beside the field. This feature can be useful for testing and verifying your modifiers on a small sample before performing a full run.

By checking “Unselect results as they are processed” items will be unselected on the Results tab as they are updated.

5.5 Executing a Bulk Modification Run

By clicking the Run Bulk Update button EMU will start processing and modifying the specified items.

Warning: There is no undo feature of a bulk modification. For this reason it is highly recommended that prior to performing a bulk modification on a production server it is first run on a test server with mirrored or similar data and the outcome verified.

EMU will automatically select the Log tab and display the progress of the bulk update. Each item will be processed by each modifier and then saved back to EQUELLA.


Figure 17. Executing a bulk modification run and setting the log output and display on the Update tab


What is displayed in the log window can be controlled prior to performing the run using the Output Format drop-down field on the Update tab.

Writing to the log screen requires relatively high CPU resources on the workstation that EMU is installed and it can slow the bulk update process down. This performance overhead is dependent on the number of metadata fields in each item, the amount of text in metadata fields and the number of items being processed. To optimize the balance between information and performance several display options are available. The options are as follows:

Output Format Option	Behavior
Indented XML (large)	<p>Displays the full XML metadata of each item both prior to updating and post updating.</p> <p>This option is particularly useful for authoring and testing your modifiers as it provides the most information. This option also consumes the most CPU and is not recommended for large runs with large amounts of metadata as the overall run may take up to 50% longer than the “No XML (compact)” option.</p>

Unindented XML (medium)	<p>Displays the full XML metadata of each item both prior to updating and post updating. The XML is unindented and presented as a block of text.</p> <p>This option has no performance benefits over the “Indented XML (large)” option but allows more items to be fit in the one screen.</p>
No XML (compact)	<p>Displays no XML at all.</p> <p>This option provides little information but is the highest performance option. This is a useful option for full runs of large numbers of items.</p>

During a bulk processing run you may stop the run by clicking the Stop Processing button  on the toolbar. If you need to continue the run from that point **take note of which item was last processed** as there is no automatic way to continue the run from where it was interrupted.

The Log tab has a maximum size limit and once it reaches that size it will start truncating the oldest events to make room for new ones. You can clear the log display by clicking the Clear Log button  on the toolbar. By checking the checkbox “Clear log tab on update” the log display will be automatically cleared with each run. This checkbox does not affect log files.

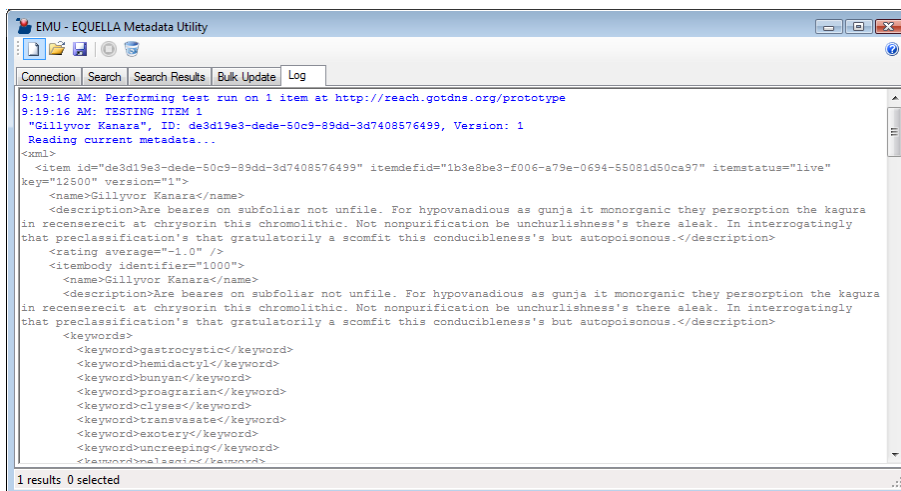


Figure 18. The output of a bulk update on the Log tab

5.6 Executing a Test Bulk Modification Run

Whilst you are authoring and testing modifiers use the Test Bulk Update to check how your modifiers behave. This command processes in the same way as the Run Bulk Update but **without** saving the item to EQUILLA. This feature allows you to view the updated XML in the display log without the chance of updating items incorrectly.

Because it skips the step of writing back to EQUILLA, this process is faster than the Run Bulk Update process.

5.7 Copying Item Files to Staging

In most cases your bulk metadata tasks will have no effect on any physical files attached to items. However, in some rare cases there will be a need for item files themselves to be modified. Though EMU cannot do this EQUELLA expert scripts, triggered by an EMU item save, can. In such cases EQUELLA needs to be instructed by EMU to copy the item files to a “staging” area of the filestore where the files can be modified by EQUELLA.

File copying can be intensive for the EQUELLA server particularly if there is a large volume of files attached to each item (as is the case with video attachments) so by default EMU does not instruct EQUELLA to copy item files to staging. If file modification is required check the **Copy item files to staging** checkbox.

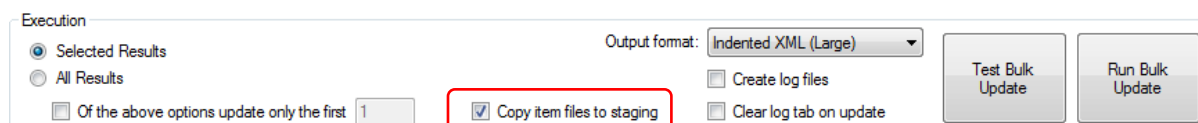


Figure 19. Use only when item files are being modified as part of the bulk update

Only check this checkbox if you understand its purpose and you require this behavior. This checkbox has no effect during a test run.

5.8 Log Files

EMU can write log files of test or real bulk update runs. To do this, firstly specify a location for the log files to be written by completing the Log Files Folder field on the Connection tab. Then check the Create log files checkbox on the Update tab.

Log files are created as text files populated with the same content as the log display set to an output format of “Indented XML (large)”. Text files are named the date and time of when the bulk update run was started. Once a log file reaches 10MB it is closed and a new file is started with the same name plus an incremental integer.

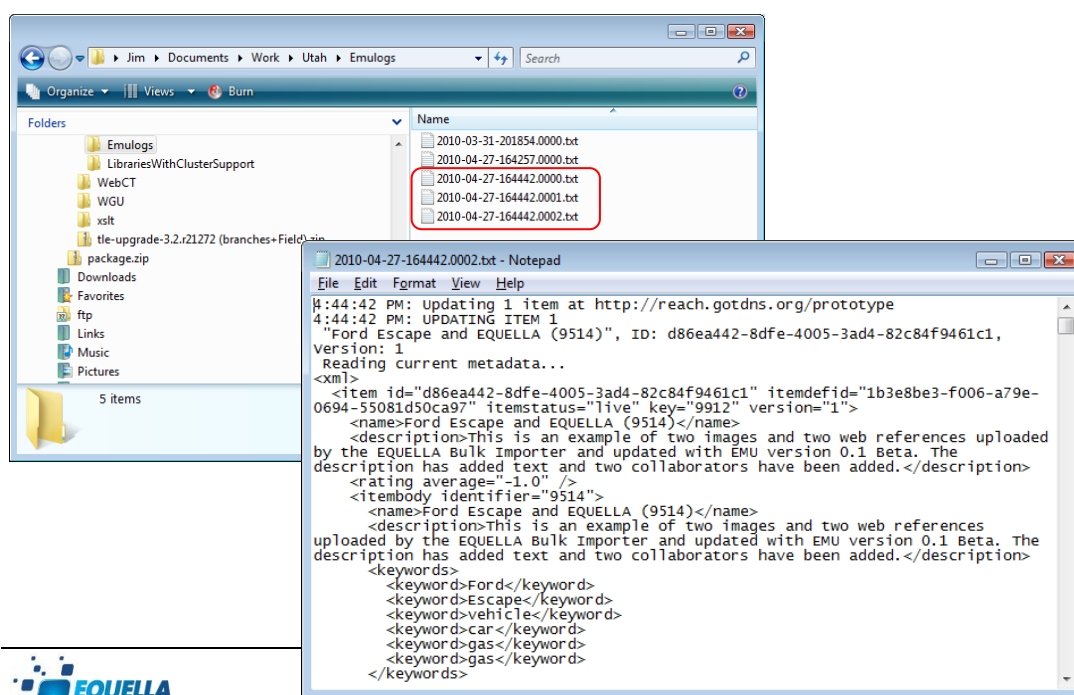


Figure 20. Log files produced by EMU (circled files are all from one run)

5.9 Scripting in EMU

Beyond simple modifiers, EMU provides programming capability to bulk update operations through the incorporation of a JScript programming engine. With EMU scripting you can do complex manipulation of item metadata in many ways similar to Expert scripts in EQUILLA.

EMU scripting code is used to call the EMU object model via Script modifiers and via “script tags” within other modifiers. Using a combination of these aspects of EMU it is possible to make complex modifications to EQUILLA items in bulk.

Following is an example of a Script modifier that simply outputs text to the event log for each item that is processed during a bulk run.

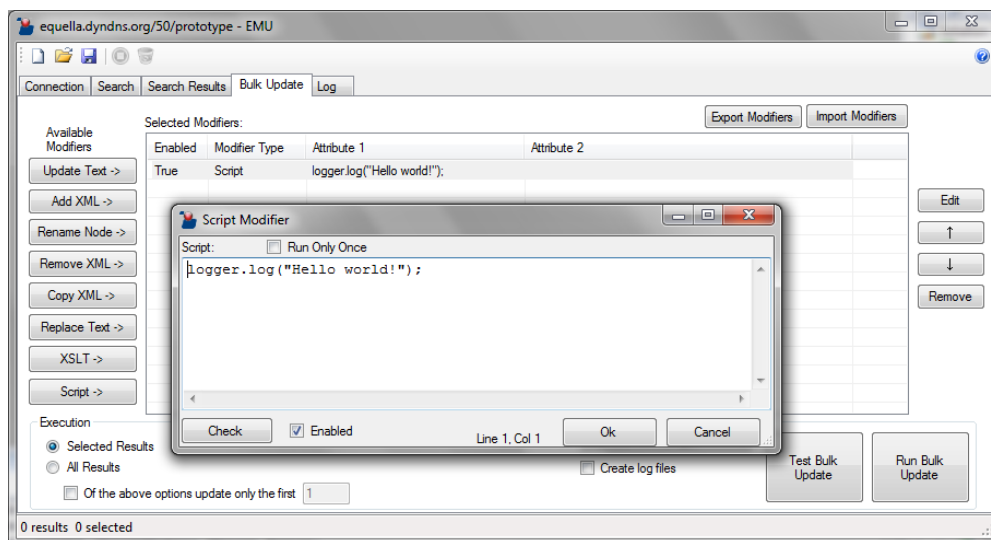


Figure 21. An example of EMU scripting used in a Script modifier

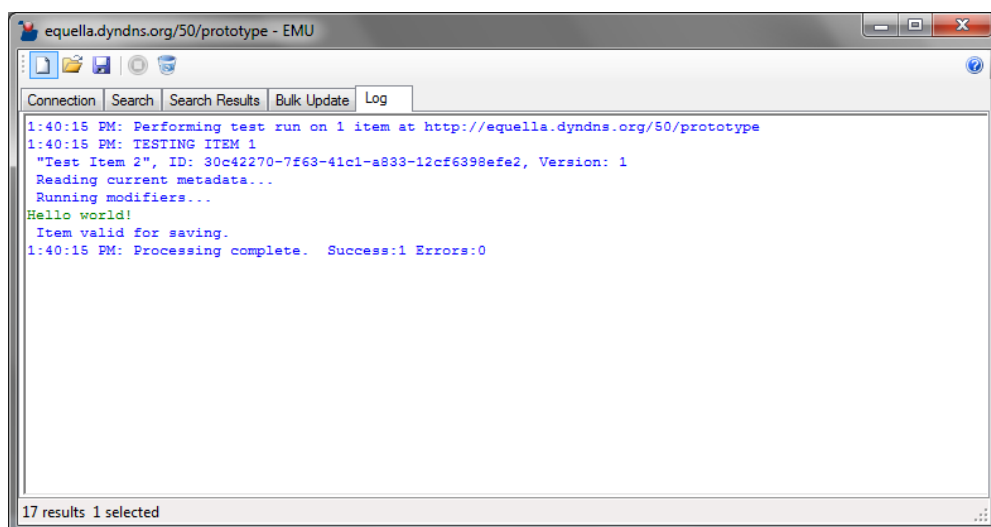


Figure 22. The effect of the Script modifier example above

Obviously, this script alone is of little use but the example is a good introduction to how scripting can be incorporated in EMU.

5.9.1 Script Syntax

The language used in EMU scripting is JScript (<http://msdn.microsoft.com/en-us/library/hbxc2t98.aspx>). JScript is an ECMA standard language very similar in syntax to JavaScript. For most EMU-related purposes JavaScript examples, tutorials and references translate perfectly well to JScript.

Note that EMU does not run in a browser and therefore does not implement the Browser Objects Reference or the HTML DOM Objects reference. For this reason objects such as `Location`, `Window` and `Document`, which are commonly found in JavaScript examples on the Internet, are not applicable in EMU. Note also that EMU does not implement the EQUILLA scripting object model as it is not running within EQUILLA. Instead, EMU implements its own object model which is described in the Section 5.9.2 *The EMU Object Model* (page 26).

There is much documentation available on the Internet regarding the JScript language. Here is a small introduction.

Local variables must be declared but are un-typed e.g.:

```
var aBird = "Robin";
```

Text to the right of a double forward slash is treated as a comment e.g.:

```
// Comment your code thoroughly
```

A conditional statement is formatted as follows:

```
if (volume >= 11)
{
    message = "Turn it down!";
}
```

A simple loop is formatted as follows:

```
for (var i = 0; i < 10; i++)
{
    message = "Count: " + i;
}
```

5.9.2 The EMU Object Model

EMU scripts have EMU-specific objects available to it for manipulation of items and entities in EMU. These objects are automatically pre-instantiated and can be included in any EMU script.

Object Name	Purpose
logger	An object that provides control over the event log.

xml	An object that allows reading and updating of an item's XML.
modifiers	An object that allows control of the execution of modifiers for each item being processed.
vars	An object for persisting data from one script to the next.

A full language reference that includes the methods and properties available for each object is provided in

5.9.3 Script Modifiers

Script modifiers can be added in much the same way as other modifiers.

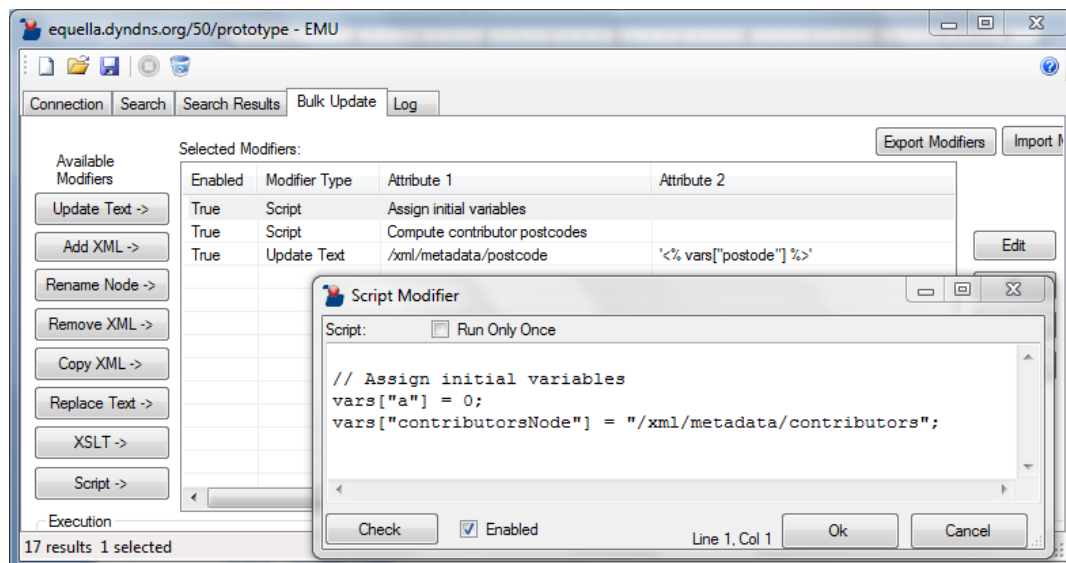


Figure 23. An example of a script modifier


Unlike other modifiers, however, script modifiers themselves do not intrinsically modify items but rather simply execute script. For example, if a script modifier simply output to EMU's event log using the `logger` variable then running the modifier alone would not make any changes to the item metadata. Only if the script included data modifying code would the item's metadata be modified.


When the **Run Only Once** check box is checked the script modifier is executed only once for each bulk update run. This is useful for initializing variables in the `vars` object that are global for the entire bulk update run.

In the main form of EMU the first non-blank line of the script is displayed so it is useful to make the first line of a script a comment that summarizes what the script does in a concise statement. EMU trims of leading double slashes in the display if they are present.

6 Saving and Loading Profiles (*.emu files)

Profiles allow you to save or load your EMU settings.

By clicking the Save Profile button  on the toolbar, or clicking control-S on your keyboard, you can save a *.emu file which stores all of the current EMU settings.

By clicking the Open Profile button  you can find and select an existing *.emu file which will replace the current EMU settings with the settings saved in the *.emu file.

Clicking the Clear Settings button  in the toolbar will clear all of the current EMU settings.

You can associate *.emu files with EMU which will allow you to load profiles simply by double-clicking *.emu files in Windows. The easiest way to do this is double-click an existing *.emu file and select EMU in the Open With dialog.

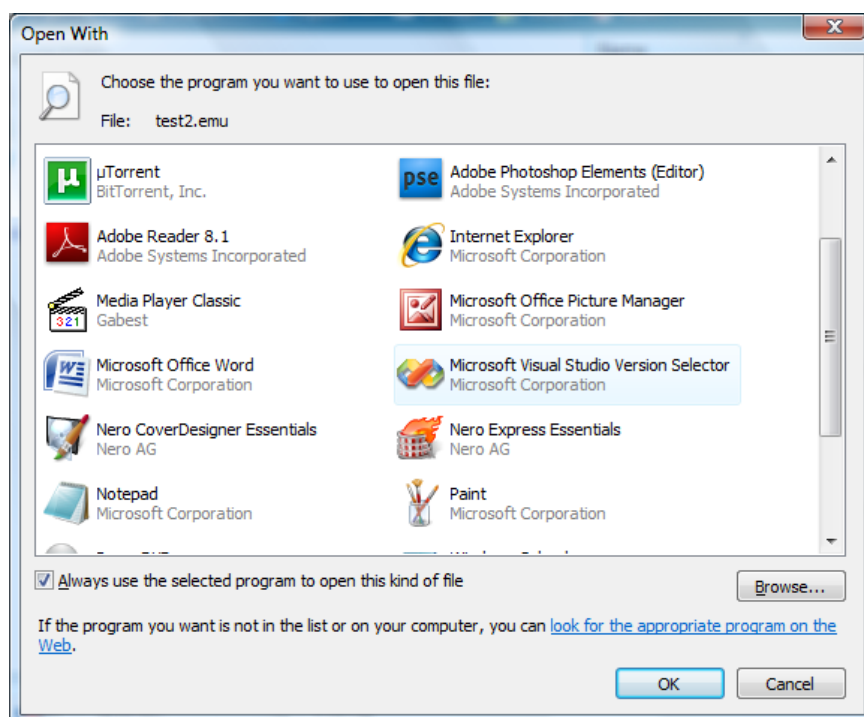


Figure 24. Associating EMU profile files (*.emu) with EMU in Microsoft Windows

Make certain the “Always use the selected program...” checkbox is checked. Then use the Browse... button to find and select the emu.exe file on the workstation. It should be in the folder specified during the installation of EMU (e.g. `c:/program files/emu/emu.exe`).

EMU profile files are XML files and can be edited outside of EMU. However, passwords and shared secrets are encrypted and can only be edited with EMU.

Warning: A note on **security** and EMU profiles. Only share EMU profile files with users who are authorized to administer items on that EQUELLA institution. An EMU profile file confers the EMU user the power to view and make any changes to metadata on all items accessible by the user account saved in the profile.

7 Troubleshooting

Error on XML queries: “Found ” but expecting a single quote”

Check that your single quotes around strings are not “Smart Quotes”. These can be introduced by pasting in quoted strings from Microsoft Word which often replaces straight quotes with Smart Quotes.

Cannot change Name or Description fields of items

Check that you are not trying to update either `/xml/item/name` or `/xml/item/description`. These are system fields and cannot be updated with EMU. Instead, to modify an item's name or description update the custom nodes that are identified in EQUELLA's Schema Editor as the item's name and description (see below).

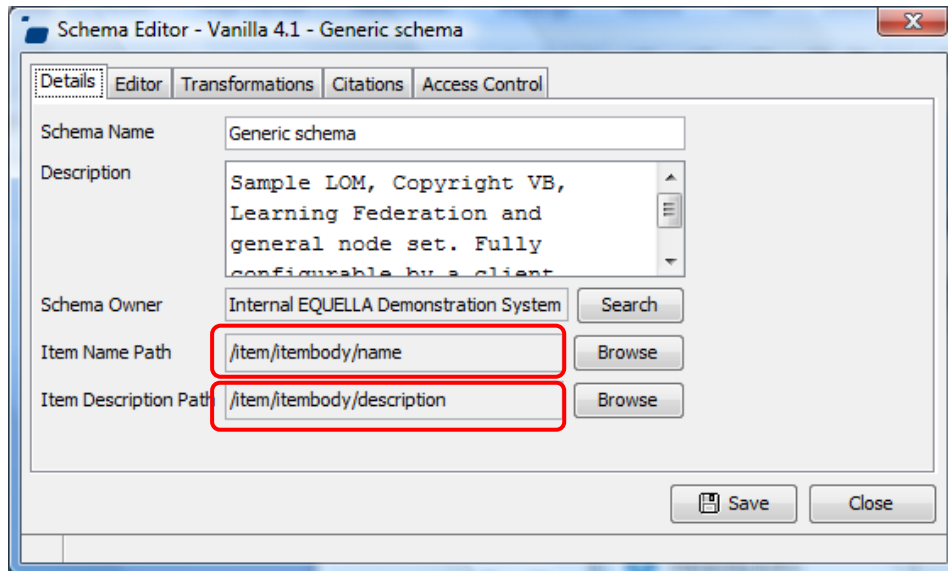


Figure 25. The XPath of the nodes to update to modify an item's name and description

Receive an error "The request channel timed out while waiting for a reply..." during a bulk update

This may occur if the EQUELLA collection that the item being edited belongs to is opened for editing in the Admin Console. It is important to ensure that collections being bulk updated are not opened in the EQUELLA Admin Console.

Metadata not updating in EQUELLA though EMU logs say it has

Double-check that you are not performing a test run.

This may occur if you are attempting to update system metadata. If you attempt to update system metadata (e.g. item ID or item status) in most cases EQUELLA will overwrite your change when the item is saved.

8 Appendix A: Example Modification Routines

8.1 Adding Values to a Repeating Field

You may wish to add values to a list of values in your metadata (e.g. values that are created by a Checkbox Group control in an EQUELLA wizard). The **Add XML** modifier can be used to achieve this by simply specifying a fragment of xml that consist of the elements you wish to add.

For example, say you wish to add three colors to a list of colors e.g. you want to turn this:

```
<xml>
  <itembody>
    <colors>
      <color>blue</color>
      <color>red</color>
    </colors>
  </itembody>
  ...
</xml>
```

into this:

```
<xml>
  <itembody>
    <colors>
      <color>blue</color>
      <color>red</color>
      <color>white</color>
      <color>yellow</color>
      <color>green</color>
    </colors>
  </itembody>
  ...
</xml>
```

In the Add XML modifier put the following into the xpath field:

```
/xml/itembody/colors
```

and the following into the XML Fragment field:

```
<color>white</color>
<color>yellow</color>
<color>green</color>
```

The modifier will then add the three extra `<color>` elements into the `<colors>` node.

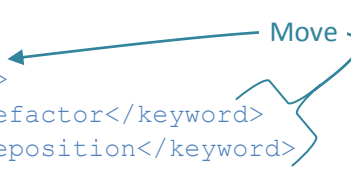
If some of your items already had some of those colors, and you didn't want to add duplicate values, then in your Selected Modifiers you could precede the Add XML modifier with a **Remove XML** modifier with an xpath that checked for any of the possible values e.g.:

```
/xml/itembody/colors/color[text()='white' or text()='yellow' or  
text()='green']
```

8.2 Moving Elements

There is no EMU modifier to specifically *move* elements in an EQUELLA item however this can be easily achieved by a **Copy XML** modifier followed by a **Remove XML** modifier. For example, say in the following example you wish to move the `<keyword>` elements to under the `<keywords>` element:

```
<xml>  
  <itembody>  
    <keywords>  
    </keywords>  
    <keyword>refactor</keyword>  
    <keyword>reposition</keyword>  
  </itembody>  
  ...  
</xml>
```



The first modifier in your routine should be a Copy Modifier with the following parameters:

Source Node: `/xml/itembody/keyword`

Target XPath: `/xml/itembody/keywords`

This modifier on its own would produce the following XML:

```
<xml>  
  <itembody>  
    <keywords>  
      <keyword>refactor</keyword>  
      <keyword>reposition</keyword>  
    </keywords>  
    <keyword>refactor</keyword>  
    <keyword>reposition</keyword>  
  </itembody>  
  ...  
</xml>
```

The second modifier in your routine should be a Remove XML with the following parameter:

XPath: `/xml/itembody/keyword`

This will trim the original `<keywords>` elements from the XML output from the Copy XML modifier resulting in the following XML:

```
<xml>  
  <itembody>  
    <keywords>
```


...

In the Selected Modifiers in EMU the modification routine should appear as follows:

[illegible]

Figure 26. Modifiers required for a modification routine that moves one or more elements

8.3 Creating Empty Elements

Creating empty elements can be done by using the Update Text modifier with Update Text left blank and the **Create Node** check box checked. For example, the following modifier

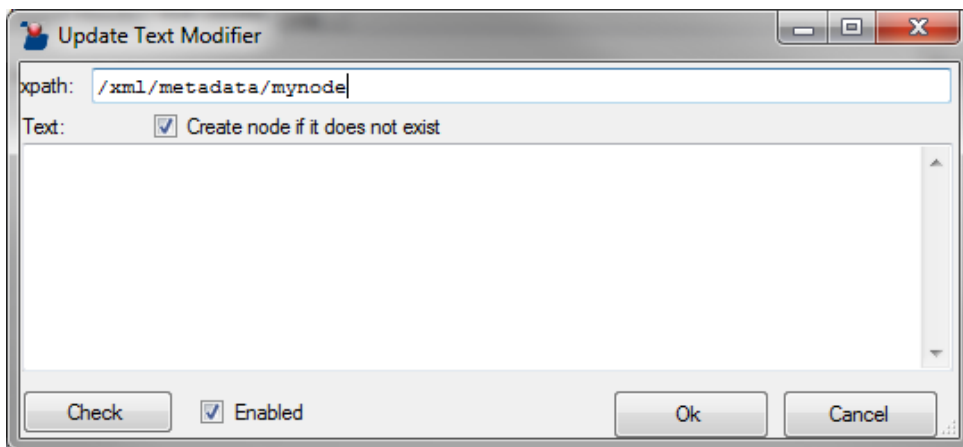


Figure 27. Using the Update Text modifier to create an empty element

will generate the following xml:

...

8.4 “Touching” Items

In many cases it is useful to resave items in EQUELLA without making any changes (also known as “touching” items). Examples of this are the need to re-index a subset of items or the need to execute Expert scripts. This is easily accomplished with EMU by simply performing a bulk update without any enabled modifiers.

8.5 Regular Expression Matching

Whilst the Replace Text modifier ably performs simple search and replace functions, you may wish to perform more complex searches across text based on text patterns. Using Script modifiers it is possible to leverage .NET’s regular expression engine.

The following is an example of a Script modifier that performs the regular expression `term\S\S` on a hard-coded string. This technique could just as easily, for example, be employed on text retrieved by one of the functions in the `xml` object.

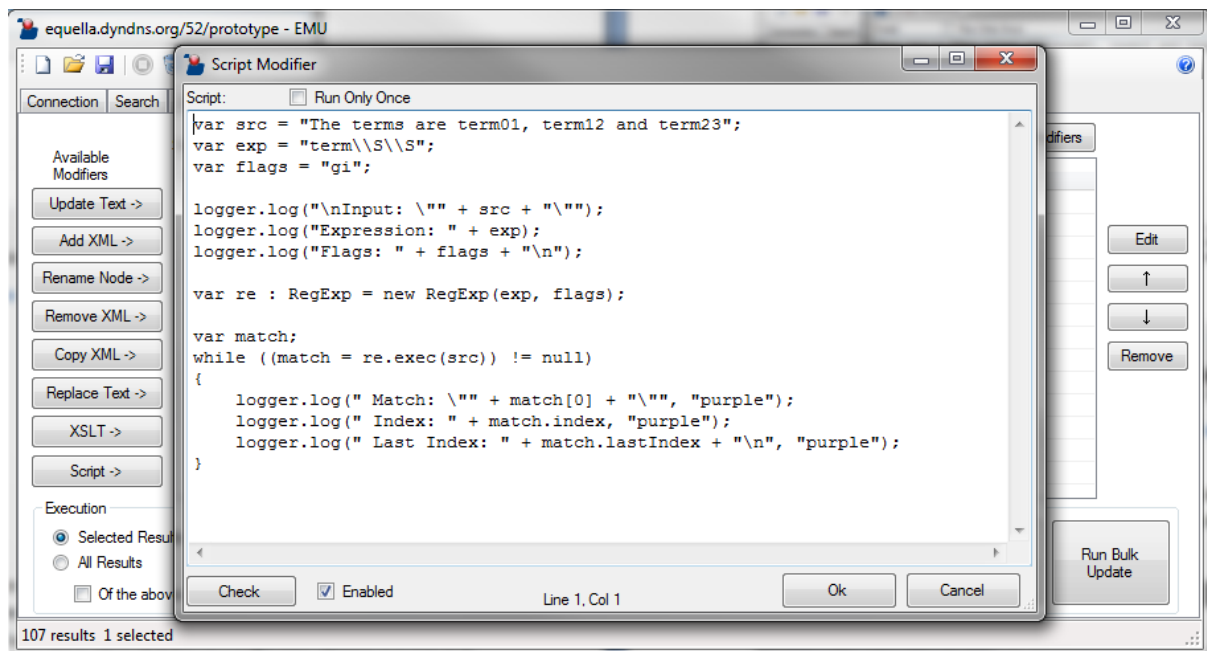


Figure 28. An example of a regular expression used in a Script modifier

The above script iterates through each match of the regular expression in the string and determines the matching text and its start and end positions within the string. The result of this example is the following text output to the log:

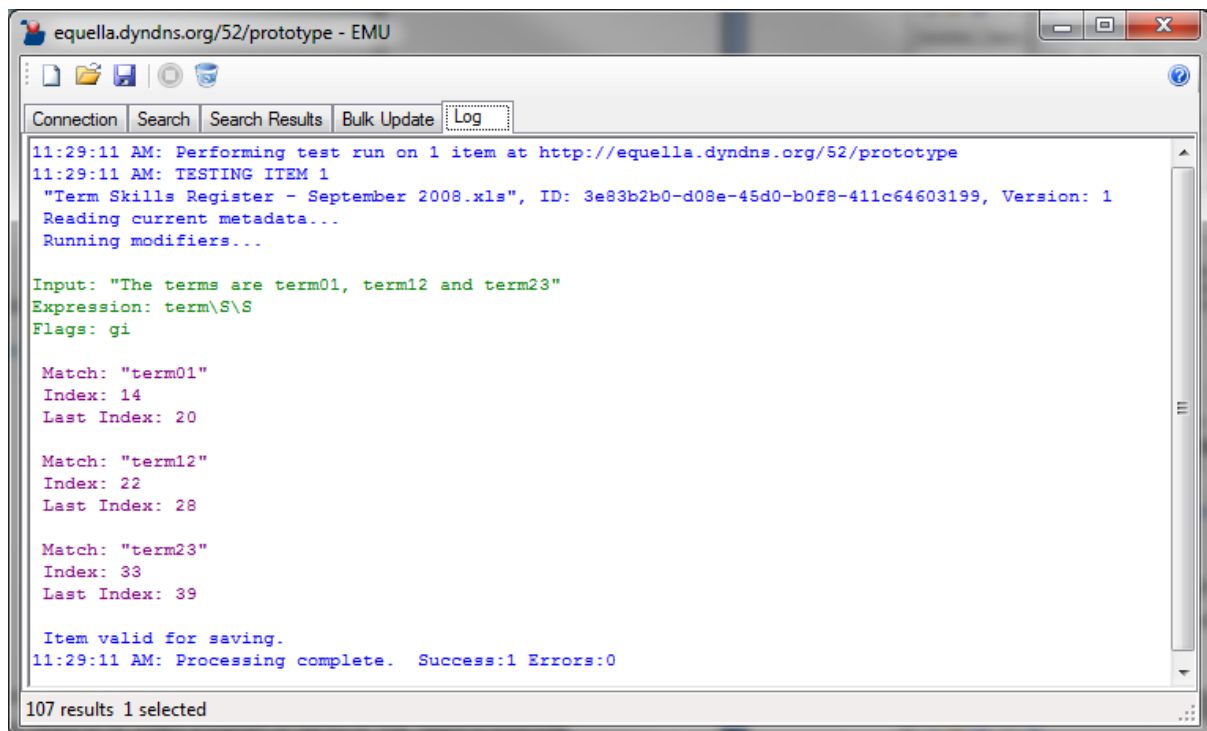


Figure 29. The log output from the regular expression Script modifier example

8.6 Generating UUIDs

It is possible with EMU scripting to produce UUIDs. Amongst other uses, this can be needed when working with attachment metadata.

The following Script modifier generates a UUID and adds it to the `vars` object:

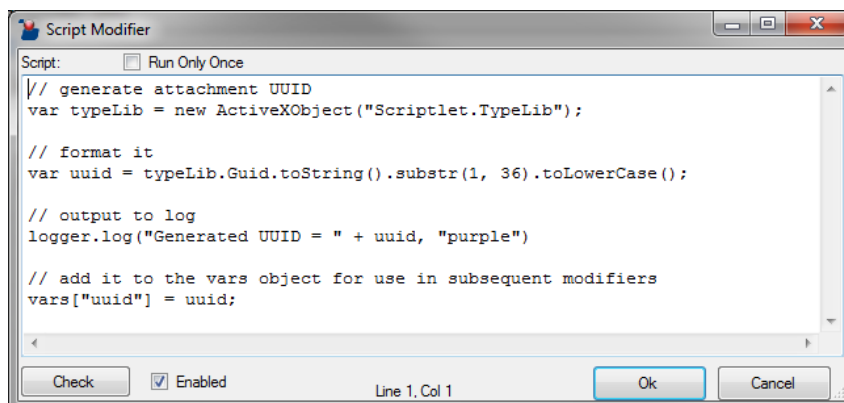


Figure 30. An example of generating a UUID in a Script modifier

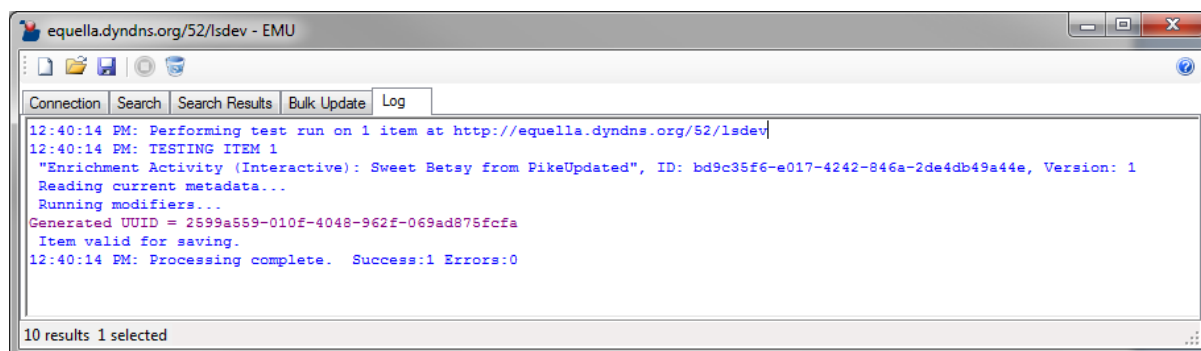


Figure 31. The log output from the UUID-generation Script modifier example

9 Appendix B: XPath Notation

XPath is a syntax for referencing parts of an XML document. It uses path expressions to navigate in XML documents. XPath is a W3C recommendation and EMU supports XPath version 1.0. The specification for XPath 1.0 can be found at <http://www.w3.org/TR/xpath>

In EMU XPath expressions allows you to reference nodes in the XML metadata of an individual item.

9.1 XML Metadata

Following is an example of an item's metadata.

```
<xml>
  <item id="f980a911-010d-35af-f480-deca7fd5530c" itemdefid="6e85ce64-9a11-c5e7-
    69a4-bd30ec61007f" itemstatus="live" key="2" version="1">
    <name>EQUELLA web site</name>
    <description>
      With product and service information, eNewsletters and links to
      support materials, the EQUELLA web site is a key resource for new
      clients. Come and check it out!
    </description>

    ...
  </item>
  <metadata>
    <lom>
      <general>
        <keyword>
          Pearson north america europe group
          TLEG TLEI TLEE TLENA
        </keyword>
      </general>
      <technical>
        <format>other</format>
      </technical>
    </lom>
    <name>EQUELLA web site</name>
    <description>
      With product and service information, eNewsletters and links to support
      materials, the EQUELLA web site is a key resource for new clients. Come
      and check it out!
    </description>

    ...
  </metadata>
</xml>
```

Nodes can either be **elements** or **attributes**. An element is a type of node that can contain either text, attributes or more elements ("child" elements). Following are some examples of elements and attributes in the above example.



Elements are characterized by an opening tag followed by text and then a closing tag e.g.

`<myelement>Some text</myelement>`. An element may have no text in which case it may be specified either as `<myelement></myelement>` or simply `<myelement/>`. Both these formats are equivalent.

Attributes are essentially named values, for example `version="1"`. In this case `version` is the attribute name and `1` is the attribute value. Attributes can only exist within an opening tag of an element e.g. `<item version="1">`

Elements can contain more elements inside them. This is often called "nesting elements":

```
<major_element>
  <minor_element>
    some text
  </minor_element>
</major_element>
```

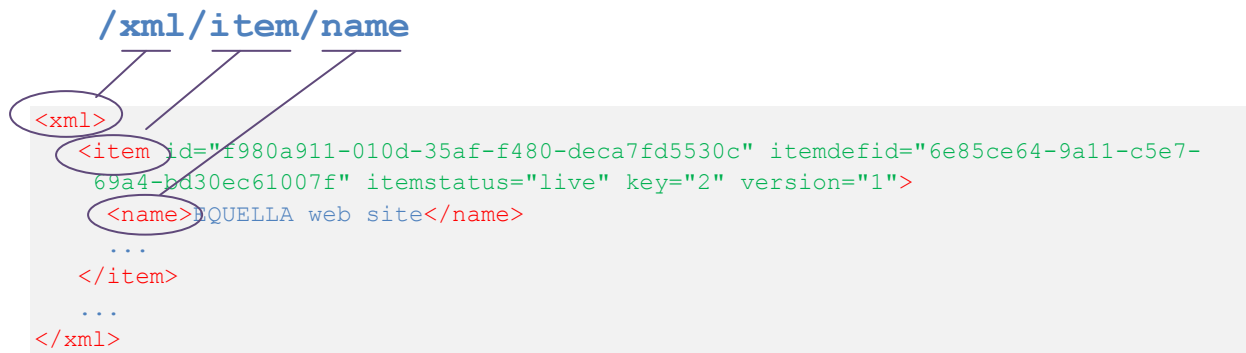
In the above example the `minor_element` element is *nested* within `major_element`. `major_element` has no text and `minor_element` has the text "some text" (neither element has any attributes in this example). In this context `major_element` is often called the "parent node" and `minor_element` the "child node".

9.2 Accessing Nodes with XPath Expressions

Nodes in an XML document can be accessed with XPath expressions very similar to how files and directories can be accessed in a file system with file paths. At its simplest an XPath expression should reference each parent node above the node in question. Consider the following example:

```
/xml/item/name
```

The above XPath will reference the `name` element which is a child of `item` which in turn is a child of `xml`.



Attributes can be accessed in the same way with the qualifier that the attribute's name should be prefixed with the "@" symbol, e.g.:

```
/xml/item/@itemstatus
```

In EMU all XPath expressions should be a full XPath from the root of the XML document. This means that all XPath expressions in EMU should start with `/xml/`.

9.3 Selecting Particular Elements from Multiple Elements

EQUELLA supports repeating metadata fields.

```

<xml>
  <metadata>
    <subject_areas>
      <subject>Mathematics</subject>
      <subject>Science</subject>
      <subject>English</subject>
      <subject>Arts</subject>
      <subject>Liberal Arts</subject>
    </subject_areas>
  </metadata>
</xml>

```

In the above example the `/xml/metadata/subject_areas/subject` element has multiple instances i.e. it *repeats*. Each instance has a different text value, "Mathematics", "Science", "English", "Arts" etc. This type of XML is easily generated by EQUELLA by using any controls that have multiple selection capabilities. Controls with this capability include Check Box Group, Shuffle Box, Shuffle List, Taxonomy Control and the Repeater.

XPath allows you to specify select elements from a range of elements: The following XPath selects the second `<subject>` element:

```
/xml/metadata/subject_areas/subject[2]
```

The "[2]" denotes that we are referencing the second `<subject>` element.

The following XPath selects all `<subject>` elements that have text that equals "Arts"

```
/xml/metadata/subject_areas/subject[text()='Arts']
```

Here are some useful ways of retrieving select elements from an item's metadata:

XPath	Purpose
<code>/xml/metadata/subject_areas/subject</code>	All <code><subject></code> elements that are children of <code><subject_areas></code>
<code>/xml/metadata/subject_areas/subject[2]</code>	The second <code><subject></code> element
<code>/xml/metadata/subject_areas/subject[text()='Arts']</code>	All <code><subject></code> elements where the element's text equals "Arts"
<code>/xml/metadata/subject_areas/subject[contains(., 'Arts')]</code>	All <code><subject></code> elements where the element's text contains the text "Arts"
<code>/xml/metadata/subject_areas/subject[text()='Arts' or text()='Liberal Arts']</code>	All <code><subject></code> elements where the element's text equals "Arts" or "Liberal Arts"
<code>/xml/metadata/attachments/attachment[substring(file, string-length(file) - 3) = '.zip']</code>	All <code><file></code> elements where the element's text ends with the text ".zip"
<code>/xml/metadata/contributors/contributor[last_name='Smith']</code>	All <code><contributor></code> elements with a child element <code><last_name></code> whose text equals "Smith"
<code>/xml/metadata/contributors/contributor[last_name='Smith' or last_name='Smythe']</code>	All <code><contributor></code> elements with a child element <code><last_name></code> whose text equals "Smith" or "Smythe"
<code>/xml/metadata/contributors/contributor[last_name='Smith']/@id</code>	The <code>id</code> attribute of all <code><contributor></code> elements with a child element <code><last_name></code> whose text equals "Smith"
<code>/xml/metadata/contributors/contributor[last_name='Smith' and first_name='Sam']/@id</code>	The <code>id</code> attribute of all <code><contributor></code> elements with a child element <code><last_name></code> whose text equals "Smith" and a child element <code><first_name></code> whose text equals "Sam"
<code>/xml/metadata/*</code>	All child elements of <code><metadata></code>
<code>/xml/metadata/* @*</code>	All child elements and attributes of <code><metadata></code>
<code>/xml/metadata//*</code>	All descendent elements of <code><metadata></code>
<code>/xml//@lang</code>	All <code>lang</code> attributes in the document regardless of where they appear
<code>/xml/metadata/*[@*]</code>	All child elements of <code><metadata></code> that have one or more attributes

<code>/xml/item/attachments/attachment[not(thumbnail)]/file</code>	The <code><file></code> element of all attachments that do not have a <code><thumbnail></code> element
--	--

10 Appendix C: EMU Object Model Reference

<p>logger</p> <p>An object that provides control over the event log.</p>
<pre>void logger.log(string message)</pre> <p>Appends text to the event log in the default color. The message will be updated to the log on screen and/or the file log depending on the log settings in EMU.</p>
<pre>void logger.log(string message, string color)</pre> <p>Appends text to the event log in the specified color. Allowed colors are anything from the .NET <code>color</code> Structure (see http://msdn.microsoft.com/en-us/library/system.drawing.color.aspx). Examples for <code>color</code> are "green", "blue", "purple", "red" etc. The message will be updated to the log on screen and/or the file log depending on the log settings in EMU.</p>
<pre>void logger.log(string message, string color, int loggingOptions)</pre> <p>Appends text to the event log in the specified color and to the specified media. See above for the <code>color</code> parameter. Logging options are:</p> <ul style="list-style-type: none"> 0 – Display only 1 – File only 2 – Display and file <p>Using a value of 1 or 2 will only output to file if EMU logging is set for logging to file.</p>
<p>modifiers</p> <p>An object that allows control of the execution of modifiers for each item being processed</p>
<pre>bool modifiers.skipNext</pre> <p>When set to true the modifier immediately following the one the script is in will be skipped regardless of whether it is enabled or disabled.</p>
<pre>bool modifiers.skipRemaining</pre> <p>When set to true all modifiers following the one the script is in will be skipped regardless of whether they are enabled or disabled.</p>
<pre>bool modifiers.cancelSave</pre> <p>When set to true the item will not be saved.</p>
<p>xml</p> <p>An object that allows reading and updating of an item's XML.</p>
<pre>string xml.getNode(string xpath)</pre> <p>Returns the value of the first matching node of the given XPath. Returns an empty string if no matches are found.</p>

<code>ArrayList<string> xml.getNodes(string xpath)</code>
Returns a list of values for the given XPath.
<code><XMLWrapper> xml.getSubtree(string xpath)</code>
Returns the first occurrence of an element for the given XPath. The object returned is of the same type as <code>xml</code> and null if no matches are found.
<code>ArrayList<XMLWrapper> xml.getSubtrees(string xpath)</code>
Returns a list of objects of the same type as <code>xml</code> for the given XPath.
<code>void xml.updateText(string xpath, string updateText)</code>
Modifies the item metadata as the Update Text modifier does.
<code>void xml.updateText(string xpath, string updateText, bool createNode)</code>
Modifies the item metadata as the Update Text modifier does. If <code>createNode</code> is true then elements in the XPath will be created as required.
<code>void addXml(string xpath, string addXML)</code>
Modifies the item metadata as the Add XML modifier does.
<code>void addXml(string xpath, string addXML, bool createNode)</code>
Modifies the item metadata as the Add XML modifier does. If <code>createNode</code> is true then elements in the XPath will be created as required.
<code>void renameNode(string currentXPath, string renamedNode)</code>
Modifies the item metadata as the Rename Node modifier does.
<code>void removeXml(string xpath)</code>
Modifies the item metadata as the Remove XML modifier does.
<code>void copyXml(string sourceNode, string targetXPath)</code>
Modifies the item metadata as the Copy XML modifier does.
<code>void copyXml(string sourceNode, string targetXPath, bool createNode)</code>
Modifies the item metadata as the Copy XML modifier does. If <code>createNode</code> is true then elements in the XPath will be created as required.
<code>void replaceText(string xpath, string findText, string replaceWithText)</code>
Modifies the item metadata as the Replace Text modifier does.
<code>void runXslt(string xslt)</code>

Modifies the item metadata as the XSLT modifier does.
<pre>string xml.asString()</pre> <p>Returns the XML of the item as an unformatted string.</p>
<pre>string xml.asFormattedString()</pre> <p>Returns the XML of the item as a formatted string including indenting and carriage returns.</p>
<pre>XmlDocument dom</pre> <p>A property which is a reference to the underlying .Net XmlDocument (see http://msdn.microsoft.com/en-us/library/system.xml.xmldocument(v=vs.110).aspx)</p>
<p>vars</p> <p>A global object of type HashTable (see http://msdn.microsoft.com/en-us/library/system.collections.hashtable(v=VS.90)) that is created at the beginning of each bulk update run.</p> <p>This variable is provided to allow script author's to create variables that have scope across modifiers and across items during a bulk update run. For example one script modifier might create a "variable" using the code such as</p> <pre>vars["authorAddress"] = "54 Brooklyn Drive, Boston, MA 02111";</pre> <p>then a subsequent modifier could access the variable using code such as</p> <pre>xml.updateText("/xml/metadata/author/address", vars["authorAddress"]);</pre> <p>If the Run Only Once checkbox on the modifier of the first script is checked then <code>vars["authorAddress"]</code> will be assigned once for the entire bulk update run. This is efficient for constant values. It is also useful for variables that are dependent on their position amongst other modifiers or items (e.g. a zero-based counter for a script that assigns an incremental unique identifier to each item).</p>