

Using MadCap Flare's Find and Replace Feature to its Full Potential

PRESENTED BY

Peter Kelley

MW

A BIT ABOUT ME



VISA

- I'm a Director in the Business Operations group within Client Services at Visa. I have 25 years of experience in the software industry managing various software applications
- I'm what I like to refer to as a "hybrid," which means I'm half business and half technical and I can translate between the two



HOW WE USE FLARE

- We produce a large PDF publication (1,100+ pages) of the Visa Rules twice annually. There are two versions, one for our clients and a redacted public version. We also produce another publication from the same source and several translations, with more of both to come.
- We have 20+ authors globally that edit our content using a controlled and auditable process.
- We don't have any web output at the moment, but that may be a possibility in the future.



THE PROBLEM

- We were converting our data to Flare from a custom software application.
- We imported that data into Flare from a DITA format, which resulted in a large project with 2,500+ topics.
- All the data we needed was there, but it wasn't all in the format that we needed it in.
- Reformatting the data manually would have been a very time consuming process to say the least!



FIND AND REPLACE TO THE RESCUE!

- We used Flare’s “Find and Replace in Files” feature along with Regular Expressions (Regex) to quickly reformat the data into the format we needed.
- This allowed us to not only save a lot of time, but it ensured a consistent format throughout all of the topics and eliminated mistakes due to human error.
- It also introduced us to a great feature that we continue to use during our publication cycle and as we bring up new projects.



WHAT I'LL COVER

- Today I'll demonstrate several examples of how we did this as well as touch on how we continue to use this powerful and versatile feature today.
- First, I'll actively show you how to apply a fairly simple change to many topics at once using *Replace All*.
- Then, I'll demo a few statements that are more complex and really show the power of this feature, and I'll walk through those in a single topic.
- Lastly, I'll provide some tips, strategies, and resources.

FIRST, A BIT ABOUT REGULAR EXPRESSIONS

- Using Regular Expressions with Find and Replace allows you to search for patterns in your *Find* statement and then use those same patterns in your *Replace with* statement. For example, let's say you need to update the date in this highlighted HTML across multiple topics:

```
<li class="table-rule-lastupdate-edition" MadCap:autonum="Edition: Oct 2018"> </li>  
<li MadCap:autonum=" | &#160;Last Updated:"> Oct 2014</li>
```

- You can use this as your *Find* statement:
`Updated:"> (.*)`

HOW IT WORKS

```
<li class="table-rule-lastupdate-edition" MadCap:autonum="Edition: Oct 2018"> </li>  
<li MadCap:autonum=" | &#160;Last Updated:"> Oct 2014</li>
```

Updated:"> (.*)

- The Updated:"> (including the trailing space) and the find exactly that text, just like they would using the *Regular Text* option.
- The (.*) is where Regular Expressions come in. This is called a capture group, and it finds any text between the matched pattern on each side. So in this case, it finds the Oct 2014 date between Updated:"> and .

HOW IT WORKS (CONT.)

```
<li class="table-rule-lastupdate-edition" MadCap:autonum="Edition: Oct 2018"> </li>  
<li MadCap:autonum=" | &#160;Last Updated:"> Oct 2014</li>
```

- The `(.*)` capture group will capture any text in between `Updated:">` and ``, so if I have this same HTML in all my topics, but with different dates in different topics, it will find all of them. Even if I have different text altogether, say `New` for example, it will find that too.
- It's called a capture group as Regular Expressions will remember this text and it can be used in our *Replace with* statement if we want to keep that text. More on that later.



Time for a Demo!

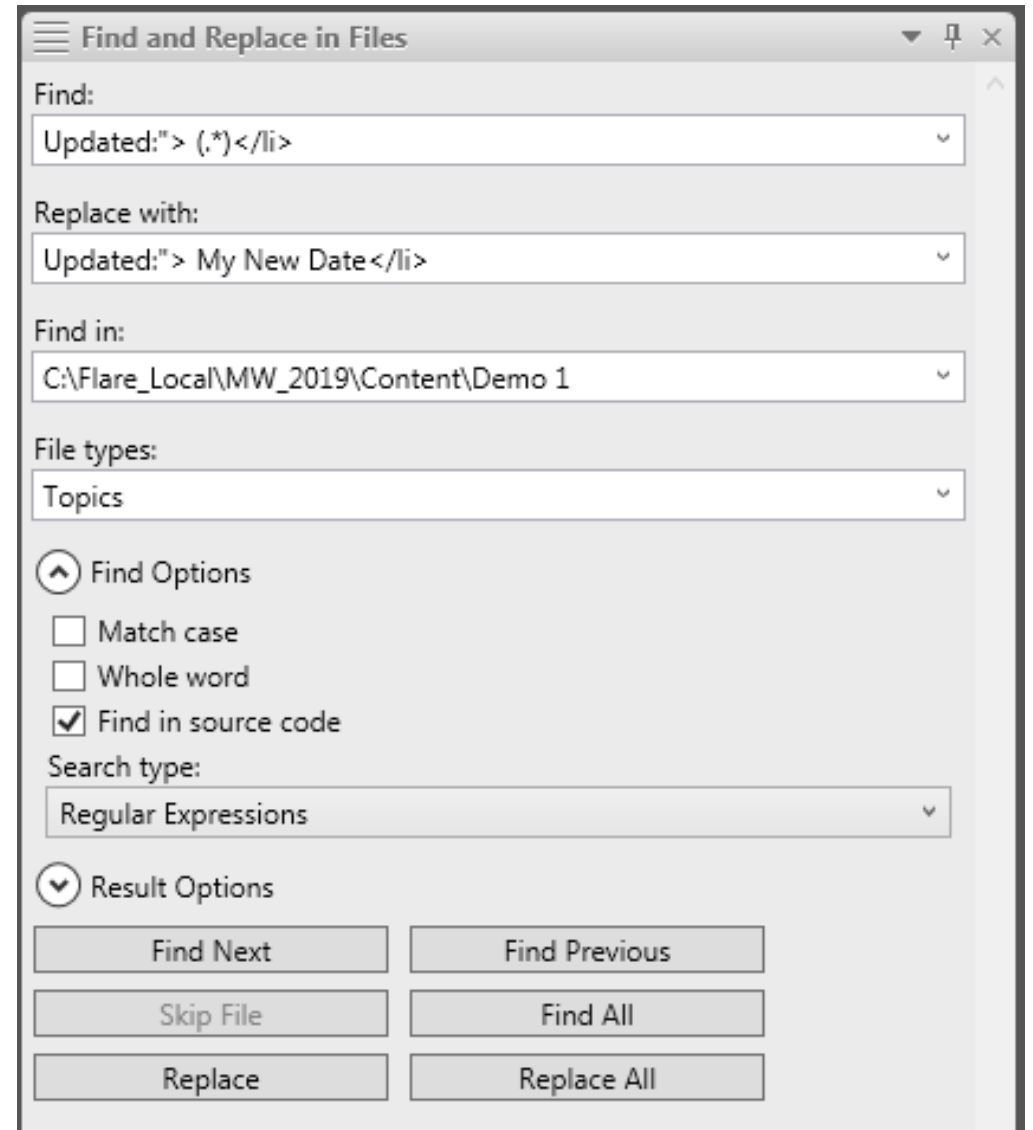
Let's run through this in Flare and see how it works.

CAPTURE GROUP DEMO

- To start, we'll do the following:
 - Open *Find and Replace in Files* from the *Home* menu
 - Select *(pick a folder)* from *Find in* and select our "Demo 1" folder
 - Select *Topics* from *File types*
 - Check *Find in source code* from the *Find Options*
 - Select *Regular Expressions* from *Search type*
- Then we'll enter our *Find and Replace with* statements:
 - **Find** Updated:"> (.*)
 - **Replace with** Updated:"> My New Date

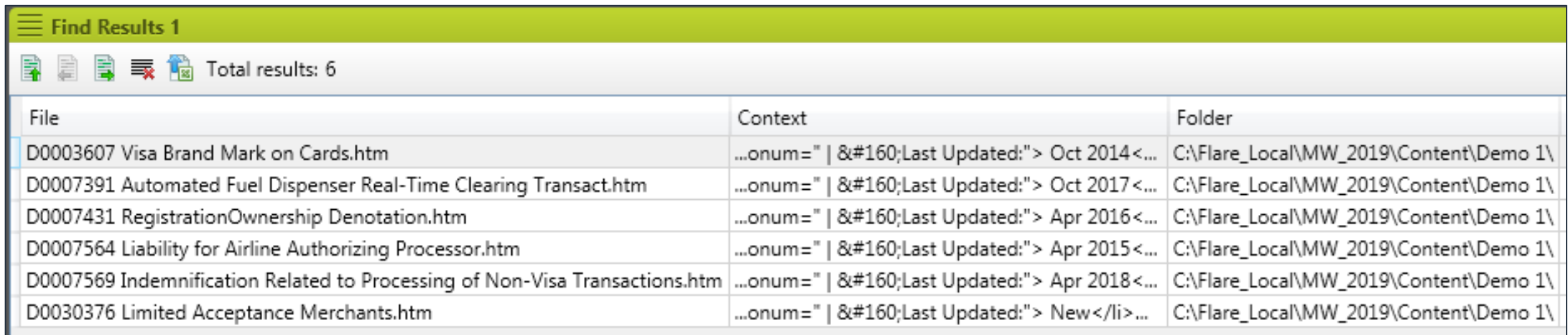
Capture Group Demo (cont.)

Here's how it
should look in Flare
once you set it up.



CAPTURE GROUP DEMO (CONT.)

- In this demo, the text of **My New Date** replaces the capture group of `(.*)` for all the values it finds.
- Before we run the *Replace All*, we'll run a *Find All* to make sure we're getting the results we expected:

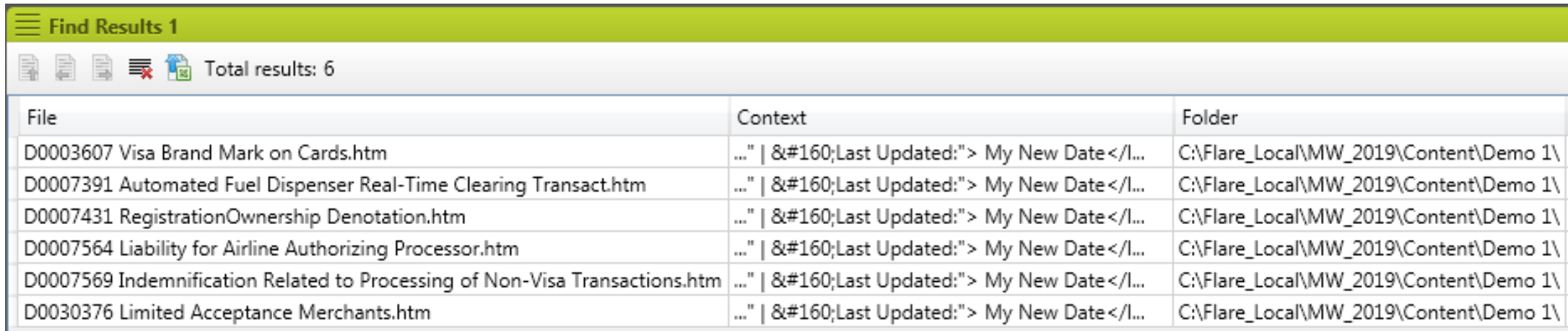


The screenshot shows a 'Find Results' window with a green header bar. Below the header, there are icons for file operations and a status bar that reads 'Total results: 6'. The main area contains a table with three columns: 'File', 'Context', and 'Folder'. The table lists six files, each with its corresponding context and folder path.

File	Context	Folder
D0003607 Visa Brand Mark on Cards.htm	...onum=" Last Updated:"> Oct 2014<...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0007391 Automated Fuel Dispenser Real-Time Clearing Transact.htm	...onum=" Last Updated:"> Oct 2017<...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0007431 RegistrationOwnership Denotation.htm	...onum=" Last Updated:"> Apr 2016<...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0007564 Liability for Airline Authorizing Processor.htm	...onum=" Last Updated:"> Apr 2015<...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0007569 Indemnification Related to Processing of Non-Visa Transactions.htm	...onum=" Last Updated:"> Apr 2018<...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0030376 Limited Acceptance Merchants.htm	...onum=" Last Updated:"> New...	C:\Flare_Local\MW_2019\Content\Demo 1\

CAPTURE GROUP DEMO (CONT.)

- Now we'll run our *Replace All*, and have a look at the results:



File	Context	Folder
D0003607 Visa Brand Mark on Cards.htm	... Last Updated:"> My New Date</l...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0007391 Automated Fuel Dispenser Real-Time Clearing Transact.htm	... Last Updated:"> My New Date</l...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0007431 RegistrationOwnership Denotation.htm	... Last Updated:"> My New Date</l...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0007564 Liability for Airline Authorizing Processor.htm	... Last Updated:"> My New Date</l...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0007569 Indemnification Related to Processing of Non-Visa Transactions.htm	... Last Updated:"> My New Date</l...	C:\Flare_Local\MW_2019\Content\Demo 1\
D0030376 Limited Acceptance Merchants.htm	... Last Updated:"> My New Date</l...	C:\Flare_Local\MW_2019\Content\Demo 1\

- You can see that all of the values have been replaced with **My New Date**.

More Complex Statements

Now I'd like to move on to two more complex examples, but first I want to walk through some of the additional syntax I'll be using.

CARRIAGE RETURNS & SPACES

- Because of the way Flare formats the HTML, we needed to find HTML that spanned multiple rows. This means accounting for carriage returns (line breaks) and spaces.
- For example, if I want to find the <head> tags that contain my titles, I would be looking for this:

```
3 | | <head><title>CPS/Rewards 1 and CPS/Rewards 2 for Visa Signature Card Transactions - US Region</title>
4 | |   <link href="../../VCR-VPSR.css" rel="stylesheet" />
5 | | </head>
```

- And to find it, I would use this:

```
<head><title>(.*?)</title>\r\s{1,}<link href="../../VCR-VPSR.css" rel="stylesheet" />\r\s{1,}</head>
```


CARRIAGE RETURNS & SPACES (CONT.)

```
3 <head><title>CPS/Rewards 1 and CPS/Rewards 2 for Visa Signature Card Transactions - US Region</title>
4   <link href="../../../VCR-VPSR.css" rel="stylesheet" />
5 </head>
```

- It looks complicated, but it's really not, let's break it down:
`<head><title>(.*?)</title>\r\s{1,}<link href="../../../VCR-VPSR.css" rel="stylesheet" />\r\s{1,}</head>`
 - Everything in blue, the `<head>` and `<title>` tags and the `link href` is just doing a regular text search.
 - The `(.*)` is my capture group which is pulling in my really long title.
 - The `\r\s{1,}` is what's new. The `\r` looks for a carriage return. The `\s` looks for a space, and the `{1,}` tells it to not look just for one space, but one to any number of spaces in a row.
- So, it's not as bad as it looks!

File Tags Example Demo

This demo will show how we removed unneeded data left over from our DITA import and reformatted our metadata into File Tags that Flare will recognize.

FILE TAGS BEFORE

We'll start with this:

```
3 <html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" class="topic" MadCap:
  sourceDocument="C:\Flare\Input\VCR-VPSR\Visa Core Rules\General\Governance\0007428 Use of the Visa
  Rules.dita">
4   <head><title>Use of the Visa Rules</title>
5     <link href="../../../VCR-VPSR.css" rel="stylesheet" />
6   </head>
7   <body>
8     <h1 class="topictitle">Use of the Visa Rules</h1>
9     <p class="tag">MadCap:fileTags="Date Effective.2010-04-01,Last Updated.2014-10-15,Redaction
  Classification.Public,Region.AP,Region.Canada,Region.CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data
  Classification.Visa Confidential,Rule Owners.Peter Kelley,Rule ID.7428"</p>
```

We needed to keep the <html> tag and move the File Tags within it, while almost everything else was unneeded.

FILE TAGS AFTER

And we'll end up with this—File Tags that Flare can use:

```
3 <html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" MadCap:fileTags="Date  
Effective.2010-04-01,Last Updated.2014-10-15,Redaction Classification.Public,Region.AP,  
Region.Canada,Region.CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data Classification.Visa  
Confidential,Rule Owners.Peter Kelley,Rule ID.7428"><body><h1>Use of the Visa Rules</h1>
```

File Tags		Comment
Private	<input type="checkbox"/>	
Public	<input checked="" type="checkbox"/>	

FILE TAGS DEMO

- To start, we'll do the following:
 - Select *(current document)* from *Find in*
 - Check *Find in source code*
 - Select *Regular Expressions* from *Search type*.
- Here are the first statements which remove the `<head>`:
 - **Find** `>\r\s{1,}<head><title>(.*?)</title>\r\s{1,}<link href="..\..\..\VCR-VPSR.css" rel="stylesheet" />\r\s{1,}</head>`
 - **Replace with** `>`
- The *Find* is almost identical to the one we just broke down.

FILE TAGS DEMO (CONT.)

- You can see the highlighted text will be removed and replaced with just a closing angle bracket (>):

```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" class="topic" MadCap:
sourceDocument="C:\Flare\Input\VCR-VPSR\Visa Core Rules\General\Governance\0007428 Use of the Visa
Rules.dita">
  <head><title>Use of the Visa Rules</title>
    <link href="../../../VCR-VPSR.css" rel="stylesheet" />
  </head>
  <body>
    <h1 class="topictitle">Use of the Visa Rules</h1>
    <p class="tag">MadCap:fileTags="Date Effective.2010-04-01,Last Updated.2014-10-15,Redaction
Classification.Public,Region.AP,Region.Canada,Region.CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data
Classification.Visa Confidential,Rule Owners.Peter Kelley,Rule ID.7428"</p>
```

FILE TAGS DEMO (CONT.)

- So now it looks like this:

```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" class="topic" MadCap:sourceDocument="C:\Flare\Input\VCR-VPSR\Visa Core Rules\General\Governance\0007428 Use of the Visa Rules.dita">
  <body>
    <h1 class="topicitle">Use of the Visa Rules</h1>
    <p class="tag">MadCap:fileTags="Date Effective.2010-04-01,Last Updated.2014-10-15,Redaction Classification.Public,Region.AP,Region.Canada,Region.CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data Classification.Visa Confidential,Rule Owners.Peter Kelley,Rule ID.7428"</p>
```

- The next statement will be:
 - Find `class="topic" MadCap:sourceDocument="(.)">`
 - Replace with `class="topic">`

FILE TAGS DEMO (CONT.)

- This *Find* statement returns the path to the source DITA document from our initial import which we no longer need. The highlighted text below will be removed and replaced with the topic class from the *Find* (`class="topic">`):

```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" class="topic" MadCap:
sourceDocument="C:\Flare\Input\VCR-VPSR\Visa Core Rules\General\Governance\0007428 Use of the Visa
Rules.dita">
  <body>
    <h1 class="topicitle">Use of the Visa Rules</h1>
    <p class="tag">MadCap:fileTags="Date Effective.2010-04-01,Last Updated.2014-10-15,Redaction
Classification.Public,Region.AP,Region.Canada,Region.CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data
Classification.Visa Confidential,Rule Owners.Peter Kelley,Rule ID.7428"</p>
```


FILE TAGS DEMO (CONT.)

- So now it looks like this:

```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" class="topic">
  <body>
    <h1 class="topicitle">Use of the Visa Rules</h1>
    <p class="tag">MadCap:fileTags="Date Effective.2010-04-01,Last Updated.2014-10-15,Redaction
Classification.Public,Region.AP,Region.Canada,Region.CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data
Classification.Visa Confidential,Rule Owners.Peter Kelley,Rule ID.7428"</p>
```

- We retained the `class="topic">` here so that we can use it in the next *Find* statement.

FILE TAGS DEMO (CONT.)

- The next set of statements will combine all of what we have used so far and will use our capture groups in the *Replace with* statement in order to save that text and rearrange it within the HTML.
 - **Find** `class="topic">\r\s{1,}<body>\r\s{1,}<h1 class="topictitle">(.*?)</h1>\r\s{1,}<p class="tag">(.*?)</p>`
 - **Replace with** `\2><body><h1>\1</h1>`
- The `\2` and the `\1` in the last replace statement is how you replace the capture groups `(.*)`

FILE TAGS DEMO (CONT.)

- Just to refresh, the blue captures regular text, our capture groups of (.*) are in red, and the green `\r\s{1,}` is finding carriage returns and spaces.
 - Find `class="topic">\r\s{1,}<body>\r\s{1,}<h1 class="topictitle">(.*?)</h1>\r\s{1,}<p class="tag">(.*?)</p>`

```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" class="topic">
  <body>
    <h1 class="topictitle">Use of the Visa Rules</h1>
    <p class="tag">MadCap:fileTags="Date Effective.2010-04-01,Last Updated.2014-10-15,Redaction
Classification.Public,Region.AP,Region.Canada,Region.CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data
Classification.Visa Confidential,Rule Owners.Peter Kelley,Rule ID.7428"</p>
```

FILE TAGS DEMO (CONT.)

- Let's review our *Replace with* of `\2><body><h1>\1</h1>`:
 - It moves the File Tags into the HTML tag using the `\2`
 - It removes the paragraph tags and class around the File Tags
 - It moves the `<body>` tag below the File Tags
 - It removes the two classes (“topic” and “topictitle”)
 - It moves the heading into the body tag using the `\1`
- So now it looks like this, which is what we want:

```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" MadCap:fileTags="Date Effective. 2010-04-01, Last Updated. 2014-10-15, Redaction Classification. Public, Region. AP, Region. Canada, Region. CEMEA, Region. LAC, Region. US, Rule Type. VCR, Data Classification. Visa Confidential, Rule Owners. Peter Kelley, Rule ID. 7428"><body><h1>Use of the Visa Rules</h1>
```

FILE TAGS DEMO (CONT.)

So we went from this: **Find** `class="topic">\r\s{1,}<body>\r\s{1,}<h1 class="topictitle">(.*?)</h1>\r\s{1,}<p class="tag">(.*?)</p>`

```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" class="topic">
  <body>
    <h1 class="topictitle">Use of the Visa Rules</h1>
    <p class="tag">MadCap:fileTags="Date Effective.2010-04-01,Last Updated.2014-10-15,Redaction
Classification.Public,Region.AP,Region.Canada,Region.CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data
Classification.Visa Confidential,Rule Owners.Peter Kelley,Rule ID.7428"</p>
```

To this: **Replace with** `\2><body><h1>\1</h1>`

```
<html xmlns:MadCap="http://www.madcapsoftware.com/Schemas/MadCap.xsd" MadCap:fileTags="Date Effective.
2010-04-01,Last Updated.2014-10-15,Redaction Classification.Public,Region.AP,Region.Canada,Region.
CEMEA,Region.LAC,Region.US,Rule Type.VCR,Data Classification.Visa Confidential,Rule Owners.Peter
Kelley,Rule ID.7428"><body><h1>Use of the Visa Rules</h1>
```

Table Style Example Demo

This next demo will show how we applied a table style to our “Rule ID Bars” to update the look and feel.

RULE ID BAR BEFORE

We'll start with this:

```
<table class="table-rule">
  <tbody>
    <tr>
      <td class="table-rule-id">ID# 0007428</td>
      <td class="table-rule-lastupdate">
        <ul>
          <li>Edition: Oct 2018</li>
          <li>Last Updated: Oct 2014</li>
        </ul>
      </td>
    </tr>
  </tbody>
</table>
```

ID# 0007428	<ul style="list-style-type: none">• Edition: Oct 2018• Last Updated: Oct 2014
-------------	--

RULE ID BAR AFTER

And end up with this:

```
<table class="TableStyle-RuleID" style="mc-table-style: url('../Resources/Tablestyles/RuleID.css');" cellspacing="0">
  <tbody>
    <tr class="TableStyle-RuleID-Body-Body1">
      <td class="TableStyle-RuleID-BodyB-Column1-Body1">ID# 0007428</td>
      <td class="TableStyle-RuleID-BodyA-Column1-Body1">
        <ul>
          <li MadCap:autonum="Edition: Oct 2018"> </li>
          <li MadCap:autonum=" | &#160;Last Updated:"> Oct 2014</li>
        </ul>
      </td>
    </tr>
  </tbody>
</table>
```

ID# 0007428

▶ Edition: Oct 2018 ◀▶ | Last Updated: Oct 2014 ◀

TABLE STYLE DEMO

- This is all done with one statement:
 - **Find** `<table class="table-rule">\r\s{1,}<tbody>\r\s{1,}<tr>\r\s{1,}<td class="table-rule-id">(.*?)</td>\r\s{1,}<td class="table-rule-lastupdate">\r\s{1,}\r\s{1,}Edition: (.*?)\r\s{1,}Last Updated: (.*?)`

```
<table class="table-rule">
  <tbody>
    <tr>
      <td class="table-rule-id">ID# 0007428</td>
      <td class="table-rule-lastupdate">
        <ul>
          <li>Edition: Oct 2018</li>
          <li>Last Updated: Oct 2014</li>
        </ul>
      </td>
    </tr>
  </tbody>
</table>
```

TABLE STYLE DEMO (CONT.)

- And our *Replace with* statement:
 - **Replace with** `<table class="TableStyle-RuleID" style="mc-table-style: url('../Resources/Tablestyles/RuleID.css');" cellpadding="0"><tbody><tr class="Body-Body1"><td class="BodyB-Column1-Body1">\1</td><td class="BodyA-Column1-Body1"><li MadCap:autonum="Edition: \2"> <li MadCap:autonum=" | Last Updated: "> \3`

```
<table class="TableStyle-RuleID" style="mc-table-style: url('../Resources/Tablestyles/RuleID.css');" cellpadding="0"><tbody><tr class="Body-Body1"><td class="BodyB-Column1-Body1">ID# 0007428</td><td class="BodyA-Column1-Body1"><ul><li MadCap:autonum="Edition: Oct 2018"> </li><li MadCap:autonum=" | Last Updated: "> Oct 2014</li></ul>
```

TABLE STYLE DEMO (CONT.)

- Flare does a lot of the formatting work for you. When you first do the replace, it looks like this:

```
<table class="TableStyle-RuleID" style="mc-table-style: url('../Resources/Tablestyles/RuleID.css');" cellspacing="0"><tbody><tr class="Body-Body1"><td class="BodyB-Column1-Body1">ID# 0007428</td><td class="BodyA-Column1-Body1"><ul><li MadCap:autonum="Edition: Oct 2018"> </li><li MadCap:autonum=" | Last Updated: "> Oct 2014</li>
```

- Click the XML Editor tab, then click back to the Text Editor tab and Flare reformats it, even adding in needed HTML

```
<table class="TableStyle-RuleID" style="mc-table-style: url('../Resources/Tablestyles/RuleID.css');" cellspacing="0">  
  <tbody>  
    <tr class="TableStyle-RuleID-Body-Body1">  
      <td class="TableStyle-RuleID-BodyB-Column1-Body1">ID# 0007428</td>  
      <td class="TableStyle-RuleID-BodyA-Column1-Body1">  
        <ul>  
          <li MadCap:autonum="Edition: Oct 2018"> </li>  
          <li MadCap:autonum=" | &#160;Last Updated:"> Oct 2014</li>
```

Unique IDs & One More Demo

Using a unique ID in each of your topics means that you can easily retrieve and update them using Find and Replace



WHY USE A UNIQUE ID?

- We found that using a unique ID in our topic content and file names makes Find and Replace tasks and using the Quick Find more efficient, and also makes topic organization easier.
- If you have a lot of topics in the same folder, and you want to run a *Replace All* on just a subset, this is a way to easily do that if you have a list of the topics you want to update.
- If you don't want the unique ID to show in your topic content, you can just condition them out of your output.

UNIQUE ID DEMO

- Let's say you have a Last Updated Date in your topics that you want to update only for those topics that you have updated since your last publication.
- If you keep a list of the IDs for those topics, then you can use a *Find All* to identify them with regular expressions.
- Just string them together using the bar character (|) which acts as an OR operator in regex as shown below.

ID# 0003607|ID# 0007391|ID# 0007431|ID# 0007564

UNIQUE ID DEMO (CONT.)

- Click on the top result in your *Find Results*, then hold down the *Shift* key and double-click on the bottom result to open them all. Once open, you'll proceed with your *Find*
- For this exercise, I'll repeat our first demo:
`Last Updated:"> (.*)`
- Next, before entering your *Replace with* statement, change the *Find in* to *(all open documents)*.
- Make sure the *Find Results* contain all of the topics that you opened. You'll see in my demo, that's not the case.

UNIQUE ID DEMO (CONT.)

- Note that in one of my topics, there is an extra space between my colon and my quotation mark, so I'll fix that.

`Last Updated: "> Oct 2014`

- Once you have figured out any discrepancies and fixed them (if appropriate), you're ready to enter your *Replace with* statement. For this demo, I'll again use:

`Last Updated:"> My New Date`

- As a refresher, “**My New Date**” replaces the capture group of `(.*)` which in this case, is the text we want to change.

UNIQUE ID DEMO (CONT.)

- Since you have all of your topics open, even if you are not using source control, Flare won't automatically save your topics. This is nice as it gives you one last chance to review your changes in case you made any mistakes.
- When you are ready to save your changes, right-click on the title tab of one of your topics and select "*Close All Documents Except This One.*" You will then be prompted to save the changes and when you do, Flare will save the changes to all of them and you can just close the last one.

Capture Group Pitfalls

Remember that Regex uses pattern matching and if you have similar repeated patterns, you might capture more data than you want.

CAPTURE GROUP UNEXPECTED RESULTS

- We wanted to capture and reformat section references:

```
<MadCap:xref href="0000652 Anti-Money Laundering Program Requirement.htm"><i>Section 1,"Anti-Money Laundering Program Requirement"</i></MadCap:xref> and <MadCap:xref href="0000653 Visa Anti-Money Laundering Program - Member Requirements.htm"><i>Section 1,"Visa Anti-Money Laundering Program - Member Requirements"</i></MadCap:xref>
```

- So I used this to capture each section reference individually:
 - `<MadCap:xref href="(.*)<i>(.*)</i></MadCap:xref>`
 - But it captured the whole thing!
 - Instead, I needed to use a separate statement for each that made them unique:
 - `<MadCap:xref href="(.*)<i>(.*)</i></MadCap:xref>` and
 - and `<MadCap:xref href="(.*)<i>(.*)</i></MadCap:xref>`

Regex Resources and Tips

Feeling overwhelmed? Thinking this is too complicated? Don't worry, there's lots of help out there.



WHERE TO GO FOR HELP

- By now, you may be thinking, “Okay, this looks pretty cool, but how am I going to figure this out for my own projects?”
- Don’t worry, there’s lots of helpful resources, and basic strategies to use so you don’t break anything!
- The key thing for me to remember when I got started, is that using Regex for Find and Replace is all about pattern matching, and when I want to figure out how to match a new pattern, I just Google it.

REGEX ONLINE RESOURCES

- Regular-Expressions.info—This is a comprehensive resource: <https://www.regular-expressions.info/index.html>
- RegExLib.com—This is a library compilation site: <http://regexlib.com/> Their cheat sheet is especially useful: <http://regexlib.com/CheatSheet.aspx>
- RegexOne—Learn Regular Expressions with simple, interactive exercises: <https://regexone.com/>
- Regex Tutorial—A quick cheat sheet by examples: <https://medium.com/factory-mind/regex-tutorial-a-simple-cheatsheet-by-examples-649dc1c3f285>

ALWAYS BACK UP YOUR DATA FIRST!

- I can't emphasize this point enough for obvious reasons.
 - Before you run a *Replace All* against many topics, run just the *Find All* and use the *Export results to csv* button on the Find Results panel to save off the results so that you have a record of which topics you changed.
 - If you are not using source control, make a backup copy of your project first, as once you run a *Replace All*, Flare will save those changes for any files that you changed that are not open.
 - If you are using source control, you do have a fallback as Flare will checkout your topics upon running a *Replace All* and you can undo your checkout if you make a mistake.

TIPS AND BASIC STRATEGIES

- Always start with one topic, using the *(current document)* option under *Find in*. Once you have it down, then you can run it against some or all of your topics at once by using the other *Find in* options and changing the *File types* option to *Topics*.
In order to run a *Replace All* against just some of the topics in the same folder, you can open the topics you want to change first and then use the *Find in* option called *(all open documents)*.

Other Things to Note

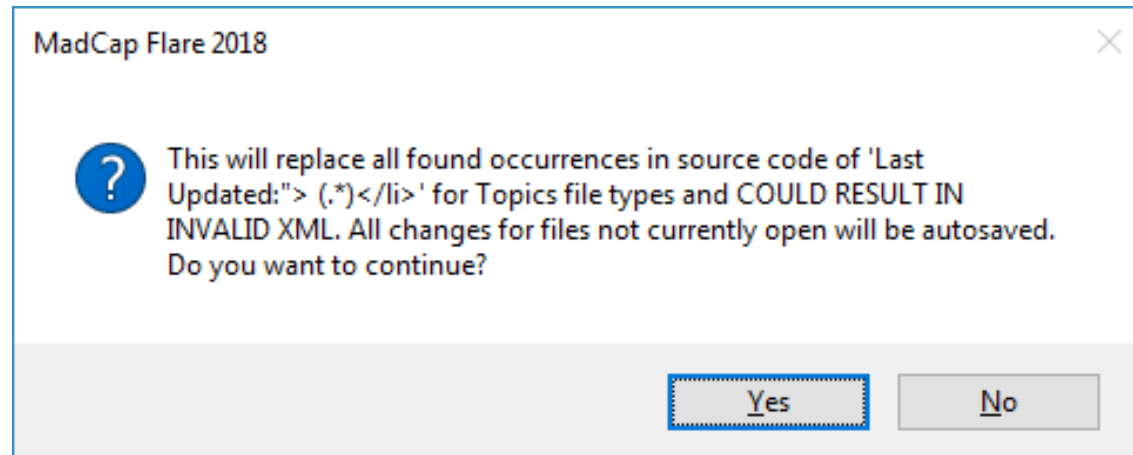
There are a few other miscellaneous items that may prove to be helpful as you use Find and Replace

“REGULAR TEXT” IS USEFUL TOO!

- The *Regular Text* option in the *Search type* is also handy.
- Every publication cycle, we use it for the following:
 - Replacing hyphens with en dashes (i.e., “–” instead of “-”).
 - Replacing regular quotes with smart quotes (e.g., “” instead of "").
We also use this for single quotes and apostrophes.
- Regular language replacement when a word or term changes in your project when you are not using a variable.
 - Remember to account for plural and past tense cases, etc.
 - This can be useful during translation reviews as well.

A FEW LAST TIPS

- Find and Replace remembers what you entered. If you click the drop-down arrow on the *Find* or *Replace with* fields, you can see (and reuse) your last 10 entries.
- The *Replace All* message is your friend! It reminds you there may be no going back...



Concluding Thoughts

I'm not a Regular Expressions expert. I figured it out with a little help from someone who already knew, and combined that with a little online research, and you can too!

Thank You!

Please feel free to contact me at pekelley@visa.com!

MW