Toolbox

Contents

DOCX2MD Converter

This Bash script is intended to prepare Baxter CCDS word files for DITA conversion. Custom `sed` expressions can be used to tune the result.

Dependencies:

- PANDOC
- BASH

Note: Ensure PATH variables for PANDOC are set

It does the following:

1. Creates a folder named output (if it doesn't already exist)
2. Runs PANDOC conversion to Markdown (incl. extensions for: extracting media, pipetables and atx-header)
3. Runs a series of GREP operations to tidy and/or mark text for further work
4. Moves extracted media from default 'media' to 'images' folder

Following the use of this script, it's expected that you will use a tool (e.g. OxgyenXML) to convert the Markdown output to DITA XML.

```
# DOCX2MD Converter
# by Owen Rowe 2021

# Create folder

if [ ! -d "output" ]; then
    mkdir output
fi

# Declare variables
declare path="output"
declare ofile="output.md"

# With any DOCX found, convert to Markdown

for filename in *.docx
do
 pandoc -f docx -t commonmark-pipe_tables --wrap=none --extract-media=$path
 --markdown-headings=atx $filename -o $ofile
done

# Normalise titles -> titlecase - needs work
# sed -i 's/#\s.*/\L&/; s/#.*[a-z]/\u&.*/g' $path/$ofile
# sed -i -E 's|(^#.*)|\L\1|;s|^(#\s*)([a-z])([^\s*) ([a-z])|\1\u\2\3\u\4|g'
# sed -i '1s#/sh$#/bash#' ftc.sh

# GREP deletions
sed -i -E 's|<!-- -->||;s|^[ \t]+<|<|' $ofile

# GREP fix image links
sed -i -E 's|output/media/|images/|g' $ofile

# Fix block quotes
sed -i -E 's|^>\s?||' $ofile

# Fix Topic IDs
```

```
#sed -i -E 's|'

# Fix bullet characters
sed -i -E 's|•|-|' $ofile

# Strip Heading Style from Table titles

# Fix level 1 headers
sed -i -E 's|^(# )(.*)|\U\1\2|' $ofile

# Normalise element text
sed -i -r -E 's|(</?)([^>]*)(>)|\1\L\2\E\3|g' $ofile

# Keyref instances of "[Product]"
sed -i -r -E 's|\\\[[Pp]roduct\\\]|<ph keyref="product_name"\/>|g' $ofile

# Fix newline at titles <=== DOES NOT WORK
# grep -P -e 's|\*\*\s+[\r\n]{1,2}*|BOSS|' $ofile

# Highlight for clean up 'See Section XX?.X?' text.
sed -i -E 's|\\?\[?\(([Ss]ee [Ss]ection [[:digit:]][[:digit:]]?\.?
[[:digit:]]?[^\n\(\)]*)\)\\?\]?|<required-cleanup remap="xref">\1</required-
cleanup>|g' $ofile

# Highlight footnotes for cleanup
sed -i -r 's|^\\\*{1,2}(\s?[^\n]*)|<required-cleanup remap="footnote">\1</
required-cleanup>|' $ofile

# Set mandatory <p> tag on bolded content
sed -i -r 's|^(-)?\s+?\*\*(.*)\*\*|\1 <p outputclass="mandatory">\2</p>|;s|
\*\*||g' $ofile

# Convert RA comments to <draft-comment> tag
sed -i -E 's|\\\[([^\\]*)\\\]|<draft-comment author="admin">\1</draft-
comment>|' $ofile

# Fix extraneous indents and header styles before tags(#)
sed -i -r -E 's|^[# ]*(<)|\1|' $ofile

# Flag <sup> tag for required-cleanup @footnote
sed -i -E 's|^\*{0,2}?(<sup>.{1,4}</sup>\s?[^\n\*]*)\*{0,2}?\.?|<required-
cleanup remap="footnote">\1</required-cleanup>|' $ofile

# Fix sections
#sed -i -r -E 's|^<u>(.*)</u>$|#### \1 {.section}|' $ofile

# Fix extraneous indents and header styles on empty lines
sed -i -r -E 's|^[# ]*$||' $ofile

# Strip blockquotes
sed -i -E 's|</?blockquote>||g' $ofile

# MODULES remove hash (#) on script (sed) lines to activate
# ==================================================
# Triple chamber bags
# sed -i -E -f resources/3Cbags.txt $ofile

# Splitter function (not generally required since Oxygen can do this better
 See: Refactoring>Convert Nested Topics to New Topics)
# csplit $path/$ofile '/^# /' {*} --suffix-format="%d.md" -f $path/section

# Move images
mv $path/media $path/images
```

Title Case Converter

Since bash's parameter expansion includes case modification, there's no need for sed. Just a short function:

```
tc() { set ${*,,} ; echo ${*^} ; }
```

Test (don't use quotes, since a title is typically no longer than a sentence, it shouldn't matter):

```
tc FOO bar
```

Output:

```
Foo Bar
```

Fancy version that avoids capitalizing some conjunctions, articles and such:

```
ftc() { set ${*,,} ; set ${*^} ; echo -n "$1 " ; shift 1 ; \
        for f in ${*} ; do \
            case $f in  A|The|Is|Of|And|Or|But|About|To|In|By) \
                    echo -n "${f,,} " ;; \
                *) echo -n "$f " ;; \
            esac ; \
        done ; echo ; }
```

Test:

```
ftc the last of the mohicans
```

Output:

The Last of the Mohicans

Supposing the script's name is `ftc.sh`, then: `sed -i '1s#/sh$#/bash#' ftc.sh` should work.

Gradle build

dita-ot-gradle

CMD `gradle dita`

```
plugins {
    id 'de.undercouch.download' version '4.1.1'
 id 'com.github.eerohele.dita-ot-gradle' version '0.7.1'
}

def ditaOtVersion = '3.6.1'

// Download and install DITA-OT

task downloadDitaOt(type: Download) {
    src "https://github.com/dita-ot/dita-ot/releases/download/
${ditaOtVersion}/dita-ot-${ditaOtVersion}.zip"
    dest new File(buildDir, "dita-ot-${ditaOtVersion}.zip")
    overwrite false
}

task extract(dependsOn: downloadDitaOt, type: Copy) {
```

```
    from zipTree(downloadDitaOt.dest)
    into buildDir
}

def ditaHome = "${buildDir}/dita-ot-${ditaOtVersion}"

// Install DITA-OT plugins from DITA-OT Plugin Registry

def plugins = ['org.lwdita', 'org.dita.normalize']

plugins.each { id ->
    tasks.create(id, Exec) {
        // Install plugin if not already installed, otherwise build will
 fail
        onlyIf { !file("$ditaHome/plugins/$id").exists() }
        outputs.dir("$ditaHome/plugins/$id")
        workingDir ditaHome
        commandLine 'bin/dita', '--install', id
    }
}

task install(dependsOn: [extract, plugins])

// Publish my.ditamap into the PDF output format.
dita {
    dependsOn install
    ditaOt ditaHome
    singleOutputDir true
    useAssociatedFilter true
    input fileTree(dir: '.', include: '*.ditamap')
    transtype 'pdf2'
}

defaultTasks 'dita'
```

General expressions

CAUTION:

these are only *lightly* tested - be sure to confirm you have the desired result before proceeding further.

Convert `[PRODUCT]` to DITA key

Find	Replace
`\[(PRODUCT|Product])\]`	`[product_name]`

Convert RA style comments to draft comments:

Find	Replace
`\[([^\]*)\]`	`<draft-comments>$1</draft-comments>`

Convert RA style conditions to DITA element for later clean up

Find	Replace
`\<([^\]*)\>`	`<required-cleanup>$1</required-cleanup>`

Standardise to capitals on Titles (#->####)

*NB: works in Notepad++, **does not** work in JAVA Regex (e.g. OxygenXML)*

Find	Replace
`^(#{1,4}\s)(.+)`	`$1\U$2`

Capitals to title case

Find	Replace
`(#[ ]*)([A-Z])(.+)`	`$1\U$2\L$3`

Find trailing spaces after * and ** (a Word conversion gotcha - can affect bold and italics)

Find	Replace
`((?<=\s)|(?<=^))(*+[\w !"#$%&'()+,-./:;<=>?@[\]^_{|}~]+)(\s+)(*+) |`	`$2$4$3`

Find extra spaces in lists (ensure a single [space] occurs at the end of the replace expression).

Find	Replace
`\n-\s+`	`\n-`

Tidies superfluous carriage returns.

Find a newline *not* after a period, followed by 2 (or more) spaces and preceding one lowercase letter, number, or "(". Remove newline, trim to one (1) space. *NB: not fully supported by all Regex engines (works for Oxygen though)

Find	Replace
`(?!<=.)\n^(\s){2,}(?=\p{Ll}|\d|()`	`$1`

Find asterisks in text and leave a comment for a fix (will drop a comment on the line above the "*")

Find	Replace
`(^.*\*)`	`<draft-comment>Clean up needed</draft-comment>\n$1`

Find newline followed by a lower case (a-z) letter (note space before `[space]$1` in Replace expression)

Find	Replace
`(?<=\w)\n^([\w])`	`$1`

Find **Notes** (or similar) and convert to DITA compliant tags (esp. device labels)

Find	Replace
`**NOTE:** ([^\n]*)`	`<note>$1</note>`

Find codeblocks ">" before an image (leave replace query blank)

Find	Replace
`^> (?=!)`	`[blank]`

Adjust table header row of table

Captures content after the `table` tag up to the end of the next `tr` tag. Replaces the first `tbody` tag with `thead`; appends closing `thead` and reopens `tbody` at this new position.

Find	Replace
`((?<=<table>)[^<]*)(<tbody>)(.*?(?<=</tr>))`	`$1<thead>$3\r<\thead><tbody>`

Find/replace `codeph` elements that can be specialised to `xmlatt` or `xmlelement`

Find	Replace
`<codeph>@([^<]*)</codeph>`	`<xmlatt>$1</xmlatt>`
`<codeph></?([^\>]*)\?></codeph>`	`<xmlelement>$1</xmlelement>`

Convert bolded titles to markdown style

Finds single line bolded strings starting with a digit and prepends with markdown style header symbol i.e. #

Find	Replace
`**(\d\d?.?\s)([^*]*)**`	`## $1$2`
`**(\d\d?.\d\d?\s)([^*]*)**`	`### $1$2`
`**(\d\d?.\d\d?.\d\d?.?\s)([^*]*)**`	`#### $1$2`

Create IDs for Headers

Find	Replace
`^(#+\s[0-9.]*)([^A-Z]+)([\w -,/]*(?!{)$)`	`$1$2$3 {#\L$3}`
`(?<={#)(.*?)[^a-z-]+(.*?)(?=})`	`$1-$2`

Convert HTML style img tags to Markdown images

Find	Replace
`<img src=("[^"]*").+?/>`	`![]($1)`

Create h5 sections

Find	Replace
`[([^]]*)]{.ul}`	`##### $1`

XML specific expressions

OxygenXML allows to constrain your expressions to your DITA structure using the XML query language XPath. This allows you to create expressions that are constrained to the structures that you define.

Warning: Regular expressions by themselves (without XPATH constraints), are unaware of XML structure and can invalidate your structure very quickly.

Remove extra spaces from a specific tag

Within OxygenXML **Find/Replace** using **XPath:** `//p/text()` (replace "p" with any other tag name). This limits the search to the text node of the p element.

Find	Replace
`(\s)\s+`	`$1`

Related information

https://en.wikipedia.org/wiki/XPath

Word to DITA (via MarkDown)

It is difficult to predict how well a conversion will go, but we can alleviate some known problems by starting with a well formatted Word doc - failing that, we can construct regular expressions to identify and correct problems in the MarkDown text. MarkDown is used here as a stepping stone to full `XML`.

Word document preparation:

- Complex tables (merged/joined cells, etc.) can be problematic - can they be simplified?
- Styles should be used consistently - heading progression is important (you cannot skip heading levels e.g. **Heading 1** > **Heading 4**)
- Optional flags include: `-raw_html` & `--extract-media`

Important: Pandoc is needed for Step 2 (see Pandoc)

1. Start with a `.docx` file, if you try to use `.doc`, the conversion will fail. Resave any `.doc` files to `.docx`)
2. Using Pandoc (via commmandline), convert `.docx` to CommonMark with extensions --atx-headers & --wrap=none options) *NB: replace []*.

```
pandoc -f docx -t commonmark --wrap=none --markdown-headings=atx
 [input_filename.docx] -o [output_filename.md]
```

3. With the generated MarkDown file, process any text artifacts in a suitable text file editor (e.g. using any appropriate Regex)
4. Test the MarkDown document for DITA compliance within OxygenXML (i.e.*does it render to DITA?*)
5. Using context menu (right click) **Export as DITA topic**
6. Save as DITA file
7. Create a new DITAMAP and associate the DITA file to the map.

8. Using XML Refactoring to **Convert nested topics to new topics**
9. Arrrange your topics appropriately and tidy up!

Related information

Markdown Syntax Reference

Learn Markdown

Git Submodules

How to update a submodule

Submodules allow you to treat the two projects as separate yet still be able to use one from within the other. When dependencies are managed in submodules (full, independant, repos in their own right), occasionally you'll need to update the submodule (NB: the submodule is pegged to a specific commit - **it does not follow HEAD**).

Note: Sourcetree allows you to update all child submodules of a repo at the same time: **Actions** > **Options** > **Git** > **[x]Perform submodule updates recursively**

Once the submodule parent repo has been updated/pushed to the remote, the steps below are for updating the hosted submodule:

1. In Sourcetree: from the host repo, double click the submodule from the submodule panel on the left
2. Pull any new updates to the submodule
3. Return to host repo and commit the submodule update

If you have the submodule parent repo open on the same branch as the submodule you may get an error (Sourcetree may complain that proceeding will delete local files). If you changed files and wish to keep these changes, save them elsewhere. Checkout a different branch on the submodule parent repo and pull updates from the host repo again

XML-capable editors

Source code editors are generally intended for a broader use case and are consequently not focused specifically on DITA XML, they can however be a powerful low/no cost option.

Source code editors can be configured to comply with the rules of the DITA schema.

Configuring VS Code as an authoring tool

Important: Before proceeding, the following should be installed and up-to-date: DITA Open Toolkit and Java.

1. Download and install VS Code https://code.visualstudio.com/
2. Install extension: XML Language Support by Red Hat

Configure VS Code

VS Code is not a DITA specific editor, but it can be configured to validate DITA XML by editing the `settings.json` file.

1. Open **Command Palette** by pressing Ctrl + Shift + P
   a. Type **Open Settings (JSON)** to directly edit the settings JSON file.
   b. Type **Open Settings (UI)** to open a UI to edit the settings JSON file indirectly.
2. Set the following [replace with your local paths]:

```
        "xml.catalogs": [
        "[path to dita-ot-version]\\catalog-dita.xml"
```

```
        ],
        "xml.java.home": "[path to Java]\\Java\\jre1.8.0_251",
        "xml.validation.resolveExternalEntities": true
```

Note: Backslashes are escaped by *an additional slash* in the settings file '\' = '\\'

To inspect whether your DITA XML document is valid (or receive suggested fixes for invalid markup), open the **Problems** view by pressing Ctrl + Shift + M

References

GOV.UK

- GOVDOWN
- Tech Docs Template

VS Code

- Snippet Generator

Pandoc

- Pandoc user's guide

CSS

DITA

Docker

XML

XSL

HTML5

Acrobat

Illustrator

Indesign

XFDF2BI XSLT

Purpose

The XSLT transforms PDF xfdf data to a regular XML format

For `data/@id`, it's a copy of f/@href at the moment. You can change this with your own function in `<data id="{ /*/f/@href }">`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsl:stylesheet
 xmlns:xsl="http://www.w3.org/1999/XSL/Transform"
 xmlns:xs="http://www.w3.org/2001/XMLSchema"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:mf="http://www.mekon.com/functions"
 xpath-default-namespace="http://ns.adobe.com/xfdf/"
 exclude-result-prefixes="#all"
 version="2.0">

 <xsl:output
  method="xml"
  encoding="UTF-8"
 />
```

```
 <xsl:template match="/">
  <data id="{ /*/f/@href }">
   <xsl:apply-templates select="//field"/>
  </data>
 </xsl:template>

 <xsl:template match="field">
  <field>
   <key>
    <xsl:value-of select="@name"/>
   </key>
   <xsl:apply-templates select="value"/>
  </field>
 </xsl:template>

 <xsl:template match="value">
  <value>
   <xsl:value-of select="."/>
  </value>
 </xsl:template>

</xsl:stylesheet>
```