docToolchain Manual

Table of Contents

```
<style>
.gravatar img {
    margin-left: 3px;
    border-radius: 4px;
}
</style>
```

1. Introduction and Goals

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/01_introduction_and_goals.adoc[]

Jump directly to the github repository

1.1. Create awesome docs!

docToolchain is an implementation of the *docs-as-code* approach for software architecture plus some additional automation. The basis of docToolchain is the philosophy that software documentation should be treated in the same way as code together with the *arc42* template for software architecture.

How it all began...

1.1.1. docs-as-code

Before this project started, I wasn't aware of the term *docs-as-code*. I just got tired of keeping all my architecture diagrams up to date by copying them from my UML tool over to my word processor.

As a lazy developer, I told myself 'there has to be a better way of doing this'. And I started to automate the diagram export and switched from a full fledged word processor over to a markup renderer. This enabled me to reference the diagrams from within my text and update them just before I render the document.

1.1.2. arc42

Since my goal was to document software architectures, I was already using arc42, a template for software architecture documentation. At that time, I used the MS Word template.

But what is arc42?

Dr. Gernot Starke and Peter Hruschka created this template in a joint effort to create a standard for software architecture documents. They dumped all their experience about software architectures not only into a structure but also explaining texts. These explanations are part of every chapter of the template and give you guidance on how to write each chapter of the document.

arc42 is available in many formats like MS Word, textile, and Confluence. All these formats are automatically generated from one *golden master* formatted in *AsciiDoc*.

1.1.3. docToolchain

In order to follow the *docs-as-code* approach, you need a build script that automates steps like exporting diagrams and rendering the used Markdown (*AsciiDoc* in case of *docToolchain*) to the target format.

Unfortunately, such a build script is not easy to create in the first place ('how do I create .docx?',

'why does lib x not work with lib y?') and it is also not too easy to maintain.

docToolchain is the result of my journey through the *docs-as-code* land. The goal is to have an easy to use build script that only has to be configured and not modified and that is maintained by a community as open source software.

The technical steps of my journey are written down in my blog: https://rdmueller.github.io.

Let's start with what you'll get when you use *docToolchain*...

1.2. Benefits of the docs-as-code Approach

You want to write technical docs for your software project. So it is likely you already have the tools and processes to work with source code in place. Why not also use it for your docs?

1.2.1. Document Management System

By using a version control system like Git, you get a perfect document management system for free. It lets you version your docs, branch them and gives you an audit trail. You are even able to check who wrote which part of the docs. Isn't that great?

Since your docs are now just plain text, it is also easy to do a diff and see exactly what has changed.

And when you store your docs in the same repository as your code, you always have both in sync!

1.2.2. Collaboration and Review Process

Git as a distributed version control system even enables collaboration on your docs. People can fork the docs and send you pull requests for the changes they made. By reviewing the pull request, you have a perfect review process out of the box - by accepting the pull request, you show that you've reviewed and accepted the changes. Most Git frontends like Bitbucket, GitLab and of course GitHub also allow you to reject pull requests with comments.

1.2.3. Image References and Code Snippets

Instead of pasting images to a binary document format, you now can reference images. This will ensure that those images are always up to date every time you rebuild your documents.

In addition, you can reference code snippets directly from your source code. This way, these snippets are also always up to date!

1.2.4. Compound and Stakeholder-Tailored Docs

Since you can not only reference images and code snippets but also sub-documents, you can split your docs into several sub-documents and a master, which brings all those docs together. But you are not restricted to one master — you can create master docs for different stakeholders that only contain the chapters needed for them.

1.2.5. Many more Features…

If you can dream it, you can script it.

- Want to include a list of open issues from Jira? ✅ Check.
- Want to include a changelog from Git? ✅ Check.
- Want to use inline, text based diagrams? ✅ Check.
- and many more…

2. How to install docToolchain

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/02_install.adoc[]

2.1. Get the tool

To start with docToolchain you need to get a copy of the current docToolchain repository. The easiest way is to clone the repository without history and remove the `.git` folder:

Linux with git clone

```
git clone --recursive https://github.com/docToolchain/docToolchain.git <docToolchain
home>
cd <docToolchain home>
rm -rf .git
rm -rf resources/asciidoctor-reveal.js/.git
rm -rf resources/reveal.js/.git
```

`--recursive` option is required because the repository contains 2 submodules - `resources/asciidoctor-reveal.js` and `resources/reveal.js`.

Another way is to download the zipped git repository and rename it:

Linux with download as zip

```
wget https://github.com/docToolchain/docToolchain/archive/master.zip
unzip master.zip

# fetching dependencies

cd docToolchain-master/resources

rm -d reveal.js
wget https://github.com/hakimel/reveal.js/archive/tags/3.9.2.zip -O reveal.js.zip
unzip reveal.js.zip
mv reveal.js-tags-3.9.2 reveal.js

rm -d asciidoctor-reveal.js
wget https://github.com/asciidoctor/asciidoctor-reveal.js/archive/v2.0.1.zip -O
asciidoctor-reveal.js.zip
unzip asciidoctor-reveal.js.zip
mv asciidoctor-reveal.js-2.0.1 asciidoctor-reveal.js

mv docToolchain-master <docToolchain home>
```

If you work (like me) on a Windows environment, just download and unzip the repository as well as its dependencies: reveal.js and asciidoctor-reveal.js.

After unzipping, put the dependencies in `resources` folder, so that the structure is the same as on GitHub.

You can add <docToolchain home>/bin to your PATH or you can run doctoolchain with full path if you prefer.

2.2. Initialize directory for documents

The next step after getting docToolchain is to initialize a directory where your documents live. In docToolchain this directory is named "newDocDir" during initialization, or just "docDir" later on.

2.2.1. Existing documents

If you already have some existing documents in AsciiDoc format in your project, you need to put the configuration file there to inform docToolchain what and how to process. You can do that manually by copying the contents of template_config directory. You can also do that by running `initExisting` task.

Linux initExisting example

```
cd <docToolchain home>
./gradlew -b init.gradle initExisting -PnewDocDir=<your directory>
```

You need to open Config.groovy file and configure names of your files properly. You may also change the PDF schema file to your taste.

docToolchain by default uses `src/docs` as the directory of your documentation. If you use a different directory you need to go to <docToolchain home> and set the `inputPath` property in `gradle.properties` to `your/path`.

2.2.2. arc42 from scratch

If you don't have existing documents yet, or if you need a fresh start, you can get the arc42 template in AsciiDoc format. You can do that by manually downloading from https://arc42.org/download. You can also do that by running the `initArc42<language>` task. Currently supported languages are:

- DE - German
- EN - English
- ES - Spanish
- RU - Russian

Linux initArc42EN example

```
cd <docToolchain home>
./gradlew -b init.gradle initArc42EN -PnewDocDir=<newDocDir>
```

The Config.groovy file is then preconfigured to use the downloaded template.

Blog-Post: Let's add Content!

2.3. Build

This should already be enough to start a first build.

By now, docToolchain should be installed as command line tool and the path to its `bin` folder should be on your path. If you now switch to your freshly initialized <newDocDir>, you should be able to execute the following commands:

Linux

```
doctoolchain <docDir> generateHTML
doctoolchain <docDir> generatePDF
```

Windows

```
doctoolchain.bat <docDir> generateHTML
doctoolchain.bat <docDir> generatePDF
```

<docDir> may be relative, e.g. ".", or absolute.

As a result, you will see the progress of your build together with some warnings which you can just ignore for the moment.

The first build generated some files within the `<docDir>/build`:

```
build
|-- html5
|   |-- arc42-template.html
|   `-- images
|       |-- 05_building_blocks-EN.png
|       |-- 08-Crosscutting-Concepts-Structure-EN.png
|       `-- arc42-logo.png
`-- pdf
    |-- arc42-template.pdf
    `-- images
        |-- 05_building_blocks-EN.png
        |-- 08-Crosscutting-Concepts-Structure-EN.png
        `-- arc42-logo.png
```

Congratulations! If you see the same folder structure, you just managed to render the standard arc42 template as html and pdf!

If you didn't get the right output, please raise an issue on github

Blog-Posts: Behind the great Firewall, Enterprise AsciiDoctor

3. Overview of available Tasks

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_tasks.adoc[]

This chapter explains all docToolchain specific tasks.

The following picture gives an overview of the whole build system:

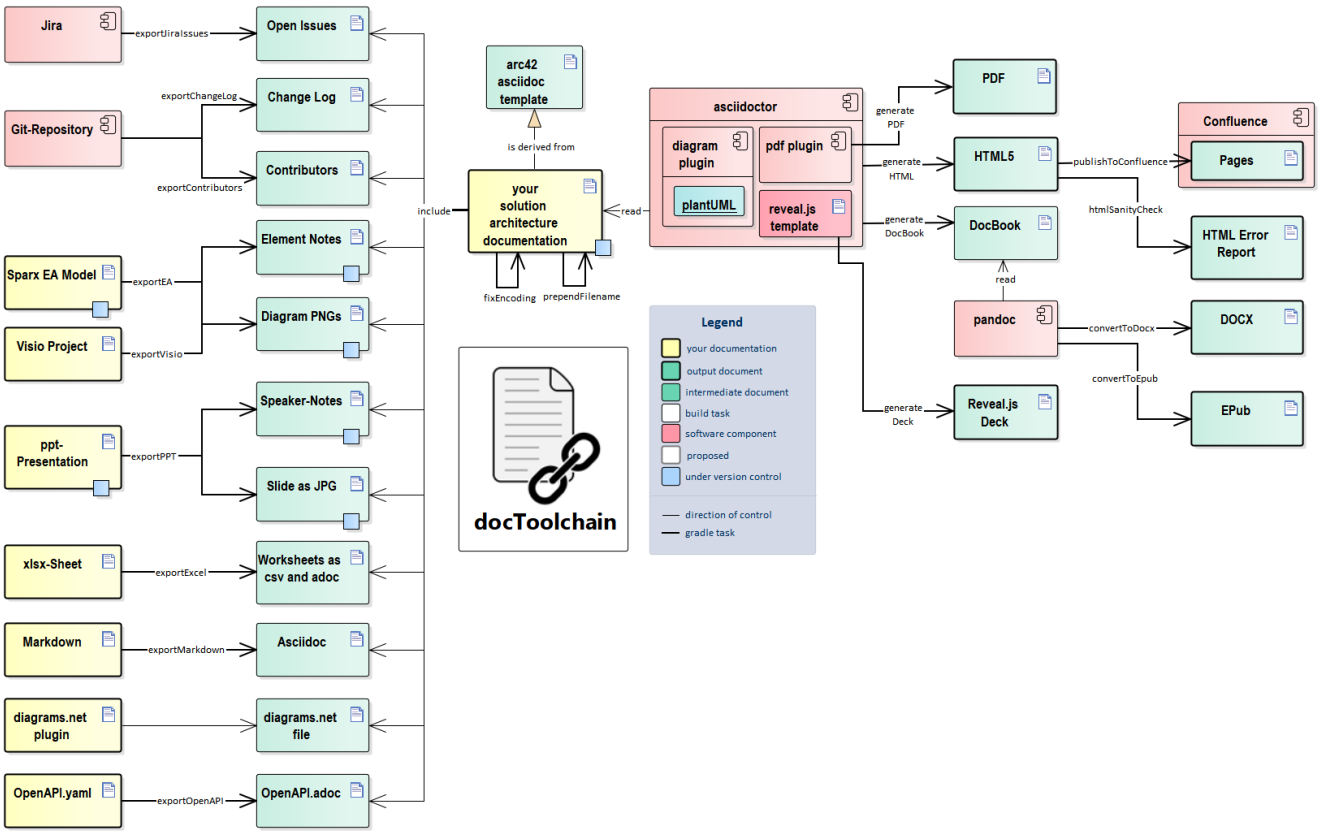


Figure 1. docToolchain

3.1. Conventions

There are some simple naming conventions for the tasks. They might be confusing at first and that's why they are explained here.

3.1.1. generateX

render would have been another good prefix, since these tasks use the plain AsciiDoctor functionality to render the source to a given format.

3.1.2. exportX

These tasks export images and AsciiDoc snippets from other systems or file formats. The resulting artifacts can then be included from your main sources.

What's different from the *generateX* tasks is that you don't need to export with each build.

It is also likely that you have to put the resulting artifacts under version control because the tools needed for the export (like Sparx Enterprise Architect or MS PowerPoint) are likely to be not available on a build server or on another contributor's machine.

3.1.3. convertToX

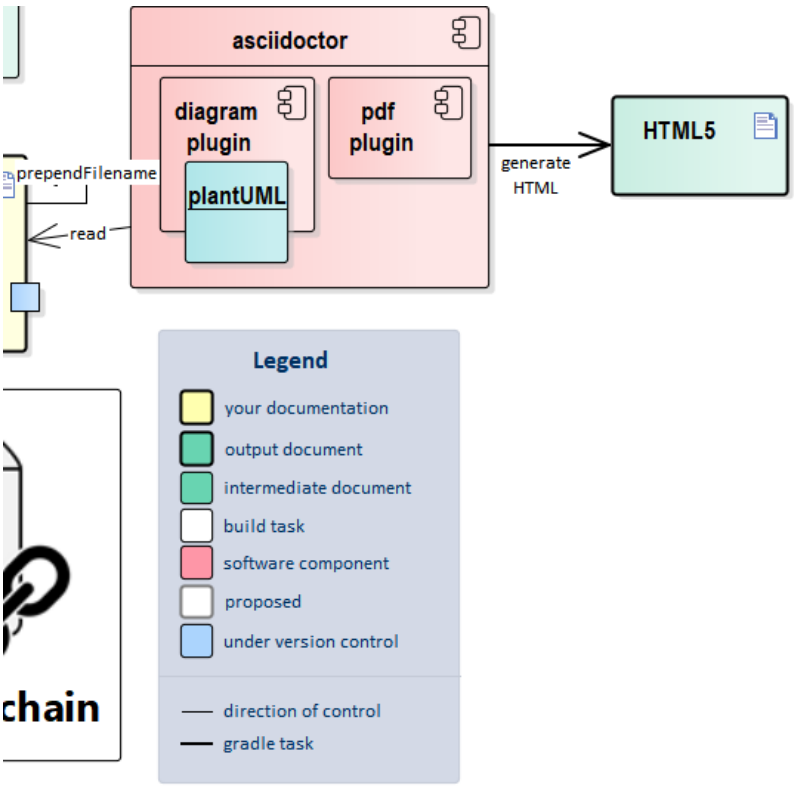
These tasks take the output from AsciiDoctor and convert it (through other tools) to the target format. This results in a dependency on a *generateX* task and another external tool (currently pandoc).

3.1.4. publishToX

These tasks not only convert your documents but also deploy/publish/move them to a remote system — currently Confluence. This means that the result is likely to be visible immediately to others.

3.2. generateHTML

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_generateHTML.adoc[]



This is the standard AsciiDoctor generator which is supported out of the box.

The result is written to `build/html5`. The HTML files need the images folder to be in the same directory to display correctly.

If you would like to have a single-file HTML as result, you can configure AsciiDoctor to store the images inline as `data-uri`.
Just set `:data-uri:` in the config of your AsciiDoc file.
But be warned - such a file can become very big easily and some browsers might get into trouble rendering them.
https://rdmueller.github.io/single-file-html/

3.2.1. Text based Diagrams

For docToolchain, it is configured to use the asciidoctor-diagram plugin which is used to create PlantUML diagrams.

The plugin also supports a bunch of other text based diagrams, but PlantUML is the most used.

To use it, just specify your PlantUML code like this:

```
.example diagram
[plantuml, "{plantUMLDir}demoPlantUML", png] ①
----
class BlockProcessor
class DiagramBlock
class DitaaBlock
class PlantUmlBlock

BlockProcessor <|-- DiagramBlock
DiagramBlock <|-- DitaaBlock
DiagramBlock <|-- PlantUmlBlock
----
```

① The element of this list specifies the diagram tool `plantuml` to be used.
The second element is the name of the image to be created and the third specifies the image type.

The `{plantUMLDir}` ensures that PlantUML also works for the generatePDF task. Without it, generateHTML works fine, but the PDF will not contain the generated images.

Make sure to specify a unique image name for each diagram, otherwise they will overwrite each other.

The above example renders as

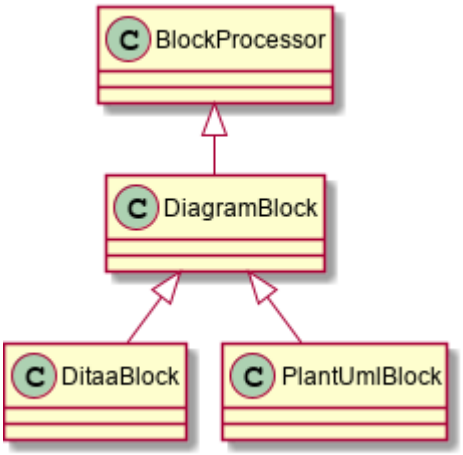


Figure 2. example diagram

If you want to control the size of the generated diagram in the output, you can configure the "width" attribute (in pixels) or "scale" attribute (floating point ratio) passed to asciidoctor-diagram. For example, if you take the example diagram above and change the declaration to one of the below versions

```
[plantuml, target="{plantUMLDir}demoPlantUMLWidth", format=png, width=250]
# rest of the diagram definition

[plantuml, target="{plantUMLDir}demoPlantUMLScale", format=png, scale=0.75]
# rest of the diagram definition
```

it will render like this:

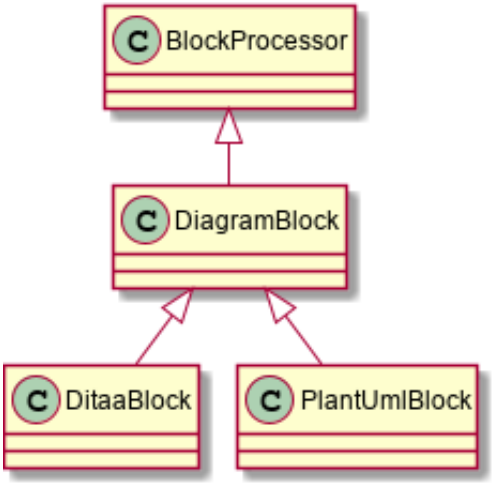


Figure 3. example diagram (with specified width)

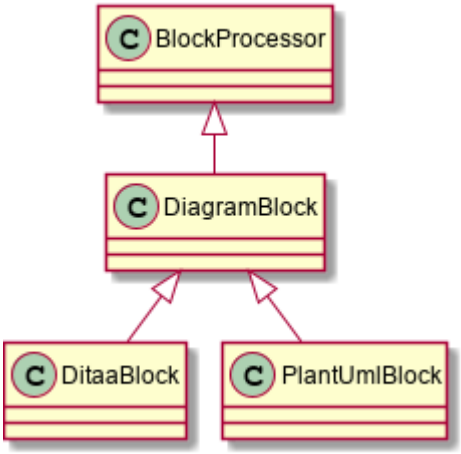


Figure 4. example diagram (with specified scale)

PlantUML needs Graphviz dot installed to work. If you can't install it, you can use the Java based version of the dot library. Just add `!pragma graphviz_dot smetana` as the first line of your diagram definition. This is still an experimental feature, but already works quite well!
https://rdmueller.github.io/plantuml-without-graphviz/

Blog-Posts: PlantUML with Gradle, plantUML with Asciidoctor-pdf, plantUML revisited, How to use PlantUML without Graphviz

3.2.2. Source

AsciiDocBasics.gradle

```
task generateHTML (
        type: AsciidoctorTask,
        group: 'docToolchain',
        description: 'use html5 as asciidoc backend') {
        attributes (
            'plantUMLDir'         : file("${docDir}/${config.outputPath}/html5").
toURI().relativize(new File("${docDir}/${config.outputPath}/html5/plantUML/").toURI()
).getPath()
        )

    onlyIf {
        !sourceFiles.findAll {
            'html' in it.formats
        }.empty
    }

    // specify output folder explicitly to avoid cleaning targetDir from other
generated content
    outputDir = file(targetDir + '/html5/')
    separateOutputDirs(false)

    sources {
        sourceFiles.findAll {
            'html' in it.formats
        }.each {
            include it.file
            println it.file
        }
    }

    backends = ['html5']
}
```

3.3. fixEncoding

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_fixencoding.adoc[]

Whenever Asciidoctor has to process a file which is not UTF-8 encoded, the underlying Ruby tries to read it and throws an error like this:

```
asciidoctor: FAILED: /home/demo/test.adoc: Failed to load AsciiDoc document - invalid
byte sequence in UTF-8
```

Unfortunately, it is hard to find the wrong encoded file if a lot of `includes::` are used - Asciidoctor only shows the name of the main document.

This is not a problem of Asciidoctor, but of the underlying ruby interpreter.

This target crawls through all `*.ad` and `*.adoc` files and checks their encoding. If it encounters a file which is not UTF-8 encoded, it will rewrite it with the UTF-8 encoding.

scripts/fixEncoding.gradle

```
import groovy.util.*
import static groovy.io.FileType.*

task fixEncoding(
        description: 'finds and converts non UTF-8 adoc files to UTF-8',
        group: 'docToolchain helper',
) {
    doLast {
        File sourceFolder = new File("${docDir}/${inputPath}")
        println("sourceFolder: " + sourceFolder.canonicalPath)
        sourceFolder.traverse(type: FILES) { file ->
            if (file.name ==~ '^.*(ad|adoc|asciidoc)$') {
                CharsetToolkit toolkit = new CharsetToolkit(file);
                // guess the encoding
                def guessedCharset = toolkit.getCharset().toString().toUpperCase();
                if (guessedCharset!='UTF-8') {
                    def text = file.text
                    file.write(text, "utf-8")
                    println("  converted ${file.name} from '${guessedCharset}' to
'UFT-8'")
                }
            }
        }
    }
}
```

3.4. prependFilename

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_prependFilename.adoc[]

When Asciidoctor renders a file, the file context only knows the name of the top-level AsciiDoc file but an include file doesn't know that it is being included. It will simply get the name of the master file and has no chance to get his own names as attribute.

This task simply crawls through all AsciiDoc files and prepends the name of the current file like this:

```
:filename: manual/03_task_prependFilename.adoc
```

This way, each file can get its own file name. This enables features like the inclusion of file contributors (see exportContributors-task).

The task skips all files named `config.*`, `_config.*`, `feedback.*` and `_feedback.*`.

scripts/prependFilename.gradle

```
import static groovy.io.FileType.*

task prependFilename(
        description: 'crawls through all AsciiDoc files and prepends the name of the
current file',
        group: 'docToolchain helper',
) {
    doLast {
        File sourceFolder = new File("${docDir}/${inputPath}")
        println("sourceFolder: " + sourceFolder.canonicalPath)
        sourceFolder.traverse(type: FILES) { file ->
            if (file.name ==~ '^.*(ad|adoc|asciidoc)$') {
                if (file.name.split('[.]')[0] in ["feedback", "_feedback", "config",
"_config"]) {
                    println "skipped "+file.name
                } else {
                    def text = file.getText('utf-8')
                    def name = file.canonicalPath - sourceFolder.canonicalPath
                    name = name.replace("\\", "/").replaceAll("^/", "")
                    if (text.contains(":filename:")) {
                        text = text.replaceAll(":filename:.*", ":filename: $name")
                        println "updated "+name
                    } else {
                        text = ":filename: $name\n" + text
                        println "added   "+name
                    }
                    file.write(text,'utf-8')
                }
            }
        }
    }
}
```

3.5. collectIncludes

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_collectIncludes.adoc[]

This tasks crawls through your whole project looking for AsciiDoc files with a certain name pattern. It then creates an AsciiDoc file which just includes all files found.

When you create modular documentation, most includes are static. For example, the arc42-template has 12 chapters and a master template in which those 12 chapters are included.

But when you work with dynamic modules like ADRs - Architecture Decision Records - you create those files on the fly. Maybe not even within your `/src/docs` folder but right beside the code file for which you wrote the ADR.

In order to include these files in your documentation, you now would have to add the file with its whole relative path to one of your AsciiDoc files.

This task will handle it for you!

Just stick with your files names to the pattern `^[A-Z]{3,}[-\_].\*` (it begins with at last three upper case letters and a dash/underscore) and this task will collect this file and write it to your build folder. You only have to include this generated file from within your documentation.

If you provide templates for the documents these templates are skipped if the name matches the pattern `^.*[-_][tT]emplate[-\_].*`.

Example:

You have a file called

```
/src/java/yourCompany/domain/books/ADR-1-whyWeUseTheAISINInsteadOFISBN.adoc
```

The task will collect this file and write another file called

```
/build/docs/_includes/ADR_includes.adoc
```

Which will look like this:

```
include::../../../src/java/yourCompany/domain/books/ADR-1-
whyWeUseTheAISINInsteadOFISBN.adoc[]
```

Obviously, you get the most benefit if you not only have one ADR file, but several ones which get collected.

You then include these files in your main documentation by using a single include:

```
include::{targetDir}/docs/_includes/ADR_includes.adoc[]
```

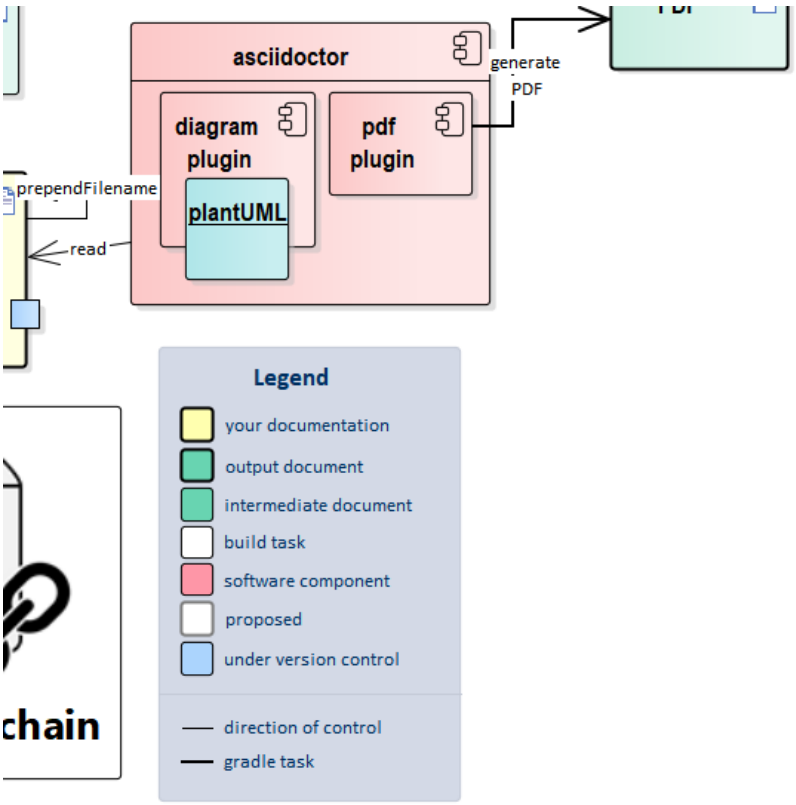
scripts/collectIncludes.gradle

```
import static groovy.io.FileType.*
import java.security.MessageDigest

task collectIncludes(
        description: 'collect all ADRs as includes in one file',
        group: 'docToolchain'
) {
    doFirst {
        new File(targetDir, '_includes').mkdirs()
    }
    doLast {
        //let's search the whole project for files, not only the docs folder
        //could be a problem with node projects :-)

        //running as subproject? set scandir to main project
        if (project.name!=rootProject.name && scanDir=='.') {
            scanDir = project(':').projectDir.path
        }
        if (docDir.startsWith('.')) {
            docDir = file(new File(projectDir, docDir).canonicalPath)
        }
        logger.info "docToolchain> docDir: ${docDir}"
        logger.info "docToolchain> scanDir: ${scanDir}"
        if (scanDir.startsWith('.')) {
            scanDir = file(new File(docDir, scanDir).canonicalPath)
        } else {
            scanDir = file(new File(scanDir, "").canonicalPath)
        }
        logger.info "docToolchain> scanDir: ${scanDir}"

        logger.info "docToolchain> includeRoot: ${includeRoot}"

        if (includeRoot.startsWith('.')) {
            includeRoot = file(new File(docDir, includeRoot).canonicalPath)
        }
        logger.info "docToolchain> includeRoot: ${includeRoot}"

        File sourceFolder = scanDir
        println "sourceFolder: " + sourceFolder.canonicalPath
        def collections = [:]
        sourceFolder.traverse(type: FILES) { file ->
            if (file.name ==~ '^[A-Z]{3,}[-_].*[.](ad|adoc|asciidoc)$') {
                def type = file.name.replaceAll('^([A-Z]{3,})[-_].*$','\$1')
                if (!collections[type]) {
                    collections[type] = []
                }
```

```
                println "file: " + file.canonicalPath
                def fileName = (file.canonicalPath - scanDir.canonicalPath)[1..-1]
                if (file.name ==~ '^.*[Tt]emplate.*$') {
                    println "ignore template file: " + fileName
                } else {
                    if (file.name ==~ '^.*[A-Z]{3,}_includes.adoc$') {
                        println "ignore generated _includes files: " + fileName
                    } else {
                        if ( fileName.startsWith('docToolchain') || fileName.replace(
"\\", "/").matches('^.*/docToolchain/.*$')) {
                            //ignore docToolchain as submodule
                        } else {
                            println "include corrected file: " + fileName
                            collections[type] << fileName
                        }
                    }
                }
            }
        }
        println "targetDir - docDir: " + (targetDir - docDir)
        println "targetDir - includeRoot: " + (targetDir - includeRoot)
        def pathDiff = '../' * ((targetDir - docDir)
                                .replaceAll('^/','')
                                .replaceAll('/$','')
                                .replaceAll("[^/]",'').size()+1)

        println "pathDiff: " + pathDiff
        collections.each { type, fileNames ->
            if (fileNames) {
                def outFile = new File(targetDir+'/_includes', type + '_includes.adoc
')
                println outFile.canonicalPath-sourceFolder.canonicalPath
                outFile.write("// this is autogenerated\n")
                fileNames.sort().each { fileName ->
                    outFile.append ("include::../"+pathDiff+fileName.replace("\\", "/
")+"[]\n\n")
                }
            }
        }
    }
}
```

3.6. generatePDF

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_generatePDF.adoc[]



This task makes use of the asciidoctor-pdf plugin to render your documents as a pretty PDF file.

The file will be written to `build/pdf`.

The used plugin is still in alpha status, but the results are already quite good. If you want to use another way to create a PDF, use PhantomJS for instance and script it.

The PDF is generated directly from your AsciiDoc sources without the need of an intermediate format or other tools. The result looks more like a nicely rendered book than a print-to-pdf HTML page.

It is very likely that you need to "theme" you PDF - change colors, fonts, page header, and footer. This can be done by creating a `custom-theme.yml` file. As a starting point, copy the file `src/docs/pdfTheme/custom-theme.yml` from docToolchain to your project and reference it from your main `.adoc`file via setting the `:pdf-stylesdir:`. For instance, insert

```
:pdf-stylesdir: ../pdfTheme
```

at the top of your document to reference the `custom-theme.yml` from the `/pdfTheme` folder.

Documentation on how to modify a theme can be found in the asciidoctor-pdf theming guide.

Blog-Post: Beyond HTML

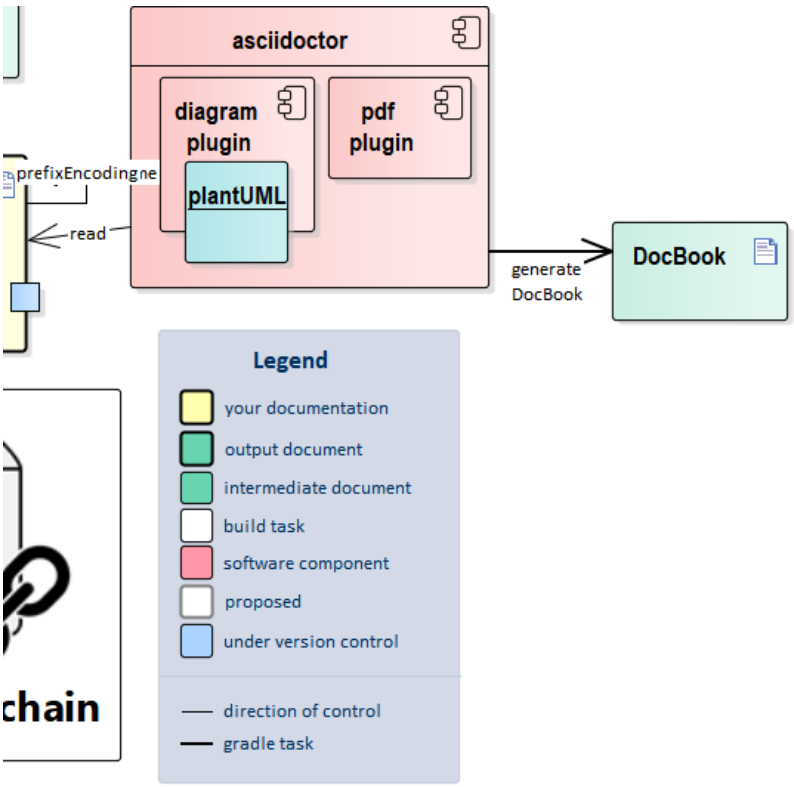
3.6.1. Source

AsciiDocBasics.gradle

```
task generatePDF (
        type: AsciidoctorTask,
        group: 'docToolchain',
        description: 'use pdf as asciidoc backend') {
        attributes (
            'plantUMLDir'         : file("${docDir}/${config.outputPath
}/pdf/images/plantUML/").path
        )

    onlyIf {
        !sourceFiles.findAll {
            'pdf' in it.formats
        }.empty
    }

    sources {
        sourceFiles.findAll {
            'pdf' in it.formats
        }.each {
            include it.file
        }
    }

    backends = ['pdf']
}
```

3.7. generateDocbook

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_generateDocBook.adoc[]



This is only a helper task - it generates the intermediate format for convertToDocx and convertToEpub.

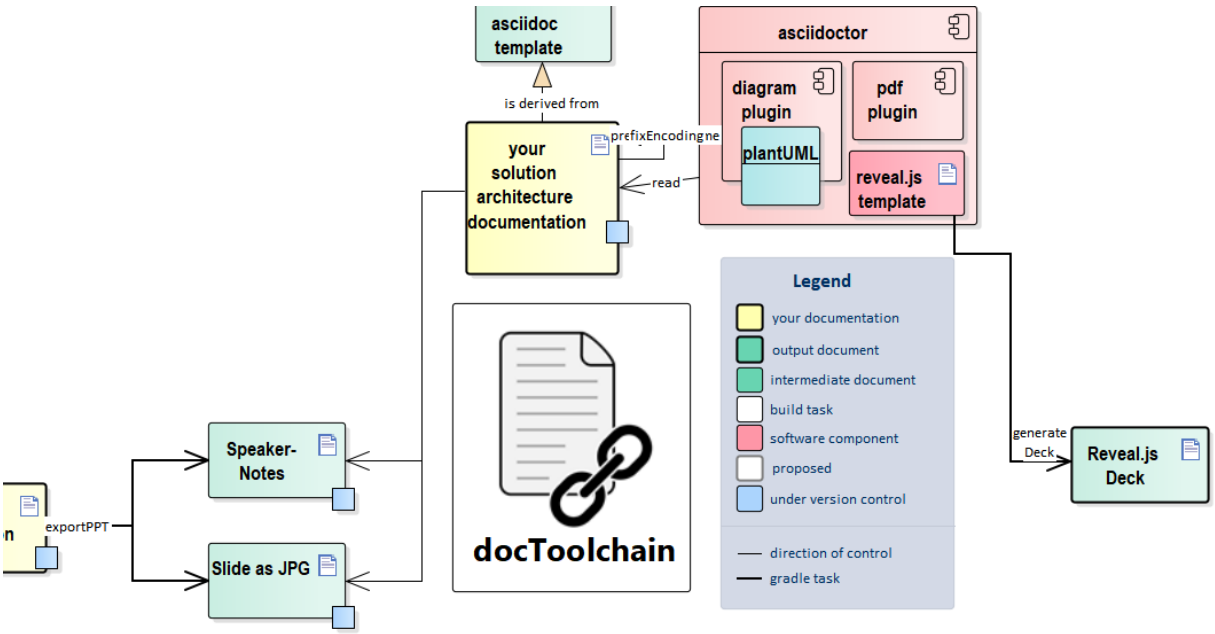
3.7.1. Source

AsciiDocBasics.gradle

```
task generateDocbook (
        type: AsciidoctorTask,
        group: 'docToolchain',
        description: 'use docbook as asciidoc backend') {

    onlyIf {
        !sourceFiles.findAll {
            'docbook' in it.formats
        }.empty
    }

    sources {
        sourceFiles.findAll {
            'docbook' in it.formats
        }.each {
            include it.file
        }
    }

    backends = ['docbook']
}
```

3.8. generateDeck

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_generateDeck.adoc[]



This task makes use of the asciidoctor-reveal.js backend to render your documents into a HTML based presentation.

This task is best used together with the exportPPT task. It creates a PowerPoint presentation and enriches it with reveal.js slide definitions in AsciiDoc within the speaker notes.

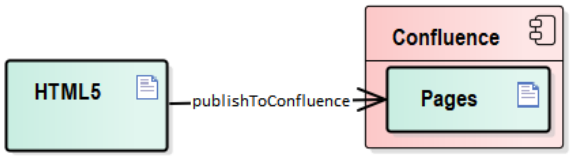
3.8.1. Source

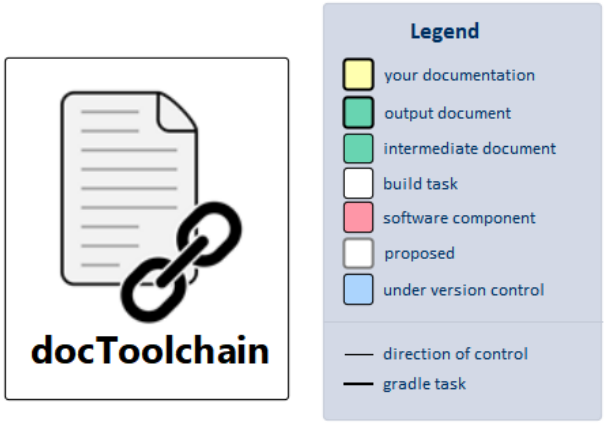
AsciiDocBasics.gradle

```
task generateDeck (
        type: AsciidoctorTask,
        group: 'docToolchain',
        description: 'use revealJs as asciidoc backend to create a presentation') {

    attributes (
            'idprefix': 'slide-',
            'idseparator': '-',
            'docinfo1': '',
            'revealjs_theme': 'black',
            'revealjs_progress': 'true',
            'revealjs_touch': 'true',
            'revealjs_hideAddressBar': 'true',
            'revealjs_transition': 'linear',
            'revealjs_history': 'true',
            'revealjs_slideNumber': 'true'
    )
    options template_dirs : [new File(new File (projectDir,'/resources/asciidoctor-
reveal.js'),'templates').absolutePath ]

    onlyIf {
        !sourceFiles.findAll {
            'revealjs' in it.formats
        }.empty
    }

    sources {
        sourceFiles.findAll {
            'revealjs' in it.formats
        }.each {
            include it.file
        }

    }

    outputDir = file(targetDir+'/decks/')

    resources {
        from('resources') {
            include 'reveal.js/**'
        }
        from(sourceDir) {
            include 'images/**'
        }
        into("")
        logger.info "${docDir}/${config.outputPath}/images"
    }
}
```

3.9. publishToConfluence

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_publishToConfluence.adoc[]





This target takes the generated HTML file, splits it by headline and pushes it to Confluence. This enables you to use the docs-as-code approach while getting feedback from non-techies through Confluence comments. And it fulfills the "requirement" of "... but all documentation is in Confluence".

Special features:

- `[source]`-blocks are converted to code-macro blocks in confluence.
- only pages and images which have changed between task runs are really published and hence only for those changes notifications are sent to watchers. This is quite important - otherwise watchers are easily annoyed by too many notifications.
- `:keywords:` Keywords are attached as labels to every generated Confluence page. The rules for page labels should be kept in mind. See https://confluence.atlassian.com/doc/add-remove-and-search-for-labels-136419.html. Several keywords are allowed. They must be separated by comma, e.g. `:keywords: label_1, label-2, label3, ⋯`. Labels (keywords) must not contain a space character! Use '_' or '-'.

code-macro-blocks in confluence render an error if the language attribute contains an unknown language. See https://confluence.atlassian.com/doc/code-block-macro-139390.html for a list of valid language and how to add further languages.

3.9.1. Configuration

We tried to make the configuration self-explaining, but there are always some note to add.

ancestorId

this is the page ID of the parent page to which you want your docs to be published. Go to this page, click on edit and the needed ID will show up in the URL. Specify the ID as string within the config file.

api

for cloud instances, `[context]` is `wiki`

preambleTitle

the title of the page containing the preamble (everything the first second level heading). Default is 'arc42'

disableToC

This boolean configuration define if the Table of Content (ToC) is disabled from the page once uploaded in confluence. (it is false by default, so the ToC is active)

pagePrefix/pageSuffix

Confluence can't handle two pages with the same name. Moreover, the script matches pages regardless of the case. It will thus refuse to replace a page whose title only differs in case with an existing page. So you should create a new confluence space for each piece of larger documentation. If you are restricted and can't create new spaces, you can use this `pagePrefix` /`pageSuffix` to define a prefix/suffix for this doc so that it doesn't conflict with other page names.

credentials

Use username and password or even better username and api-token. You can create new API-tokens in your profile. To avoid having your password or api-token versioned through git, you can store it outside of this configuration as environment variable or in another file - the key here is that the config file is a groovy script. e.g. you can do things like `credentials = "user:${new File("/home/me/apitoken").text}".bytes.encodeBase64().toString()`

To simplify the injection of credentials from external sources there is a fallback. Should you leave the credentials field empty, the variables `confluenceUser` and `confluencePassword` from the build environment will be used for authentication. You can set these through any means allowed by gradle like the `gradle.properties` file in the project or your home directory, environment variables or command-line flags. For all ways to set these variables, have a look at the gradle manual.

apikey

In cases where you have to use full user authorization because of internal confluence permission handling, you need to add the API-token in addition to the credentials. The API-token cannot be added to the credentials as it is used for user and password exchange. Therefore the API-token can be added as parameter apikey, which makes the addition of the token as a separate header field with key: `keyId` and value of `apikey`. Example including storing of the real value outside this configuration: `apikey = "${new File("/home/me/apitoken").text}"`.

extraPageContent

If you need to prefix your pages with a warning that this is generated content - this is the right place.

enableAttachments

If value is set to **true**, your links to local file references will be uploaded as attachments. The current implementation only supports a single folder. This foldername will be used as a prefix to validate if your file should be uploaded or not. In case you enable this feature, and use a folder which starts with "attachment*", an adaption of this prefix is required.

jiraServerId

the jira server id your confluence instance is connected to. If value is set, all anchors pointing to a jira ticket will be replaced by the confluence jira macro. To function properly jiraRoot has to be configured (see exportJiraIssues).

Example:

All files to attach will require to be linked inside the document.

```
link:attachement/myfolder/myfile.json[My API definition]
```

attachmentPrefix

The expected foldername of your output dir. **Default**: `attachment`

proxy

If you need to provide a proxy to access Confluence, you may set a map with keys `host` (e.g. `'my.proxy.com'`), `port` (e.g. `'1234'`) and `schema` (e.g. `'http'`) of your proxy.

useOpenapiMacro

If this option is present and equal to "confluence-open-api" then any source block marked with class openapi will be wrapped in Elitesoft Swagger Editor macro: (see Elitesoft Swagger Editor)
For backward compatibility: If this option is present and equal to `true`, then again the Elitesoft Swagger Editor macro will be used.
If this option is present and equal to "open-api" then any source block marked with class openapi will be wrapped in Open API Documentation for Confluence macro: (see Open API Documentation for Confluence). A download source (yaml) button is shown by default.
This is how you'd include your openapi YAML file:

```
[source.openapi,yaml]
----
include::myopeanapi.yaml[]
----
```

publishToConfluence.gradle

```
//Configureation for publishToConfluence

confluence = [:]

// 'input' is an array of files to upload to Confluence with the ability
//          to configure a different parent page for each file.
//
// Attributes
```

```
// - 'file': absolute or relative path to the asciidoc generated html file to be
exported
// - 'url': absolute URL to an asciidoc generated html file to be exported
// - 'ancestorName' (optional): the name of the parent page in Confluence as string;
//                              this attribute has priority over ancestorId, but if
page with given name doesn't exist,
//                              ancestorId will be used as a fallback
// - 'ancestorId' (optional): the id of the parent page in Confluence as string; leave
this empty
//                            if a new parent shall be created in the space
// - 'preambleTitle' (optional): the title of the page containing the preamble
(everything
//                            before the first second level heading). Default is
'arc42'
//
// The following four keys can also be used in the global section below
// - 'spaceKey' (optional): page specific variable for the key of the confluence space
to write to
// - 'createSubpages' (optional): page specific variable to determine whether ".sect2"
sections shall be split from the current page into subpages
// - 'pagePrefix' (optional): page specific variable, the pagePrefix will be a prefix
for the page title and it's sub-pages
//                            use this if you only have access to one confluence space
but need to store several
//                            pages with the same title - a different pagePrefix will
make them unique
// - 'pageSuffix' (optional): same usage as prefix but appended to the title and it's
subpages
// only 'file' or 'url' is allowed. If both are given, 'url' is ignored
confluence.with {
    input = [
            [ file: "build/docs/html5/arc42-template-de.html" ],
    ]

    // endpoint of the confluenceAPI (REST) to be used
    // verfiy that you got the correct endpoint by browsing to
    // https://[yourServer]/[context]/rest/api/user/current
    // you should get a valid json which describes your current user
    // a working example is https://arc42-
template.atlassian.net/wiki/rest/api/user/current
    api = 'https://[yourServer]/[context]/rest/api/'

//    Additionally, spaceKey, createSubpages, pagePrefix and pageSuffix can be
globally defined here. The assignment in the input array has precedence

    // the key of the confluence space to write to
    spaceKey = 'asciidoc'

    // the title of the page containing the preamble (everything the first second
level heading). Default is 'arc42'
    preambleTitle = ''
```

```
    // variable to determine whether ".sect2" sections shall be split from the current
page into subpages
    createSubpages = false

    // the pagePrefix will be a prefix for each page title
    // use this if you only have access to one confluence space but need to store
several
    // pages with the same title - a different pagePrefix will make them unique
    pagePrefix = ''

    pageSuffix = ''

    /*
    WARNING: It is strongly recommended to store credentials securely instead of
commiting plain text values to your git repository!!!

    Tool expects credentials that belong to an account which has the right permissions
to to create and edit confluence pages in the given space.
    Credentials can be used in a form of:
     - passed parameters when calling script (-PconfluenceUser=myUsername
-PconfluencePass=myPassword) which can be fetched as a secrets on CI/CD or
     - gradle variables set through gradle properties (uses the 'confluenceUser' and
'confluencePass' keys)
    Often, same credentials are used for Jira & Confluence, in which case it is
recommended to pass CLI parameters for both entities as
    -Pusername=myUser -Ppassword=myPassword
    */

    //optional API-token to be added in case the credentials are needed for user and
password exchange.
    //apikey = "[API-token]"

    // HTML Content that will be included with every page published
    // directly after the TOC. If left empty no additional content will be
    // added
    // extraPageContent = '<ac:structured-macro ac:name="warning"><ac:parameter
ac:name="title" /><ac:rich-text-body>This is a generated page, do not edit!</ac:rich-
text-body></ac:structured-macro>
    extraPageContent = ''

    // enable or disable attachment uploads for local file references
    enableAttachments = false

    // default attachmentPrefix = attachment - All files to attach will require to be
linked inside the document.
    // attachmentPrefix = "attachment"



    // Optional proxy configuration, only used to access Confluence
    // schema supports http and https
```

```
    // proxy = [host: 'my.proxy.com', port: 1234, schema: 'http']

    // Optional: specify which Confluence OpenAPI Macro should be used to render
OpenAPI definitions
    // possible values: ["confluence-open-api", "open-api", true]. true is the same as
"confluence-open-api" for backward compatibility
    // useOpenapiMacro = "confluence-open-api"
}
```

3.9.2. CSS Styling

Some AsciiDoctor features depend on particular CSS style definitions. Unless these styles are defined, some formatting that is present in the HTML version will not be represented when published to Confluence.

To configure Confluence to include additional style definitions:

1. Log in to Confluence as a space admin
2. Go to the desired space
3. Select "Space tools", "Look and Feel", "Stylesheet"
4. Click "Edit", enter the desired style definitions, and click "Save"

The default style definitions can be found in the AsciiDoc project as asciidoctor-default.css. Note that you likely do **NOT** want to include the whole thing, as some of the definitions are likely to disrupt Confluence's layout.

The following style definitions are believed to be Confluence-compatible, and enable use of the built-in roles (`big`/`small`, `underline`/`overline`/`line-through`, `COLOR`/`COLOR-background` for the sixteen HTML color names):

```
.big{font-size:larger}
.small{font-size:smaller}
.underline{text-decoration:underline}
.overline{text-decoration:overline}
.line-through{text-decoration:line-through}
.aqua{color:#00bfbf}
.aqua-background{background-color:#00fafa}
.black{color:#000}
.black-background{background-color:#000}
.blue{color:#0000bf}
.blue-background{background-color:#0000fa}
.fuchsia{color:#bf00bf}
.fuchsia-background{background-color:#fa00fa}
.gray{color:#606060}
.gray-background{background-color:#7d7d7d}
.green{color:#006000}
.green-background{background-color:#007d00}
.lime{color:#00bf00}
.lime-background{background-color:#00fa00}
.maroon{color:#600000}
.maroon-background{background-color:#7d0000}
.navy{color:#000060}
.navy-background{background-color:#00007d}
.olive{color:#606000}
.olive-background{background-color:#7d7d00}
.purple{color:#600060}
.purple-background{background-color:#7d007d}
.red{color:#bf0000}
.red-background{background-color:#fa0000}
.silver{color:#909090}
.silver-background{background-color:#bcbcbc}
.teal{color:#006060}
.teal-background{background-color:#007d7d}
.white{color:#bfbfbf}
.white-background{background-color:#fafafa}
.yellow{color:#bfbf00}
.yellow-background{background-color:#fafa00}
```

3.9.3. Source

publishToConfluence.gradle

```
task publishToConfluence(
        description: 'publishes the HTML rendered output to confluence',
        group: 'docToolchain'
) {
    doLast {
        logger.info("docToolchain> docDir: "+docDir)

        binding.setProperty('config',config)
        binding.setProperty('docDir',docDir)
        evaluate(new File(projectDir, 'scripts/asciidoc2confluence.groovy'))
    }
}
```

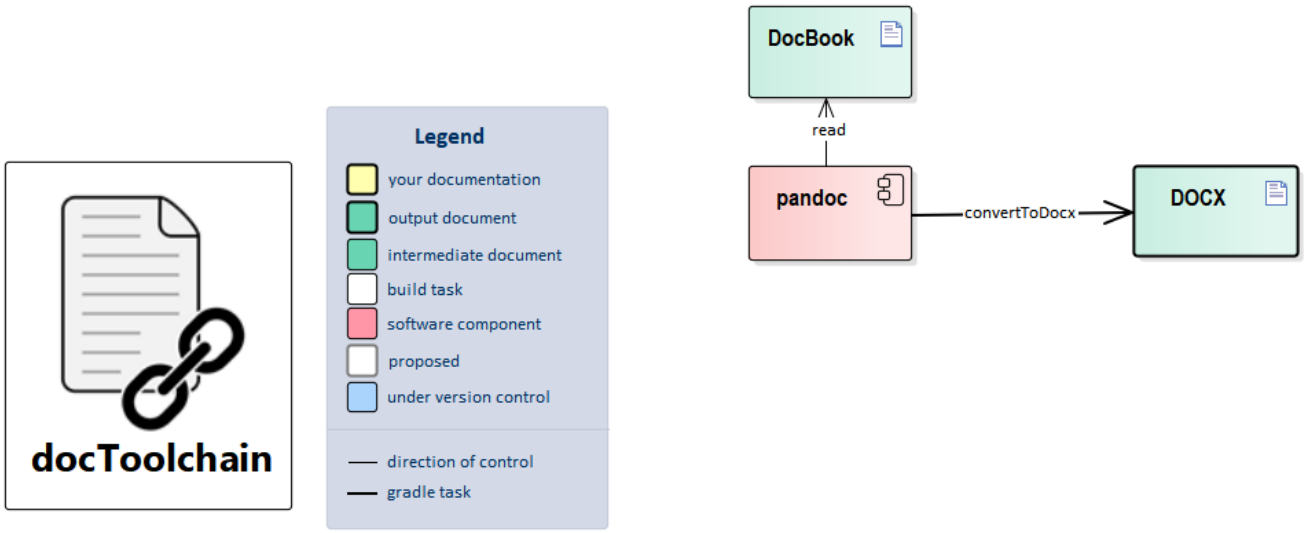
scripts/asciidoc2confluence.groovy

```
```

3.10. convertToDocx

- Needs pandoc installed
- Please make sure that 'docbook' and 'docx' are added to the inputFiles formats in Config.groovy
- Optional: you can specify a reference doc file with custom stylesheets (see task createReferenceDoc)

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_convertToDocx.adoc[]



Blog-Post: Render AsciiDoc to docx (MS Word)

3.10.1. Source

pandoc.gradle

```
task convertToDocx (
        group: 'docToolchain',
        description: 'converts file to .docx via pandoc. Needs pandoc installed.',
        type: Exec
) {
    // All files with option `docx` in config.groovy is converted to docbook and then
to docx.
    def sourceFilesDocx = sourceFiles.findAll { 'docx' in it.formats }
    sourceFilesDocx.each {
        def sourceFile = it.file.replace('.adoc', '.xml')
        def targetFile = sourceFile.replace('.xml', '.docx')

        workingDir "$targetDir/docbook"
        executable = "pandoc"

        if(referenceDocFile?.trim()) {
            args = ["-r","docbook",
                    "-t","docx",
                    "-o","../docx/$targetFile",
                    "--reference-doc=${docDir}/${referenceDocFile}",
                    sourceFile]
        } else {
            args = ["-r","docbook",
                    "-t","docx",
                    "-o","../docx/$targetFile",
                    sourceFile]
        }
    }
    doFirst {
        new File("$targetDir/docx/").mkdirs()
    }
}
```

3.11. createReferenceDoc

- Needs pandoc installed

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_createReferenceDoc.adoc[]

Creates a reference docx file used by pandoc during conversion from docbook to docx. You can edit this file to use your prefered styles in task convertToDocx.

Please make sure that 'referenceDocFile' property is set to your custom reference file in Config.groovy

Excerpt of Config.groovy with property 'referenceDocFile'

```
inputPath = '.'

// use a style reference file in the input path for conversion from docbook to docx
referenceDocFile = "${inputPath}/my-ref-file.docx"
```

The contents of the reference docx are ignored, but its stylesheets and document properties (including margins, page size, header, and footer) are used in the new docx.

See Pandoc User's Guide: Options affecting specific writers (--reference-doc)

If you have problems with changing the default table style: see https://github.com/jgm/pandoc/issues/3275.

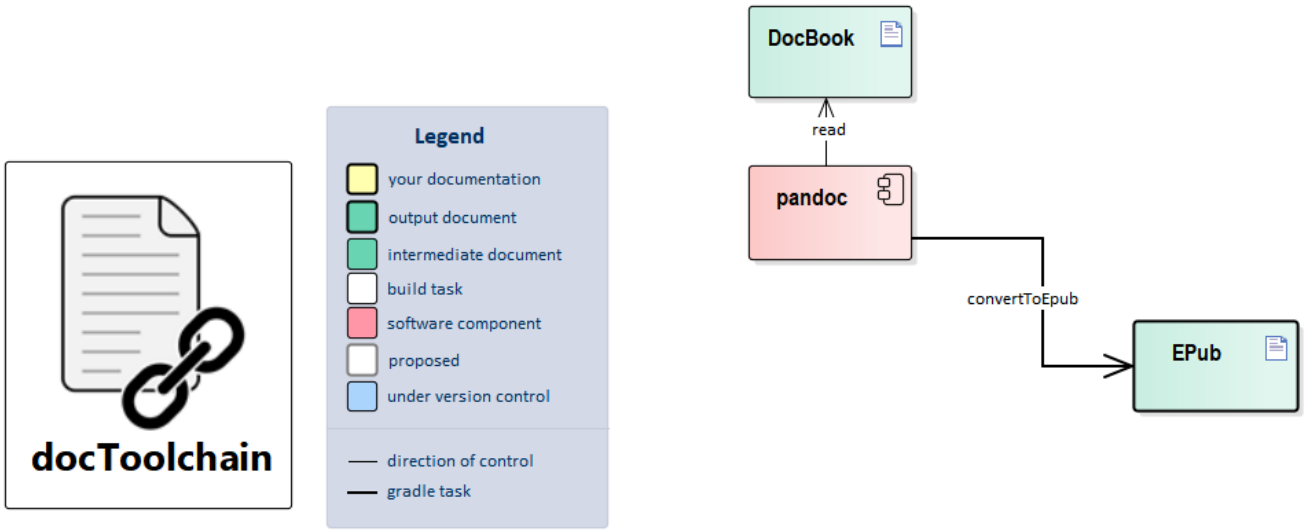
3.11.1. Source

pandoc.gradle

```
task createReferenceDoc (
        group: 'docToolchain helper',
        description: 'creates a docx file to be used as a format style reference in
task convertToDocx. Needs pandoc installed.',
        type: Exec
) {
    workingDir "$docDir"
    executable = "pandoc"
    args = ["-o", "${docDir}/${referenceDocFile}",
            "--print-default-data-file",
            "reference.docx"]

    doFirst {
        if(!(referenceDocFile?.trim())) {
            throw new GradleException("Option `referenceDocFile` is not defined in
config.groovy or has an empty value.")
        }
    }
}
```

3.12. convertToEpub

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_convertToEpub.adoc[]



Dependency: generateDocbook

This task uses pandoc to convert the DocBook output from AsciiDoctor to ePub. This way, you can read your documentation in a convenient way on an eBook-reader.

The resulting file can be found in `build/docs/epub`

Blog-Post: Turn your Document into an Audio-Book

3.12.1. Source

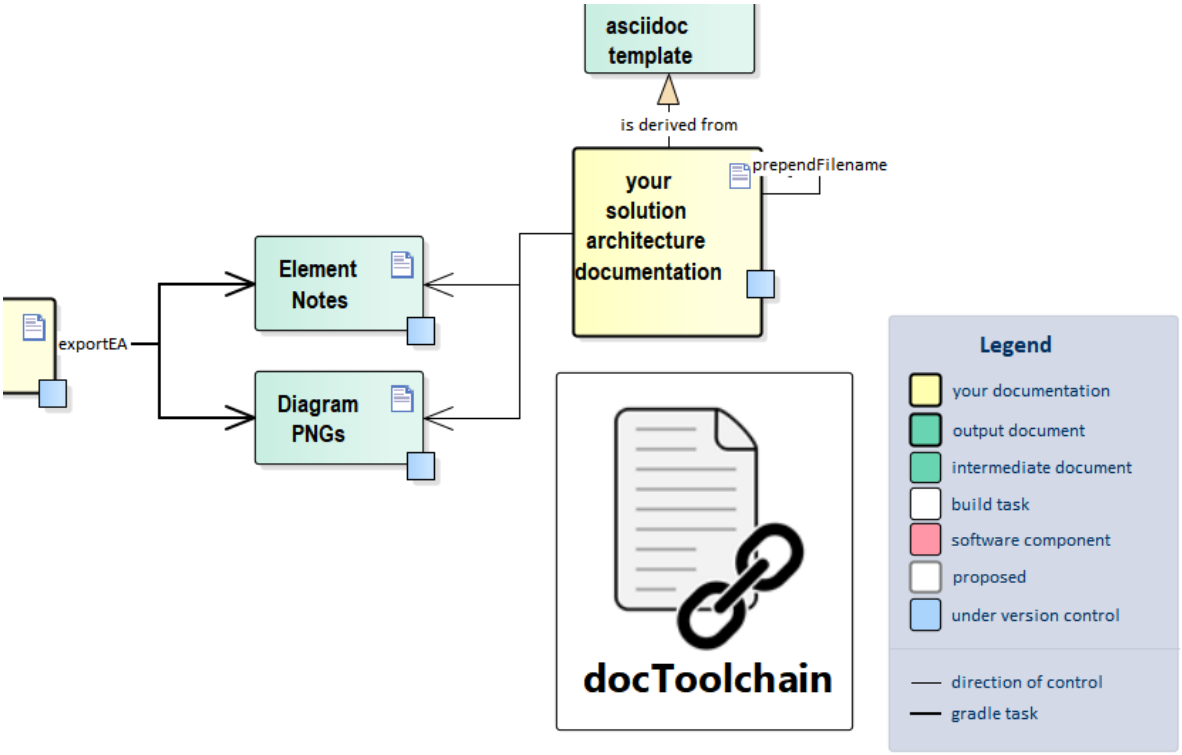
pandoc.gradle

```
task convertToEpub (
        group: 'docToolchain',
        description: 'converts file to .epub via pandoc. Needs pandoc installed.',
        type: Exec
) {
    // All files with option `epub` in config.groovy is converted to docbook and then
to epub.
    def sourceFilesEpub = sourceFiles.findAll { 'epub' in it.formats }
    sourceFilesEpub.each {
        def sourceFile = it.file.replace('.adoc', '.xml')
        def targetFile = sourceFile.replace('.xml', '.epub')

        workingDir "$targetDir/docbook"
        executable = "pandoc"
        args = ['-r','docbook',
                '-t','epub',
                '-o',"../epub/$targetFile",
                sourceFile]
    }
    doFirst {
        new File("$targetDir/epub/").mkdirs()
    }
}
```

3.13. exportEA

Currently this feature is WINDOWS-only. See related issue

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportEA.adoc[]



Blog-Posts: JIRA to Sparx EA, Did you ever wish you had better Diagrams?

3.13.1. Configuration

By default no special configuration is necessary. But, to be more specific on the project and its packages to be used for export, six optional parameter configurations are available. The parameters can be used independently from each other. A sample how to edit your projects Config.groovy is given in the 'Config.groovy' of the docToolchain project itself.

connection

Set the connection to a certain project or comment it out to use all project files inside the src folder or its child folder.

packageFilter

Add one or multiple packageGUIDs to be used for export. All packages are analysed, if no packageFilter is set.

exportPath

Relative path to base 'docDir' to which the diagrams and notes are to be exported. Default: "src/docs". Example: docDir = 'D:\work\mydoc\' ; exportPath = 'src/pdocs' ; Images will be

exported to 'D:\work\mydoc\src\pdocs\images\ea', Notes will be exported to 'D:\work\mydoc\src\pdocs\ea',

searchPath

Relative path to base 'docDir', in which Enterprise Architect project files are searched Default: "src/docs". Example: docDir = 'D:\work\mydoc\' ; exportPath = 'src/projects' ; Lookup for eap and eapx files starts in 'D:\work\mydoc\src\projects' and goes down the folder structure. **Note**: In case parameter 'connection' is already defined, the searchPath value is used, too. exportEA starts opening the database parameter 'connection' first and looks afterwards for further project files either in the searchPath (if set) or in the docDir folder of the project.

glossaryAsciiDocFormat

Depending on this configuration option, the EA project glossary is exported. If it is not set or an empty string, no glossary is exported. The glossaryAsciiDocFormat string is used to format each glossary entry in a certain asciidoc format. Following placeholder for the format string are defined: ID, TERM, MEANING, TYPE. One or many of these placeholder can be used by the output format.

Example: A valid output format to include the glossary as a flat list. The file can be included where needed in the documentation.

```
glossaryAsciiDocFormat = "TERM:: MEANING"
```

Other format strings can be used to include it as a table row. The glossary is sorted by terms in alphabetical order.

glossaryTypes

This parameter is used in case a glossaryAsciiDocFormat is defined, otherwise it is not evaluated. It is used to filter for certain types. If the glossaryTypes list is empty, all entries will be used. Example: glossaryTypes = ["Business", "Technical"]

diagramAttributes

Beside the diagram image, an EA diagram offers several useful attributes which could be required in the resulting document. If set, the string is used to create and store the diagram attributes to be included in the document. These placeholders are defined and filled with the diagram attributes if used in the diagramAttributes string: %DIAGRAM_AUTHOR%, %DIAGRAM_CREATED%, %DIAGRAM_GUID%, %DIAGRAM_MODIFIED%, %DIAGRAM_NAME%. %DIAGRAM_NOTES%, Example: diagramAttributes = "Last modification: %DIAGRAM_MODIFIED%"

You can add the string %NEWLINE% where a line break shall be added.

The resulting text is stored beside the diagram image using same path and file named, but different file extension (.ad). This can included in the document if required. If diagramAttributes is not set or string is empty, no file is written.

3.13.2. Glossary export

By setting the glossaryAsciiDocFormat the glossary terms stored in the EA project is exported into a

folder named 'glossary' below the configured exportPath. In case multiple EA projects were found for export, one glossary per project is exported. Each named with the projects GUID plus extension '.ad'. Each single file will be filtered (see glossaryTypes) and sorted in alphabetical order. In addition, a global glossary is created by using all single glossary files. This global file is named 'glossary.ad' and is also placed in the glossary folder. The global glossary is also filtered and sorted. In case there is one EA project only, the global glossary is written only.

3.13.3. Source

build.gradle

```
task exportEA(
        dependsOn: [streamingExecute],
        description: 'exports all diagrams and some texts from EA files',
        group: 'docToolchain'
) {
    doFirst {
    }
    doLast {
        logger.info("docToolchain > exportEA: "+docDir)
        logger.info("docToolchain > exportEA: "+mainConfigFile)
        def configFile = new File(docDir, mainConfigFile)
        def config = new ConfigSlurper().parse(configFile.text)
        def scriptParameterString = ""
        def exportPath = ""
        def searchPath = ""
        def glossaryPath = ""
        def readme = """This folder contains exported diagrams or notes from
Enterprise Architect.

Please note that these are generated files but reside in the `src`-folder in order to
be versioned.

This is to make sure that they can be used from environments other than windows.

# Warning!

**The contents of this folder   will be overwritten with each re-export!**

use `gradle exportEA` to re-export files
"""

        if(!config.exportEA.connection.isEmpty()){
            logger.info("docToolchain > exportEA: found "+config.exportEA.connection)
            scriptParameterString = scriptParameterString + "-c \"${config.exportEA
.connection}\""
        }
        if (!config.exportEA.packageFilter.isEmpty()){
            def packageFilterToCreate = config.exportEA.packageFilter as List
            logger.info("docToolchain > exportEA: package filter list size: "
+packageFilterToCreate.size())
```

```
            packageFilterToCreate.each { packageFilter ->
               scriptParameterString = scriptParameterString + " -p \"${packageFilter
}\""
            }
        }
        if (!config.exportEA.exportPath.isEmpty()){
            exportPath = new File(docDir, config.exportEA.exportPath).getAbsolutePath
()
        }
        else {
            exportPath = new File(docDir, 'src/docs').getAbsolutePath()
        }
        if (!config.exportEA.searchPath.isEmpty()){
            searchPath = new File(docDir, config.exportEA.searchPath).getAbsolutePath
()
        }
        else {
            searchPath = new File(docDir, 'src').getAbsolutePath()
        }
        scriptParameterString = scriptParameterString + " -d \"$exportPath\""
        scriptParameterString = scriptParameterString + " -s \"$searchPath\""
        logger.info("docToolchain > exportEA: exportPath: "+exportPath)

        //remove old glossary files/folder if exist
        new File(exportPath , 'glossary').deleteDir()
        //set the glossary file path in case an output format is configured, other no
glossary is written
        if (!config.exportEA.glossaryAsciiDocFormat.isEmpty()) {
            //create folder to store glossaries
            new File(exportPath , 'glossary/.').mkdirs()
            glossaryPath = new File(exportPath , 'glossary').getAbsolutePath()
            scriptParameterString = scriptParameterString + " -g \"$glossaryPath\""
        }
        //configure additional diagram attributes to be exported
        if (!config.exportEA.diagramAttributes.isEmpty()) {
            scriptParameterString = scriptParameterString + " -da \"$
config.exportEA.diagramAttributes\""
        }
        //make sure path for notes exists
        //and remove old notes
        new File(exportPath , 'ea').deleteDir()
        //also remove old diagrams
        new File(exportPath , 'images/ea').deleteDir()

        //create a readme to clarify things
        new File(exportPath , 'images/ea/.').mkdirs()
        new File(exportPath , 'images/ea/readme.ad').write(readme)
        new File(exportPath , 'ea/.').mkdirs()
        new File(exportPath , 'ea/readme.ad').write(readme)

        //execute through cscript in order to make sure that we get WScript.echo right
```

```
        "%SystemRoot%\\System32\\cscript.exe //nologo ${projectDir
}/scripts/exportEAP.vbs ${scriptParameterString}".executeCmd()
        //the VB Script is only capable of writing iso-8859-1-Files.
        //we now have to convert them to UTF-8
        new File(exportPath, 'ea/.').eachFileRecurse { file ->
            if (file.isFile()) {
                println "exported notes " + file.canonicalPath
                file.write(file.getText('iso-8859-1'), 'utf-8')
            }
        }

        //sort, filter and reformat a glossary if an output format is configured
        if (!config.exportEA.glossaryAsciiDocFormat.isEmpty()) {
            def glossaryTypes

            if (!config.exportEA.glossaryTypes.isEmpty()){
                glossaryTypes = config.exportEA.glossaryTypes as List
            }
            new GlossaryHandler().execute(glossaryPath, config.exportEA
.glossaryAsciiDocFormat, glossaryTypes);
        }
    }
}
```

scripts/exportEAP.vbs

```
    ' based on the "Project Interface Example" which comes with EA
    ' http://stackoverflow.com/questions/1441479/automated-method-to-export-
enterprise-architect-diagrams



    Dim EAapp 'As EA.App
    Dim Repository 'As EA.Repository
    Dim FS 'As Scripting.FileSystemObject

    Dim projectInterface 'As EA.Project

    Const   ForAppending = 8
    Const   ForWriting = 2

    ' Helper
    ' http://windowsitpro.com/windows/jsi-tip-10441-how-can-vbscript-create-multiple-
folders-path-mkdir-command
    Function MakeDir (strPath)
      Dim strParentPath, objFSO
      Set objFSO = CreateObject("Scripting.FileSystemObject")
      On Error Resume Next
      strParentPath = objFSO.GetParentFolderName(strPath)

      If Not objFSO.FolderExists(strParentPath) Then MakeDir strParentPath
```

```
      If Not objFSO.FolderExists(strPath) Then objFSO.CreateFolder strPath
      On Error Goto 0
      MakeDir = objFSO.FolderExists(strPath)
    End Function

    ' Replaces certain characters with '_' to avoid unwanted file or folder names
causing errors or structure failures.
    ' Regular expression can easily be extended with further characters to be
replaced.
    Function NormalizeName(theName)
       dim re : Set re = new regexp
       re.Pattern = "[\\/\[\]\s]"
       re.Global = True
       NormalizeName = re.Replace(theName, "_")
    End Function

    Sub WriteNote(currentModel, currentElement, notes, prefix)
        If (Left(notes, 6) = "{adoc:") Then
            strFileName = Mid(notes,7,InStr(notes,"}")-7)
            strNotes = Right(notes,Len(notes)-InStr(notes,"}"))
            set objFSO = CreateObject("Scripting.FileSystemObject")
            If (currentModel.Name="Model") Then
              ' When we work with the default model, we don't need a sub directory
              path = objFSO.BuildPath(exportDestination,"ea/")
            Else
              path =
objFSO.BuildPath(exportDestination,"ea/"&NormalizeName(currentModel.Name)&"/")
            End If
            MakeDir(path)

            post = ""
            If (prefix<>"") Then
                post = "_"
            End If
            MakeDir(path&prefix&post)

            set objFile =
objFSO.OpenTextFile(path&prefix&post&strFileName&".ad",ForAppending, True)
            name = currentElement.Name
            name = Replace(name,vbCr,"")
            name = Replace(name,vbLf,"")

            if (Left(strNotes, 3) = vbCRLF&"|") Then
                ' content should be rendered as table - so don't interfere with it
                objFile.WriteLine(vbCRLF)
            else
                'let's add the name of the object
                objFile.WriteLine(vbCRLF&vbCRLF&"."&name)
            End If
            objFile.WriteLine(vbCRLF&strNotes)
            objFile.Close
```

```
            if (prefix<>"") Then
                ' write the same to a second file
                set objFile = objFSO.OpenTextFile(path&prefix&".ad",ForAppending,
True)
                objFile.WriteLine(vbCRLF&vbCRLF&"."&name&vbCRLF&strNotes)
                objFile.Close
            End If
        End If
    End Sub

    Sub SyncJira(currentModel, currentDiagram)
        notes = currentDiagram.notes
        set currentPackage = Repository.GetPackageByID(currentDiagram.PackageID)
        updated = 0
        created = 0
        If (Left(notes, 6) = "{jira:") Then
            WScript.echo " >>>> Diagram jira tag found"
            strSearch = Mid(notes,7,InStr(notes,"}")-7)
            Set objShell = CreateObject("WScript.Shell")
            'objShell.CurrentDirectory = fso.GetFolder("./scripts")
            Set objExecObject = objShell.Exec ("cmd /K  groovy
./scripts/exportEAPJiraPrintHelper.groovy """ & strSearch &""" & exit")
            strReturn = ""
            x = 0
            y = 0
            Do While Not objExecObject.StdOut.AtEndOfStream
                output = objExecObject.StdOut.ReadLine()
                ' WScript.echo output
                jiraElement = Split(output,"|")
                name = jiraElement(0)&":"&vbCR&vbLF&jiraElement(4)
                On Error Resume Next
                Set requirement = currentPackage.Elements.GetByName(name)
                On Error Goto 0
                if (IsObject(requirement)) then
                    ' element already exists
                    requirement.notes = ""
                    requirement.notes = requirement.notes&"<a
href='"&jiraElement(5)&"'>"&jiraElement(0)&"</a>"&vbCR&vbLF
                    requirement.notes = requirement.notes&"Priority:
"&jiraElement(1)&vbCR&vbLF
                    requirement.notes = requirement.notes&"Created:
"&jiraElement(2)&vbCR&vbLF
                    requirement.notes = requirement.notes&"Assignee:
"&jiraElement(3)&vbCR&vbLF
                    requirement.Update()
                    updated = updated + 1
                else
                    Set requirement =
currentPackage.Elements.AddNew(name,"Requirement")
                    requirement.notes = ""
                    requirement.notes = requirement.notes&"<a
```

```
href='"&jiraElement(5)&"'>"&jiraElement(0)&"</a>"&vbCR&vbLF
                    requirement.notes = requirement.notes&"Priority:
"&jiraElement(1)&vbCR&vbLF
                    requirement.notes = requirement.notes&"Created:
"&jiraElement(2)&vbCR&vbLF
                    requirement.notes = requirement.notes&"Assignee:
"&jiraElement(3)&vbCR&vbLF
                    requirement.Update()
                    currentPackage.Elements.Refresh()
                    Set dia_obj =
currentDiagram.DiagramObjects.AddNew("l="&(10+x*200)&";t="&(10+y*50)&";b="&(10+y*50+44
)&";r="&(10+x*200+180),"")
                    x = x + 1
                    if (x>3) then
                      x = 0
                      y = y + 1
                    end if
                    dia_obj.ElementID = requirement.ElementID
                    dia_obj.Update()
                    created = created + 1
                end if
            Loop
            Set objShell = Nothing
            WScript.echo "created "&created&" requirements"
            WScript.echo "updated "&updated&" requirements"
        End If
    End Sub

    ' This sub routine checks if the format string defined in diagramAttributes
    ' does contain any characters. It replaces the known placeholders:
    ' %DIAGRAM_AUTHOR%, %DIAGRAM_CREATED%, %DIAGRAM_GUID%, %DIAGRAM_MODIFIED%,
    ' %DIAGRAM_NAME%, %DIAGRAM_NOTES%
    ' with the attribute values read from the EA diagram object.
    ' None, one or multiple number of placeholders can be used to create a diagram
attribute
    ' to be added to the document. The attribute string is stored as a file with the
same
    ' path and name as the diagram image, but with suffix .ad. So, it can
    ' easily be included in an asciidoc file.
    Sub SaveDiagramAttribute(currentDiagram, path, diagramName)
        If Len(diagramAttributes) > 0 Then
            filledDiagAttr = diagramAttributes
            set objFSO = CreateObject("Scripting.FileSystemObject")
            filename = objFSO.BuildPath(path, diagramName & ".ad")
            set objFile = objFSO.OpenTextFile(filename, ForWriting, True)
            filledDiagAttr = Replace(filledDiagAttr, "%DIAGRAM_AUTHOR%",
currentDiagram.Author)
            filledDiagAttr = Replace(filledDiagAttr, "%DIAGRAM_CREATED%",
currentDiagram.CreatedDate)
            filledDiagAttr = Replace(filledDiagAttr, "%DIAGRAM_GUID%",
currentDiagram.DiagramGUID)
```

```
            filledDiagAttr = Replace(filledDiagAttr, "%DIAGRAM_MODIFIED%",
currentDiagram.ModifiedDate)
            filledDiagAttr = Replace(filledDiagAttr, "%DIAGRAM_NAME%",
currentDiagram.Name)
            filledDiagAttr = Replace(filledDiagAttr, "%DIAGRAM_NOTES%",
currentDiagram.Notes)
            filledDiagAttr = Replace(filledDiagAttr, "%NEWLINE%",          vbCrLf)
            objFile.WriteLine(filledDiagAttr)
            objFile.Close
        End If
    End Sub
    Sub SaveDiagram(currentModel, currentDiagram)
        ' Open the diagram
        Repository.OpenDiagram(currentDiagram.DiagramID)

        ' Save and close the diagram
        set objFSO = CreateObject("Scripting.FileSystemObject")
        If (currentModel.Name="Model") Then
            ' When we work with the default model, we don't need a sub directory
            path = objFSO.BuildPath(exportDestination,"/images/ea/")
        Else
            path = objFSO.BuildPath(exportDestination,"/images/ea/" &
NormalizeName(currentModel.Name) & "/")
        End If
        path = objFSO.GetAbsolutePathName(path)
        MakeDir(path)

        diagramName = currentDiagram.Name
        diagramName = Replace(diagramName,vbCr,"")
        diagramName = Replace(diagramName,vbLf,"")
        diagramName = NormalizeName(diagramName)
        filename = objFSO.BuildPath(path, diagramName & ".png")

        projectInterface.SaveDiagramImageToFile(filename)
        WScript.echo " extracted image to " & filename
        If Not IsEmpty(diagramAttributes) Then
            SaveDiagramAttribute currentDiagram, path, diagramName
        End If
        Repository.CloseDiagram(currentDiagram.DiagramID)

        ' Write the note of the diagram
        WriteNote currentModel, currentDiagram, currentDiagram.Notes,
diagramName&"_notes"

        For Each diagramElement In currentDiagram.DiagramObjects
            Set currentElement = Repository.GetElementByID(diagramElement.ElementID)
            WriteNote currentModel, currentElement, currentElement.Notes,
diagramName&"_notes"
        Next
        For Each diagramLink In currentDiagram.DiagramLinks
            set currentConnector =
```

```
Repository.GetConnectorByID(diagramLink.ConnectorID)
            WriteNote currentModel, currentConnector, currentConnector.Notes,
diagramName&"_links"
        Next
    End Sub
    '
    ' Recursively saves all diagrams under the provided package and its children
    '
    Sub DumpDiagrams(thePackage,currentModel)
        Set currentPackage = thePackage

        ' export element notes
        For Each currentElement In currentPackage.Elements
            WriteNote currentModel, currentElement, currentElement.Notes, ""
            ' export connector notes
            For Each currentConnector In currentElement.Connectors
                ' WScript.echo currentConnector.ConnectorGUID
                if (currentConnector.ClientID=currentElement.ElementID) Then
                    WriteNote currentModel, currentConnector, currentConnector.Notes,
""
                End If
            Next
            if (Not currentElement.CompositeDiagram Is Nothing) Then
                SyncJira currentModel, currentElement.CompositeDiagram
                SaveDiagram currentModel, currentElement.CompositeDiagram
            End If
            if (Not currentElement.Elements Is Nothing) Then
                DumpDiagrams currentElement,currentModel
            End If
        Next


        ' Iterate through all diagrams in the current package
        For Each currentDiagram In currentPackage.Diagrams
            SyncJira currentModel, currentDiagram
            SaveDiagram currentModel, currentDiagram
        Next

        ' Process child packages
        Dim childPackage 'as EA.Package
        ' otPackage = 5
        if (currentPackage.ObjectType = 5) Then
            For Each childPackage In currentPackage.Packages
                call DumpDiagrams(childPackage, currentModel)
            Next
        End If
    End Sub

    Function SearchEAProjects(path)
      For Each folder In path.SubFolders
        SearchEAProjects folder
```

```
      Next

      For Each file In path.Files
            If fso.GetExtensionName (file.Path) = "eap" OR fso.GetExtensionName
(file.Path) = "eapx" Then
                WScript.echo "found "&file.path
                If (Left(file.name, 1) = "_") Then
                    WScript.echo "skipping, because it start with `_` (replication)"
                Else
                    OpenProject(file.Path)
                End If
            End If
      Next
    End Function

    'Gets the package object as referenced by its GUID from the Enterprise Architect
project.
    'Looks for the model node, the package is a child of as it is required for the
diagram export.
    'Calls the Sub routine DumpDiagrams for the model and package found.
    'An error is printed to console only if the packageGUID is not found in the
project.
    Function DumpPackageDiagrams(EAapp, packageGUID)
        WScript.echo "DumpPackageDiagrams"
        WScript.echo packageGUID
        Dim package
        Set package = EAapp.Repository.GetPackageByGuid(packageGUID)
        If (package Is Nothing) Then
          WScript.echo "invalid package - as package is not part of the project"
        Else
          Dim currentModel
          Set currentModel = package
          while currentModel.IsModel = false
            Set currentModel = EAapp.Repository.GetPackageByID(currentModel.parentID)
          wend
          ' Iterate through all child packages and save out their diagrams
          ' save all diagrams of package itself
          call DumpDiagrams(package, currentModel)
        End If
    End Function

    Function FormatStringToJSONString(inputString)
        outputString = Replace(inputString, "\", "\\")
        outputString = Replace(outputString, """", "\""")
        outputString = Replace(outputString, vbCrLf, "\n")
        outputString = Replace(outputString, vbLf, "\n")
        outputString = Replace(outputString, vbCr, "\n")
        FormatStringToJSONString = outputString
    End Function

    'If a valid file path is set, the glossary terms are read from EA repository,
```

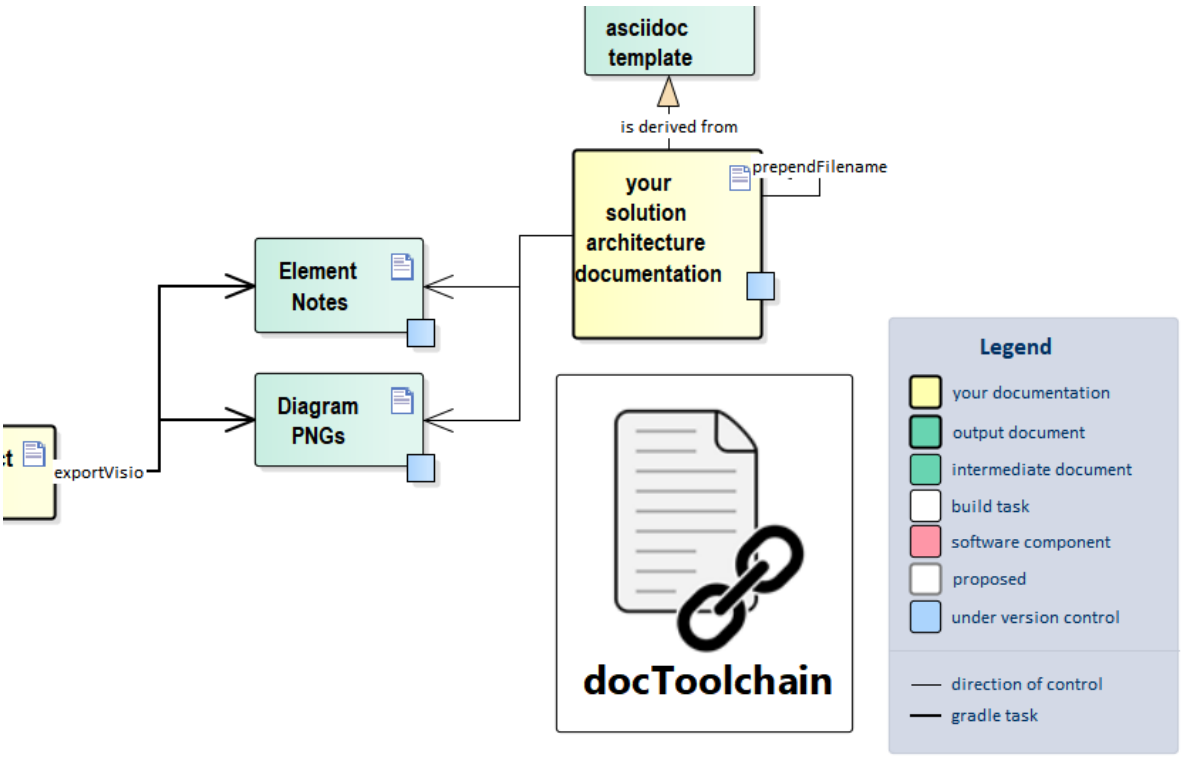
```
    'formatted in a JSON compatible format and written into file.
    'The file is read and reformatted by the exportEA gradle task afterwards.
    Function ExportGlossaryTermsAsJSONFile(EArepo)
        If (Len(glossaryFilePath) > 0) Then
            set objFSO = CreateObject("Scripting.FileSystemObject")
            GUID = Replace(EArepo.ProjectGUID,"{","")
            GUID = Replace(GUID,"}","")
            currentGlossaryFile = objFSO.BuildPath(glossaryFilePath,"/"&GUID&".ad")
            set objFile = objFSO.OpenTextFile(currentGlossaryFile,ForAppending, True)

            Set glossary = EArepo.Terms()
            objFile.WriteLine("[")
            dim counter
            counter = 0
            For Each term In glossary
                if (counter > 0) Then
                    objFile.Write(",")
                end if
                objFile.Write("{ ""term"" :
"""&FormatStringToJSONString(term.term)&""", ""meaning"" :
"""&FormatStringToJSONString(term.Meaning)&""",")
                objFile.WriteLine(" ""termID"" :
"""&FormatStringToJSONString(term.termID)&""", ""type"" :
"""&FormatStringToJSONString(term.type)&""" }")
                counter = counter + 1
            Next
            objFile.WriteLine("]")

            objFile.Close
        End If
    End Function

    Sub OpenProject(file)
      ' open Enterprise Architect
      Set EAapp = CreateObject("EA.App")
      WScript.echo "opening Enterprise Architect. This might take a moment..."
      ' load project
      EAapp.Repository.OpenFile(file)
      ' make Enterprise Architect to not appear on screen
      EAapp.Visible = False

      ' get repository object
      Set Repository = EAapp.Repository
      ' Show the script output window
      ' Repository.EnsureOutputVisible("Script")
      call ExportGlossaryTermsAsJSONFile(Repository)

      Set projectInterface = Repository.GetProjectInterface()

      Dim childPackage 'As EA.Package
      ' Iterate through all model nodes
```

```
      Dim currentModel 'As EA.Package
      If (InStrRev(file,"{") > 0) Then
        ' the filename references a GUID
        ' like {04C44F80-8DA1-4a6f-ECB8-982349872349}
        WScript.echo file
        GUID = Mid(file, InStrRev(file,"{")+0,38)
        WScript.echo GUID
        ' Iterate through all child packages and save out their diagrams
        call DumpPackageDiagrams(EAapp, GUID)
      Else
        If packageFilter.Count = 0 Then
          WScript.echo "done"
        ' Iterate through all model nodes
        For Each currentModel In Repository.Models
            ' Iterate through all child packages and save out their diagrams
            For Each childPackage In currentModel.Packages
                call DumpDiagrams(childPackage,currentModel)
            Next
        Next
        Else
          ' Iterate through all packages found in the package filter given by script
parameter.
          For Each packageGUID In packageFilter
            call DumpPackageDiagrams(EAapp, packageGUID)
          Next
        End If
      End If
      EAapp.Repository.CloseFile()
      ' Since EA 15.2 the Enterprise Architect background process hangs without
calling Exit explicitly
      EAapp.Repository.Exit()
    End Sub

  Private connectionString
  Private packageFilter
  Private exportDestination
  Private searchPath
  Private glossaryFilePath
  Private diagramAttributes

  exportDestination = "./src/docs"
  searchPath = "./src"
  Set packageFilter = CreateObject("System.Collections.ArrayList")
  Set objArguments = WScript.Arguments

  Dim argCount
  argCount = 0
  While objArguments.Count > argCount+1
    Select Case objArguments(argCount)
      Case "-c"
        connectionString = objArguments(argCount+1)
```

```
      Case "-p"
        packageFilter.Add objArguments(argCount+1)
      Case "-d"
        exportDestination = objArguments(argCount+1)
      Case "-s"
        searchPath = objArguments(argCount+1)
      Case "-g"
        glossaryFilePath = objArguments(argCount+1)
      Case "-da"
        diagramAttributes = objArguments(argCount+1)
    End Select
    argCount = argCount + 2
  WEnd
  set fso = CreateObject("Scripting.fileSystemObject")
  WScript.echo "Image extractor"

  ' Check both types in parallel - 1st check Enterprise Architect database connection,
2nd look for local project files
  If Not IsEmpty(connectionString) Then
     WScript.echo "opening database connection now"
     OpenProject(connectionString)
  End If
  WScript.echo "looking for .eap(x) files in " & fso.GetAbsolutePathName(searchPath)
  ' Dim f As Scripting.Files
  SearchEAProjects fso.GetFolder(searchPath)

  WScript.echo "finished exporting images"
```

3.14. exportVisio

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportVisio.adoc[]



This tasks searches for Visio files in the `/src/docs` folder. It then exports all diagrams and element notes to `/src/docs/images/visio` and `/src/docs/visio`.

- Images are stored as `/images/visio/[filename]-[pagename].png`
- Notes are stored as `/visio/[filename]-[pagename].adoc`

You can specify a file name to which the notes of a diagram are exported by starting any comment with `{adoc:[filename].adoc}`. It will then be written to `/visio/[filename].adoc`.

Currently, only Visio files stored directly in `/src/docs` are supported. For all others, the exported files will be in the wrong location.

Please close any running Visio instance before starting this task.

Todos: issue #112

3.14.1. Source

exportVisio.gradle

```
task exportVisio(
        dependsOn: [streamingExecute],
        description: 'exports all diagrams and notes from visio files',
        group: 'docToolchain'
) {
    doLast {
        //make sure path for notes exists
        //and remove old notes
        new File(docDir, 'src/docs/visio').deleteDir()
        //also remove old diagrams
        new File(docDir, 'src/docs/images/visio').deleteDir()
        //create a readme to clarify things
        def readme = """This folder contains exported diagrams and notes from visio
files.

Please note that these are generated files but reside in the `src`-folder in order to
be versioned.

This is to make sure that they can be used from environments other than windows.

# Warning!

**The contents of this folder   will be overwritten with each re-export!**

use `gradle exportVisio` to re-export files
"""
        new File(docDir, 'src/docs/images/visio/.').mkdirs()
        new File(docDir, 'src/docs/images/visio/readme.ad').write(readme)
        new File(docDir, 'src/docs/visio/.').mkdirs()
        new File(docDir, 'src/docs/visio/readme.ad').write(readme)
        def sourcePath = new File(docDir, 'src/docs/.').canonicalPath
        def scriptPath = new File(projectDir, 'scripts/VisioPageToPngConverter.ps1')
.canonicalPath
        "powershell ${scriptPath} -SourcePath ${sourcePath}".executeCmd()
    }
}
```

scripts/VisioPageToPngConverter.ps1

```
# Convert all pages in all visio files in the given directory to png files.
# A Visio windows might flash shortly.
# The converted png files are stored in the same directory
# The name of the png file is concatenated from the Visio file name and the page name.
# In addtion all the comments are stored in adoc files.
# If the Viso file is named "MyVisio.vsdx" and the page is called "FirstPage"
# the name of the png file will be "MyVisio-FirstPage.png" and the comment will
# be stored in "MyVisio-FirstPage.adoc".
# But for the name of the adoc files there is an alternative. It can be given in the
```

```
first
# line of the comment. If it is given in the comment it has to be given in curly
brackes
# with the prefix "adoc:", e.g. {adoc:MyCommentFile.adoc}
# Prerequisites: Viso and PowerShell has to be installed on the computer.
# Parameter: SourcePath where visio files can be found
# Example powershell VisoPageToPngConverter.ps1 -SourcePath c:\convertertest\

Param
(
    [Parameter(Mandatory=$true,ValueFromPipeline=$true,Position=0)]
    [Alias('p')][String]$SourcePath
)

Write-Output "starting to export visio"

If (!(Test-Path -Path $SourcePath))
{
    Write-Warning "The path ""$SourcePath"" does not exist or is not accessible,
please input the correct path."
    Exit
}

# Extend the source path to get only Visio files of the given directory and not in
subdircetories
If ($SourcePath.EndsWith("\"))
{
    $SourcePath  = "$SourcePath"
}
Else
{
    $SourcePath  = "$SourcePath\"
}

$VisioFiles = Get-ChildItem -Path "$SourcePath*" -Recurse -Include
*.vsdx,*.vssx,*.vstx,*.vxdm,*.vssm,*.vstm,*.vsd,*.vdw,*.vss,*.vst

If(!($VisioFiles))
{
    Write-Warning "There are no Visio files in the path ""$SourcePath""."
    Exit
}

$VisioApp = New-Object -ComObject Visio.Application
$VisioApp.Visible = $false

# Extract the png from all the files in the folder
Foreach($File in $VisioFiles)
{
    $FilePath = $File.FullName
    Write-Output "found ""$FilePath"" ."
```

```
    $FileDirectory = $File.DirectoryName   # Get the folder containing the Visio file.
Will be used to store the png and adoc files
    $FileBaseName = $File.BaseName -replace '[ :/\\*?|<>]','-'   # Get the filename to
be used as part of the name of the png and adoc files

    Try
    {
        $Document = $VisioApp.Documents.Open($FilePath)
        $Pages = $VisioApp.ActiveDocument.Pages
        Foreach($Page in $Pages)
        {
            # Create valid filenames for the png and adoc files
            $PngFileName = $Page.Name -replace '[ :/\\*?|<>]','-'
            $PngFileName = "$FileBaseName-$PngFileName.png"
            $AdocFileName = $PngFileName.Replace(".png", ".adoc")

            #TODO: this needs better logic
            Write-Output("$SourcePath\images\visio\$PngFileName")
            $Page.Export("$SourcePath\images\visio\$PngFileName")

            $AllPageComments = ""
            ForEach($PageComment in $Page.Comments)
            {
                # Extract adoc filename from comment text if the syntax is valid
                # Remove the filename from the text and save the comment in a file
with a valid name
                $EofStringIndex = $PageComment.Text.IndexOf(".adoc}")
                if ($PageComment.Text.StartsWith("{adoc") -And ($EofStringIndex -gt
6))
                {
                    $AdocFileName = $PageComment.Text.Substring(6, $EofStringIndex -1)
                    $AllPageComments += $PageComment.Text.Substring($EofStringIndex +
6)
                }
                else
                {
                    $AllPageComments += $PageComment.Text+"`n"
                }
            }
            If ($AllPageComments)
            {

                $AdocFileName = $AdocFileName -replace '[:/\\*?|<>]','-'
                #TODO: this needs better logic
                $stream = [System.IO.StreamWriter] "$SourcePath\visio\$AdocFileName"
                $stream.WriteLine($AllPageComments)
                $stream.close()
            }
        }
        $Document.Close()
    }
```

```
    Catch
    {
        if ($Document)
        {
            $Document.Close()
        }
        Write-Warning "One or more visio page(s) in file ""$FilePath"" have been lost
in this converting."
        Write-Warning "Error was: $_"
    }
}
$VisioApp.Quit()
```

3.15. exportDrawIo

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportDrawIo.adoc[]

There is no `exportDrawIo` task because it is not required. You can continue to use draw.io as an editor for your diagrams by making a change to your diagram authoring workflow.

Export your draw.io diagrams as a PNG with the source embedded in the file metadata. Using this approach allows you to embed your diagrams into AsciiDoc source as you normally do with the `image::` macro, with the added advantage of storing the diagram source with the image itself.

If you are converting a Confluence page with embedded draw.io diagrams to AsciiDoc, you can use this export workflow to continue to enjoy the draw.io editing experience.

Export an editable PNG diagram from Confluence

1. Load the diagram you want to export from Confluence.
2. Click **File › Export as › PNG...**
3. In the Image modal, make sure that "Include a copy of my diagram" is selected.
4. Click Export to save the PNG file with the pattern `[file].dio.png`.

Specifying `.dio` (short for "**d**raw**io**") in the name will help you identify PNG files containing embedded XML diagram source.

5. Commit the exported PNG file to source control.

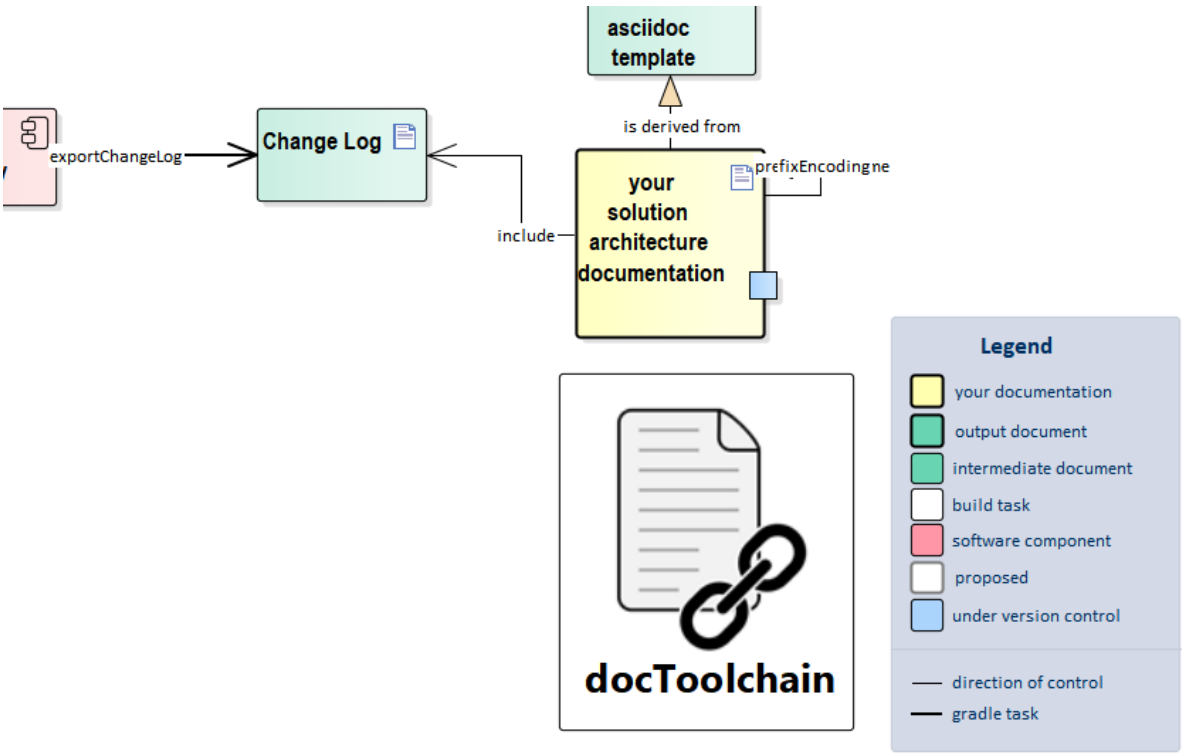
You now have a diagram that can be managed in source control, added to your documentation source, and edited using a draw.io Desktop version.

Draw.io offers free and open source desktop editors for all major operating system platforms. See https://about.draw.io/integrations/ to find a desktop editor application compatible with your operating system. When you use the desktop version, just create your diagram with the `.png` or even better `.dio.png` extension and draw.io will always save your diagam as PNG with the source as meta data.

NEW! Draw.io is now called diagrams.net and there is a free plugin for VS Code and IntelliJ to edit your diagrams even offline!

3.16. exportChangeLog

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportChangeLog.adoc[]



As the name says, this task exports the changelog to be referenced from within your documentation - if needed. The changelog is written to `build/docs/changelog.adoc`.

This task can be configured to use different source control system or different directory. To configure the task, copy `template_config/scripts/ChangelogConfig.groovy` to your directory and modify to your needs. Then give the path to your configuration file to the task using -PchangelogConfigFile=<your config file>. See the description inside the template for more details.

By default, the source is the Git changelog for the path `src/docs` - it only contains the commit messages for changes on the documentation. All changes on the build or other sources from the repository will not show up. By default, the changelog contains the changes with *date*, *author* and *commit message* already formatted as AsciiDoc table content:

```
| 09.04.2017
| Ralf D. Mueller
| fix #24 template updated to V7.0

| 08.04.2017
| Ralf D. Mueller
| fixed typo
```

You simply include it like this:

```
.Changes
[options="header",cols="1,2,6"]
|====
| Date
| Author
| Comment

include::../../build/docs/changelog.adoc[]

|====
```

By excluding the table definition, you can easily translate the table headings through different text snippets.

it might make sense to only include certain commit messages from the change log or exclude others (starting with # or //?). But this isn't implemented yet.

Blog-Post: The only constant in life is change

3.16.1. Source

exportChangelog.gradle

```
task exportChangeLog(
        description: 'exports the change log from a git subpath',
        group: 'docToolchain'
) {
    doFirst {
        new File(targetDir).mkdirs()
    }
    doLast {
        logger.info("docToolchain> docDir: "+docDir)
        logger.info("docToolchain> mainConfigFile: "+mainConfigFile)
        def config = new ConfigSlurper().parse(new File(docDir, mainConfigFile).text)

        def cmd = "${config.changelog.cmd} ."
        def changes = cmd.execute(null, new File(docDir, config.changelog.dir)).text
        def changelog = new File(targetDir, 'changelog.adoc')
        logger.info "> changelog exported ${changelog.canonicalPath}"
        changelog.write(changes)
    }
}
```

3.17. exportContributors

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportContributors.adoc[]

This tasks crawls through all Asciidoctor source files and extracts a list of contributors. This list is then rendered as AsciiDoc images of the contributor's gravatar picture.

The extracted lists are stored in `C:\doctoolchain\./build/test3/contributors/manual/03_task_exportContributors.adoc` to be easily included in your documents.

Best way to use this feature is to create a `feedback.adoc` file which might look like this:

feedback.adoc

```
ifndef::backend-pdf[] ①

    image::https://img.shields.io/badge/improve-this%20doc-
orange.svg[link={manualdir}{filename}, float=right] ②
    image::https://img.shields.io/badge/create-an%20issue-
blue.svg[link="https://github.com/docToolchain/documentation/issues/new?title=&body=%0
A%0A%5BEnter%20feedback%20here%5D%0A%0A%0A---%0A%23page:{filename}", float=right] ③

endif::[]

include::{targetDir}/contributors/{filename}[] ④
```

① do not show this section when rendered as PDF

② create an "improve this doc" button which links to the git sources

③ create a "create-an-issue" button which links to your issue tracker

④ include the list of contributors created by this task

3.17.1. Reading Time

The task also appends to the list of contributors the estimated reading time.

3.17.2. File Attributes

The task now also exports some attributes of the git files. The extracted attributes are stored in `C:\doctoolchain\./build/test3/fileattribs/manual/03_task_exportContributors.adoc`.

```
:lastUpdated: 16.05.2019 06:22
:lastAuthorName: Ralf D. Müller
:lastAuthorEmail: ralf.d.mueller@gmail.com
:lastAuthorAvatar:
http://www.gravatar.com/avatar/cc5f3bf8b3cb91c985ed4fd046aa451d?d=identicon[32,32,role
='gravatar',alt='Ralf D. Müller',title='Ralf D. Müller']
:lastMessage: #310 started to document config options
```

This enables you to import these attributes the same way you import the contributors list and use these attributes.

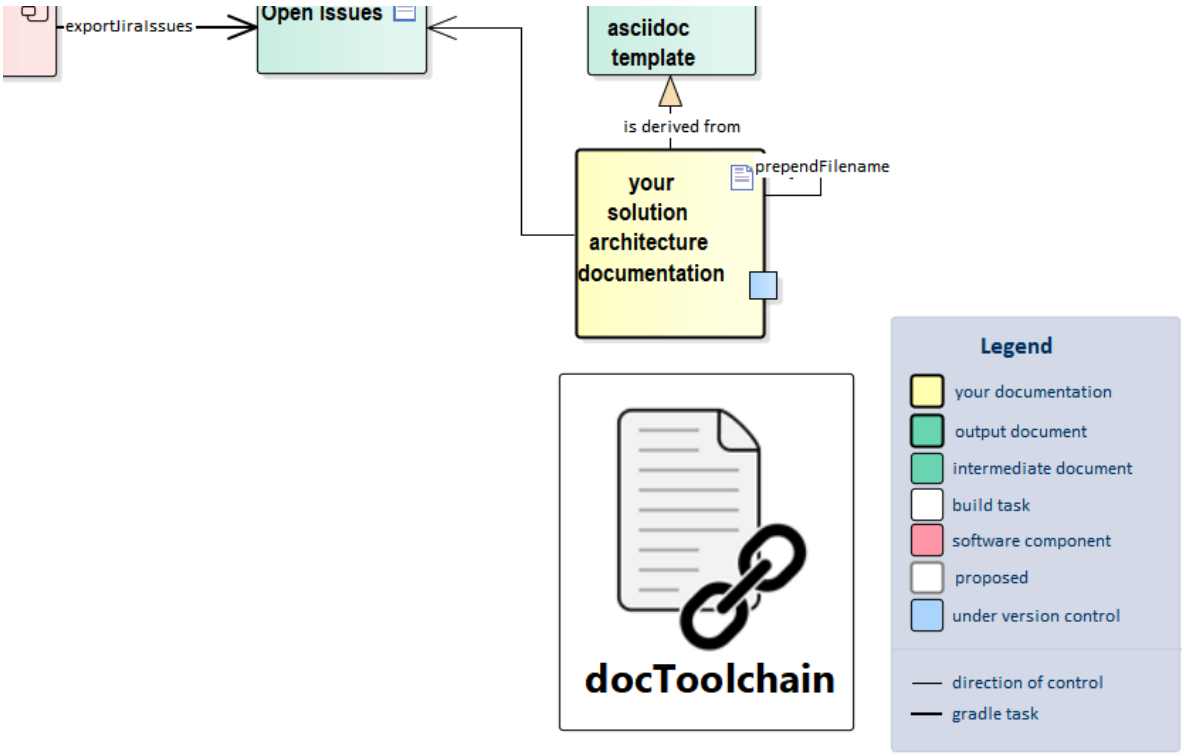
Here is an example:

feedback.adoc

```
include::{targetDir}/fileattribs/{filename}[]

Last updated {lastUpdated} by {lastAuthorName}
```

3.18. exportJiraIssues

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportJiraIssues.adoc[]



This task exports all issues for a given query(s) from Jira as AsciiDoc table or Excel file.

The configuration for this task can be found within `Config.gradle` (`gradle.properties` can be used as a fallback configuration). Username/Password is deprecated, so you have to use username/API-token instead. An API-Token can be created through https://id.atlassian.com/manage/api-tokens. It is encouraged to keep username and API token out of git repository and to pass it as environment variables to the docToolchain.

Blog-Post: Living Documents for Agile Projects

3.18.1. Configuration

Jira configuration support list requests to Jira where results of each requests will be saved in a file with specifies filename. Flags *saveAsciidoc* & *saveExcel* allow users to easily configure in which format results should be saved.

Old configuration based on **single Jira query is deprecated** (single 'jql' parameter). Support for it will be removed in the near future. Please migrate to the new configuration that allows multiple Jira queries.

Config.groovy

```
// Configuration for Jira related tasks
jira = [:]

jira.with {

    // endpoint of the JiraAPI (REST) to be used
    api = 'https://your-jira-instance'

    /*
    WARNING: It is strongly recommended to store credentials securely instead of
commiting plain text values to your git repository!!!

    Tool expects credentials that belong to an account which has the right permissions
to read the JIRA issues for a given project.
    Credentials can be used in a form of:
     - passed parameters when calling script (-PjiraUser=myUsername
-PjiraPass=myPassword) which can be fetched as a secrets on CI/CD or
     - gradle variables set through gradle properties (uses the 'jiraUser' and
'jiraPass' keys)
    Often, Jira & Confluence credentials are the same, in which case it is recommended
to pass CLI parameters for both entities as
    -Pusername=myUser -Ppassword=myPassword
    */

    // the key of the Jira project
    project = 'PROJECTKEY'

    // the format of the received date time values to parse
    dateTimeFormatParse = "yyyy-MM-dd'T'H:m:s.SSSz" // i.e. 2020-07-24'T'9:12:40.999
CEST

    // the format in which the date time should be saved to output
    dateTimeFormatOutput = "dd.MM.yyyy HH:mm:ss z" // i.e. 24.07.2020 09:02:40 CEST

    // the label to restrict search to
    label = 'label1'

    // Legacy settings for Jira query. This setting is deprecated & support for it
will soon be completely removed. Please use JiraRequests settings
    jql = "project='%jiraProject%' AND labels='%jiraLabel%' ORDER BY priority DESC,
duedate ASC"

    // Base filename in which Jira query results should be stored
    resultsFilename = 'JiraTicketsContent'

    saveAsciidoc = true // if true, asciidoc file will be created with *.adoc
extension
    saveExcel = true // if true, Excel file will be created with *.xlsx extension

    // Output folder for this task inside main outputPath
    resultsFolder = 'JiraRequests'
```

```
    /*
    List of requests to Jira API:
    These are basically JQL expressions bundled with a filename in which results will
be saved.
    User can configure custom fields IDs and name those for column header,
    i.e. customfield_10026:'Story Points' for Jira instance that has custom field with
that name and will be saved in a coloumn named "Story Points"
    */
    requests = [
        new JiraRequest(
            filename:"File1_Done_issues",
            jql:"project='%jiraProject%' AND status='Done' ORDER BY duedate ASC",
            customfields: [customfield_10026:'Story Points']
        ),
        new JiraRequest(
            filename:'CurrentSprint',
            jql:"project='%jiraProject%' AND Sprint in openSprints() ORDER BY priority
DESC, duedate ASC",
            customfields: [customfield_10026:'Story Points']
        ),
    ]
}

@groovy.transform.Immutable
class JiraRequest {
    String filename  //filename (without extension) of the file in which JQL results
will be saved. Extension will be determined automatically for Asciidoc or Excel file
    String jql // Jira Query Language syntax
    Map<String,String> customfields // map of customFieldId:displayName values for
Jira fields which don't have default names, i.e. customfield_10026:StoryPoints
}
```

3.18.2. Source

exportJiraIssues.gradle

```
task exportJiraIssues(
        description: 'exports all jira issues from a given search',
        group: 'docToolchain'
) {
    doLast {
        final String taskSubfolderName = config.jira.resultsFolder
        final File targetFolder = new File(targetDir + File.separator +
taskSubfolderName)
        if (!targetFolder.exists()) targetFolder.mkdirs()
        logger.debug("Output folder for 'exportJiraIssues' task is: '${targetFolder}'
")

        // map configuration from Config.groovy to existing variables for
```

```
compatibility with naming of Jira settings in gradle.properties
        def jiraRoot = config.jira.api
        def jiraProject = config.jira.project
        def jiraLabel = config.jira.label
        def jiraResultsFilename = config.jira.resultsFilename
        def jiraDateTimeFormatParse = config.jira.dateTimeFormatParse
        def jiraDateTimeOutput = config.jira.dateTimeFormatOutput
        def defaultFields = 'priority,created,resolutiondate,summary,assignee,status'

        def jira = new groovyx.net.http.RESTClient(jiraRoot + '/rest/api/2/')
        jira.encoderRegistry = new groovyx.net.http.EncoderRegistry(charset: 'utf-8')
        def headers = [
                'Authorization': "Basic " + config.jira.credentials,
                'Content-Type' : 'application/json; charset=utf-8'
        ]

        def jiraRequests = config.jira.requests

        if (config.jira.jql) {
            logger.warn(">>>Found legacy Jira requests. Please migrate to the new Jira
configuration ASAP. Old config with jql will be removed soon")
            writeAsciiDocFileForLegacyConfiguration(targetFolder, jira, headers,
config.jira)
        }

        jiraRequests.each {rq ->
            logger.quiet("Request to Jira API for '${rq.filename}' with query: '${rq
.jql}'")

            def allHeaders = "${defaultFields},${rq.customfields.values().join(",")}"
            def allFieldIds = "${defaultFields},${rq.customfields.keySet().join(",")}"
            logger.quiet("Preparing headers for default & custom fields: ${allHeaders
}")
            logger.quiet("Preparing field IDs for default & custom fields:
${allFieldIds}")

            // Save AsciiDoc file
            if (config.jira.saveAsciidoc) {
                def extension = 'adoc'
                jiraResultsFilename = "${rq.filename}.${extension}"
                logger.info("Results will be saved in '${rq.filename}.${extension}'
file")

                def jiraDataAsciidoc = new File(targetFolder, "${rq.filename}.
${extension}")
                jiraDataAsciidoc.write(".${rq.filename}\n", 'utf-8')
                jiraDataAsciidoc.append("|=== \n")

                // AsciiDoc table headers (custom fields map needs values here)
                jiraDataAsciidoc.append("|Key ", 'utf-8')
                allHeaders.split(",").each {field ->
```

```
                    jiraDataAsciidoc.append("|${field.capitalize()} ", 'utf-8')
                }
                jiraDataAsciidoc.append("\n", 'utf-8')

                jira.get(path: 'search',
                    query: ['jql'       : rq.jql.replaceAll('%jiraProject%',
jiraProject).replaceAll('%jiraLabel%', jiraLabel),
                            'maxResults': 1000,
                            fields: "${allFieldIds}"
                    ],
                    headers: headers
                ).data.issues.each { issue ->
                    //logger.quiet(">> Whole issue ${issue.key}:\n   ${issue.fields}")
                    jiraDataAsciidoc.append("| ${jiraRoot}/browse/${issue.key}[${
issue.key}] ", 'utf-8')
                    jiraDataAsciidoc.append("| ${issue.fields.priority.name} ", 'utf-
8')
                    jiraDataAsciidoc.append("| ${Date.parse(jiraDateTimeFormatParse,
issue.fields.created).format(jiraDateTimeOutput)} ", 'utf-8')
                    jiraDataAsciidoc.append("| ${issue.fields.resolutiondate ? Date
.parse(jiraDateTimeFormatParse, issue.fields.resolutiondate).format(
jiraDateTimeOutput) : ''} ", 'utf-8')
                    jiraDataAsciidoc.append("| ${issue.fields.summary} ", 'utf-8')
                    jiraDataAsciidoc.append("| ${issue.fields.assignee ? issue.fields
.assignee.displayName : 'not assigned'}", 'utf-8')
                    jiraDataAsciidoc.append("| ${issue.fields.status.name} ", 'utf-8')

                    rq.customfields.each { field ->
                        def foundCustom = issue.fields.find {it.key == field.key}
                        //logger.quiet("Examining issue '${issue.key}' for custom
field '${field.key}' has found: '${foundCustom}'")
                        jiraDataAsciidoc.append("| ${foundCustom ? foundCustom.value :
'-'}\n", 'utf-8')
                    }
                }
                jiraDataAsciidoc.append("|=== \n")
            } else {
                logger.quiet("Set saveAsciidoc=true in '${configFile.name}' to save
results in AsciiDoc file")
            }

            // Save Excel file
            if (config.jira.saveExcel) {
                def extension = 'xlsx'
                jiraResultsFilename = "${rq.filename}.${extension}"
                logger.quiet(">> Results will be saved in '${rq.filename}.${extension
}' file")

                //def jiraDataAsciidoc = new File(targetFolder,
"${rq.filename}.${extension}")
```

```
                def jiraDataXls = new File(targetFolder, jiraResultsFilename)
                def jiraFos = new FileOutputStream(jiraDataXls)

                Workbook wb = new XSSFWorkbook();
                CreationHelper hyperlinkHelper = wb.getCreationHelper();
                def sheetName = "${rq.filename}"
                def ws = wb.createSheet(sheetName)

                String rgbS = "A7A7A7"
                byte[] rgbB = Hex.decodeHex(rgbS)
                XSSFColor color = new XSSFColor(rgbB, null) //IndexedColorMap has no
usage until now. So it can be set null.
                XSSFCellStyle headerCellStyle = (XSSFCellStyle) wb.createCellStyle()
                headerCellStyle.setFillForegroundColor(color)
                headerCellStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND)

                def titleRow = ws.createRow(0);
                int cellNumber = 0;
                titleRow.createCell(cellNumber).setCellValue("Key")
                allHeaders.split(",").each {field ->
                    titleRow.createCell(++cellNumber).setCellValue("${field.
capitalize()}")
                }
                def lastRow = titleRow.getRowNum()
                titleRow.setRowStyle(headerCellStyle)

                jira.get(path: 'search',
                    query: ['jql'       : rq.jql.replaceAll('%jiraProject%',
jiraProject).replaceAll('%jiraLabel%', jiraLabel),
                            'maxResults': 1000,
                            fields: "${allFieldIds}"
                    ],
                    headers: headers
                ).data.issues.each { issue ->
                    int cellPosition = 0
                    def row = ws.createRow(++lastRow)
                    Hyperlink link = hyperlinkHelper.createHyperlink(HyperlinkType.
URL)
                    link.setAddress("${jiraRoot}/browse/${issue.key}")
                    Cell cellWithUrl = row.createCell(cellPosition)
                    cellWithUrl.setCellValue("${issue.key}")
                    cellWithUrl.setHyperlink(link)

                    row.createCell(++cellPosition).setCellValue("${issue.fields
.priority.name}")
                    row.createCell(++cellPosition).setCellValue("${Date.parse
(jiraDateTimeFormatParse, issue.fields.created).format(jiraDateTimeOutput)}")
                    row.createCell(++cellPosition).setCellValue("${issue.fields
.resolutiondate ? Date.parse(jiraDateTimeFormatParse, issue.fields.resolutiondate)
.format(jiraDateTimeOutput) : ''}")
                    row.createCell(++cellPosition).setCellValue("${issue.fields
```

```
.summary}")
                    row.createCell(++cellPosition).setCellValue("${issue.fields
.assignee ? issue.fields.assignee.displayName : ''}")
                    row.createCell(++cellPosition).setCellValue("${issue.fields.
status.name}")

                    // Custom fields
                    rq.customfields.each { field ->
                        def position = ++cellPosition
                        def foundCustom = issue.fields.find {it.key == field.key}
                        row.createCell(position).setCellValue("${foundCustom ?
foundCustom.value : '-'}")
                    }
                }

                // set jira issue key column fits the content width

                for(int colNum = 0; colNum<allHeaders.size()+1;colNum++) {
                    ws.autoSizeColumn(colNum)
                }
                // Set summary column width slightly wider but fixed size, so it
doesn't change with every summary update
                ws.setColumnWidth(4, 25*384)


                wb.write(jiraFos)
            } else {
                logger.quiet("Set saveExcel=true in '${configFile.name}' to save
results in Excel file")
            }
        }
    }
}

// This method can be removed when support for legacy Jira configuration is gone
def writeAsciiDocFileForLegacyConfiguration(def targetFolder, def restClient, def
headers, def jiraConfig) {
    def resultsFilename = "${jiraConfig.resultsFilename}_legacy.adoc"

    def openIssues = new File(targetFolder, "${resultsFilename}")
    openIssues.write(".Table {Title}\n", 'utf-8')
    openIssues.append("|=== \n")
    openIssues.append("|Key |Priority |Created | Assignee | Summary\n", 'utf-8')
        def legacyJql = jiraConfig.jql.replaceAll('%jiraProject%', config.jira.
project).replaceAll('%jiraLabel%', config.jira.label)
        println ("Results for legacy query '${legacyJql}' will be saved in '
${resultsFilename}' file")

        restClient.get(path: 'search',
                query: ['jql'       : legacyJql,
                        'maxResults': 1000,
```

```
                        'fields'    :
'created,resolutiondate,priority,summary,timeoriginalestimate, assignee'
                ],
                headers: headers
      ).data.issues.each { issue ->
        openIssues.append("| ${jiraRoot}/browse/${issue.key}[${issue.key}] ", 'utf-8')
        openIssues.append("| ${issue.fields.priority.name} ", 'utf-8')
        openIssues.append("| ${Date.parse(jiraConfig.dateTimeFormatParse, issue.
fields.created).format(jiraConfig.dateTimeFormatOutput)} ", 'utf-8')
        openIssues.append("| ${issue.fields.assignee ? issue.fields.assignee
.displayName : 'not assigned'}", 'utf-8')
        openIssues.append("| ${issue.fields.summary} ", 'utf-8')
    }
    openIssues.append("|=== \n")
}
```

3.19. exportJiraSprintChangelogIssues

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportJiraSprintChangelog.adoc[]

This task exports simplified (Key & summary) list of Jira issues for specific Sprint. Few additional fields (i.e. assignee etc..) can be switched using configuration flags.

Users can define one specific sprint in which they are interested. This will result in generating Asciidoc and Excel file. In case sprint defined in users configuration isn't found, changelogs for all sprints that match configuration will be saved in separate Asciidoc files and in a different tabs of an Excel file.

The configuration for this task can be found within `Config.gradle`. In addition to the below listed configuration snippet, it is important to configure Jira API and credentials in Jira section of the configuration inside same file.

3.19.1. Configuration

Config.groovy

```
// Sprint changelog configuration generate changelog lists based on tickets in sprints
of an Jira instance.
// This feature requires at least Jira API & credentials to be properly set in Jira
section of this configuration
sprintChangelog = [:]
sprintChangelog.with {
    sprintState = 'closed' // it is possible to define multiple states, i.e. 'closed,
active, future'
    ticketStatus = "Done, Closed" // it is possible to define multiple ticket
statuses, i.e. "Done, Closed, 'in Progress'"

    showAssignee = false
    showTicketStatus = false
    showTicketType = true
    sprintBoardId = 12345  // Jira instance probably have multiple boards; here it can
be defined which board should be used

    // Output folder for this task inside main outputPath
    resultsFolder = 'Sprints'

    // if sprintName is not defined or sprint with that name isn't found, release
notes will be created on for all sprints that match sprint state configuration
    sprintName = 'PRJ Sprint 1' // if sprint with a given sprintName is found, release
notes will be created just for that sprint
    allSprintsFilename = 'Sprints_Changelogs' // Extension will be automatically
added.
}
```

3.19.2. Source

exportJiraSprintChangelog.gradle

```
task exportJiraSprintChangelog(
        description: 'exports all jira issues from Sprint for release notes',
        group: 'docToolchain'
) {
    doLast {
        // Pre defined ticket fields for Changelog based on Jira Sprints
        def defaultTicketFields = 'summary,status,assignee,issuetype'

        // retrieving sprints for a given board
        def sprints = { apiSprints, headers, boardId, sprintState ->
                apiSprints.get(path: "agile/latest/board/${boardId}/sprint",
                    query:[state: "${sprintState}"],
                    headers: headers
                ).data
        }

        // retrieving issues for given sprint
        def issues = { apiIssues, headers, boardId, sprintId, status ->
            apiIssues.get(path: "agile/latest/board/${boardId}/sprint/${sprintId
}/issue",
                    query: ['jql'       : "status in (${status}) ORDER BY type DESC,
status ASC",
                            'maxResults': 1000,
                            fields: defaultTicketFields
                    ],
                    headers: headers
            ).data
        }

        // preparing target folder for generated files
        final String taskSubfolderName = config.sprintChangelog.resultsFolder
        final File targetFolder = new File(targetDir + File.separator +
taskSubfolderName)
        if (!targetFolder.exists()) targetFolder.mkdirs()
        logger.debug("Output folder for 'exportJiraSprintChangelog' task is: '
${targetFolder}'")

        // Getting configuration
        def jiraRoot = config.jira.api
        def jiraProject = config.jira.project

        def sprintState = config.sprintChangelog.sprintState
        def ticketStatusForReleaseNotes = config.sprintChangelog.ticketStatus
        def sprintBoardId = config.sprintChangelog.sprintBoardId
        def showAssignee = config.sprintChangelog.showAssignee
        def showTicketStatus = config.sprintChangelog.showTicketStatus
        def showTicketType = config.sprintChangelog.showTicketType
```

```
        def sprintName = config.sprintChangelog.sprintName
        def allSprintsFilename = config.sprintChangelog.allSprintsFilename

        logger.info("\n=========================\nJira Release notes config
\n=========================")
        logger.info("Spring Board ID: ${sprintBoardId}")
        logger.info("Show assignees: ${showAssignee}. Show ticket status:
${showTicketStatus}. Show ticket type: ${showTicketType}")
        logger.info("Filtering for sprints with configured state: '${sprintState}'")
        logger.info("Filtering for issues with configured statuses:
${ticketStatusForReleaseNotes}")
        logger.info("Attempt to generate release notes for sprint with a name: '
${sprintName}'")
        logger.info("Filename used for all sprints: '${allSprintsFilename}'")

        def api = new groovyx.net.http.RESTClient(jiraRoot + '/rest/')
        api.encoderRegistry = new groovyx.net.http.EncoderRegistry(charset: 'utf-8')
        def headers = [
                'Authorization': "Basic " + config.jira.credentials,
                'Content-Type' : 'application/json; charset=utf-8'
        ]

        def allChangelogsFilename = "${allSprintsFilename}.xlsx"
        logger.quiet("Changelogs of all sprints will be saved in '
${allChangelogsFilename}' file")

        def changelogsXls = new File(targetFolder, allChangelogsFilename)
        def changelogsXlsFos = new FileOutputStream(changelogsXls)
        Workbook wb = new XSSFWorkbook();
        CreationHelper hyperlinkHelper = wb.getCreationHelper();

        String rgbS = "A7A7A7"
        byte[] rgbB = Hex.decodeHex(rgbS) // get byte array from hex string
        XSSFColor color = new XSSFColor(rgbB, null) //IndexedColorMap has no usage
until now. So it can be set null.
        XSSFCellStyle headerCellStyle = (XSSFCellStyle) wb.createCellStyle()
        headerCellStyle.setFillForegroundColor(color)
        headerCellStyle.setFillPattern(FillPatternType.SOLID_FOREGROUND)

        // prepare tickets according to configuration
        def columns = ['key'].plus(defaultTicketFields.split(',').collect())
        if (!showAssignee) { columns = columns.minus('assignee')}
        if (!showTicketStatus) { columns = columns.minus('status')}
        if (!showTicketType) { columns = columns.minus('issuetype')}
        logger.info("Release notes will contain following info: ${columns}")

        logger.info("\n====================\n      Sprints\n====================")
        //
        def allMatchedSprints = sprints(api, headers, sprintBoardId, sprintState)
.values
        def foundExactSprint = allMatchedSprints.any {it.name == sprintName}
```

```
        logger.info("All sprints that matched configuration: ${allMatchedSprints.size
()}")

        def sprintsForChangelog = foundExactSprint ? allMatchedSprints.stream().
filter() {it.name == sprintName} : allMatchedSprints
        logger.info("Found exact Sprint with name '${sprintName}': ${foundExactSprint
}.")

        sprintsForChangelog.each { sprint ->
            logger.quiet("\nSprint: $sprint.name [id: $sprint.id] state <$
sprint.state>")

            /* ===========================================
                Create new worksheet inside existing excel file
               =========================================== */
            String safeSprintName = WorkbookUtil.createSafeSheetName("${sprint.name}")
            def ws = wb.createSheet(safeSprintName)

            // Add titles (typically key & summary, but assignee, ticket status,
ticket type can be configured in Config.groovy too)
            def titleRow = ws.createRow(0);
            int cellNumber = 0;
            columns.each {columnTitle -> titleRow.createCell(cellNumber++)
.setCellValue("${columnTitle.capitalize()}")}

            def lastRow = titleRow.getRowNum()
            titleRow.setRowStyle(headerCellStyle)
            // set summary (at position 1) column wider than other columns
            ws.setColumnWidth(1, 35*256)

            /* ========================================
                    AsciiDoc file for each sprint
               ======================================== */
            def asciidocFilename = "${sprint.name.replaceAll(" ", "_")}.adoc"
            logger.info("Results will be saved in '${asciidocFilename}' file")

            def changeLogAdoc = new File(targetFolder, "${asciidocFilename}")
            changeLogAdoc.write(".Table ${sprint.name} Changelog\n", 'utf-8')
            changeLogAdoc.append("|=== \n")

            // AsciiDoc table columns
            columns.each {columnTitle -> changeLogAdoc.append("|${columnTitle} ",
'utf-8')}

            /* ========================================
                    Add tickets for the sprint
               ======================================== */
            issues(api, headers, sprintBoardId, sprint.id,
ticketStatusForReleaseNotes).issues.each {issue ->
                def assignee = "${issue.fields.assignee ? issue.fields.assignee
.displayName : 'unassigned'} "
```

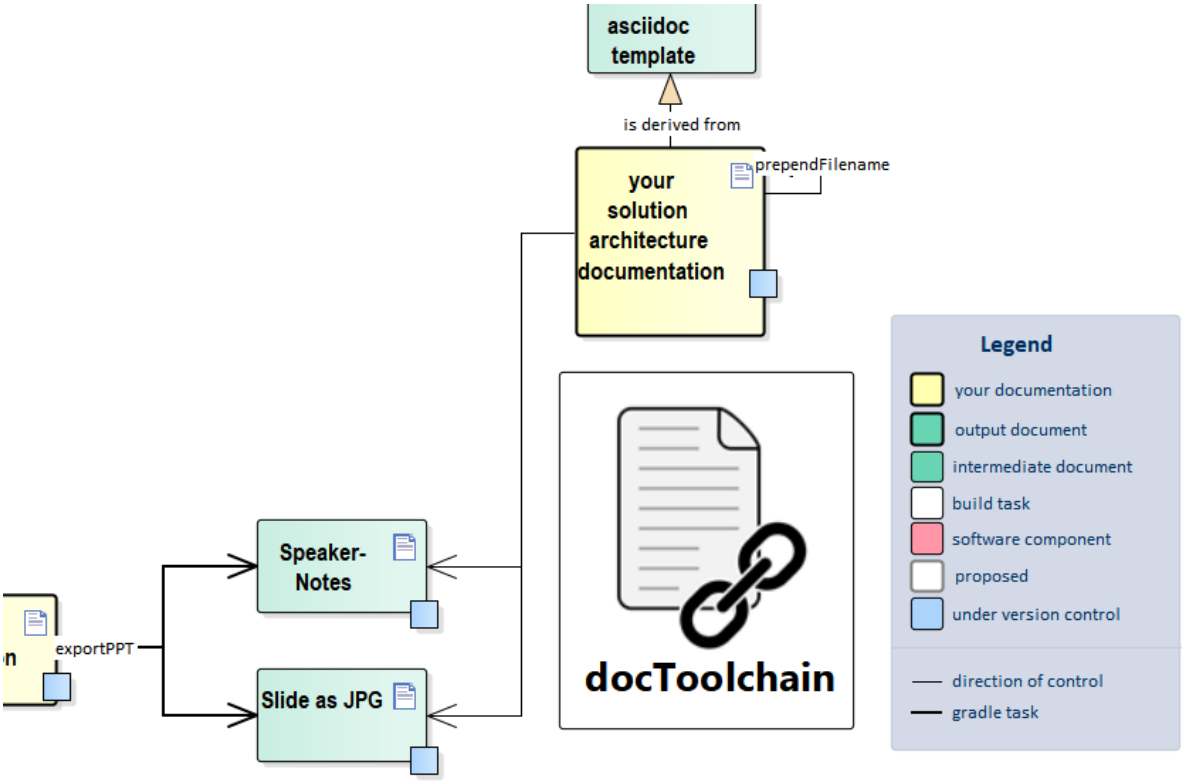
```
                def message = showAssignee ? "by ${assignee} ": ""
                logger.quiet("Issue: [$issue.key] '$issue.fields.summary' ${message}<
$issue.fields.status.name>")

                /* ============================
                      Write ticket to Excel
                   ============================ */
                int cellPosition = 0
                def row = ws.createRow(++lastRow)

                Hyperlink link = hyperlinkHelper.createHyperlink(HyperlinkType.URL)
                link.setAddress("${jiraRoot}/browse/${issue.key}")
                Cell cellWithUrl = row.createCell(cellPosition)
                cellWithUrl.setCellValue("${issue.key}")
                cellWithUrl.setHyperlink(link)

                row.createCell(++cellPosition).setCellValue("${issue.fields.summary}")

                /* ============================
                      Write ticket to Asciidoc
                   ============================ */
                changeLogAdoc.append("\n", 'utf-8')
                changeLogAdoc.append("| ${jiraRoot}/browse/${issue.key}[${issue.key}]
", 'utf-8')
                changeLogAdoc.append("| ${issue.fields.summary} ", 'utf-8')

                /* === Write ticket status, assignee, ticket typee if configured to
both Asciidoc & Excel files === */
                if (showTicketStatus) {
                    row.createCell(++cellPosition).setCellValue("${issue.fields.
status.name}")
                    changeLogAdoc.append("| ${issue.fields.status.name} ", 'utf-8')
                }

                if (showAssignee) {
                    row.createCell(++cellPosition).setCellValue("${assignee}")
                    changeLogAdoc.append("| ${assignee}", 'utf-8')
                }

                if (showTicketType) {
                    row.createCell(++cellPosition).setCellValue("${issue.fields
.issuetype.name}")
                    changeLogAdoc.append("| ${issue.fields.issuetype.name} ", 'utf-8')
                }
            }
            // Close the asciidoc table
            changeLogAdoc.append("\n|=== \n",'utf-8')

            // Set auto-width to KEY column
            ws.autoSizeColumn(0);
        }
```

```
        // Write to Excel file
        wb.write(changelogsXlsFos)
    }
}
```

3.20. exportPPT

Currently this feature is WINDOWS-only. See related issue

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportPPT.adoc[]



Blog-Post: Do more with Slides

see asciidoctorj-office-extension for another way how you can use PPT slides in your docs.

3.20.1. Source

exportPPT.gradle

```
task exportPPT(
        dependsOn: [streamingExecute],
        description: 'exports all slides and some texts from PPT files',
        group: 'docToolchain'
) {
    doLast {
        //make sure path for notes exists
        //and remove old notes
        new File(projectDir, 'src/docs/ppt').deleteDir()
        //also remove old diagrams
        new File(projectDir, 'src/docs/images/ppt').deleteDir()
        //create a readme to clarify things
        def readme = """This folder contains exported slides or notes from .ppt
presentations.

Please note that these are generated files but reside in the `src`-folder in order to
be versioned.

This is to make sure that they can be used from environments other than windows.

# Warning!

**The contents of this folder   will be overwritten with each re-export!**

use `gradle exportPPT` to re-export files
"""
        new File(projectDir, 'src/docs/images/ppt/.').mkdirs()
        new File(projectDir, 'src/docs/images/ppt/readme.ad').write(readme)
        new File(projectDir, 'src/docs/ppt/.').mkdirs()
        new File(projectDir, 'src/docs/ppt/readme.ad').write(readme)
        //execute through cscript in order to make sure that we get WScript.echo right
        "%SystemRoot%\\System32\\cscript.exe //nologo ${projectDir
}/scripts/exportPPT.vbs".executeCmd()
    }
}
```

exportPPT.vbs

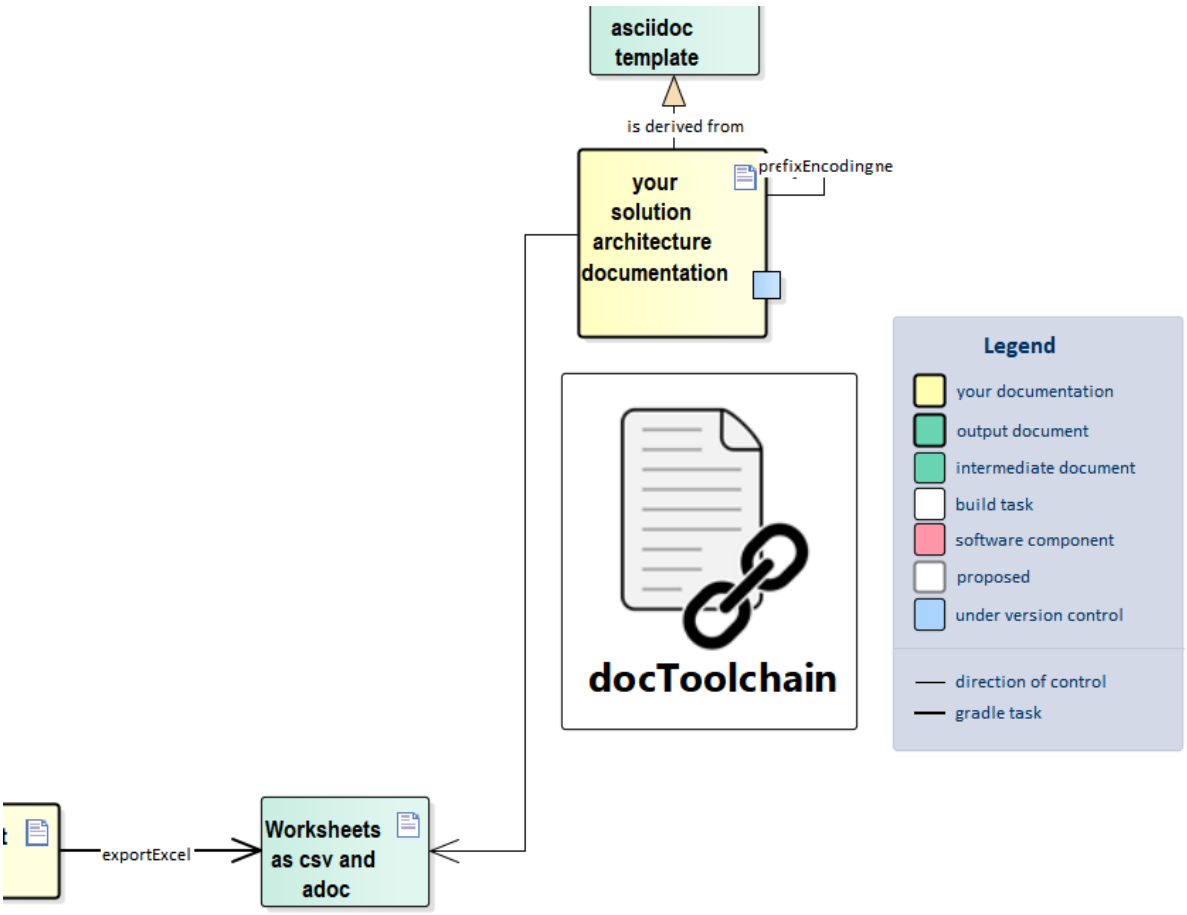
```
    Const   ForAppending = 8
    Const   ppPlaceholderBody = 2

    ' Helper
    ' http://windowsitpro.com/windows/jsi-tip-10441-how-can-vbscript-create-multiple-
folders-path-mkdir-command
    Function MakeDir (strPath)
      Dim strParentPath, objFSO
      Set objFSO = CreateObject("Scripting.FileSystemObject")
      On Error Resume Next
```

```
      strParentPath = objFSO.GetParentFolderName(strPath)

      If Not objFSO.FolderExists(strParentPath) Then MakeDir strParentPath
      If Not objFSO.FolderExists(strPath) Then objFSO.CreateFolder strPath
      On Error Goto 0
      MakeDir = objFSO.FolderExists(strPath)

    End Function

    Function SearchPresentations(path)

          For Each folder In path.SubFolders
            SearchPresentations folder
          Next

          For Each file In path.Files
                If (Left(fso.GetExtensionName (file.Path), 3) = "ppt") OR
(Left(fso.GetExtensionName (file.Path), 3) = "pps")  Then
                    WScript.echo "found "&file.path
                    ExportSlides(file.Path)
                End If
          Next

    End Function

    Sub ExportSlides(sFile)
        Set objRegEx = CreateObject("VBScript.RegExp")
        objRegEx.Global = True
        objRegEx.IgnoreCase = True
        objRegEx.MultiLine = True
        ' "." doesn't work for multiline in vbs, "[\s,\S]" does...
        objRegEx.Pattern = "[\s,\S]*{adoc}"
        ' http://www.pptfaq.com/FAQ00481_Export_the_notes_text_of_a_presentation.htm
        strFileName = fso.GetFIle(sFile).Name
        Set oPPT = CreateObject("PowerPoint.Application")
        Set oPres = oPPT.Presentations.Open(sFile, True, False, False) ' Read Only, No
Title, No Window
        Set oSlides = oPres.Slides
        strNotesText = ""
        strImagePath = "/src/docs/images/ppt/" & strFileName & "/"
        MakeDir("." & strImagePath)
        strNotesPath = "/src/docs/ppt/"
        MakeDir("." & strNotesPath)
        For Each oSl In oSlides
           strSlideName = oSl.Name
           ' WScript.echo fso.GetAbsolutePathName(".") & strImagePath & strSlideName &
".jpg"
           oSl.Export fso.GetAbsolutePathName(".") & strImagePath & strSlideName &
".jpg", ".jpg"
            For Each oSh In oSl.NotesPage.Shapes
                If oSh.PlaceholderFormat.Type = ppPlaceholderBody  Then
```

```
                    If oSh.HasTextFrame Then
                        If oSh.TextFrame.HasText Then
                            strCurrentNotes = oSh.TextFrame.TextRange.Text
                            strCurrentNotes = Replace(strCurrentNotes,vbVerticalTab,
vbCrLf)
                            strCurrentNotes =
Replace(strCurrentNotes,"{slide}","image::ppt/"&strFileName&"/"&strSlideName&".jpg[]")
                            ' remove speaker notes before marker "{adoc}"
                            strCurrentNotes = objRegEx.Replace(strCurrentNotes,"")
                            strNotesText = strNotesText  & vbCrLf & strCurrentNotes &
vbCrLf & vbCrLf
                        End If
                    End If
                End If
            Next
        Next
        ' WScript.echo fso.GetAbsolutePathName(".") &
strNotesPath&""&strFileName&".ad"
        ' http://stackoverflow.com/questions/2524703/save-text-file-utf-8-encoded-
with-vba

        Set fsT = CreateObject("ADODB.Stream")
        fsT.Type = 2 'Specify stream type - we want To save text/string data.
        fsT.Charset = "utf-8" 'Specify charset For the source text data.
        fsT.Open 'Open the stream And write binary data To the object
        fsT.WriteText "ifndef::imagesdir[:imagesdir:
../../images]"&vbCrLf&CStr(strNotesText)
        fsT.SaveToFile fso.GetAbsolutePathName(".") &
strNotesPath&""&strFileName&".ad", 2 'Save binary data To disk
        oPres.Close()
        oPPT.Quit()
    End Sub

  set fso = CreateObject("Scripting.fileSystemObject")
  WScript.echo "Slide extractor"
  WScript.echo "looking for .ppt files in " & fso.GetAbsolutePathName(".") & "/src"
  SearchPresentations fso.GetFolder("./src")
  WScript.echo "finished exporting slides"
```

3.21. exportExcel

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportExcel.adoc[]



Sometimes you have tabular data to be included in your documentation. Then it is likely that the data is available as Excel sheet or you would like to use MS Excel to create and edit it.

Either way, this task lets you export your Excel sheet and include it directly in your docs.

The task searches for `.xlsx` files and exports each contained worksheet as `.csv` and as `.adoc`.

Formulas contained in your worksheet are evaluated and exported statically.

The generated files are written to `src/excel/[filename]/[worksheet].(adoc|cvs)` . The `src` folder is chosen over the `build` folder to get a better history for all changes on the worksheets.

The files can be included either as AsciiDoc

```
include::excel/Sample.xlsx/Numerisch.adoc[]
```

or as a CSV file

```
[options="header",format="csv"]
|===
include::excel/Sample.xlsx/Numerisch.csv[]
|===
```

The AsciiDoc version gives you a bit more control:

- horizontal and vertical alignment is preserved
- col- and row-spans are preserved
- line breaks are preserved
- column width relative to other columns is preserved
- background colors are preserved.

see asciidoctorj-office-extension for another way how you can use Excel sheets in your docs.

3.21.1. Source

build.gradle

```
task exportExcel(
        description: 'exports all excelsheets to csv and AsciiDoc',
        group: 'docToolchain'
) {
    doFirst {
        File sourceDir = file(srcDir)

        def tree = fileTree(srcDir).include('**/*.xlsx').exclude('**/~*')

        def exportFileDir = new File(sourceDir, 'excel')

        //make sure path for notes exists
        exportFileDir.deleteDir()
        //create a readme to clarify things
        def readme = """This folder contains exported workbooks from Excel.

Please note that these are generated files but reside in the `src`-folder in order to
be versioned.

This is to make sure that they can be used from environments other than windows.

# Warning!

**The contents of this folder will be overwritten with each re-export!**

use `gradle exportExcel` to re-export files
"""
```

```
        exportFileDir.mkdirs()
        new File(exportFileDir, '/readme.ad').write(readme)
    }
    doLast {
        File sourceDir = file(srcDir)
        def exportFileDir = new File(sourceDir, 'excel')
        def tree = fileTree(srcDir).include('**/*.xlsx').exclude('**/~*')

        def nl = System.getProperty("line.separator")

        def export = { sheet, evaluator, targetFileName ->
            def targetFileCSV = new File(targetFileName + '.csv')
            def targetFileAD = new File(targetFileName + '.adoc')
            def df = new org.apache.poi.ss.usermodel.DataFormatter();
            def regions = []
            sheet.numMergedRegions.times {
                regions << sheet.getMergedRegion(it)
            }
            logger.debug "sheet contains ${regions.size()} regions"
            def color = ''
            def resetColor = false
            def numCols = 0
            def headerCreated = false
            def emptyRows = 0
            for (int rowNum=0; rowNum<=sheet.lastRowNum; rowNum++) {
                def row = sheet.getRow(rowNum)
                if (row && !headerCreated) {
                    headerCreated = true
                    // create AsciiDoc table header
                    def width = []
                    numCols = row.lastCellNum
                    numCols.times { columnIndex ->
                        width << sheet.getColumnWidth((int) columnIndex)
                    }
                    //lets make those numbers nicer:
                    width = width.collect { Math.round(100 * it / width.sum()) }
                    targetFileAD.append('[options="header",cols="' + width.join(',') +
'"]' + nl)
                    targetFileAD.append('|===' + nl)
                }
                def data = []
                def style = []
                def colors = []
                // For each row, iterate through each columns
                if (row && (row?.lastCellNum!=-1)) {
                    numCols.times { columnIndex ->
                        def cell = row.getCell(columnIndex)
                        if (cell) {
                            def cellValue = df.formatCellValue(cell, evaluator)
                            if (cellValue.startsWith('*') && cellValue.endsWith(
'\u20AC')) {
```

```
                                // Remove special characters at currency
                                cellValue = cellValue.substring(1).trim();
                            }
                            def cellStyle = ''
                            def region = regions.find { it.isInRange(cell.rowIndex,
cell.columnIndex) }
                            def skipCell = false
                            if (region) {
                                //check if we are in the upper left corner of the
region
                                if (region.firstRow == cell.rowIndex && region
.firstColumn == cell.columnIndex) {
                                    def colspan = 1 + region.lastRow - region.firstRow
                                    def rowspan = 1 + region.lastColumn - region
.firstColumn
                                    if (rowspan > 1) {
                                        cellStyle += "${rowspan}"
                                    }
                                    if (colspan > 1) {
                                        cellStyle += ".${colspan}"
                                    }
                                    cellStyle += "+"
                                } else {
                                    skipCell = true
                                }
                            }
                            if (!skipCell) {
                                switch (cell.cellStyle.alignmentEnum.toString()) {
                                    case 'RIGHT':
                                        cellStyle += '>'
                                        break
                                    case 'CENTER':
                                        cellStyle += '^'
                                        break
                                }
                                switch (cell.cellStyle.verticalAlignmentEnum.toString
()) {
                                    case 'BOTTOM':
                                        cellStyle += '.>'
                                        break
                                    case 'CENTER':
                                        cellStyle += '.^'
                                        break
                                }
                                color = cell.cellStyle.fillForegroundXSSFColor?.RGB?
.encodeHex()
                                color = color != null ? nl + "{set:cellbgcolor:#
${color}}" : ''
                                data << cellValue
                                if (color == '' && resetColor) {
                                    colors << nl + "{set:cellbgcolor!}"
```

```
                                resetColor = false
                            } else {
                                colors << color
                            }
                            if (color != '') {
                                resetColor = true
                            }
                            style << cellStyle
                        } else {
                            data << ""
                            colors << ""
                            style << "skip"
                        }
                    } else {
                        data << ""
                        colors << ""
                        style << ""
                    }

                }
                emptyRows = 0
            } else {
                if (emptyRows<3) {
                    //insert empty row
                    numCols.times {
                        data << ""
                        colors << ""
                        style << ""
                    }
                    emptyRows++
                } else {
                    break
                }
            }

            targetFileCSV.append(data
                    .collect {
                "\"${it.replaceAll('"', '""')}\""
            }
            .join(',') + nl, 'UTF-8')
            targetFileAD.append(data
                    .withIndex()
                    .collect { value, index ->
                if (style[index] == "skip") {
                    ""
                } else {
                    style[index] + "| ${value.replaceAll('[|]', '{vbar}')
.replaceAll("\n", ' +$0') + colors[index]}"
                }
            }
            .join(nl) + nl * 2, 'UTF-8')
```

```
            }
            targetFileAD.append('|===' + nl)
        }


        tree.each { File excel ->
            println excel
            def excelDir = new File(exportFileDir, excel.getName())
            excelDir.mkdirs()
            InputStream inp
            inp = new FileInputStream(excel)
            def wb = org.apache.poi.ss.usermodel.WorkbookFactory.create(inp);
            def evaluator = wb.getCreationHelper().createFormulaEvaluator();
            for (int wbi = 0; wbi < wb.getNumberOfSheets(); wbi++) {
                def sheetName = wb.getSheetAt(wbi).getSheetName()
                println sheetName
                def targetFile = new File(excelDir, sheetName)
                export(wb.getSheetAt(wbi), evaluator, targetFile.getAbsolutePath())
            }
            inp.close();
        }
    }
}
```

Unresolved directive in manual/03_task_exportMarkdown.adoc - include::../../../build/docs/exportMarkdownDocs.adoc[]

3.21.2. Source

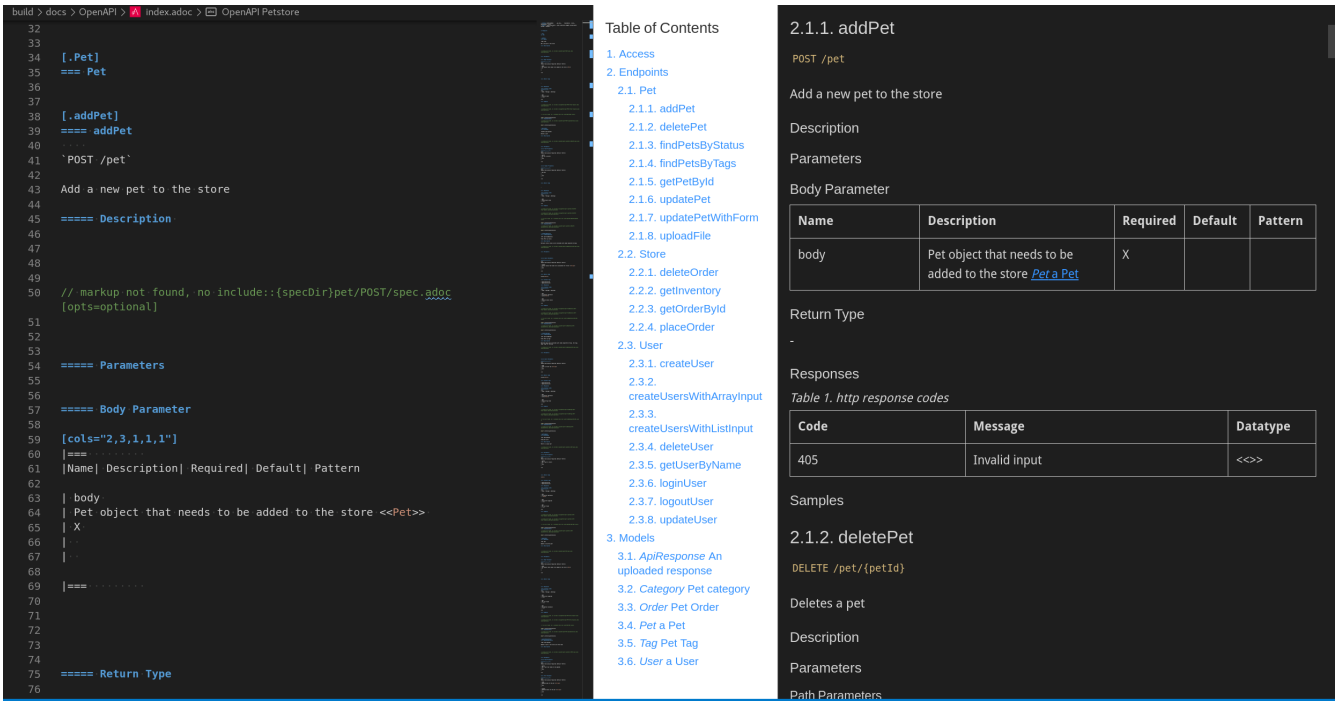
exportMarkdown.gradle

```
task exportMarkdown(
        description: 'exports all markdown files to AsciiDoc',
        group: 'docToolchain',
        type: Copy
) {
    from srcDir

    include("**/*.md") //include only markdown files
    includeEmptyDirs = false
    rename(/(.+).md/, '$1.adoc') //rename all files from *.md to *.adoc
    filter(Markdown2AdocFilter) // convert the content of the files

    into targetDir
}

class Markdown2AdocFilter extends FilterReader {
    Markdown2AdocFilter(Reader input) {
        super(new StringReader(nl.jworks.markdown_to_asciidoc.Converter
.convertMarkdownToAsciiDoc(input.text)))
    }
}
```

3.22. exportOpenAPI

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_exportOpenApi.adoc[]

This task exports OpenAPI specification defined in a yaml file to AsciiDoc document. Currently it relies on OpenAPI Generator (v4.3.1) and its gradle plugin.



3.22.1. Configuration

Config.groovy

```
// Configuration for OpenAPI related task
openApi = [:]

// 'specFile' is the name of OpenAPI specification yaml file. Tool expects this file
inside working dir (as a filename or relative path with filename)
// 'infoUrl' and 'infoEmail' are specification metadata about further info related to
the API. By default this values would be filled by openapi-generator plugin
placeholders
//

openApi.with {
    specFile = 'src/docs/petstore-v2.0.yaml' // i.e. 'petstore.yaml',
'src/doc/petstore.yaml'
    infoUrl = 'https://my-api.company.com'
    infoEmail = 'info@company.com'
}
```

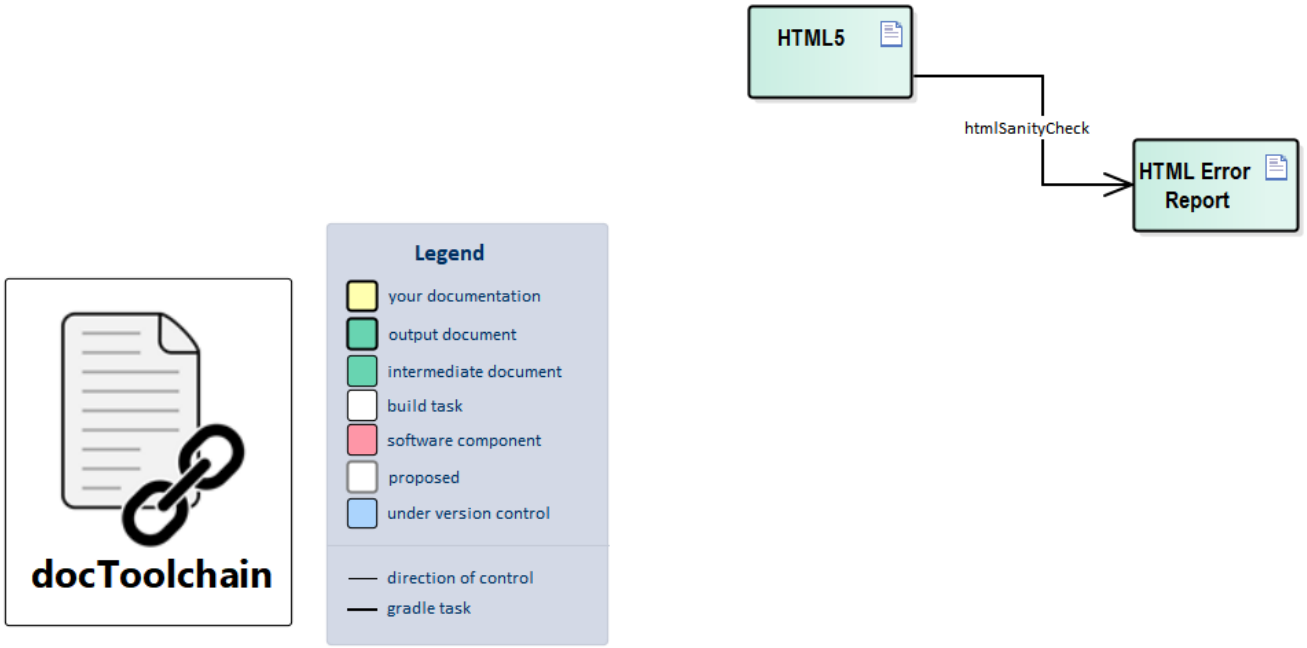
3.22.2. Source

exportOpenApi.gradle

```
task exportOpenApi (
        type: org.openapitools.generator.gradle.plugin.tasks.GenerateTask,
        group: 'docToolchain',
        description: 'exports OpenAPI specification to the asciidoc file') {

    if (!specFile) {
        logger.info("\n---> OpenAPI specification file not found in Config.groovy
(https://doctoolchain.github.io/docToolchain/#_exportopenapi)")
        return
    } else {
        logger.info("Found OpenAPI specification in Config.groovy")
    }

    outputs.upToDateWhen { false }
    outputs.cacheIf { false }
    generatorName = 'asciidoc'
    outputDir = "${targetDir}/OpenAPI".toString()
    inputSpec = "${docDir}/${specFile}" // plugin is not able to find file if
inputPath is defined as '.'

    logger.debug("\n====================\nProject Config:\n====================")
    logger.debug("Docdir: ${docDir}")
    logger.debug("Target: ${targetDir}")
    logger.info("\n====================\nOpenAPI Config:\n====================")
    logger.info("Specification file: ${specFile}")
    logger.info("inputSpec: ${inputSpec}")
    logger.info("outputDir: ${outputDir}\n")

    additionalProperties = [
        infoEmail:"${config.openApi.infoEmail}",
        infoUrl:"${config.openApi.infoUrl}"
    ]
}
```

3.23. htmlSanityCheck

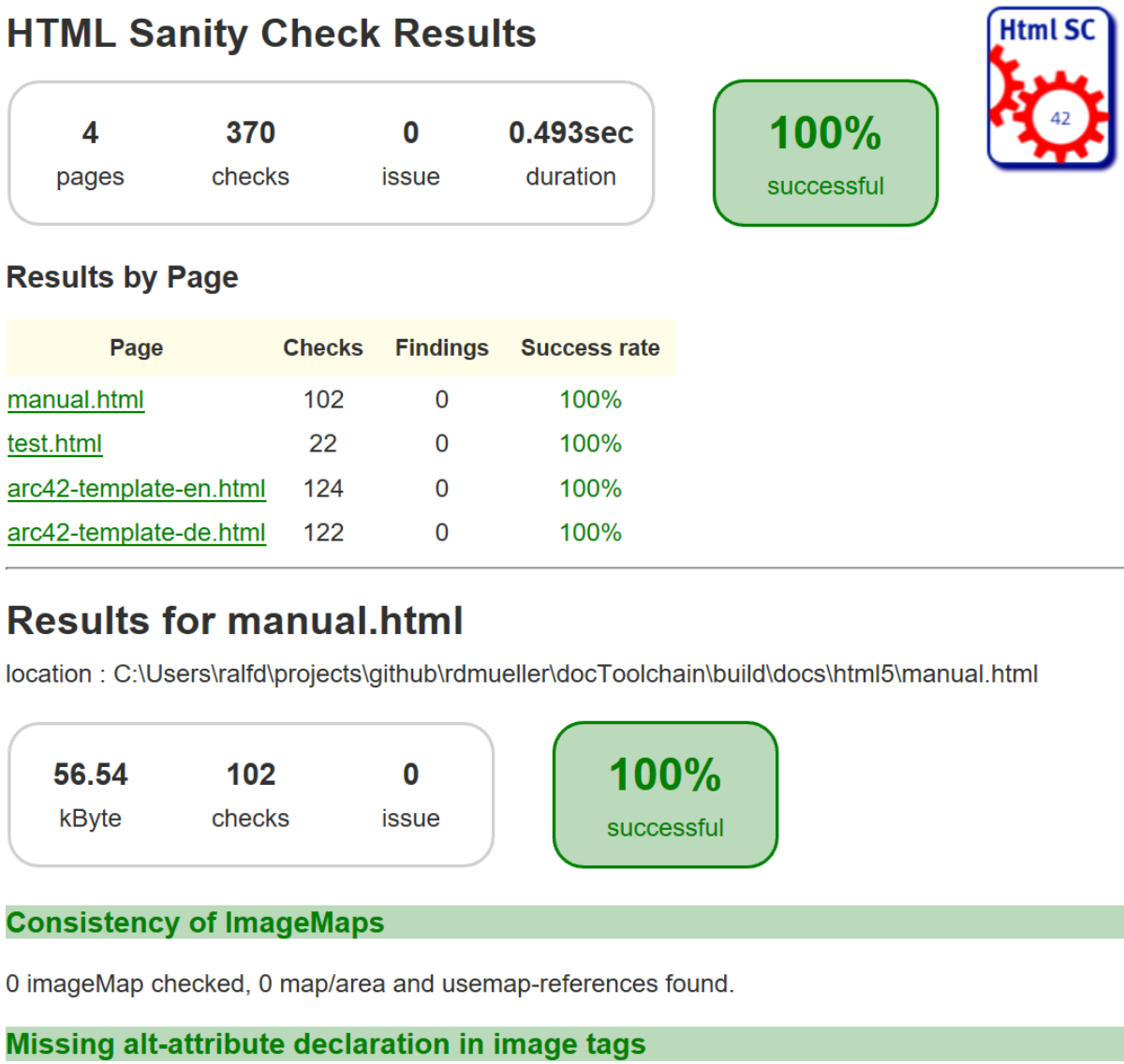
Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_htmlSanityCheck.adoc[]



This task invokes the htmlSanityCheck gradle plugin. It is a standalone (batch- and command-line) html sanity checker - it detects missing images, dead links, and duplicate bookmarks.

In docToolchain, this task is used to ensure that the generated HTML contains no missing links or other problems.

This task is the last default task and creates a report in `build/report/htmlchecks/index.html`



HTML Sanity Check Results

4 pages | 370 checks | 0 issue | 0.493sec duration

100% successful

Results by Page

Page	Checks	Findings	Success rate
manual.html	102	0	100%
test.html	22	0	100%
arc42-template-en.html	124	0	100%
arc42-template-de.html	122	0	100%

Results for manual.html

location : C:\Users\ralfd\projects\github\rdmueller\docToolchain\build\docs\html5\manual.html

56.54 kByte | 102 checks | 0 issue

100% successful

Consistency of ImageMaps

0 imageMap checked, 0 map/area and usemap-references found.

Missing alt-attribute declaration in image tags

3 image tags checked, 0 missing alt attributes found.

Broken Internal Links Check

56 href checked, 0 missing id found.

Missing Local Images Check

3 img src attributes checked, 0 missing image files found.

Missing Local Resources Check

0 anchor tag href attribute checked, 0 missing local resources found.

Duplicate Definition of id Check

40 id checked, 0 duplicate id found.

Figure 5. sample report

Further information can be found on GitHub: https://github.com/aim42/htmlSanityCheck

Blog-Post: Automated Quality-Checks

3.23.1. Source

htmlSanityCheck.gradle

```
htmlSanityCheck {
    sourceDir = new File(config.htmlSanityCheck.sourceDir?:"$targetDir/html5")
    // files to check - in Set-notation
    //sourceDocuments = [ "one-file.html", "another-file.html", "index.html"]

    // where to put results of sanityChecks...
    checkingResultsDir = new File(config.htmlSanityCheck.checkingResultsDir
?:checkingResultsPath)
    // directory where the results written to in JUnit XML format
    junitResultsDir = new File(config.htmlSanityCheck.junitResultsDir?:"$
targetDir/test-results/htmlchecks")

    // which statuscodes shall be interpreted as warning, error or success defaults to
standard
    httpSuccessCodes = config.htmlSanityCheck.httpSuccessCodes?:[]
    httpWarningCodes = config.htmlSanityCheck.httpWarningCodes?:[]
    httpErrorCodes = config.htmlSanityCheck.httpErrorCodes?:[]

    // fail build on errors?
    failOnErrors = config.htmlSanityCheck.failOnErrors?:false

    logger.info "docToolchain> HSC sourceDir: ${sourceDir}"
    logger.info "docToolchain> HSC checkingResultsDir: ${checkingResultsDir}"
}
```

3.24. dependencyUpdates

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/03_task_dependencyUpdates.adoc[]

This task uses the Gradle versions plugin created by Ben Manes to check for outdated build dependencies. It is quite helpful to keep all dependencies up-to-date.

if you discover newer version, this does not mean that they will play nicely together. If you want to make sure that everything works, rely on the versions which the docToolchain contributors have selected.

Blog-Post: Handle Dependency Updates the easy Way

4. Development

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/08_development.adoc[]

INFO: this chapter is still work in progress

4.1. How to run Tests

4.1.1. Prerequisites

- make sure that Git and Graphviz are installed
- make sure that your Gradle setup is able to work with proxies
- use Java 8

4.1.2. Prepare the Project

```
git clone git@github.com:docToolchain/docToolchain.git
cd docToolchain/
git checkout V1.0.0         ①
git submodule update -i
```

① the version to test. Not needed if you work on the HEAD revision on Master

4.1.3. Execute Tests

```
rm -r build && ./gradlew test --info
```

The `rm` command ensures that you really have a clean test running. Otherwise you might get false positives because Gradle will skip steps ('Up-to-date') because artifacts of an older test run still exist.

4.1.4. Proxy Setting for Tests

The docToolchain test setup is based on the Gradle-Test-Kit and makes use of the Spock test execution framework.

The gradle test runner is started in its own test environment and its own JVM instance. As a result the global proxy settings are ignored.

As workarund to execute the test with the correct proxy settings it is necessary to copy the proxy setting normally done in the in gradle.properties located in the user directory to the gradle.properties file located in the docToolchain folder itself.

The files downloaded by the Gradle test Runner are placed in a different folder than the default gradle cache. You'll find them in the Tmp folder: C:\Users\YOUR_USER_NAME\AppData\Local\Temp\.gradle-test-kit-YOUR_USER_NAME\caches

4.2. Create new Release

We use semantic versioning and keep a changelog. Bot on a best effort base.

A release consists of four parts:

4.2.1. Github

- update the changelog
  - create a section for the version
  - copy all unreleased features which will be in the release to the created section
  - commit and push the new version
- Draft a new release
- copy the content of the changelog for this version to the description and submit
- set version as **v**X.Y.Z
- run `./gradlew createDist`
- this creates a zip of the source in `build` which is the distribution file
- add this zip and submit the new release

4.2.2. Dockerhub

The image build for rdmueller/doctoolchain depends on the github repository docToolchain/docker-image

- update the Dockerfile to reflect the new version
- create a new release
- reference the github release for changelog
- the build on dockerhub will be automatically triggered

4.2.3. sdkman

A github action has been created to deploy to sdkman: sdkman deploy.

- set version to the same as for the other relases but without the pre-pended *v*: X.Y.Z
- use as download link the link to the `docToolchain-dist.zip` from the github release (hint: looks like https://github.com/docToolchain/docToolchain/releases/download/v1.3.1/docToolchain-dist.zip)

5. FAQ: Frequently asked Questions

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/06_Frequently_asked_Questions.adoc[]

This section tries to answer the most common and frequently asked questions about how to work with docToolchain. It will also contain questions relevant to the tools used to build docToolchain, but the main focus is docToolchain itself.

If you are stuck, make sure that you also check other sources like Stack Overflow.

There is also a great FAQ for all your arc42 questions: https://faq.arc42.org/home/

If you have a question or problem for which you can't find a solution, you can

- for this repo, add your question and create a pull request
- raise the issue through the GitHub issue tracker
- ask your question on Stack Overflow
- discuss the problem on Slack

5.1. Images

- Asciidoctor User Manual on images
- Asciidoctor Quick Reference on images
- AsciiDoctor Writer Guide on images

5.1.1. Q: Why are images not shown in the preview of my editor?

A: this is most likely because your editor doesn't know where they are stored. If you follow the default settings, you probably store your images in a subfolder `images`. The build script knows about it, because the attribute `imagesdir` has been set to `./images`, but your editor doesn't care about the build script - it only checks the currently opened AsciiDoc file.

The solution is to add a line to each file which checks if the `imagesdir` is set and if not, sets it to a valid value:

```
ifndef::imagesdir[:imagesdir: ../images]
```

5.1.2. Q: Which image format should I use?

A: AsciiDoc and AsciiDoctor support several formats like GIF, PNG, JPG and SVG. However, if you want to use most features, some formats are better to use than others:

GIF

is not supported by the PDF renderer. Use JPG or PNG instead.

JPG

is great for photos but not for diagrams (you might get compression artifacts). So, if you want to use photos from your flipcharts - JPG might work for you.

SVG

great for high resolution diagrams, but not good supported by DOCX as output format. OpenOffice Writer might display the image a bit stretched, MS Word didn't display it at all in some experiments. PDF output might display a warning that newer SVG versions are not supported (happends especially with diagrams.net images).

PNG

this is the preferred format for images used with docToolchain. All output formats support it and if diagrams are rendered with a resolution high enough to display all details, it will also be scaled well with all output formats.

5.1.3. Q: Why are my images rotated in the output?

A: This most likely happens when you've taken photos with a mobile device and include them in you docs. A mobile device does not rotate the image itself, it only stores the orientation of the device in the meta data of the photo. Your operating system will show you the image as expected, but the rendered AsciiDoc will not. This can be „fixed" with Imagemagick, by using `convert -auto -orient` or `mogrify -auto-orient` (thanx to @rotnroll666 for this tip). You can also try to just open the image in your favourite editor and re-save it.

5.2. exportVisio

5.2.1. Q: I get an error message saying that a library is not registered when I try to run the `exportVisio`-task.

```
Ausnahme beim Festlegen von "Visible": "Das COM-Objekt des Typs
"Microsoft.Office.Interop.Visio.ApplicationClass" kann nicht in den Schnittstellentyp
"Microsoft.Office.Interop.Visio.IVApplication" umgewandelt werden. Dieser Vorgang
konnte nicht durchgeführt werden, da der QueryInterface-Aufruf an die
COM-Komponente für die Schnittstelle mit der IID "{000D0700-0000-0000-C000-
000000000046}" aufgrund des folgenden Fehlers nicht durchgeführt werden konnte:
Bibliothek nicht registriert. (Ausnahme von HRESULT: 0x8002801D
(TYPE_E_LIBNOTREGISTERED))."
In ...\scripts\VisioPageToPngConverter.ps1:48 Zeichen:1
+ $VisioApp.Visible = $false
+ ~~~~~~~~~~~~~~~~~~~~~~~~~~
    + CategoryInfo          : NotSpecified: (:) [], SetValueInvocationException
    + FullyQualifiedErrorId : ExceptionWhenSetting
```

A: When Visio is installed, it registers itself as a com library. It seems that this registration can break. You can fix this by visiting the windows system settings → install or uninstall a program, select `visio`, select `change` and then `repair`.

5.3. Sparx Enterprise Architect

5.3.1. Q: Sparx Enterprise Architect is a Windows tool, isn't it?

Yes, it is, but it is written to support CrossOver in order to run on Linux based systems. Wine, the open source branch of CrossOver, seems to work as well.

Take a look at this page to see how to install it on a linux based system: https://www.sparxsystems.com/support/faq/ea_on_linux.html

I (Ralf) once gave it a try and even managed to get remote control over EA via VBS and COM up and running (which is the pre-requisite for docToolchain).

5.4. Known error Messages

5.4.1. Q: I get the error saying 'Failed to create MD5 hash for file content'.

```
* What went wrong:
Failed to capture snapshot of input files for task ':generateHTML' property
'sourceDir' during up-to-date check.
> Failed to create MD5 hash for file content.`
```

There are two known reasons for this error.

1. One of the .adoc files is opened in an editor, so that windows can't get the lock for that file. → Close all .adoc files.
2. You use the Bash script `doctoolchain` on a windows system. → Use `doctoolchain.bat` instead. It works even in a Bash Shell.

5.4.2. Q: I get the error saying 'Unsupported major.minor version 52.0'

This is a sign that you use an outdated version of Java. Please upgrade to Java 8 at least.

5.4.3. Q: I get an error message saying 'Error occurred during initialization of VM'

```
Starting a Gradle Daemon, 1 incompatible Daemon could not be reused, use --status for
details

FAILURE: Build failed with an exception.

* What went wrong:
Unable to start the daemon process.
…
Error occurred during initialization of VM
Could not reserve enough space for 2097152KB object heap
```

Somehow docToolchain can't allocate the memory which is configured out of the box. Try to free up some memory or comment out the line `org.gradle.jvmargs=-Xmx2048m` in `gradle.properties`

5.4.4. Q: I get the error saying 'Could not initialize class java.awt.GraphicsEnvironment$LocaleGE'

This seems to be a problem with WSL on Windows. Some sources mention to run Java in headless mode, but in this case, it doesn't solve the problem. The root cause seems to be plantUML trying to get some font information.

Only real solution seems to be to shutdown WSL from a powershell window wiht `wsl --shutdown` and retry.

this will kill all your WSL terminals without warning.

6. Further Reading

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/04_further_reading.adoc[]

This chapter lists some additional references to interesting resources.

6.1. Links

- **AsciiDoc**
  - Asciidoctor User-Manual
  - AsciiDoc Syntax Quick Reference
- **arc42**
  - arc42
  - arc42 Tips & Tricks
  - arc42 FAQ
- the docToolchain Blog

6.2. Books

links to Amazon are affiliate links

6.2.1. English Books

- DOCS-LIKE-CODE by Anne Gentle
- Modern technical writing: An Introduction to Software Documentation by Andrew Etter
- arc42 by Example by Gernot Starke, Stefan Zörner, Michael Simons
- Communicating Software Architectures with arc42 by Gernot Starke und Peter Hruschka
- Visualise, document and explore your software architecture by Simon Brown

6.2.2. German Books

- arc42 in Aktion: Praktische Tipps zur Architekturdokumentation von Gernot Starke und Peter Hruschka
- Softwarearchitekturen dokumentieren und kommunizieren: Entwürfe, Entscheidungen und Lösungen nachvollziehbar und wirkungsvoll festhalten von Stefan Zörner

6.3. Past and upcoming Talks

6.3.1. Dokumentation am (Riesen-)Beispiel – arc42, AsciiDoc und Co. in Aktion

Gernot Starke and Ralf D. Müller

Anhand eines großen Systems zeigen Gernot und Ralf, wie Sie mit ziemlich wenig Aufwand angemessene und vernünftige Dokumentation für unterschiedliche Stakeholder produzieren – sodass Entwicklungsteams dabei auch noch Spaß haben.

Unser Rezept: AsciiDoc mit arc42 mischen, Automatisierung mit Gradle und Maven hinzufügen und mit Diagramm- oder Modellierungstools Ihrer Wahl kombinieren. Schon sind schicke HTML- und reviewfähige PDF-Dokumente fertig. Auf Wunsch gibts DOCX und Confluence als Zugabe.

Wir zeigen, wie Sie Doku genau wie Quellcode verwalten können, stakeholderspezifische Dokumente erzeugen und Diagramme automatisiert integrieren können. Einige Teile dieser Doku können Sie sogar automatisiert testen.

Zwischendurch bekommen Sie zahlreiche Tipps, wie und wo Sie systematisch den Aufwand für Dokumentation reduzieren können und trotzdem lesbare, verständliche und praxistaugliche Ergebnisse produzieren.

6.3.2. Gesunde Dokumentation mit AsciiDoctor

Alexander Schwartz

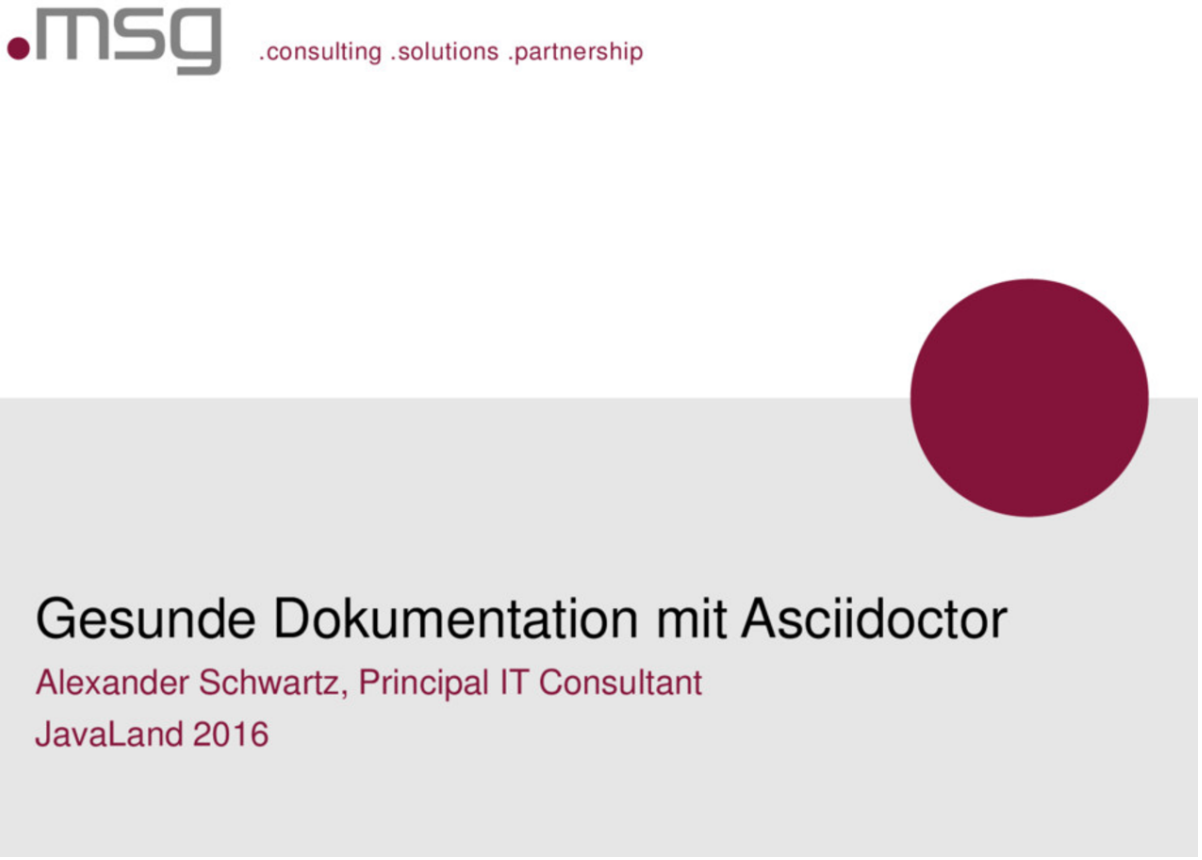
Autoren möchten Inhalte effizient dokumentieren und vorhandene Inhalte wiederverwenden. Ein Leser möchte das Dokument in einem ansprechenden Layout präsentiert bekommen.

Das textbasierte AsciiDoc-Format bietet für einen Entwickler oder technischen Redakteur alle notwendigen Auszeichnungselemente, um auch anspruchsvolle Dokumente zu schreiben. So werden unter anderem Tabellen, Fußnoten und annotierte Quellcodes unterstützt. Gleichzeitig ist es

ähnlich leichtgewichtig wie z.B. das Markdown Format. Für die Leser wird HTML, PDF oder EPUB generiert.

Da AsciiDoc wie Programmcode eingecheckt wird und Merge-Operationen einfach möglich sind, können Programmcode und Dokumentation zusammen versioniert und auf einem einheitlichen Stand gehalten werden.

Der Vortrag gibt eine kurze Einführung in AsciiDoc und die dazugehörigen Werkzeuge.



7. Acknowledgements and Contributors

Unresolved directive in manual/feedback.adoc - include::C:\doctoolchain\./build/test3/contributors/manual/05_contributors.adoc[]

This project is an open source project which is based on community efforts.

Many people are involved in the underlying technologies like AsciiDoc, AsciiDoctor, Gradle, arc42 etc. This project depends and builds on them.

But it even more depends on the direct contributions made through giving feedback, creating issues, answering questions, or sending pull requests.

Here is an incomplete and unordered list of contributors:

- Stefan Bodewig
- MoePad
- Niels
- wschaef
- Gernot Starke
- Alexander Schwartz
- Alexander Heusingfeld
- Dan Allen
- Stefan Pfeiffer
- isidorotrevino
- Jakub Jablonski
- Frank Pohl
- Ixchel Ruiz
- Schalk Cronjé
- Mario García
- Joe
- David M. Carr
- Fabian Nonnenmacher
- Christoph Stoettner
- Roman Funk
- ghp-dev
- Christoph Raaflaub
- Jorge Aguilera
- Stefan Bohn

- Jochen Kraushaar
- Luis Muniz
- Andreas Offenhaeuser
- Daniel Bast
- Sabatmonk
- Maarten Gribnau
- Michael Prieß
- Heiko Stehli
- Peter Stange
- Nils Mahlstädt @ hmmh
- Kevin Werner
- J. Staub
- Vladi Bjelakovic
- Daniel Kessel
- Björn Seebeck
- Txemanu
- Nikolay Orozov
- Andrea Macaluso
- Michael Roßner
- Jan Hendriks

(please update your entry to match your preferences! :-)

Appendix A: Configuration

This appendix covers all configuration introduced by docToolchain. AsciiDoc, AsciiDoctor, Gradle and other tools and libraries used know of more configuration settings and you can read about those in the corresponding documentation.

A.1. `mainConfigFile` and `docDir`

docToolchain should be easy to use. That's why the goal is to have one config file with all settings for each project. But first of all, docToolchain has to know where your documentation project is located.

If `docDir` is defined, the default for `mainConfigFile` is `Config.groovy` in the root folder of your `docDir`.

You have several options to specify the location of your documentation project (`docDir`) and the location of your config file (`mainConfigFile`).

A.1.1. build.gradle

If you use docToolchain as Gradle sub-project as described in our tutorial, you will have some code in your main `build.gradle` which overwrites the location of the config file:

```
//configure docToolchain to use the main project's config
project('docToolchain') {
    if (project.hasProperty('docDir')) {
        docDir = '../.'
        mainConfigFile = 'Config.groovy'
    } else {
        println "="*80
        println "  please initialize the docToolchain submodule"
        println "  by executing git submodule update -i"
        println "="*80
    }
}
```

This way, you can place the config file wherever you like in your project.

Since the `mainConfigFile` is a standard Gradle-Property, you can also overwrite it through a command line option.

it is not possible to overwrite properties of a sub project throught the `gradle.properties` of the main project.

A.1.2. Commandline

Specify the property on the commandline

```
./gradlew generateHTML -PmainConfigFile=Config.groovy
```

and if you use the doctoolchain script

```
doctoolchain <docDir> generateHTML -PmainConfigFile=Config.groovy
```

you can verify the location of your `Config.groovy` by executing docToolchain with the `--info` parameter which sets the loglevel to `info`. It will print the location on the command line (among other settings)

A.1.3. Content of the `mainConfigFile`

```
outputPath = 'build/test3'

// Path where the docToolchain will search for the input files.
// This path is appended to the docDir property specified in gradle.properties
// or in the command line, and therefore must be relative to it.

inputPath = 'src/test3';


inputFiles = [
        [file: 'manual.adoc',       formats: ['html','pdf']],
        [file: 'test2.adoc',       formats: ['html','pdf']],

]


taskInputsDirs = ["${inputPath}/images"]

taskInputsFiles = []

//*************************************************************************************
*****

//Configuration for exportChangelog

exportChangelog = [:]

changelog.with {

    // Directory of which the exportChangelog task will export the changelog.
    // It should be relative to the docDir directory provided in the
    // gradle.properties file.
    dir = 'src/docs'

    // Command used to fetch the list of changes.
```

```
    // It should be a single command taking a directory as a parameter.
    // You cannot use multiple commands with pipe between.
    // This command will be executed in the directory specified by changelogDir
    // it the environment inherited from the parent process.
    // This command should produce asciidoc text directly. The exportChangelog
    // task does not do any post-processing
    // of the output of that command.
    //
    // See also https://git-scm.com/docs/pretty-formats
    cmd = 'git log --pretty=format:%x7c%x20%ad%x20%n%x7c%x20%an%x20%n%x7c%x20%s%x20%n
--date=short'

}

//*************************************************************************************
*****

//Configureation for publishToConfluence

confluence = [:]

// 'input' is an array of files to upload to Confluence with the ability
//          to configure a different parent page for each file.
//
// Attributes
// - 'file': absolute or relative path to the asciidoc generated html file to be
exported
// - 'url': absolute URL to an asciidoc generated html file to be exported
// - 'ancestorName' (optional): the name of the parent page in Confluence as string;
//                              this attribute has priority over ancestorId, but if
page with given name doesn't exist,
//                              ancestorId will be used as a fallback
// - 'ancestorId' (optional): the id of the parent page in Confluence as string; leave
this empty
//                            if a new parent shall be created in the space
// - 'preambleTitle' (optional): the title of the page containing the preamble
(everything
//                            before the first second level heading). Default is
'arc42'
//
// The following four keys can also be used in the global section below
// - 'spaceKey' (optional): page specific variable for the key of the confluence space
to write to
// - 'createSubpages' (optional): page specific variable to determine whether ".sect2"
sections shall be split from the current page into subpages
// - 'pagePrefix' (optional): page specific variable, the pagePrefix will be a prefix
for the page title and it's sub-pages
//                            use this if you only have access to one confluence space
but need to store several
//                            pages with the same title - a different pagePrefix will
make them unique
```

```
// - 'pageSuffix' (optional): same usage as prefix but appended to the title and it's
subpages
// only 'file' or 'url' is allowed. If both are given, 'url' is ignored
confluence.with {
    input = [
            [ file: "build/docs/html5/arc42-template-de.html" ],
    ]

    // endpoint of the confluenceAPI (REST) to be used
    // verfiy that you got the correct endpoint by browsing to
    // https://[yourServer]/[context]/rest/api/user/current
    // you should get a valid json which describes your current user
    // a working example is https://arc42-
template.atlassian.net/wiki/rest/api/user/current
    api = 'https://[yourServer]/[context]/rest/api/'

//    Additionally, spaceKey, createSubpages, pagePrefix and pageSuffix can be
globally defined here. The assignment in the input array has precedence

    // the key of the confluence space to write to
    spaceKey = 'asciidoc'

    // the title of the page containing the preamble (everything the first second
level heading). Default is 'arc42'
    preambleTitle = ''

    // variable to determine whether ".sect2" sections shall be split from the current
page into subpages
    createSubpages = false

    // the pagePrefix will be a prefix for each page title
    // use this if you only have access to one confluence space but need to store
several
    // pages with the same title - a different pagePrefix will make them unique
    pagePrefix = ''

    pageSuffix = ''

    /*
    WARNING: It is strongly recommended to store credentials securely instead of
commiting plain text values to your git repository!!!

    Tool expects credentials that belong to an account which has the right permissions
to to create and edit confluence pages in the given space.
    Credentials can be used in a form of:
     - passed parameters when calling script (-PconfluenceUser=myUsername
-PconfluencePass=myPassword) which can be fetched as a secrets on CI/CD or
     - gradle variables set through gradle properties (uses the 'confluenceUser' and
'confluencePass' keys)
    Often, same credentials are used for Jira & Confluence, in which case it is
recommended to pass CLI parameters for both entities as
```

```
    -Pusername=myUser -Ppassword=myPassword
    */

    //optional API-token to be added in case the credentials are needed for user and
password exchange.
    //apikey = "[API-token]"

    // HTML Content that will be included with every page published
    // directly after the TOC. If left empty no additional content will be
    // added
    // extraPageContent = '<ac:structured-macro ac:name="warning"><ac:parameter
ac:name="title" /><ac:rich-text-body>This is a generated page, do not edit!</ac:rich-
text-body></ac:structured-macro>
    extraPageContent = ''

    // enable or disable attachment uploads for local file references
    enableAttachments = false

    // default attachmentPrefix = attachment - All files to attach will require to be
linked inside the document.
    // attachmentPrefix = "attachment"



    // Optional proxy configuration, only used to access Confluence
    // schema supports http and https
    // proxy = [host: 'my.proxy.com', port: 1234, schema: 'http']

    // Optional: specify which Confluence OpenAPI Macro should be used to render
OpenAPI definitions
    // possible values: ["confluence-open-api", "open-api", true]. true is the same as
"confluence-open-api" for backward compatibility
    // useOpenapiMacro = "confluence-open-api"
}
//*****************************************************************************
*****
//Configuration for the export script 'exportEA.vbs'.
// The following parameters can be used to change the default behaviour of 'exportEA'.
// All parameter are optionally.
// -  connection: Parameter allows to select a certain database connection by
//    using the ConnectionString as used for directly connecting to the project
//    database instead of looking for EAP/EAPX files inside and below the 'src'
folder.
// - 'packageFilter' is an array of package GUID's to be used for export. All
//    images inside and in all packages below the package represented by its GUID
//    are exported. A packageGUID, that is not found in the currently opened
//    project, is silently skipped. PackageGUID of multiple project files can
//    be mixed in case multiple projects have to be opened.
// -  exportPath: relative path to base 'docDir' to which the diagrams and notes are
to be exported
// -  searchPath: relative path to base 'docDir', in which Enterprise Architect
project files are searched
```

```
// -  glossaryAsciiDocFormat: if set, the EA glossary is exported into exportPath as
'glossary.ad'
// -  glossaryTypes: if set and glossary is exported, used to filter for certain
types.
//    Not set or empty list will cause no filtered glossary.
// -  diagramAttributes: if set, the diagram attributes are exported and formatted as
specified

exportEA.with {
// OPTIONAL: Set the connection to a certain project or comment it out to use all
project files inside the src folder or its child folder.
// connection = "DBType=1;Connect=Provider=SQLOLEDB.1;Integrated Security=SSPI;Persist
Security Info=False;Initial Catalog=[THE_DB_NAME_OF_THE_PROJECT];Data
Source=[server_hosting_database.com];LazyLoad=1;"
// OPTIONAL: Add one or multiple packageGUIDs to be used for export. All packages are
analysed, if no packageFilter is set.
// packageFilter = [
//                    "{A237ECDE-5419-4d47-AECC-B836999E7AE0}",
//                    "{B73FA2FB-267D-4bcd-3D37-5014AD8806D6}"
//                  ]
// OPTIONAL: export diagrams, notes, etc. below folder src/docs
// exportPath = "src/docs/"
// OPTIONAL: EA project files are expected to be located in folder src/projects
// searchPath = "src/projects/"
// OPTIONAL: terms will be exported as asciidoc 'Description, single-line'
// glossaryAsciiDocFormat = "TERM:: MEANING"
// OPTIONAL: only terms of type Business and Technical will be exported.
// glossaryTypes = ["Business", "Technical"]
// OPTIONAL: Additional files will be exported containing diagram attributes in the
given asciidoc format
// diagramAttributes = "Modified: %DIAGRAM_AUTHOR%, %DIAGRAM_MODIFIED%,
%DIAGRAM_NAME%,
// %DIAGRAM_GUID%, %DIAGRAM_CREATED%, %DIAGRAM_NOTES%"
}

htmlSanityCheck.with {
    //sourceDir = "build/html5/site"

    // where to put results of sanityChecks...
    //checkingResultsDir =

    // OPTIONAL: directory where the results written to in JUnit XML format
    //junitResultsDir =

    // OPTIONAL: which statuscodes shall be interpreted as warning, error or success
defaults to standard
    //httpSuccessCodes = []
    //httpWarningCodes = []
    //httpErrorCodes = []

    // fail build on errors?
```

```
    failOnErrors = false
}

// Configuration for Jira related tasks
jira = [:]

jira.with {

    // endpoint of the JiraAPI (REST) to be used
    api = 'https://your-jira-instance'

    /*
    WARNING: It is strongly recommended to store credentials securely instead of
commiting plain text values to your git repository!!!

    Tool expects credentials that belong to an account which has the right permissions
to read the JIRA issues for a given project.
    Credentials can be used in a form of:
     - passed parameters when calling script (-PjiraUser=myUsername
-PjiraPass=myPassword) which can be fetched as a secrets on CI/CD or
     - gradle variables set through gradle properties (uses the 'jiraUser' and
'jiraPass' keys)
    Often, Jira & Confluence credentials are the same, in which case it is recommended
to pass CLI parameters for both entities as
    -Pusername=myUser -Ppassword=myPassword
    */

    // the key of the Jira project
    project = 'PROJECTKEY'

    // the format of the received date time values to parse
    dateTimeFormatParse = "yyyy-MM-dd'T'H:m:s.SSSz" // i.e. 2020-07-24'T'9:12:40.999
CEST

    // the format in which the date time should be saved to output
    dateTimeFormatOutput = "dd.MM.yyyy HH:mm:ss z" // i.e. 24.07.2020 09:02:40 CEST

    // the label to restrict search to
    label = 'label1'

    // Legacy settings for Jira query. This setting is deprecated & support for it
will soon be completely removed. Please use JiraRequests settings
    jql = "project='%jiraProject%' AND labels='%jiraLabel%' ORDER BY priority DESC,
duedate ASC"

    // Base filename in which Jira query results should be stored
    resultsFilename = 'JiraTicketsContent'

    saveAsciidoc = true // if true, asciidoc file will be created with *.adoc
extension
    saveExcel = true // if true, Excel file will be created with *.xlsx extension
```

```
    // Output folder for this task inside main outputPath
    resultsFolder = 'JiraRequests'

    /*
    List of requests to Jira API:
    These are basically JQL expressions bundled with a filename in which results will
be saved.
    User can configure custom fields IDs and name those for column header,
    i.e. customfield_10026:'Story Points' for Jira instance that has custom field with
that name and will be saved in a coloumn named "Story Points"
    */
    requests = [
        new JiraRequest(
            filename:"File1_Done_issues",
            jql:"project='%jiraProject%' AND status='Done' ORDER BY duedate ASC",
            customfields: [customfield_10026:'Story Points']
        ),
        new JiraRequest(
            filename:'CurrentSprint',
            jql:"project='%jiraProject%' AND Sprint in openSprints() ORDER BY priority
DESC, duedate ASC",
            customfields: [customfield_10026:'Story Points']
        ),
    ]
}

@groovy.transform.Immutable
class JiraRequest {
    String filename  //filename (without extension) of the file in which JQL results
will be saved. Extension will be determined automatically for Asciidoc or Excel file
    String jql // Jira Query Language syntax
    Map<String,String> customfields // map of customFieldId:displayName values for
Jira fields which don't have default names, i.e. customfield_10026:StoryPoints
}

// Configuration for OpenAPI related task
openApi = [:]

// 'specFile' is the name of OpenAPI specification yaml file. Tool expects this file
inside working dir (as a filename or relative path with filename)
// 'infoUrl' and 'infoEmail' are specification metadata about further info related to
the API. By default this values would be filled by openapi-generator plugin
placeholders
//

openApi.with {
    specFile = 'src/docs/petstore-v2.0.yaml' // i.e. 'petstore.yaml',
'src/doc/petstore.yaml'
    infoUrl = 'https://my-api.company.com'
    infoEmail = 'info@company.com'
```

```
}

// Sprint changelog configuration generate changelog lists based on tickets in sprints
of an Jira instance.
// This feature requires at least Jira API & credentials to be properly set in Jira
section of this configuration
sprintChangelog = [:]
sprintChangelog.with {
    sprintState = 'closed' // it is possible to define multiple states, i.e. 'closed,
active, future'
    ticketStatus = "Done, Closed" // it is possible to define multiple ticket
statuses, i.e. "Done, Closed, 'in Progress'"

    showAssignee = false
    showTicketStatus = false
    showTicketType = true
    sprintBoardId = 12345  // Jira instance probably have multiple boards; here it can
be defined which board should be used

    // Output folder for this task inside main outputPath
    resultsFolder = 'Sprints'

    // if sprintName is not defined or sprint with that name isn't found, release
notes will be created on for all sprints that match sprint state configuration
    sprintName = 'PRJ Sprint 1' // if sprint with a given sprintName is found, release
notes will be created just for that sprint
    allSprintsFilename = 'Sprints_Changelogs' // Extension will be automatically
added.
}
```

A.2. AsciiDoc config

A.3. Command Line Parameters