

# Multi-file Specifications in Gitlab using SpecPress

Conference Call on 3GPP Spec Modernization #8

2026-06-25

TDoc: 6GSM-260316

Source: Ericsson

Agenda item: 5.2 - Trial of Git and FORGE for 3GPP specifications

Related to...

- Proposal #5.4: Multiple formats for a single specification with Markdown as baseline
- Proposal #6.4: OneM2M (Gitlab) workflow



## Scope and Purpose

- This document visualizes how a multi-file-type-specification could be maintained in Gitlab
- Delegates may use this document to familiarize with the proposal and the tool prior to the conference call.
- At the conference call we would like to give a **live-demo of the SpecPress extension**
- We also prepared a pCR (see *6GSM-260310*) to the respective sections of the TR. The changes therein reflect our findings during SpecPress development.

## Outline

- SpecPress as PoC for 3GPP's rendering pipeline
- Folder/file structure for multi-file specifications
- Specification elements and recommended file formats
- Change tracking
- Exporting to DOCX
- Add-ons...
  - Bubble comments, RAN4 BC handler

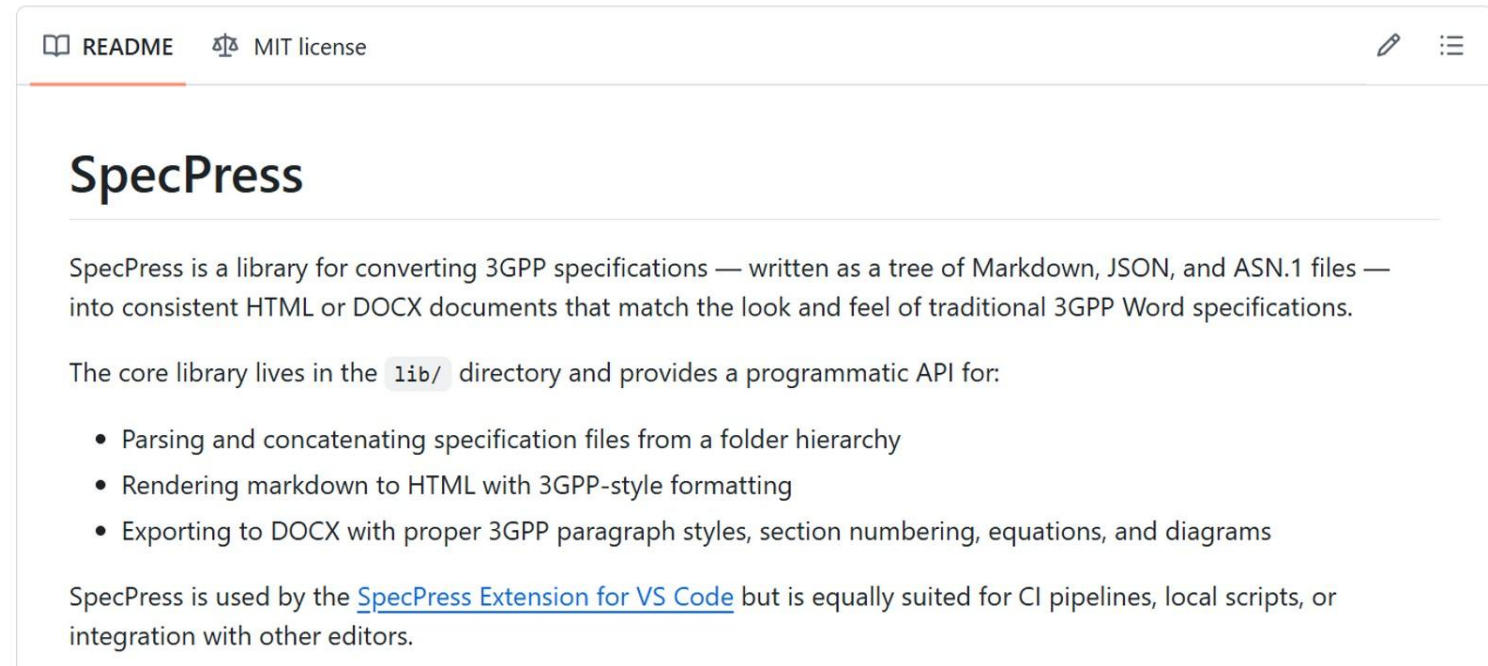


# SpecPress

## Library and extension for VS-Codium/Code



- SpecPress
  - A library that takes multi-file specifications as input and renders them as HTML or DOCX with 3GPP-like formatting.
  - Developed as a **proof-of-concept**
  - Open-source  
<https://github.com/Ericsson/specpress>
- SpecPress Extension
  - An extension for open-source IDEs Visual Studio Code/Codium

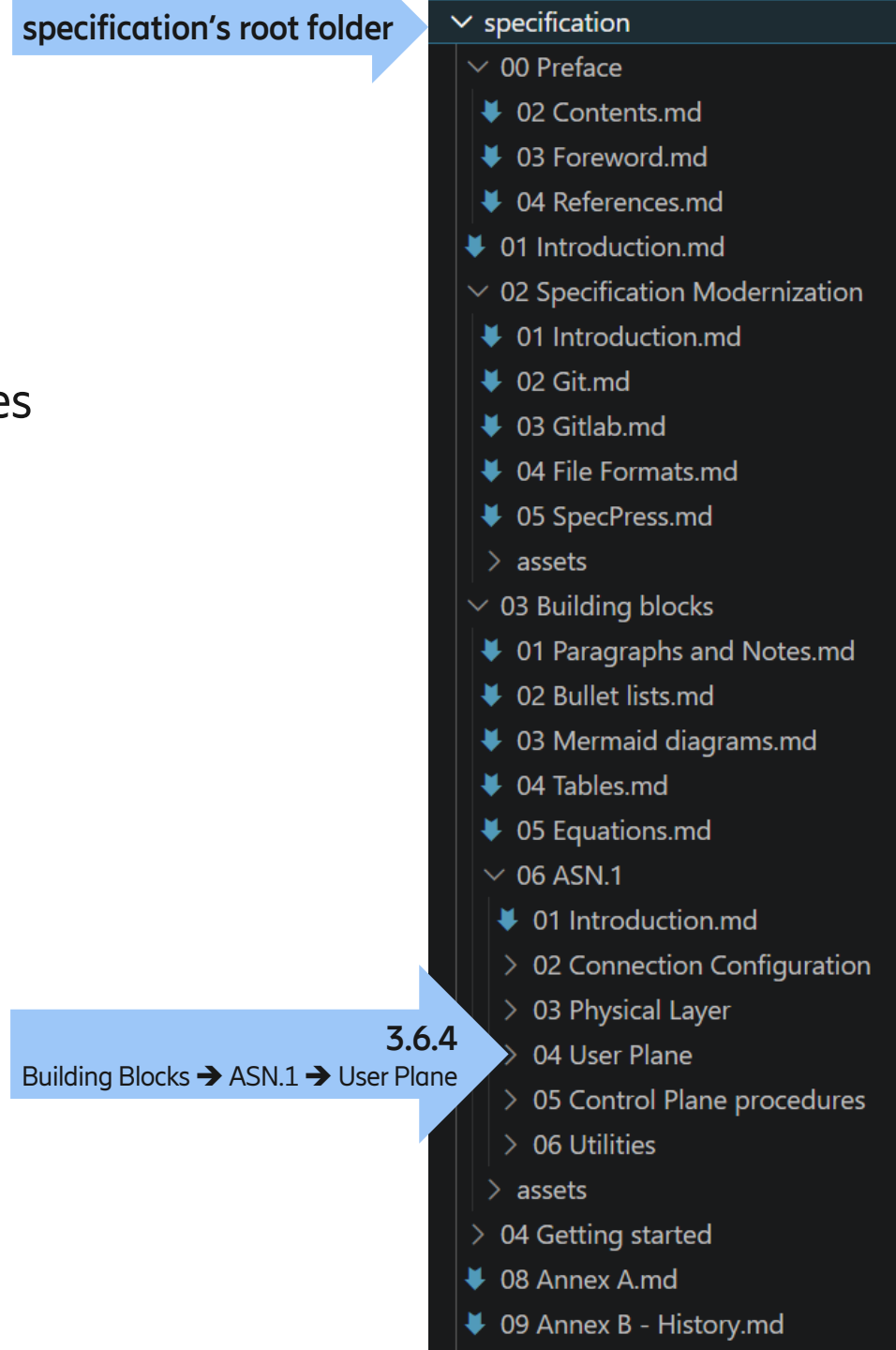


The screenshots on the following slides show the SpecPress extension and the example specification available at:  
[https://forge.3gpp.org/rep/fs\\_6gspecs\\_new/ericsson\\_multifiletypes\\_onem2m\\_example](https://forge.3gpp.org/rep/fs_6gspecs_new/ericsson_multifiletypes_onem2m_example)



# File- and Folder structure

- A specification comprises of several files and multiple file types
- **Clauses** of the specification
  - are defined by a folder structure
  - **leading numbers in folder- and file names**
- Additional non-standalone files in "*assets*" sub-directories





# Markdown by default

- **Markdown** supports intentionally just a small set of styles and formatting
  - Normal text, bold, italic, bullet lists, headings, ...
- Markdown allows **embedding** other elements
  - Images (by linking)
  - Equations (LaTeX mathematics)
  - Tables (MD specific format)
  - Mermaid sequence charts and block diagrams
- **SpecPress** auto-applies additional styles for 3GPP-specific elements:
  - Figure- and table captions
  - NOTE, Editor's Note, Example ...

Table 2.5.2-1: Supported DOCX styles

DOCX Style	Usage, Description	Supported	HTML equivalent	Comment
Heading 1 - Heading 6	Numbered clauses	yes	h1-h6	
Heading 8	Annex title for TS	yes	h1	Auto assigned by SpecPress when heading level 1 and text begins with 'Annex'
Heading 9	Annex title for TR	no		Purpose unclear. Using H8 also in TR
Normal	Standard paragraph, Definition	yes	body	
EN	For editor's notes	yes	.editors-note	
EQ	Equation	yes	katex	Style EQ is auto-assigned to block equations.
EW	(list of) Symbol, Abbreviation, Example continuation in text	<i>not yet</i>		Consider alternatives, see next table
EX	Reference, Example	yes	.example	But currently using <i>Bn</i> for references.
Bn	List element level n	yes	li, ol	
FP	Free paragraph (left justified)	yes		Used e.g. on generated cover page.
NO	Note integrated in the text	yes	.note	
NW	Note continuation in text	no		Purpose unclear. NO is more readable.
NF	Note in figure	<i>not yet</i>		Purpose unclear. Used anywhere?
TAN	Note in table	yes		
TH	Table title, Figures	yes	.table-caption, img	
TAH	Heading within table	yes	table, th	
TAC	Centred text within table	yes	table, td	
TAL	Left justified text within table	yes	table, td	
TAR	Right justified text within table	yes	table, td	
TF	Figure title	yes	.figure-caption	
TT	Contents list title	no		SpecPress uses H1 for content heading
PL	Programming language	yes	code	
Header	Header (portrait and landscape pages)	yes		In DOCX page header



# JSON for (complex) tables



- SpecPress' own format: **JsonTable**
- Captures complex tables in simple JSON files
  - Embedded in or linked from Markdown files
- Merge cells across columns and rows
- Left-/Right-/Centered Alignment
- Equations and MD-formatted text
- A GUI Editor in SpecPress extension

```
### x.x.x JSON-Table from linked file

This section shows also a JSON table but here it is defined in a separate JSON
file which linked by this MD document. SpecPress loads the JSON file and converts
it into a table if...

- the link's name is JsonTable, and
- the linked filename ends with "json"

Table x.x.x-1: A JSON-table defined in another file and embedded via a link.

[JsonTable](assets/JsonTable.json)

This table looks like the table in the previous sub-section. However, if you
check the JSON file you may notice that it declares per-column default-merge
settings ('"mergeOnAbsence": "above"' and '"mergeOnAbsence": "left"').
This simplifies tables where absent entries imply consistently (or mostly) "same
as above".
```

### 3.4.3 JSON-Table from linked file

This section shows also a JSON table but here it is defined in a separate JSON file which linked by this MD document. SpecPress loads the JSON file and converts it into a table if...

- the link's name is JsonTable, and
- the linked filename ends with "json"

Table 3.4.3-1: A JSON-table defined in another file and embedded via a link.

TPC Command Field	Accumulated $\delta_{PUSCH,b,f,c}$ or $\delta_{SRS,b,f,c}$ [dB]	Absolute $\delta_{PUSCH,b,f,c}$ or $\delta_{SRS,b,f,c}$ [dB]
0	-1	-4
1	Markdown text with <b>bold</b> and <i>italic</i> text and with a manual line break before the word 'manual'.	Some more text here.
2	This cell spans two rows and two columns and contains an equation $\delta_{PUSCH,b,f,c}$	
3		

This table looks like the table in the previous sub-section. However, if you check the JSON file you may notice that it declares per-column default-merge settings ("mergeOnAbsence": "above" and "mergeOnAbsence": "left"). This simplifies tables where absent entries imply consistently (or mostly) "same as above".

+ Row

+ Column

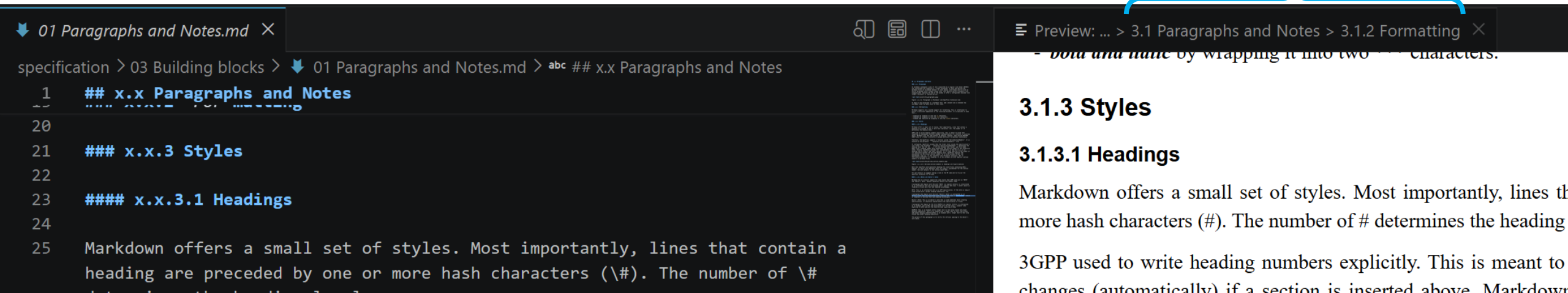
TPC Command Field	Accumulated $\delta_{PUSCH,b,f,c}$ or $\delta_{SRS,b,f,c}$ [dB]	Absolute $\delta_{PUSCH,b,f,c}$ or $\delta_{SRS,b,f,c}$ [dB]	
<div>cmd</div> <div>Right No merge</div>	<div>acc</div> <div>Center Merge</div>	<div>abs</div> <div>Center Merge</div>	
0	-1	-4	
1	Markdown text with <b>bold</b> and <i>italic</i> text and with a manual line break before the word 'manual'.	Some more text here.	
2	This cell spans two rows and two columns and contains an equation $\delta_{PUSCH,b,f,c}$		
3			



# Semi-Automated Section Numbering

- Multi-File-Specifications require sorting files in folders by leading numbers (see above)
  - ➔ Risk of ambiguity between file/folder names and section numbers inside MD-files
- SpecPress ...
  - derives section numbers from parent folder- and file names
  - injects derived section numbers into matching placeholders in MD files
  - verifies heading level and style (H1, H2, H3...) and reports errors
  - injects section numbers into figure- and table captions

SpecPress preview shows fully qualified path



The screenshot shows a code editor with a file named "01 Paragraphs and Notes.md". The editor content is as follows:

```
1  ## x.x Paragraphs and Notes
20
21  ### x.x.3 Styles
22
23  #### x.x.3.1 Headings
24
25  Markdown offers a small set of styles. Most importantly, lines that contain a
   heading are preceded by one or more hash characters (\#). The number of \#
```

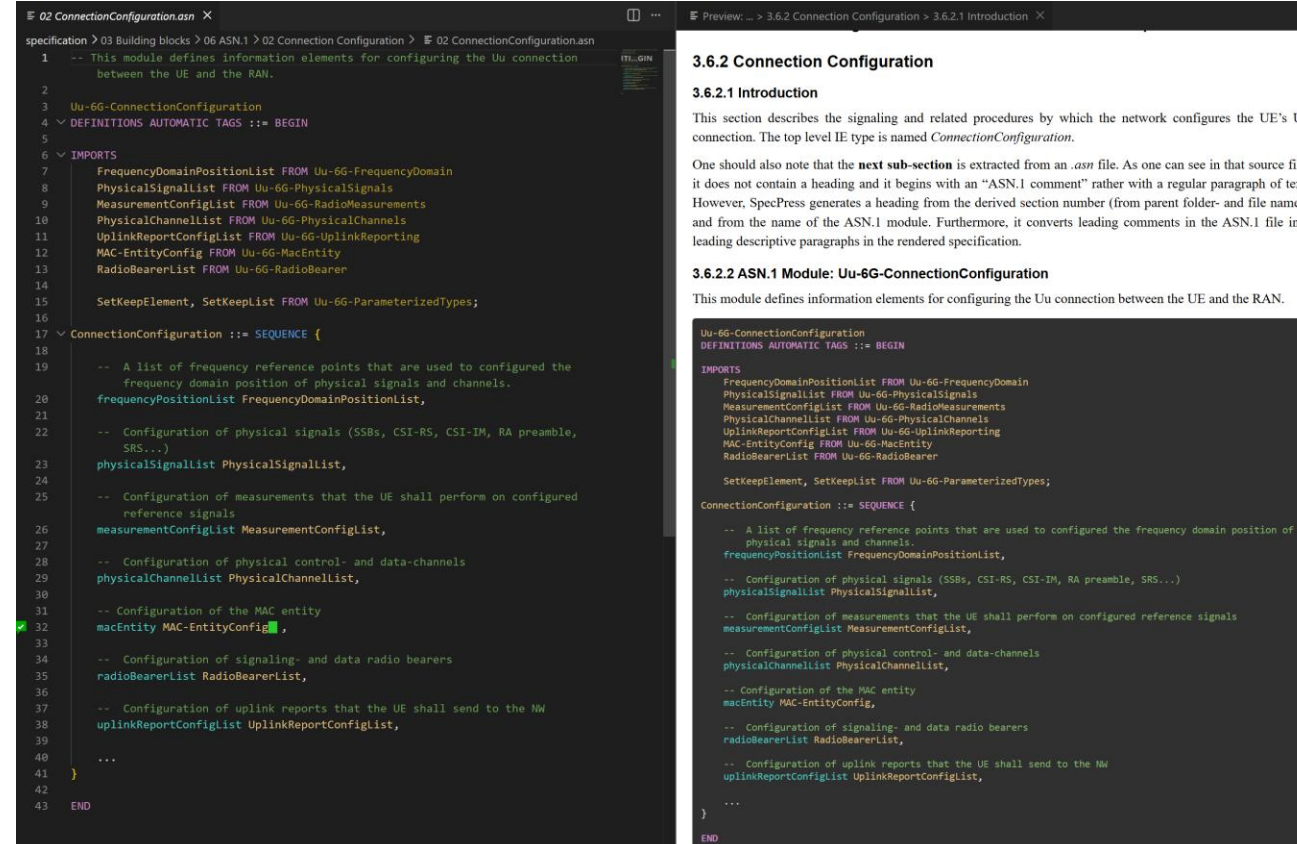
The preview window on the right shows the rendered output of the Markdown file. It displays the heading "3.1.3 Styles" and "3.1.3.1 Headings". Below the heading, the text "Markdown offers a small set of styles. Most importantly, lines that contain a heading are preceded by one or more hash characters (\#). The number of \# characters determines the heading level." is shown. A blue bracket highlights the text "SpecPress preview shows fully qualified path" pointing to the preview window.



# ASN.1

## Stored in separate \*.asn files

- Separate ASN.1 files allow using IDE plug-ins that support syntax checking, auto-completion and context sensitive search for ASN.1 elements
- SpecPress loads ASN.1 files from the file tree and...
  - Assigns the *PL* style
  - Applies additional color-highlighting
  - Wraps and indents long lines
  - Extracts modules name and leading comments to generate section heading and descriptive text



The screenshot displays two side-by-side windows from an IDE. The left window, titled '02 ConnectionConfiguration.asn', shows the source ASN.1 code. It includes a module header comment, imports for various ASN.1 modules like 'FrequencyDomainPositionList' and 'PhysicalSignalList', and a 'ConnectionConfiguration' sequence type definition. The code is syntax-highlighted with colors for comments, keywords, and identifiers. The right window, titled 'Preview: ... > 3.6.2 Connection Configuration > 3.6.2.1 Introduction', shows the rendered HTML output. It features a section heading '3.6.2 Connection Configuration', a sub-section '3.6.2.1 Introduction', and descriptive paragraphs about the module's purpose and the SpecPress tool's processing of the ASN.1 file.



# CR cover page

## Stored in JSON file within the specification repository

- JSON schema for ...
  - syntax checking,
  - auto-completion
  - tool-tips
- SpecPress renders a traditional cover page
- SpecPress auto-generates a history table from CR meta data

The following table is auto-generated from the CR meta data in the *history* folder of this specification.

NOTE: It includes only the data from JSON files with an actual number. The draft meta data for a not-yet-approved CR (*CRxxxx.json*) is omitted.

Table B.2: Change history of this specification					
CR	Rev	Cat	Title	Clauses affected	Source
0001	3	F	A first CR	6.3.2, 6.3.3	SomeOne, Some one else Inc.
0002	1	F	A second CR	5.2.1	Super Inc.

To add a change history table to the specification, add an Annex including an appropriate heading and place the `{ChangeHistory}` keyword therein. When rendering the HTML or DOCX version of the specification, specpress will crawl the CR meta data from previously agreed CRs and compile a table.

CRxxxx.json M X

specification >

```
1 {
2   Related work item code(s). Common 6G work items:
3   - FS_6GSpecs: Study on Modernization of Specification Format and Procedures for 6G
4   - FS_6G_REQ: Study on 6G Use Cases and Service Requirements
5   - FS_6G_RAN_Scen_Req: Study on 6G Scenarios and Requirements
6   - FS_6G_ARC: Study on Architecture for 6G System
7   - FS_6G_Radio: Study on 6G Radio
8   - FS_6G_APP: Study on 6G Application Enablement
9   - FS_6G_SEC: Study on Security for the 6G System
10  - FS_6G_MED: Study on Media Aspects for 6G System
11  - FS_6G_CH: Study on Charging Aspects of 6G System
12  - FS_6G_OAM: Study on 6G Management and Orchestration
13  - FS_6G_LI: Study on Lawful Interception for 6G
14  - FS_6G_CPCN_CT: Study on Control Plane Protocols in Core Network of the 6G System
15  - FS_6G_UPCN_CT: Study on Control Plane Protocols for User Plane in Core Network of 6G System
16  "Work item code": ["FS_6G_Radio"],
17  "Date": "2026-06-01",
18  "Category": "B",
19  "Release": "21",
20  "Reason for change": "1) Editing CR cover pages was error prone for CR authors.\n2) Maintaining history information in 3GPP was a time consuming task for MCC. \n3) Searching for previously approved CRs and their impact on the specification was difficult.",
21  "Summary of change": "1) Specified a JSON schema representing the required meta data for a CR cover page.\n2) Renderers that convert the JSON data into HTML- and DOCX format.\n3) A placeholder '{ChangeHistory}' that specpress replaces with an auto-generated table of all previously agreed CRs in Annex B",
22  "Impact analysis": {
23    "UE implements - RAN does not": "None, the change affects only how the specifications are maintained.",
24  },
25  "Consequences if not approved": "Error prone and time consuming specification work.",
26  "Clauses affected": ["5", "7.1"],
27  "Other specs affected": {
28    "Other core specifications": [],
29    "Test specifications": [],
30    "O&M Specifications": []
31  },
32  "Other comments": ""
33 }
```

Preview: CR Cover Page X

R2-2600239

CHANGE REQUEST

67.331

CR

-

rev

-

Current version

2.0.0

For HELP on using this form: comprehensive instructions can be found at <https://www.3gpp.org/Change-Requests>.

Proposed change affects:

UICC apps

ME

X

Radio Access Network

X

Core Network

Title:

Support auto-generated CR cover pages

Source to WG:

Ericsson

Source to TSG:

RAN2

Work item code:

FS\_6G\_Radio

Date:

2026-06-01

Category:

B

Release:

Rel-21

Reason for change:

1) Editing CR cover pages was error prone for CR authors. 2) Maintaining history information in 3GPP was a time consuming task for MCC. 3) Searching for previously approved CRs and their impact on the specification was difficult.

Summary of change:

1) Specified a JSON schema representing the required meta data for a CR cover page. 2) Renderers that convert the JSON data into HTML- and DOCX format. 3) A placeholder '{ChangeHistory}' that specpress replaces with an auto-generated table of all previously agreed CRs in Annex B

Impact analysis:

UE implements - RAN does not: None, the change affects only how the specifications are maintained.

Consequences if not approved:

Error prone and time consuming specification work.

Clauses affected:

5, 7.1

Y

N

Other specs affected:

X

Other core specifications

TS/TR ... CR ...

X

Test specifications

TS/TR ... CR ...

(show related CRs)

X

O&M Specifications

TS/TR ... CR ...

Other comments:

Forge related attachments:

| Ericsson | Uen | 2026-06-21 | Open | Page 9



# Specification Front Page

- SpecPress auto-generates the specification front page from meta data stored in a JSON file

```
{ specFrontPage.json X
specification > assets > {} specFrontPage.json > ...
1  {
2    "SPEC_NUMBER": "3GPP TR 67.123",
3    "VERSION": "V1.0.0",
4    "DATE": "2026-03",
5    "TSG": "Technical Specification Group Radio Access Network",
6    "TITLE": "Specification Modernization Example",
7    "DOC_TYPE": "Technical Report",
8    "RELEASE": "Release 20",
9    "KEYWORDS": "GSM, UMTS, LTE, 5G, methodology"
10 }
```

### 3rd Generation Partnership Project; Technical Specification Group Radio Access Network; Specification Modernization Example (Release 20)

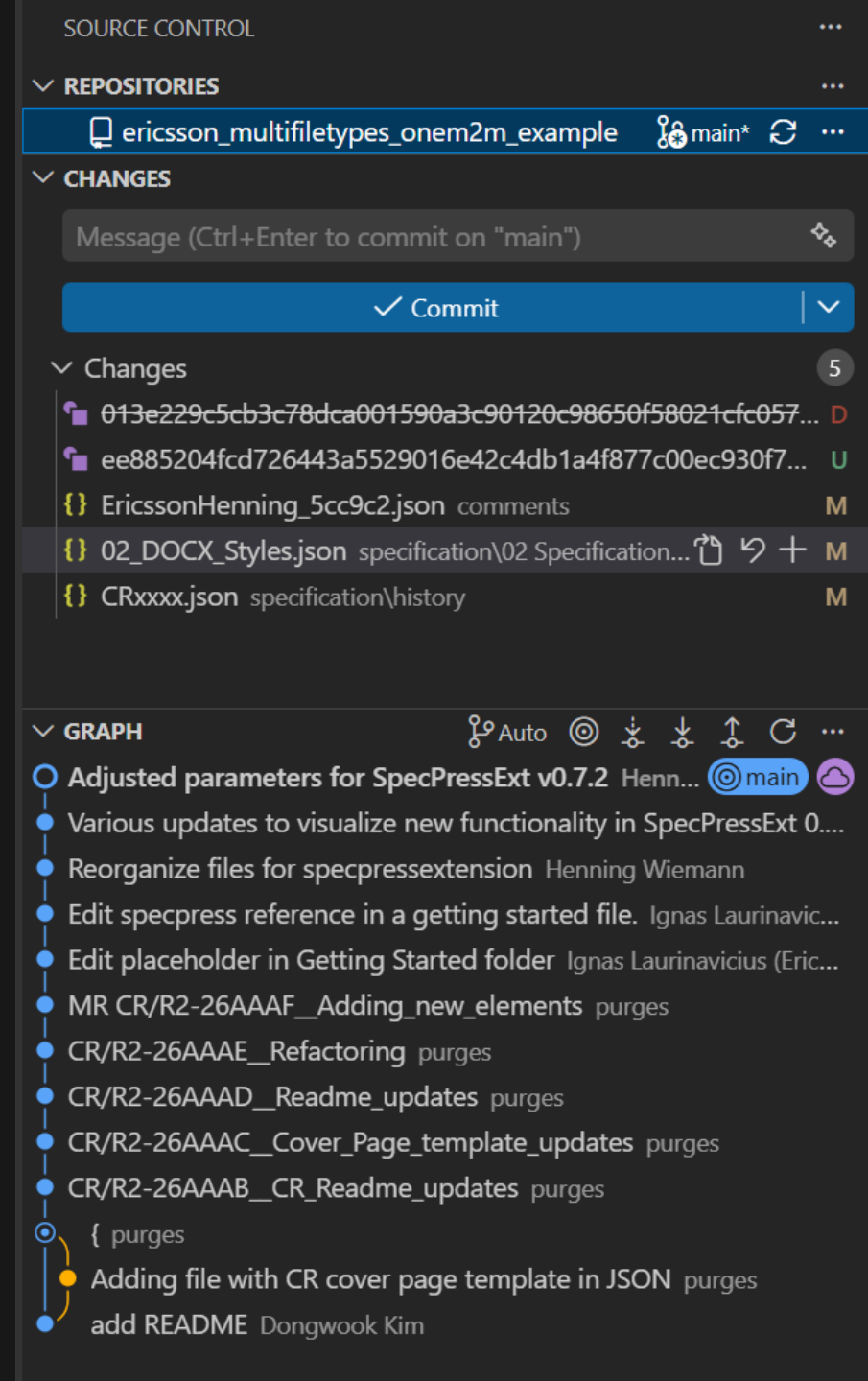


The present document has been developed within the 3rd Generation Partnership Project (3GPP™) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPP Organizational Partners and shall not be implemented. This Technical Report is provided for future development work within 3GPP only. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP™ system should be obtained via the 3GPP Organizational Partners' Publications Offices.



# Tracked Changes

- Markdown, ASN.1 and JSON are text files
  - They cannot carry “diff” information
- However, *git* shows
  - the local files that have been modified.
  - a list of previous versions (commits)
- And *git* can **generate a “diff”** between any two versions of a file which IDEs show side-by-side
- In addition, SpecPress can show a live-diff ... (see next slide)





# Live Change-Tracking in SpecPress



22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51

```
},
{
  "style": "Heading 9",
  "usage": "Annex title for TR",
  "supported": "no",
  "comment": "Purpose unclear. Using H8
    also in TR"
},
{
  "style": "Normal",
  "usage": "Standard paragraph, Definition",
  "supported": "***yes***",
  "html": "body"
},
{
  "style": "EN",
  "usage": "For editor's notes",
  "supported": "***yes***",
  "html": ".editors-note"
},
{
  "style": "EQ",
  "usage": "Equation",
  "supported": "***yes***",
  "html": "katex",
  "comment": "Style EQ is auto-assigned to
    block equations."
},
{
  "style": "EW",
  "usage": "(list of) Symbol, Abbreviation,
    Example continuation in text",
  "supported": "**not yet**",
  "comment": "SpecPress auto-assigns the style
    to paragraphs beginning with the term
    'Editor's Note'"
}
```

→

+

22  
23  
24  
25  
26  
27  
28  
29  
30  
31  
32  
33  
34  
35  
36  
37  
38  
39  
40  
41  
42  
43  
44  
45  
46  
47  
48  
49  
50  
51

```
},
{
  "style": "Heading 9",
  "usage": "Annex title for TR",
  "supported": "no",
  "comment": "Purpose still unclear. Using H8
    also in TR"
},
{
  "style": "Normal",
  "usage": "Standard paragraph, Definition",
  "supported": "***yes***",
  "html": "body"
},
{
  "style": "EN",
  "usage": "For editor's notes",
  "supported": "***yes***",
  "html": ".editors-note",
  "comment": "SpecPress auto-assigns the style
    to paragraphs beginning with the term
    'Editor's Note'"
},
{
  "style": "EQ",
  "usage": "Equation",
  "supported": "***yes***",
  "html": "katex",
  "comment": "Style EQ is auto-assigned to
    block equations."
},
{
  "style": "EW",
  "usage": "(list of) Symbols, Abbreviations,
    Examples",
  "supported": "**not yet**",
  "comment": "SpecPress auto-assigns the style
    to paragraphs beginning with the term
    'Editor's Note'"
}
```

DOCX styles that should be used according to 21.801.

Table 2.5.2-1: Supported DOCX styles

DOCX Style	Usage, Description	Supported	HTML equivalent	Comment
Heading 1 - Heading 6	Numbered clauses	yes	h1-h6	
Heading 8	Annex title for TS	yes	h1	Auto assigned by SpecPress when heading level 1 and text begins with 'Annex'
Heading 9	Annex title for TR	no		Purpose <u>still</u> unclear. Using H8 also in TR
Normal	Standard paragraph, Definition	yes	body	
EN	For editor's notes	yes	.editors-note	<u>SpecPress auto-assigns the style to paragraphs beginning with the term 'Editor's Note'</u>
EQ	Equation	yes	katex	Style EQ is auto-assigned to block equations.
EW	(list of) <u>Symbol</u> <u>Symbols</u> , <u>Abbreviation</u> <u>Abbreviations</u> , <u>Example continuation in text</u> <u>Examples</u>	not yet		Consider alternatives, see next table
EX	Reference, Example	yes	.example	But currently using <i>Bn</i> for references.
Bn	List element level n	yes	li, ol	
FD	Front cover, back cover			Used e.g. on generated cover



# Exporting to DOCX

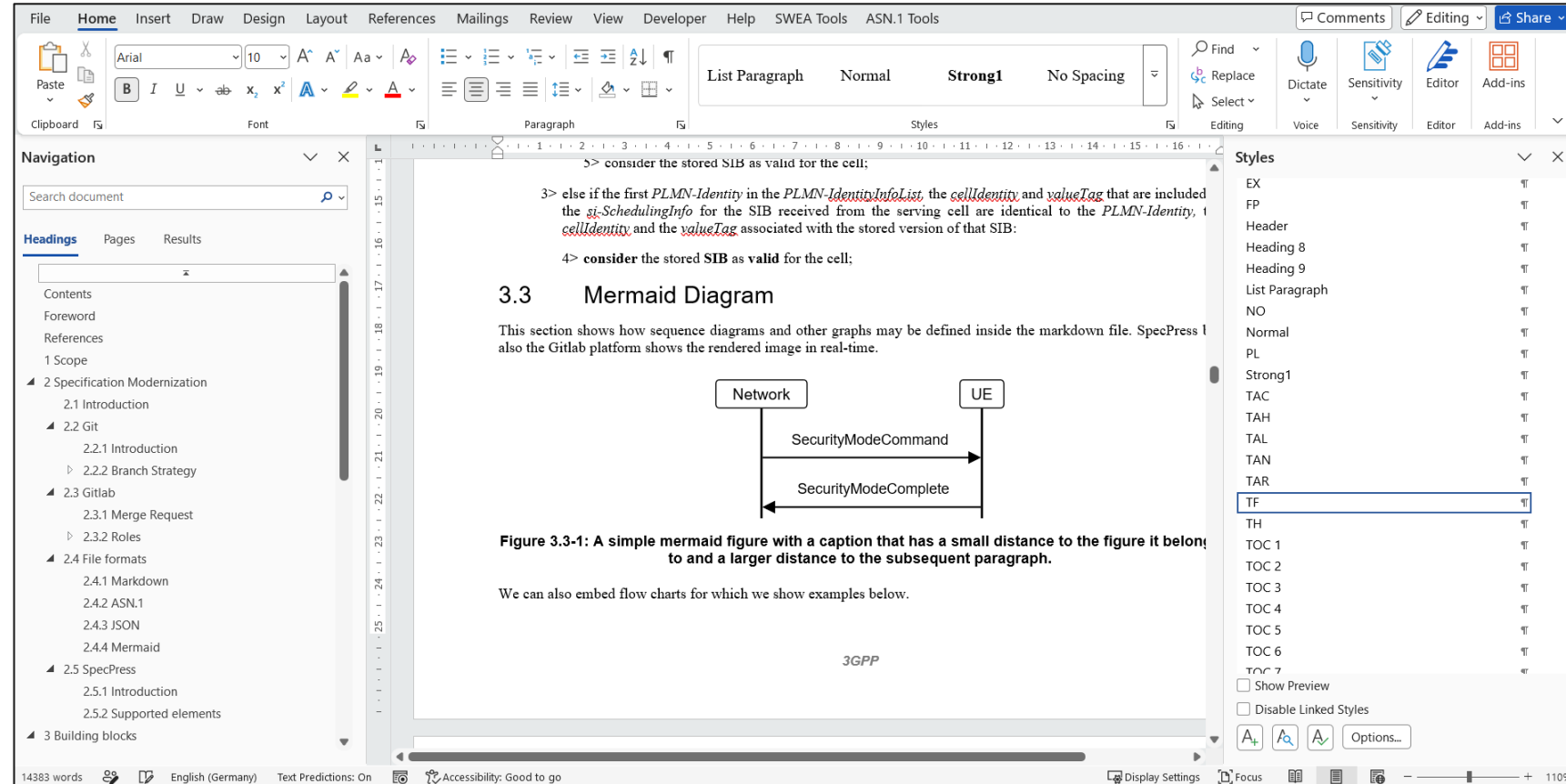
- The final specifications must be exported to DOCX



# Exporting specification to DOCX

## Using SpecPress

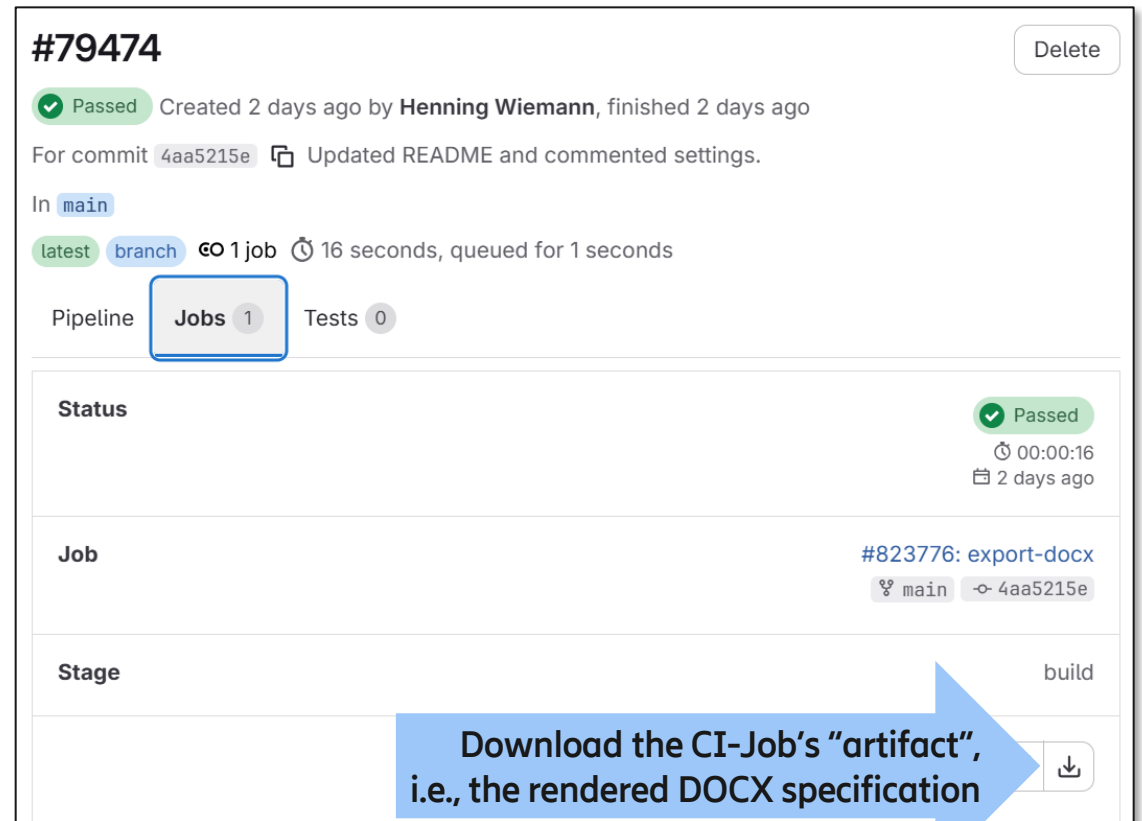
- Select the folder(s) to be exported
- Choose a (git) version to export
- Decide whether to include the specification front page
- Choose a destination folder
- Open the DOCX which shows...
  - 3GPP styles
  - Section outline
  - Table of Content
  - Images and tables
  - ...





# Gitlab auto-generates DOCX

- SpecPress' DOCX export can be run as CI Job
  - Upon every commit to a specification repository
  - Downloadable via Gitlab's web interface



**#79474** Delete

✓ **Passed** Created 2 days ago by **Henning Wiemann**, finished 2 days ago

For commit `4aa5215e` Updated README and commented settings.

In `main`

`latest` `branch` 1 job 16 seconds, queued for 1 seconds

Pipeline **Jobs** 1 Tests 0

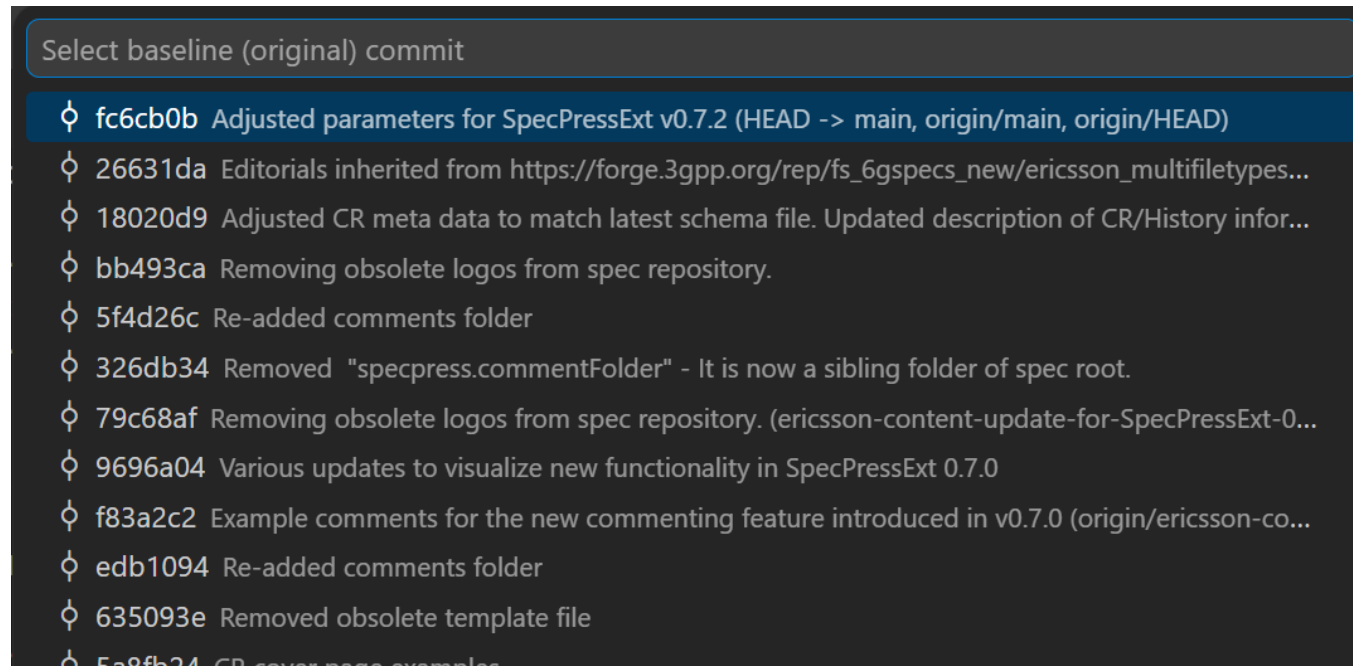
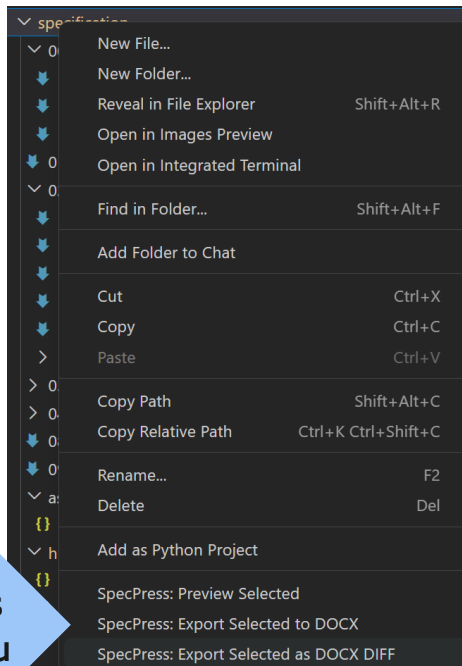
<b>Status</b>	<b>Passed</b> 00:00:16 2 days ago
<b>Job</b>	<a href="#">#823776: export-docx</a> <code>main</code> <code>4aa5215e</code>
<b>Stage</b>	build

Download the CI-Job's "artifact",  
i.e., the rendered DOCX specification



# Exporting a CR in DOCX format

- Choose the folder(s) or file(s) that the CR should contain
- Choose *SpecPress: Export Selected as DOCX DIFF*
- Choose base- and target version(s) from git history
- Set author name(s)
- Optionally prepend specification front page or CR cover page





# Exporting a CR in DOCX format

- SpecPress ...
  - generates DOCX of each selected version
  - uses MS Word to generate a “DIFF” of the DOCX files including the author names.

The following text is taken from 3GPP TS38.331 (NR RRC) with the usual "I>" bullet with appropriate changes. 74756ea\_minor-edits-QC-test, 02/06/2026 10:25:00 deleted: may

The UE ~~shall~~may:

1> delete any stored version of a SIB after 3 hours from the moment it was successfully confirmed as valid;

1> for each stored version of a SIB:

2> if the areaScope is associated and its value for the stored version of the SIB is the same as the value of the si-SchedulingInfo for that SIB from the serving cell;

3> if the UE is NPN capable and the cell is an NPN-only cell:

4> if the first ~~and second~~ NPN identity included in the NPN-IdentityInfoList, the systemInformationAreaID and the valueTag that are included in the si-SchedulingInfo for the SIB received from the serving cell are identical to the NPN identity, the systemInformationAreaID and the valueTag associated with the stored version of that SIB;

R2-2600067

CHANGE REQUEST					
67.331	CR	0001	rev	1	Current version: 1.0.0
For HELP on using this form: comprehensive instructions can be found at <a href="https://www.3gpp.org/Change-Requests">https://www.3gpp.org/Change-Requests</a> .					
Proposed change affects:		UICC apps	ME	X	Radio Access Network X Core Network
Title:	Support auto-generated CR cover pages				
Source to WG:	Ericsson				
Source to TSG:	RAN2				
Work item code:	FS_6G_Radio			Date:	2026-06-01
Category:	B			Release:	Rel-21
Reason for change:	1) Editing CR cover pages was error prone for CR authors. 2) Maintaining history information in 3GPP was a <u>time consuming</u> task for MCC. 3) Searching for previously approved CRs and their impact on the specification was difficult.				
Summary of change:	1) Specified a JSON schema representing the required meta data for a CR cover page. 2) Renderers that convert the JSON data into HTML- and DOCX format. 3) A placeholder <code>{ChangeHistory}</code> that <u>specpress</u> replaces with an auto-generated table of all previously agreed CRs in Annex B  <b>Impact analysis:</b> <b>UE implements - RAN does not:</b> None, the change affects only how the specifications are maintained.				
Consequences if not approved:	Error prone and <u>time consuming</u> specification work.				
Clauses affected:	5				
	Y	N			
Other specs affected:		X	Other core specifications	TS/TR ... CR ...	
		X	Test specifications	TS/TR ... CR ...	
		X	O&M Specifications	TS/TR ... CR ...	
Other comments:					
Forge related attachments:					



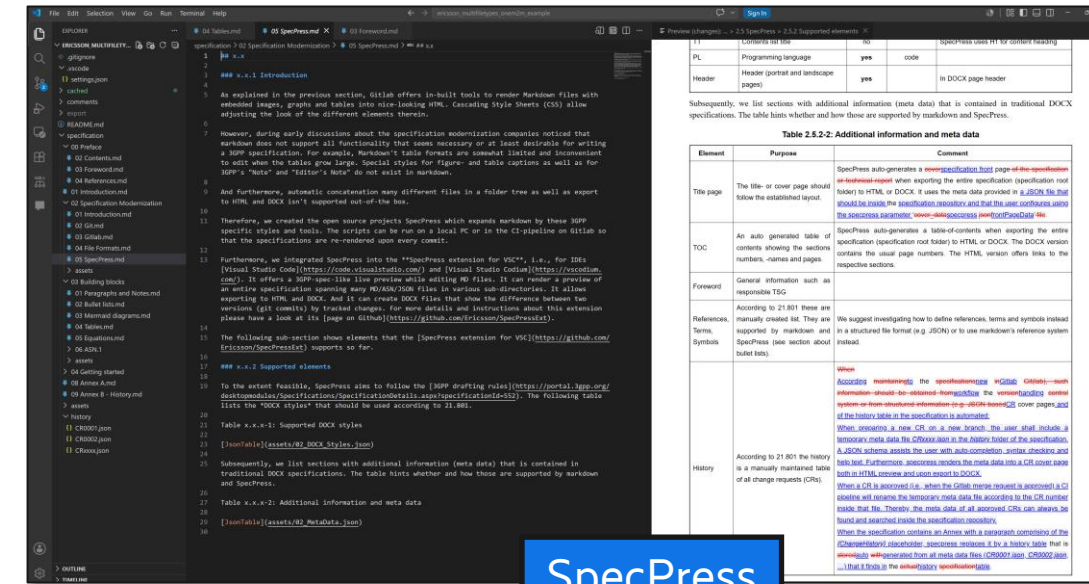
# Preparing and reviewing CRs prior to submission

- The following slides outline possible ways for drafting, sharing, reviewing and commenting CRs, i.e., the workflow prior to the actual submission of the contribution.
  - 1: SpecPress-rendered **DOCX by email**
  - 2: **Network drive** (Samba, Sharepoint, ...)
  - 3: **Private fork** on 3GPP's Gitlab
  - 4: Company-internal **mirror of 3GPP's Gitlab** repositories



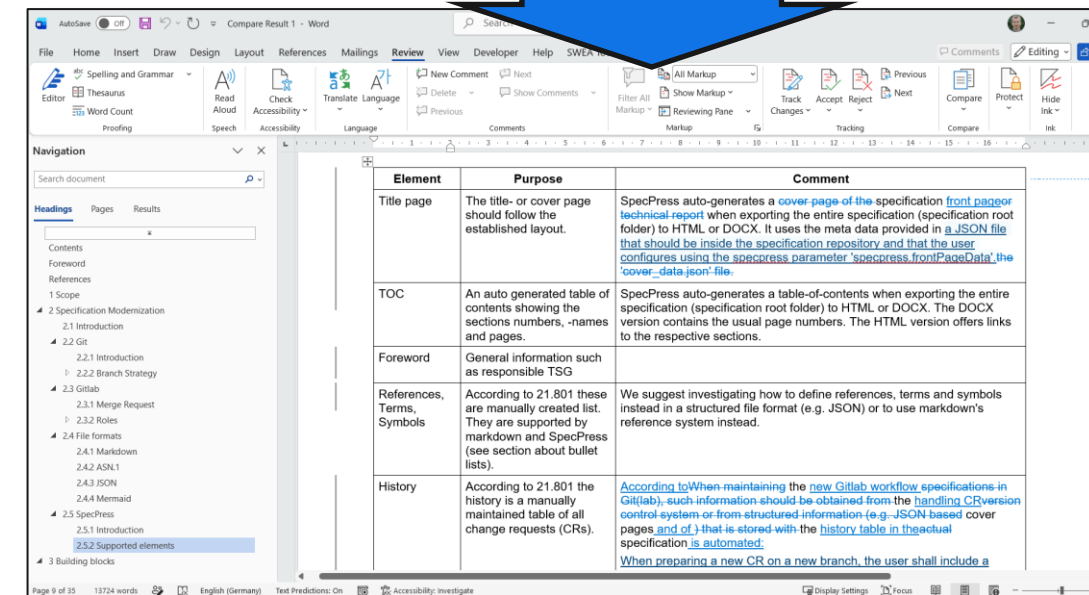
# 1: Share a rendered DOCX by email

- Prepare CR in a local git repository
- Render a DOCX DIFF using SpecPress
- Share DOCX with colleagues using email and obtain feedback by mail, in bubble-comments or as tracked changes.
- Incorporate changes into local CR
- Upon submission deadline, (git-)push final version to 3GPP's Gitlab server



The screenshot shows the SpecPress application interface. On the left, a file explorer displays the project structure, including folders for 'Specification', 'Assets', and 'History'. The main window shows a rendered DOCX diff with various sections like 'Introduction', 'Gitlab', and 'File formats'. On the right, a 'Table 2.5.2-2: Additional information and meta data' is displayed, detailing the purpose and comment for each element (Title page, TOC, Foreword, References, Terms, Symbols, History).

Element	Purpose	Comment
Title page	The title- or cover page should follow the established layout.	SpecPress auto-generates a <a href="#">cover page of the specification</a> when exporting the entire specification (specification root folder) to HTML or DOCX. It uses the meta data provided in a <a href="#">JSON file</a> that should be inside the specification repository and that the user configures using the <code>specpress.frontPageData</code> parameter.
TOC	An auto-generated table of contents showing the sections numbers, -names and pages.	SpecPress auto-generates a table-of-contents when exporting the entire specification (specification root folder) to HTML or DOCX. The DOCX version contains the usual page numbers. The HTML version offers links to the respective sections.
Foreword	General information such as responsible TSG.	According to 21.801 these are manually created list. They are supported by markdown and SpecPress (see section about bullet lists).
References, Terms, Symbols	According to 21.801 these are manually created list. They are supported by markdown and SpecPress (see section about bullet lists).	We suggest investigating how to define references, terms and symbols instead in a structured file format (e.g. JSON) or to use markdown's reference system instead.
History	According to 21.801 the history is a manually maintained table of all change requests (CRs).	<a href="#">According to</a> When maintaining the <a href="#">new Gitlab workflow</a> specifications in <a href="#">Gitlab</a> , such information should be obtained from the <a href="#">handling CR version control system</a> or from structured information (e.g. JSON-based cover pages and of j) that is stored with the <a href="#">history table in the actual specification</a> is automated. When preparing a new CR on a new branch, the user shall include a <a href="#">removals meta data</a> the <a href="#">CR data</a> as the <a href="#">history table</a> of the <a href="#">specification</a> . A JSON system avoids the user with <a href="#">links</a> <a href="#">version</a> <a href="#">control</a> and <a href="#">tools</a> <a href="#">base</a> . Furthermore, <a href="#">accesses</a> <a href="#">renders</a> the <a href="#">meta data</a> into a <a href="#">CR</a> <a href="#">base</a> <a href="#">data</a> <a href="#">table</a> in <a href="#">HTML</a> <a href="#">version</a> <a href="#">report</a> to <a href="#">DOCX</a> . When a <a href="#">CR</a> is <a href="#">accepted</a> (i.e. when the <a href="#">Gitlab merge request</a> is <a href="#">accepted</a> ) a <a href="#">CR</a> <a href="#">element</a> will <a href="#">remove</a> the <a href="#">historical</a> <a href="#">CR</a> <a href="#">data</a> <a href="#">table</a> <a href="#">according</a> to the <a href="#">CR</a> <a href="#">number</a> <a href="#">table</a> <a href="#">that</a> <a href="#">the</a> <a href="#">file</a> <a href="#">thereby</a> <a href="#">the</a> <a href="#">meta data</a> of <a href="#">all</a> <a href="#">associated</a> <a href="#">CRs</a> <a href="#">can</a> <a href="#">always</a> <a href="#">be</a> <a href="#">found</a> <a href="#">and</a> <a href="#">searched</a> <a href="#">inside</a> the <a href="#">specification</a> <a href="#">repository</a> . When the <a href="#">specification</a> <a href="#">contains</a> an <a href="#">Anex</a> with a <a href="#">relevant</a> <a href="#">comment</a> of the <a href="#">Change request</a> <a href="#">discriminator</a> , <a href="#">accesses</a> <a href="#">renders</a> it in a <a href="#">history</a> <a href="#">table</a> <a href="#">that</a> <a href="#">is</a> <a href="#">automatically</a> <a href="#">generated</a> from <a href="#">all</a> <a href="#">meta data</a> files ( <a href="#">CR0001</a> <a href="#">also</a> <a href="#">CR0002</a> <a href="#">and</a> <a href="#">...</a> that it finds in the <a href="#">subdirectory</a> <a href="#">specification</a> <a href="#">table</a> ).



The screenshot shows a Microsoft Word document with the rendered DOCX diff. The interface includes the Word ribbon (File, Home, Insert, Draw, Design, Layout, References, Mailings, Review, View, Developer, Help) and a navigation pane on the left. The main content area displays the rendered DOCX diff, including sections like 'Introduction', 'Gitlab', and 'File formats'. A table of contents is visible on the right side of the document.

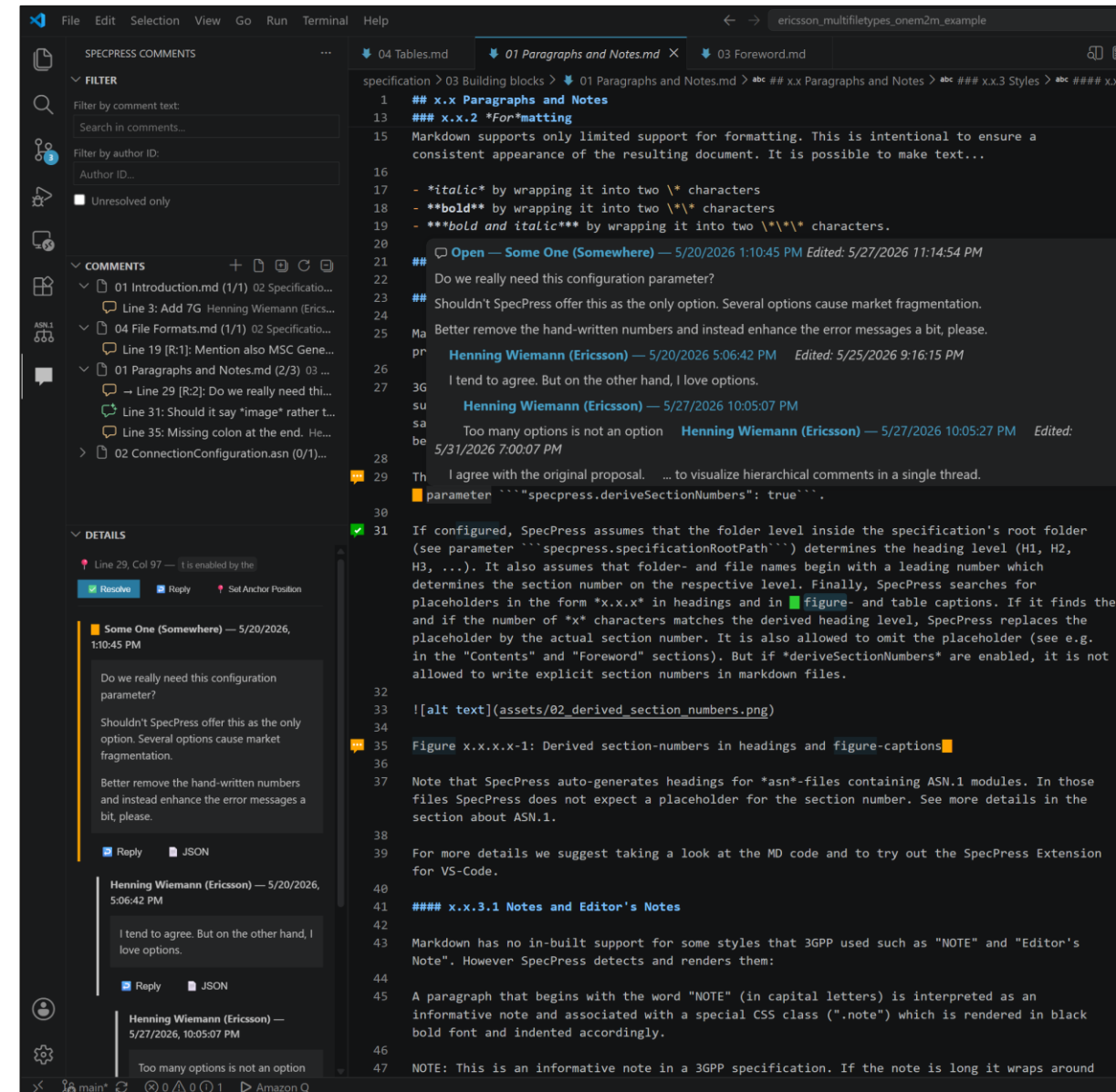
Element	Purpose	Comment
Title page	The title- or cover page should follow the established layout.	SpecPress auto-generates a <a href="#">cover page of the specification</a> when exporting the entire specification (specification root folder) to HTML or DOCX. It uses the meta data provided in a <a href="#">JSON file</a> that should be inside the specification repository and that the user configures using the <code>specpress.frontPageData</code> parameter.
TOC	An auto-generated table of contents showing the sections numbers, -names and pages.	SpecPress auto-generates a table-of-contents when exporting the entire specification (specification root folder) to HTML or DOCX. The DOCX version contains the usual page numbers. The HTML version offers links to the respective sections.
Foreword	General information such as responsible TSG.	According to 21.801 these are manually created list. They are supported by markdown and SpecPress (see section about bullet lists).
References, Terms, Symbols	According to 21.801 these are manually created list. They are supported by markdown and SpecPress (see section about bullet lists).	We suggest investigating how to define references, terms and symbols instead in a structured file format (e.g. JSON) or to use markdown's reference system instead.
History	According to 21.801 the history is a manually maintained table of all change requests (CRs).	<a href="#">According to</a> When maintaining the <a href="#">new Gitlab workflow</a> specifications in <a href="#">Gitlab</a> , such information should be obtained from the <a href="#">handling CR version control system</a> or from structured information (e.g. JSON-based cover pages and of j) that is stored with the <a href="#">history table in the actual specification</a> is automated. When preparing a new CR on a new branch, the user shall include a <a href="#">removals meta data</a> the <a href="#">CR data</a> as the <a href="#">history table</a> of the <a href="#">specification</a> . A JSON system avoids the user with <a href="#">links</a> <a href="#">version</a> <a href="#">control</a> and <a href="#">tools</a> <a href="#">base</a> . Furthermore, <a href="#">accesses</a> <a href="#">renders</a> the <a href="#">meta data</a> into a <a href="#">CR</a> <a href="#">base</a> <a href="#">data</a> <a href="#">table</a> in <a href="#">HTML</a> <a href="#">version</a> <a href="#">report</a> to <a href="#">DOCX</a> . When a <a href="#">CR</a> is <a href="#">accepted</a> (i.e. when the <a href="#">Gitlab merge request</a> is <a href="#">accepted</a> ) a <a href="#">CR</a> <a href="#">element</a> will <a href="#">remove</a> the <a href="#">historical</a> <a href="#">CR</a> <a href="#">data</a> <a href="#">table</a> <a href="#">according</a> to the <a href="#">CR</a> <a href="#">number</a> <a href="#">table</a> <a href="#">that</a> <a href="#">the</a> <a href="#">file</a> <a href="#">thereby</a> <a href="#">the</a> <a href="#">meta data</a> of <a href="#">all</a> <a href="#">associated</a> <a href="#">CRs</a> <a href="#">can</a> <a href="#">always</a> <a href="#">be</a> <a href="#">found</a> <a href="#">and</a> <a href="#">searched</a> <a href="#">inside</a> the <a href="#">specification</a> <a href="#">repository</a> . When the <a href="#">specification</a> <a href="#">contains</a> an <a href="#">Anex</a> with a <a href="#">relevant</a> <a href="#">comment</a> of the <a href="#">Change request</a> <a href="#">discriminator</a> , <a href="#">accesses</a> <a href="#">renders</a> it in a <a href="#">history</a> <a href="#">table</a> <a href="#">that</a> <a href="#">is</a> <a href="#">automatically</a> <a href="#">generated</a> from <a href="#">all</a> <a href="#">meta data</a> files ( <a href="#">CR0001</a> <a href="#">also</a> <a href="#">CR0002</a> <a href="#">and</a> <a href="#">...</a> that it finds in the <a href="#">subdirectory</a> <a href="#">specification</a> <a href="#">table</a> ).



# 2: Network drive for source files

- Clone specification into **network drive** (e.g. Samba, Sharepoint, ...)
- Share link with colleagues who open the shared folder in their Visual Studio Code/Codium
- **Edit and add files collaboratively** (changes appear almost instantly)
- Add, reply and resolve comments using [SpecPress extension](#)'s commenting function
- Upon submission deadline, (git-)push final version to 3GPP's Gitlab server

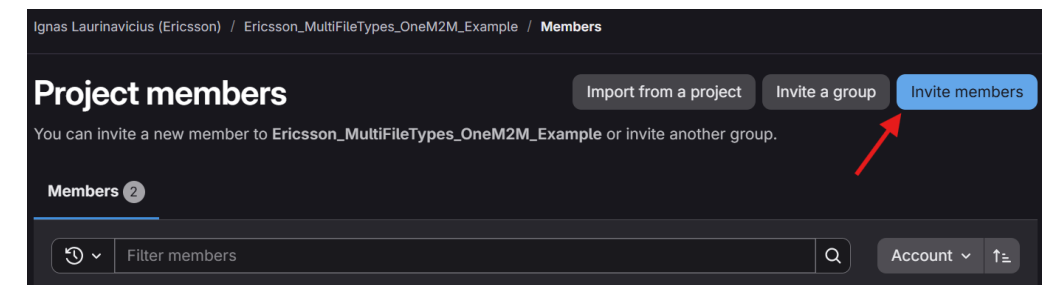
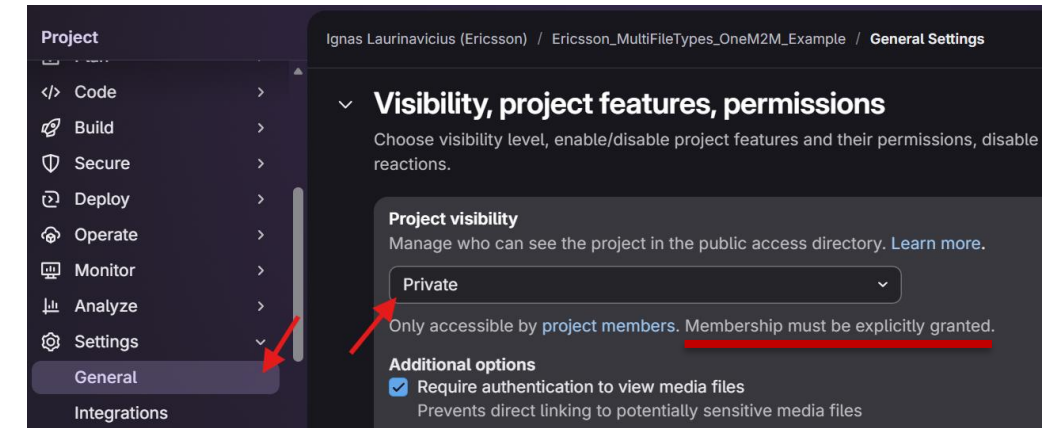
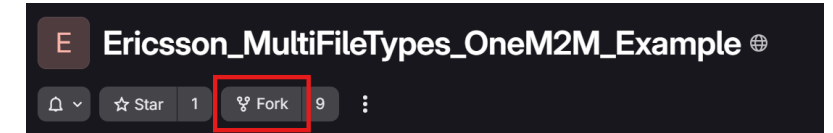
Note that this work-flow and the commenting function may also be used to collect company-internal reviews on running CRs.





# 3: Private fork on 3GPP's Gitlab

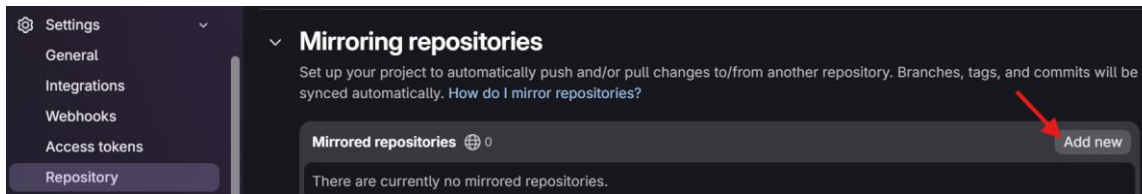
- Create a **private fork** on 3GPP's Gitlab server
- Gitlab allows one member to own one fork
  - **Full rights** as if they were the administrator
  - **May grant access to other Gitlab users**
    - Users with Developer rights on the fork see what the owner of the fork can see: branches, repository filesystem and comments by other developers on fork's merge requests
- **Permitted users collaborate like on public repositories** (git branching and commits, Gitlab merge requests and commenting function, ...)
- Upon submission deadline, (git-)push final version to the public repository instead



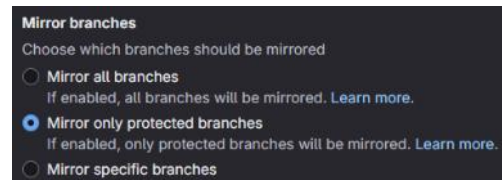


# 4: Internal mirror of 3GPP's Gitlab repo

- A company-internal Gitlab server can **pull periodically** all changes from a repository on 3GPP Forge to a repository on the internal Gitlab server ("pull mirroring")



- Preferably, **mirror protected branches** only
  - "main", "Rel-20", ...
- Don't push changes into mirrored branches as they would be overridden by subsequent changes that are pulled from 3GPP's Gitlab



- To prepare a CR...
  - draft CR using VSC + SpecPress
  - push it to a new branch on internal Gitlab server
  - create a merge request for internal discussion and collaboration
- Upon submission deadline ...
  - change the "remote" and push the (so-far internal) branch to 3GPP's Gitlab server
  - create a merge request on 3GPP's Gitlab server



# Other tools

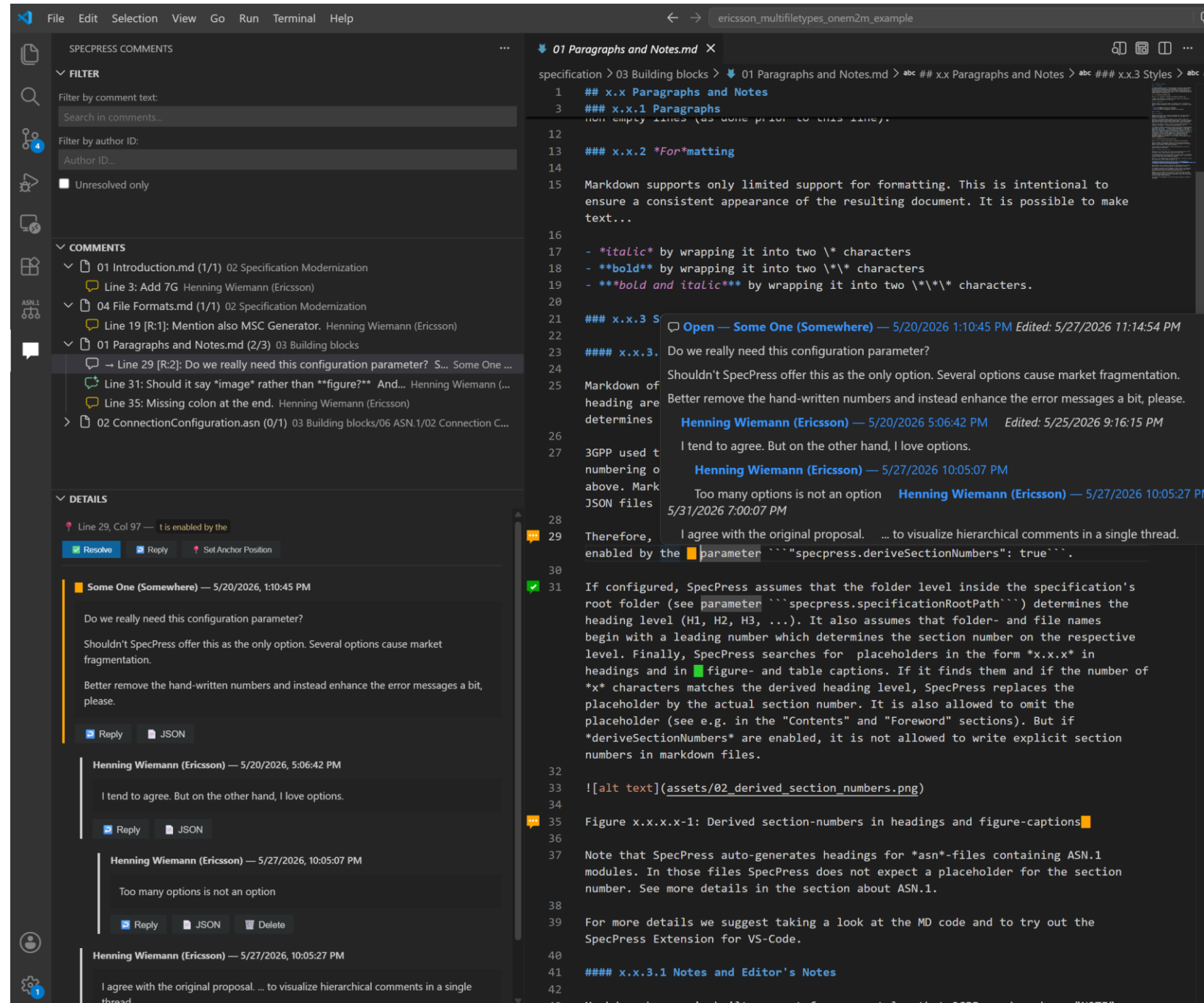
- Multi-file-type specifications and open-source tools allow 3GPP to adjust their ways-of-working and to develop their own tools

... some examples ...



# Bubble Comments

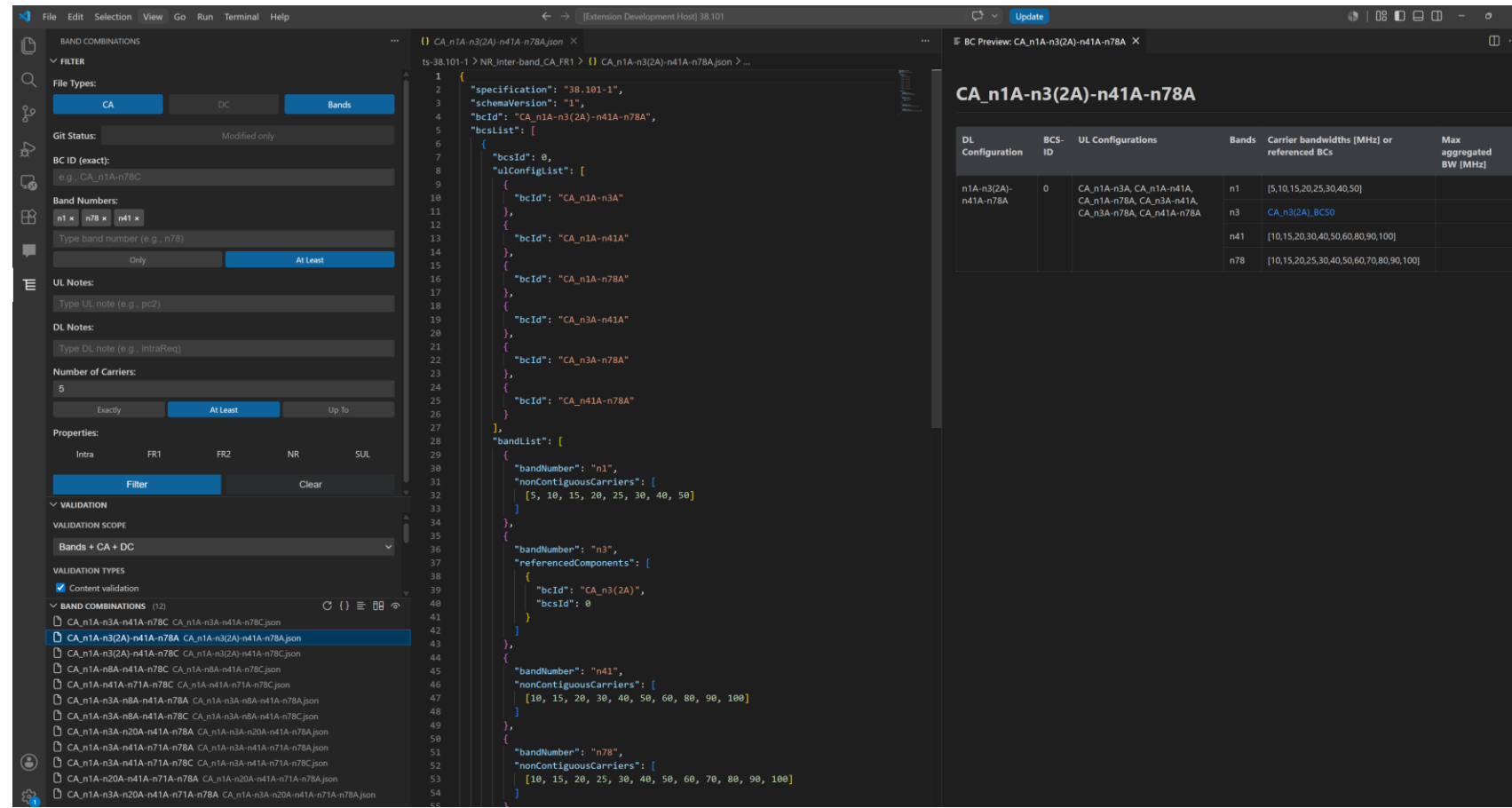
- The SpecPress extension for VSC offers bubble comments
- Stored in separate JSON files in the specification repository
  - Filter by user or content
  - List view, in-line icons, hovering text bubbles
  - Reply, (un-)resolve
  - ...





# RAN4 Band-Combination GUI

- The SpecPress extension for VSC will soon offer tools for working with 38.101's JSON data
  - Advanced Filtering
  - Sorting
  - Preview in table format
  - Content Validation
  - Format normalization
  - Export as DIFF file
  - ...



The screenshot shows the RAN4 Band-Combination GUI integrated into the Visual Studio Code editor. The interface is divided into three main sections:

- Left Sidebar (Filter and Validation):** Contains a 'BAND COMBINATIONS' section with a 'FILTER' button and a 'BANDS' button. Below this is a 'Git Status' section showing 'Modified only'. The 'BC ID (exact):' field is set to 'CA\_n1A-n3(2A)-n41A-n78A'. The 'Band Numbers' section shows 'n1', 'n3(2A)', and 'n41'. The 'UL Notes' and 'DL Notes' sections are empty. The 'Number of Carriers' is set to '5'. The 'Properties' section shows 'Intra', 'FR1', 'FR2', 'NR', and 'SUL'. The 'Filter' button is highlighted. Below this is a 'VALIDATION' section with a 'VALIDATION SCOPE' dropdown set to 'Bands + CA + DC'. The 'VALIDATION TYPES' section shows 'Content validation' checked. The 'BAND COMBINATIONS (12)' list shows the selected combination 'CA\_n1A-n3(2A)-n41A-n78A' highlighted.
- Central Editor:** Displays the JSON data for the selected band combination. The JSON structure includes 'specification', 'schemaVersion', 'bcId', 'bcsList', and 'bandList'.
- Right Sidebar (Table Preview):** Shows a table preview for the selected band combination. The table has columns: 'DL Configuration', 'BCS-ID', 'UL Configurations', 'Bands', 'Carrier bandwidths [MHz] or referenced BCs', and 'Max aggregated BW [MHz]'. The table contains three rows of data.

DL Configuration	BCS-ID	UL Configurations	Bands	Carrier bandwidths [MHz] or referenced BCs	Max aggregated BW [MHz]
n1A-n3(2A)-n41A-n78A	0	CA_n1A-n3A, CA_n1A-n41A, CA_n1A-n78A, CA_n3A-n41A, CA_n3A-n78A, CA_n41A-n78A	n1	[5,10,15,20,25,30,40,50]	
			n3	CA_n3(2A)_BCS0	
			n41	[10,15,20,30,40,50,60,80,90,100]	
			n78	[10,15,20,25,30,40,50,60,70,80,90,100]	





**ERICSSON**