Vibe Authoring with Oxygen AI Positron 8.0

Sorin Carbunaru
Software Developer, Syncro Soft
sorin_carbunaru@oxygenxml.com

An Oxygen XML webinar. © 2026 Syncro Soft SRL. All rights reserved.

Ruffly the Agenda

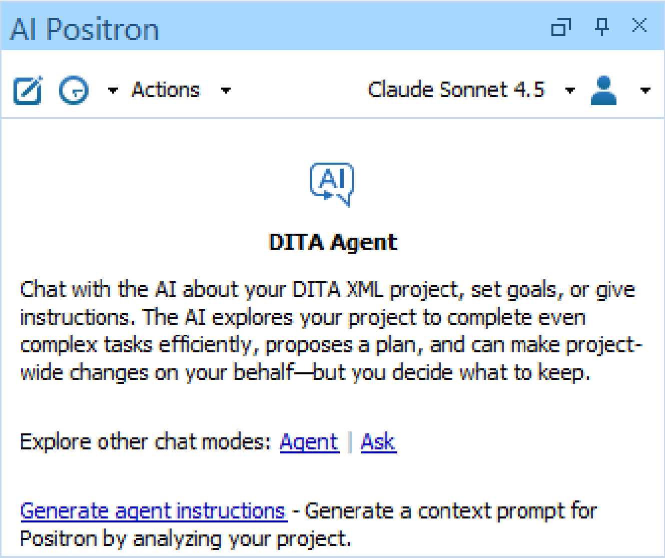
- What we’re digging into
- On-leash walk (screenshots & explanations)
- Off-leash run (live demo)
- The road ahead
- Q&A

What We’re Digging Into

- Generate agent instructions
- Automatic validation and correction
- Attach files to edited message prompts
- AI-powered file creation based on templates
- Create pull requests using GitHub's MCP server
- Find precise matches in project and replace them
- Transform XML content using AI-generated stylesheets

Note: some features require Oxygen 28.

Generate Agent Instructions

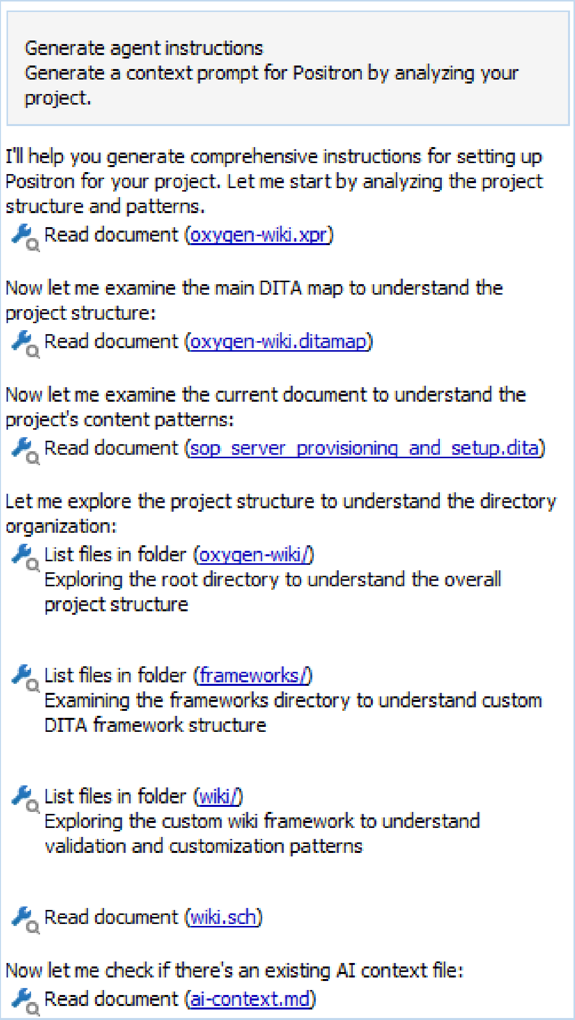


DITA Agent

Chat with the AI about your DITA XML project, set goals, or give instructions. The AI explores your project to complete even complex tasks efficiently, proposes a plan, and can make project-wide changes on your behalf—but you decide what to keep.

Explore other chat modes: Agent | Ask

Generate agent instructions - Generate a context prompt for Positron by analyzing your project.



Generate agent instructions
Generate a context prompt for Positron by analyzing your project.

I'll help you generate comprehensive instructions for setting up Positron for your project. Let me start by analyzing the project structure and patterns.

Read document (oxygen-wiki.xpr)

Now let me examine the main DITA map to understand the project structure:

Read document (oxygen-wiki.ditamap)

Now let me examine the current document to understand the project's content patterns:

Read document (sop_server_provisioning_and_setup.dita)

Let me explore the project structure to understand the directory organization:

List files in folder (oxygen-wiki/)
Exploring the root directory to understand the overall project structure

List files in folder (frameworks/)
Examining the frameworks directory to understand custom DITA framework structure

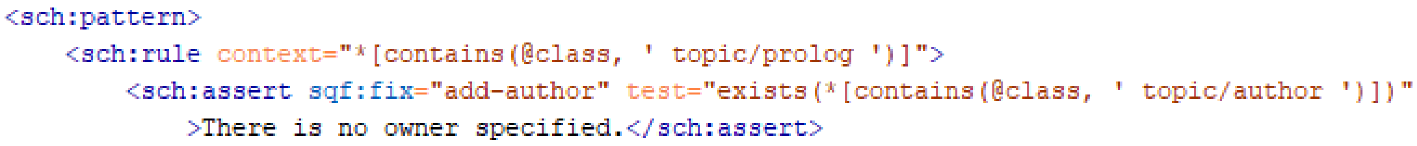
List files in folder (wiki/)
Exploring the custom wiki framework to understand validation and customization patterns

Read document (wiki.sch)

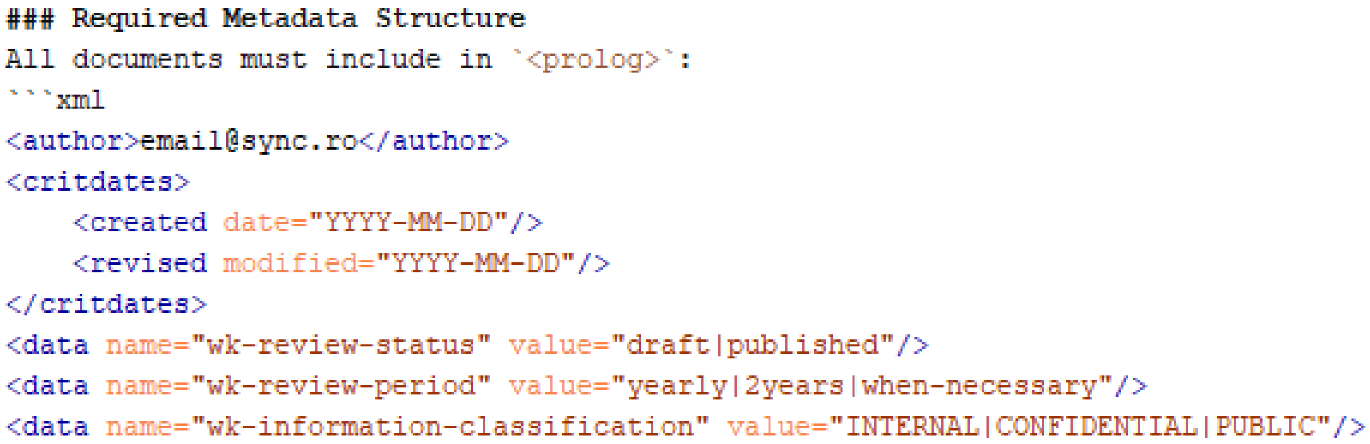
Now let me check if there's an existing AI context file:

Read document (ai-context.md)

Generate Agent Instructions

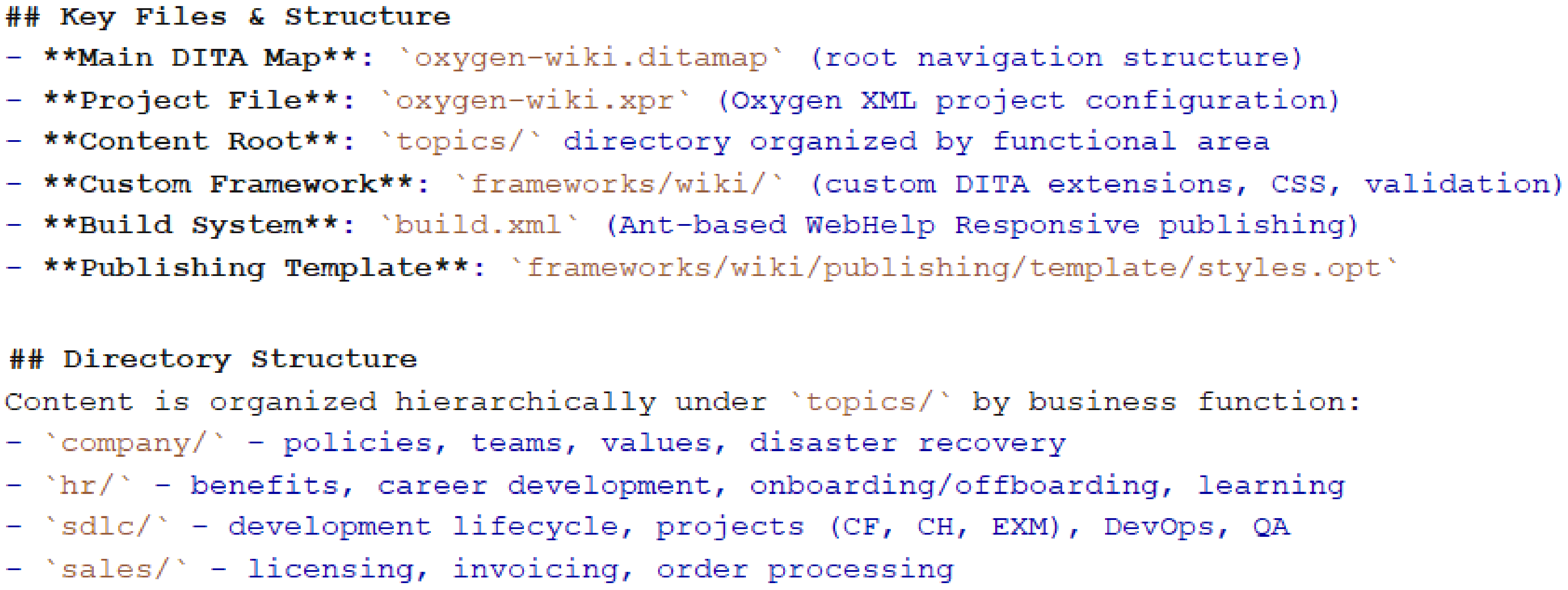


```xml
<sch:pattern>
    <sch:rule context="*[contains(@class, ' topic/prolog ')]">
        <sch:assert sqf:fix="add-author" test="exists(*[contains(@class, ' topic/author ')])"
            >There is no owner specified.</sch:assert>
```



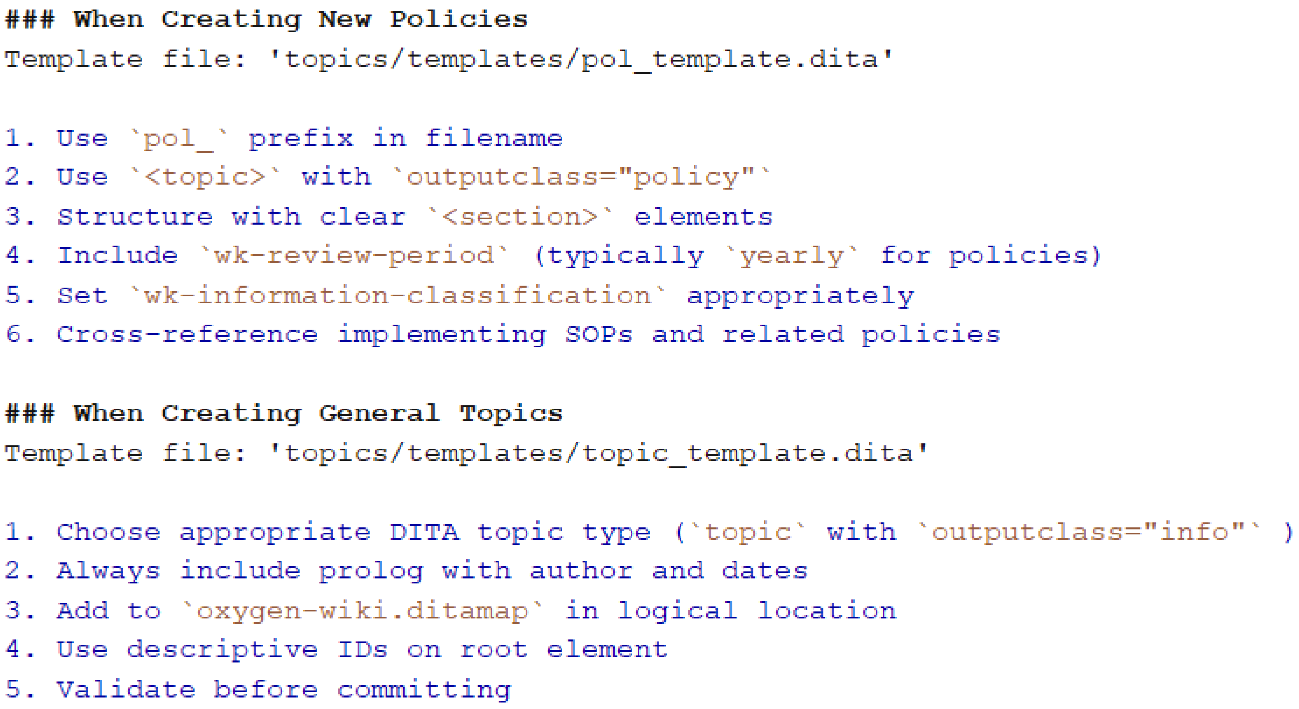
~~~
### Required Metadata Structure
All documents must include in `<prolog>`:
```xml
<author>email@sync.ro</author>
<critdates>
    <created date="YYYY-MM-DD"/>
    <revised modified="YYYY-MM-DD"/>
</critdates>
<data name="wk-review-status" value="draft|published"/>
<data name="wk-review-period" value="yearly|2years|when-necessary"/>
<data name="wk-information-classification" value="INTERNAL|CONFIDENTIAL|PUBLIC"/>
~~~

Generate Agent Instructions



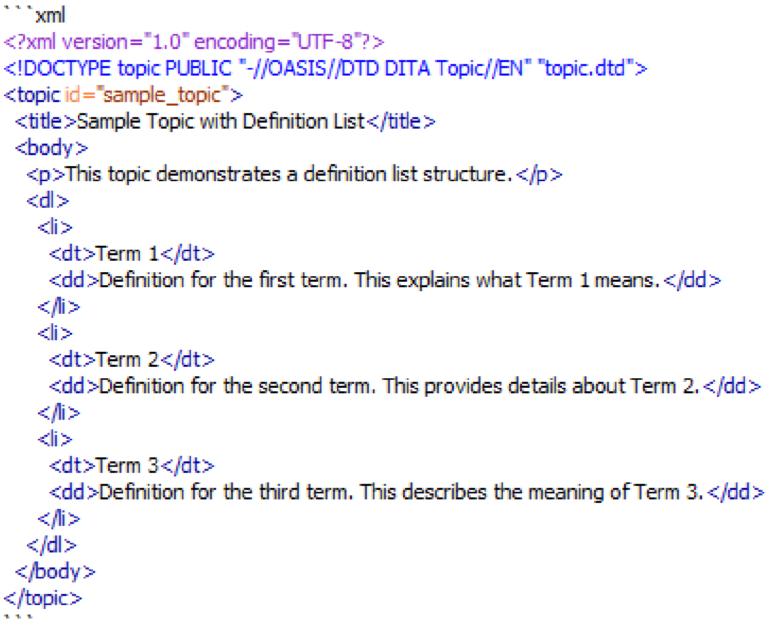
```
## Key Files & Structure
- **Main DITA Map**: `oxygen-wiki.ditamap` (root navigation structure)
- **Project File**: `oxygen-wiki.xpr` (Oxygen XML project configuration)
- **Content Root**: `topics/` directory organized by functional area
- **Custom Framework**: `frameworks/wiki/` (custom DITA extensions, CSS, validation)
- **Build System**: `build.xml` (Ant-based WebHelp Responsive publishing)
- **Publishing Template**: `frameworks/wiki/publishing/template/styles.opt`

## Directory Structure
Content is organized hierarchically under `topics/` by business function:
- `company/` - policies, teams, values, disaster recovery
- `hr/` - benefits, career development, onboarding/offboarding, learning
- `sdlc/` - development lifecycle, projects (CF, CH, EXM), DevOps, QA
- `sales/` - licensing, invoicing, order processing
```

Generate Agent Instructions

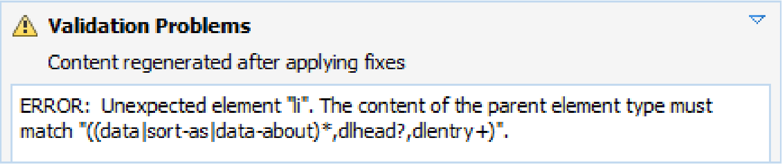


```
### When Creating New Policies
Template file: 'topics/templates/pol_template.dita'

1. Use `pol_` prefix in filename
2. Use `<topic>` with `outputclass="policy"`
3. Structure with clear `<section>` elements
4. Include `wk-review-period` (typically `yearly` for policies)
5. Set `wk-information-classification` appropriately
6. Cross-reference implementing SOPs and related policies

### When Creating General Topics
Template file: 'topics/templates/topic_template.dita'

1. Choose appropriate DITA topic type (`topic` with `outputclass="info"` )
2. Always include prolog with author and dates
3. Add to `oxygen-wiki.ditamap` in logical location
4. Use descriptive IDs on root element
5. Validate before committing
```

Automatic Validation And Correction



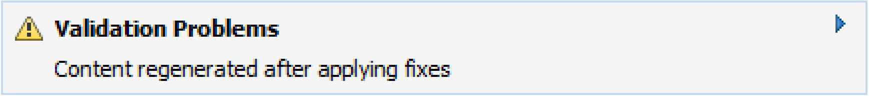
~~~
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="sample_topic">
  <title>Sample Topic with Definition List</title>
  <body>
    <p>This topic demonstrates a definition list structure.</p>
    <dl>
      <li>
        <dt>Term 1</dt>
        <dd>Definition for the first term. This explains what Term 1 means.</dd>
      </li>
      <li>
        <dt>Term 2</dt>
        <dd>Definition for the second term. This provides details about Term 2.</dd>
      </li>
      <li>
        <dt>Term 3</dt>
        <dd>Definition for the third term. This describes the meaning of Term 3.</dd>
      </li>
    </dl>
  </body>
</topic>
```
~~~



Validation Problems

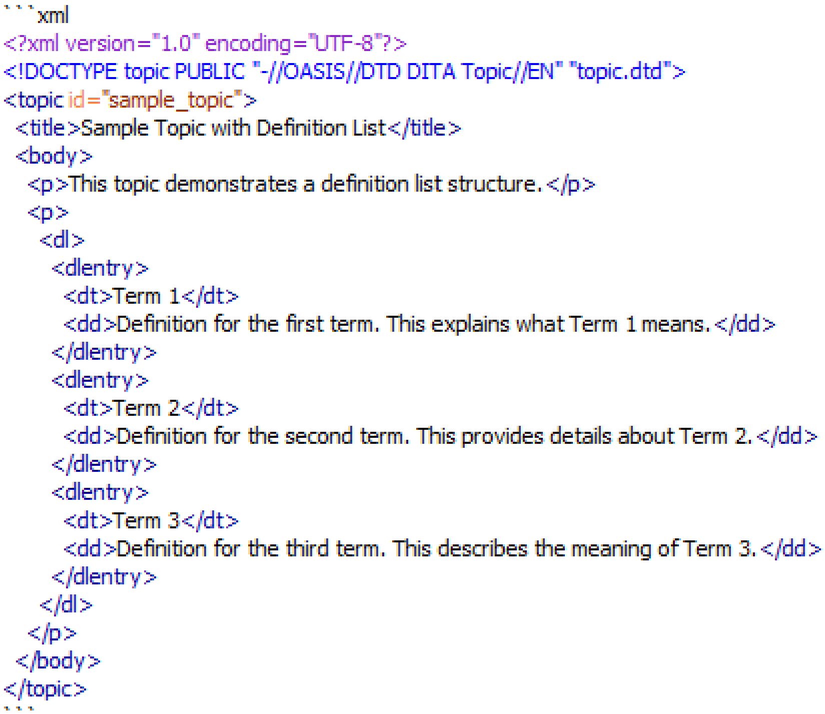
Content regenerated after applying fixes

ERROR: Unexpected element "li". The content of the parent element type must match "((data|sort-as|data-about)*,dlhead?,dlentry+)".



Validation Problems

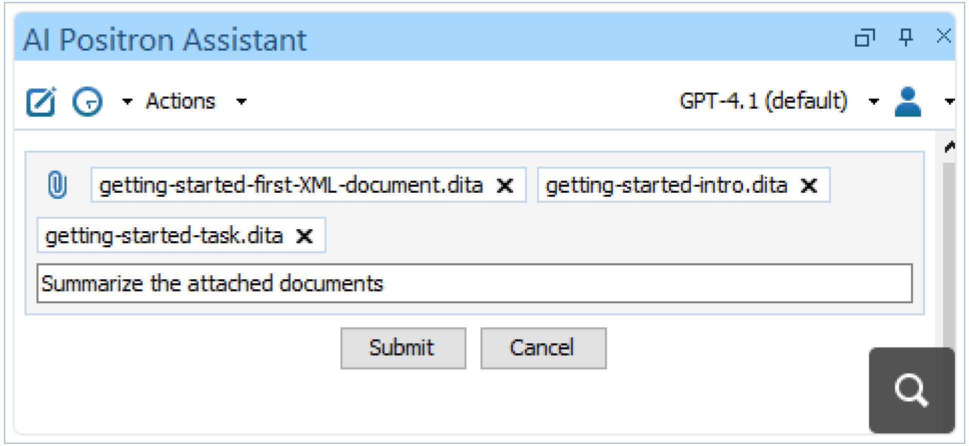
Content regenerated after applying fixes



~~~
```xml
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE topic PUBLIC "-//OASIS//DTD DITA Topic//EN" "topic.dtd">
<topic id="sample_topic">
  <title>Sample Topic with Definition List</title>
  <body>
    <p>This topic demonstrates a definition list structure.</p>
    <p>
      <dl>
        <dlentry>
          <dt>Term 1</dt>
          <dd>Definition for the first term. This explains what Term 1 means.</dd>
        </dlentry>
        <dlentry>
          <dt>Term 2</dt>
          <dd>Definition for the second term. This provides details about Term 2.</dd>
        </dlentry>
        <dlentry>
          <dt>Term 3</dt>
          <dd>Definition for the third term. This describes the meaning of Term 3.</dd>
        </dlentry>
      </dl>
    </p>
  </body>
</topic>
```
~~~

Attach Files To Edited Message Prompts

- We all need second chances... and so do our prompt attachments.



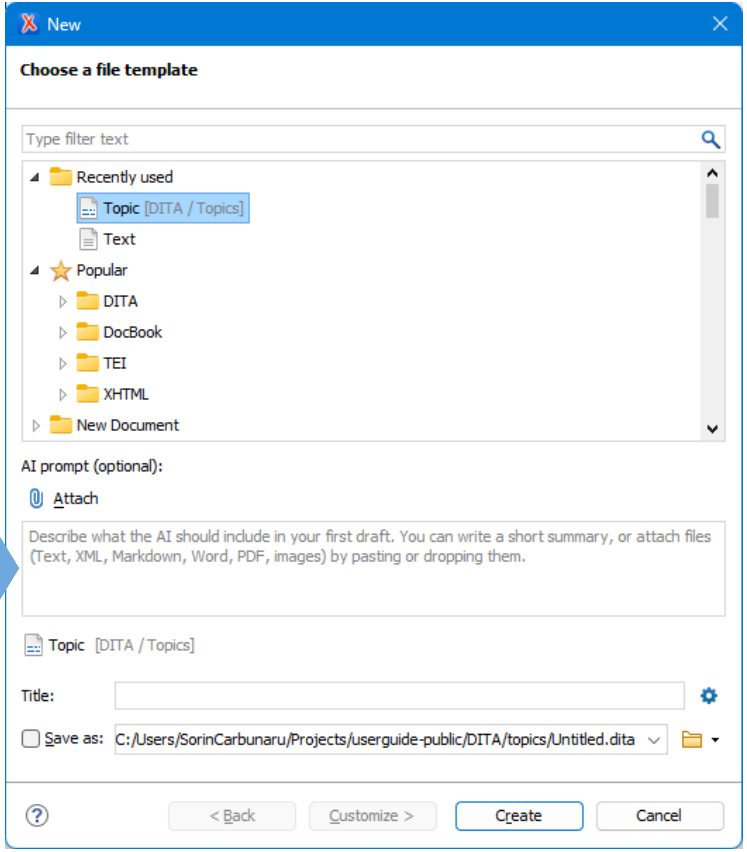
getting-started-first-XML-document.dita ✕ getting-started-intro.dita ✕ getting-started-task.dita ✕

Summarize the attached documents

Demo Time!

AI-Powered File Creation Based On Templates

- Probably the most intuitive way to create AI-generated documents exactly where you want them.
- Give instructions and attach resources (including screenshots).



Describe what the AI should include in your first draft. You can write a short summary, or attach files (Text, XML, Markdown, Word, PDF, images) by pasting or dropping them.

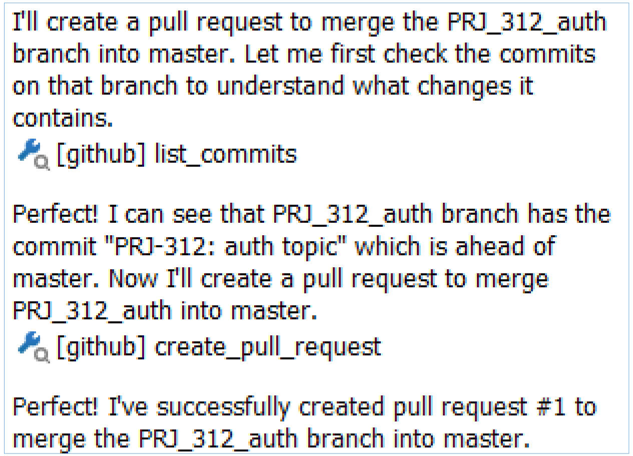
Topic [DITA / Topics]

Title:

☐ Save as: C:/Users/SorinCarbunaru/Projects/userguide-public/DITA/topics/Untitled.dita

Create Pull Requests Using GitHub's MCP Server

- The official GitHub MCP Server connects Positron to GitHub's platform.



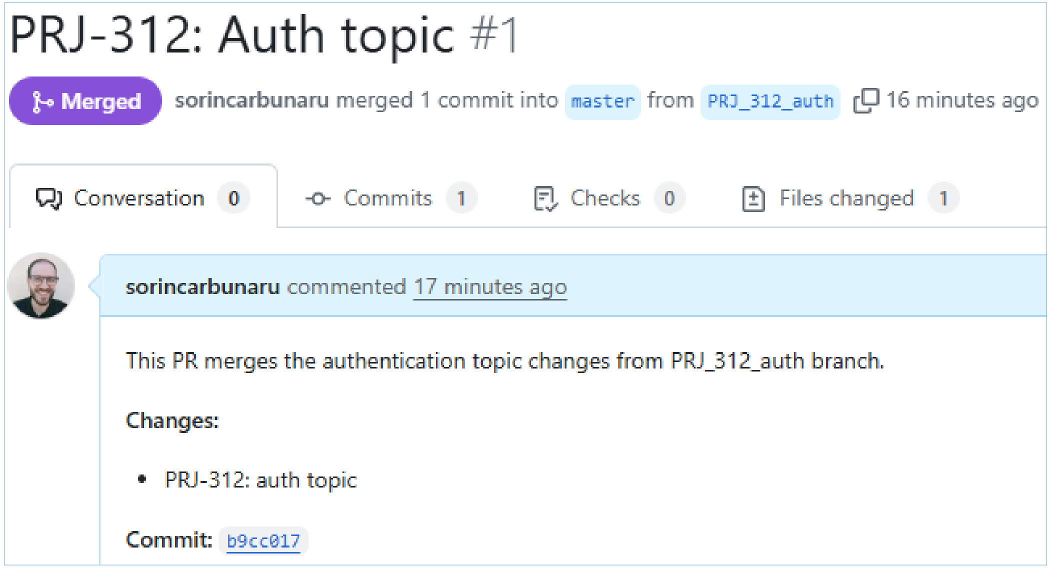
I'll create a pull request to merge the PRJ_312_auth branch into master. Let me first check the commits on that branch to understand what changes it contains.

[github] list_commits

Perfect! I can see that PRJ_312_auth branch has the commit "PRJ-312: auth topic" which is ahead of master. Now I'll create a pull request to merge PRJ_312_auth into master.

[github] create_pull_request

Perfect! I've successfully created pull request #1 to merge the PRJ_312_auth branch into master.



PRJ-312: Auth topic #1

Merged **sorincarbunaru** merged 1 commit into `master` from `PRJ_312_auth` 16 minutes ago

sorincarbunaru commented 17 minutes ago

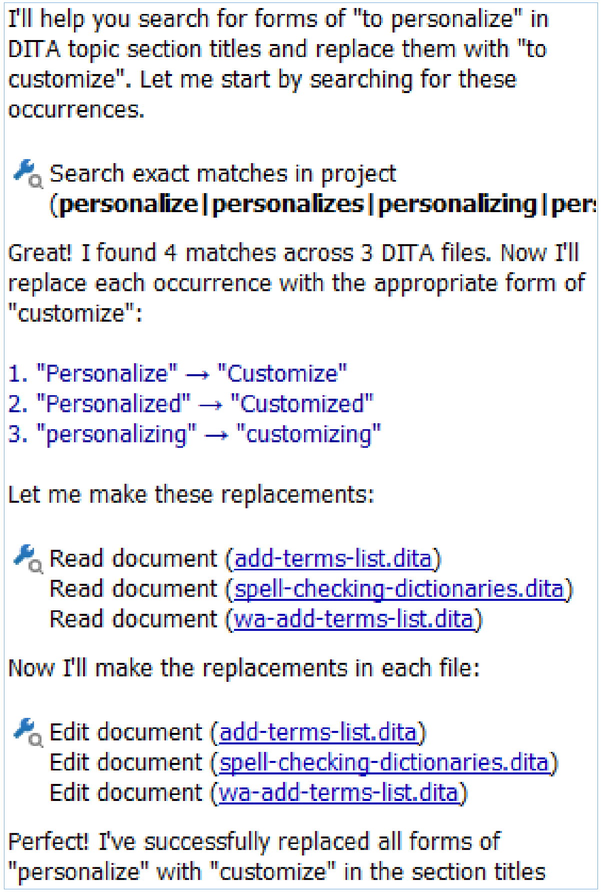
This PR merges the authentication topic changes from PRJ_312_auth branch.

Changes:

- PRJ-312: auth topic

Commit: `b9cc017`

Find Precise Matches In Project And Replace Them



I'll help you search for forms of "to personalize" in DITA topic section titles and replace them with "to customize". Let me start by searching for these occurrences.

Search exact matches in project (**personalize|personalizes|personalizing|per**

Great! I found 4 matches across 3 DITA files. Now I'll replace each occurrence with the appropriate form of "customize":

1. "Personalize" → "Customize"
2. "Personalized" → "Customized"
3. "personalizing" → "customizing"

Let me make these replacements:

Read document (add-terms-list.dita)
Read document (spell-checking-dictionaries.dita)
Read document (wa-add-terms-list.dita)

Now I'll make the replacements in each file:

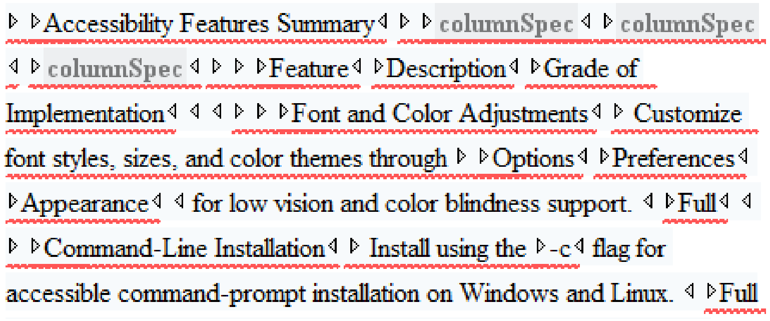
Edit document (add-terms-list.dita)
Edit document (spell-checking-dictionaries.dita)
Edit document (wa-add-terms-list.dita)

Perfect! I've successfully replaced all forms of "personalize" with "customize" in the section titles

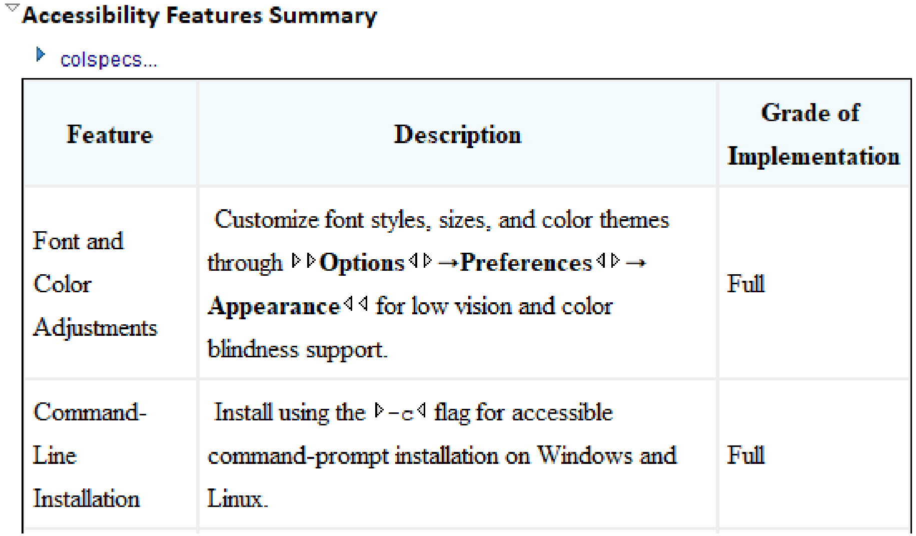
- The AI explores your project and executes precise replacements to boost efficiency.
- Editing requires selecting an ***agent* chat mode**.

Transform XML Content Using AI-Generated Stylesheets

- The AI can now use a tool that applies an XSL over an XML.



Accessibility Features Summary columnSpec columnSpec
columnSpec Feature Description Grade of
Implementation Font and Color Adjustments Customize
font styles, sizes, and color themes through Options Preferences
Appearance for low vision and color blindness support. Full
Command-Line Installation Install using the -c flag for
accessible command-prompt installation on Windows and Linux. Full

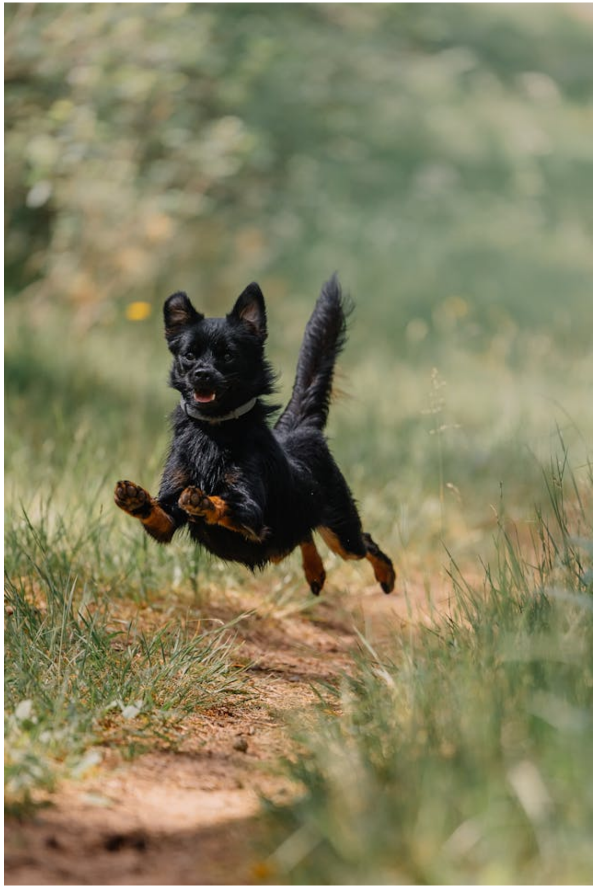


Accessibility Features Summary

colspecs...

Feature	Description	Grade of Implementation
Font and Color Adjustments	Customize font styles, sizes, and color themes through **Options** → **Preferences** → **Appearance** for low vision and color blindness support.	Full
Command-Line Installation	Install using the -c flag for accessible command-prompt installation on Windows and Linux.	Full

The Road Ahead



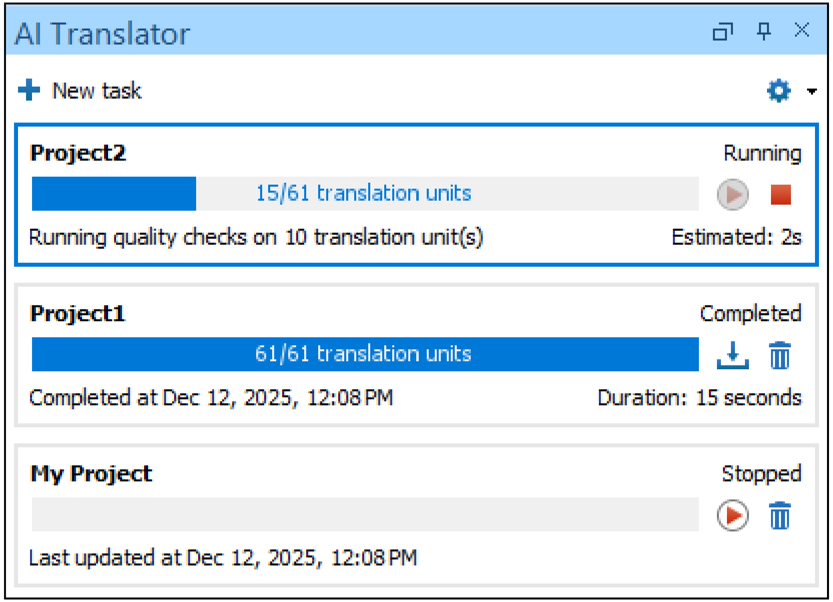
- Chat modes:
  - *Planner*
  - Create your own
  - Manage tools (enable/disable)
  - Tool to invoke other chat modes (e.g., Planner → Agent)
- Tool to run CLI commands
- Tool to rename, copy, or move resources
- Simplified change review flow
- Atomic change acceptance
- …and more—all to enhance the vibe authoring experience.

Helpful Resources

- Detailed list of what's new in Oxygen AI Positron
  https://www.oxygenxml.com/ai_positron/whats_new.html
- Oxygen AI Positron for Desktop Docs
  https://www.oxygenxml.com/doc/ug-positron-desktop/index.html
- Project
  https://github.com/sorincarbunaru/userguide-public

BONUS PREVIEW: Oxygen AI Translator Add-On

- Provides AI-powered translation capabilities for XLIFF.
- Works great with *Fluenta DITA Translation Add-on*.
- Features translation memory integration.
- Translation tasks use the AI connectors and models set up in *Oxygen AI Positron*.
- Translation tasks include a quality check that reviews the accuracy and quality of translations.
- If the quality check finds issues, the system can automatically try again using different models.
- Allows you to run multiple translation tasks in parallel.
- The translation settings are configurable.



Project2 Running
15/61 translation units
Running quality checks on 10 translation unit(s) Estimated: 2s

Project1 Completed
61/61 translation units
Completed at Dec 12, 2025, 12:08 PM Duration: 15 seconds

My Project Stopped
Last updated at Dec 12, 2025, 12:08 PM

THANK YOU!

Q&A time

Sorin Carbunaru

sorin_carbunaru@oxygenxml.com

© 2026 Syncro Soft SRL. All rights reserved.