

- [Edit article](#)
- [View stats](#)
- [View post](#)

The screenshot shows a Visual Studio Code interface with several tabs open. The main editor tab contains ABAP code for a class named CL_ABAP_PDF_IMPLEMENTATION. The code includes methods like set_text_color and set_forest_color, and various global declarations. A status bar at the bottom indicates the file is 100% complete. The right side of the screen shows a 'CodeLens' panel with several error and warning messages from the SAP ABAP Language Server, such as 'Global 00 Types(jav1) next pass: 100%', 'Run Syntax - 201_OPEN_ABAP_PDF', and 'Finding Issues - CLAS_CL_ABAP_PDF'. Below the editor, a terminal window shows the command 'rpg test' being run.

vscode screenshot during development

ABAP: Vibe coding a PDF generation library



Lars Hvam Petersen Senior SAP/ABAP Consultant at Heliconia Labs



January 14, 2026

Some [years ago](#), I got an idea to create a PDF generation library in ABAP. In order to,

- Run it on-stack, no extra communication or infrastructure
- Not have the data leave the system => data remains secure
- Something upgrade stable. SAP Script, SmartForms, Adobe Forms has been changing too much over the years
- Easy versioning, all the ABAP code is versioned
- Testable on Continuous Integration
- Easy modification/fixes from LLMs

So the other day I choose to let Claude Opus 4.5 via [GitHub Copilot](#) in [vscode](#) do the job for me, the result is open source at <https://github.com/open-abap/open-abap-pdf>

I did an initial setup for standalone ABAP development, but no other scaffolding or custom instructions. The magic prompt was:

"implement a library to build and render PDF files"

It did have a few problems with the ABAP syntax, but after some time trying various stuff it worked out. The first version did not work, so I asked it to:

"the demo program returns an invalid PDF, go fix, be very careful"

It found out that `sy-tabix` was used in a wrong spot, and the space character field is trimmed, after that it produced a valid PDF file.

[demo.pdf](#)

How to get started?

The [initial commit](#) in the [pull request](#) sets up continuous integration with standalone static analysis and unit tests. This provides a feedback mechanism to Humans and LLMs on what is wrong and gives both a chance to fix and test various approaches.

The library can be installed using [abapGit](#), but do consider if you need [multiple installations of the PDF library](#), in order to limit regressions. Or setup a continuous integration pipeline that renders and compares the generated PDF files after each code change.

LLMs can write ABAP, and many vendors provides a free plan, so its possible to get started!



Comments

Like Comment Share

Add a comment...

No comments, yet.
Be the first to comment.
[Start the conversation](#)



Lars Hvam Petersen

Senior SAP/ABAP Consultant at Heliconia Labs