PharmaSUG 2026 – Paper AI-240

DoxySAS: An End-to-End AI-Powered SAS Documentation Pipeline

Saikrishnareddy Yengannagari, Bristol Myers Squibb

ABSTRACT

What if your SAS macro library could document itself and answer programmer questions? DoxySAS makes this possible by combining AI-powered documentation generation with interactive chat capabilities, revolutionizing how pharmaceutical organizations manage macro documentation.

DoxySAS is a web-based tool that analyzes SAS macro code using Azure OpenAI to automatically generate comprehensive Doxygen-compatible headers including descriptions, parameter documentation, usage examples, and called macro references. The tool intelligently distinguishes positional from keyword parameters, processes bulk uploads simultaneously, and produces consistent, professional documentation in seconds rather than hours.

The complete pipeline transforms raw macros into searchable HTML documentation deployed via RStudio Connect, with AI chat injection enabling programmers to interactively query the macro library. This paper presents the architecture, implementation challenges, and real-world results from deploying DoxySAS across an enterprise SAS macro repository, demonstrating 95% reduction in documentation time while significantly improving accessibility and usability.

INTRODUCTION

In pharmaceutical organizations, well-documented SAS macro libraries are essential for regulatory compliance, traceability, and team productivity. Yet most teams accumulate hundreds of macros over years with inconsistent or missing documentation—and the "I'll document it later" mentality rarely materializes under deadline pressure.

Our GMTI team explored Doxygen—a mature, open-source documentation generator adapted for SAS by the SASjs project and introduced through Tom Bellmer's 2021 SAS Global Forum paper (Bellmer, 2021). Doxygen parses annotated SAS macro headers and produces professional, searchable HTML documentation. However, every macro needs a structured header with specific @ tags before Doxygen can process it. For hundreds of macros lacking headers, manually writing these would require weeks. What if AI could generate these headers automatically? That insight became DoxySAS—a complete pipeline from raw SAS code to searchable, AI-chatbot-enhanced documentation.

FROM DOXYGEN TO DOXYSAS

Doxygen (van Heesch, 1997) extracts specially formatted comments from source code and produces HTML documentation. While originally built for C/C++, Doxygen can process SAS files through its extension mapping: setting EXTENSION_MAPPING = sas=Java and FILE_PATTERNS = *.sas in the Doxyfile causes Doxygen to extract documentation from JavaDoc-style / ... */ comment blocks in SAS files. The SASjs project (core.sasjs.io) demonstrated this approach in production, and Bellmer's paper provided a practical adoption tutorial.

THE MANUAL HEADER PROBLEM

A properly formatted Doxygen header for a SAS macro includes @file, @brief, @details, @syntax, @usage, @param (with direction indicators), @calledmacros, @version, and @author. Figure 1 shows an example.

```
/**
 @file   mymacro.sas
 @brief  Merges two datasets by key variables

 @details
 Performs an inner merge of two input datasets using
 specified key variables with duplicate key handling.

 @syntax
 %mymacro(ds1=, ds2=, byvar=, out=);

 @param ds1=   [in]  First input dataset
 @param ds2=   [in]  Second input dataset
 @param byvar= [in]  BY variable(s) for the merge
 @param out=   [out] Output dataset name

 @version 1.0
 @author  Jane Smith
*/
```

Figure 1: Example Doxygen Header for a SAS Macro

Writing this requires reading the entire macro, understanding its purpose, classifying parameters by direction (input/output), identifying called macros, and composing descriptions and usage examples. For a complex macro with 15+ parameters, this takes 10–15 minutes. Across hundreds of macros, the effort is prohibitive. DoxySAS uses AI to generate these headers automatically.

ARCHITECTURE AND IMPLEMENTATION

DoxySAS is a Flask-based Python web application with three core modules.

- **Module 1: SAS Code Parser (doxysas.py)**

Before any AI call, the parser extracts structural information using regex to match the %MACRO statement, capturing the macro name and parameter list. It classifies each parameter as positional (no equals sign , “=”) or keyword (with “=”)—critical for generating correct usage syntax. The parser also detects called macros (filtering out 60+ standard SAS functions like %IF, %LET, %SYSFUNC,) and identifies inline function macros using heuristics based on name length and absence of output parameters.

- **Module 2: AI Documentation Engine (openai_helper.py)**

This module sends the extracted code structure plus source code to Azure OpenAI (GPT-4.1) via a carefully engineered prompt. The AI returns a JSON object with brief, details, usage, param_descriptions, and param_directions ([in], [out], [in,out]). A robust response cleaner handles common LLM output issues markdown fences, conversational prefixes, malformed JSON).

- **Module 3: Web Application (app.py)**

The Flask web app supports single file upload/paste, bulk upload (returned as ZIP), and batch directory processing. Bulk mode uses ThreadPoolExecutor with up to 25 concurrent workers. Files with existing Doxygen headers are auto-detected and skipped.

GENERATED HEADER FORMAT

Parameters are grouped into **Positional Parameters** and **Keyword Parameters** sections using HTML definition lists for proper rendering in Doxygen HTML output. Each parameter includes a direction indicator ([in], [out], [in,out]) determined by AI analysis, and default values are appended when present.

THE DOXYGEN BUILD PIPELINE

Once DoxySAS generates headers for all macros, Doxygen transforms these annotated files into a navigable HTML documentation site. The key Doxyfile configuration settings for SAS:

Tag	Setting	Purpose
EXTENSION_MAPPING	sas=Java	Use Java parser for .sas files
FILE_PATTERNS	*.sas	Process .sas file extensions
EXTRACT_ALL	YES	Process all files even without classes
ENABLE_PREPROCESSING	NO	Disable C preprocessor evaluation
SOURCE_BROWSER	YES	Include source code in output
GENERATE_TREEVIEW	YES	Create navigable tree-view sidebar

Table 1. Key Doxygen Configuration Settings for SAS

The EXTENSION_MAPPING = sas=Java setting is the critical trick: Doxygen's Java parser recognizes /... */ blocks, which is exactly the comment syntax used in SAS. Running "doxygen Doxyfile" produces a complete HTML site with file listings, individual documentation pages, source browser, tree-view navigation, and search functionality. The generated documentation can be deployed to any web server or documentation platform for team-wide access.

AI CHATBOT INTEGRATION

Static HTML documentation still requires programmers to know what they are looking for. A new team member might need to "merge two datasets and check for duplicates" but not know which macro does that. We addressed this by injecting an AI-powered chatbot directly into the Doxygen HTML site.

The implementation uses Retrieval-Augmented Generation (RAG): the Doxygen HTML files are parsed, chunked, and embedded into a vector store. A JavaScript chat widget is injected into the Doxygen output via the HTML_FOOTER configuration. When a user asks a question, the system retrieves the most relevant documentation chunks and feeds them as context to GPT-4.1, which generates a grounded answer. Example interactions include:

- **"Which macro creates a summary table?"** → Returns matching macros with usage examples.
- **"What parameters does %FREQ accept?"** → Retrieves the parameter list with descriptions.
- **"How do I merge LB with SUPPLB ?"** → Identifies the macro and provides sample code.
- **"What macros are called by %CREATE_REPORT?"** → Extracts the @calledmacros dependency list.

This transforms the documentation from a passive reference into an active assistant that understands the entire macro library.

IMPLEMENTATION CHALLENGES AND SOLUTIONS

- **Positional vs. Keyword Formatting.** Early versions generated usage examples with incorrect calling syntax. This was solved by classifying parameters during parsing and including type-specific instructions in the AI prompt.
- **AI Response Consistency.** LLM responses sometimes included markdown fences, conversational prefixes, or malformed JSON. A robust response cleaner was developed iteratively, and explicit prompt instructions ("Return ONLY valid JSON") reduced formatting issues.
- **Commented-Out Code.** Early called-macro detection picked up %macro(patterns inside comments, producing incorrect dependency lists. Stripping all comment blocks before scanning solved this.
- **Underscore Handling.** SAS variable names with underscores (e.g., _ALL_) were rendered as italics by Doxygen's Markdown parser. An escaping function converting _ to _ resolved this.
- **Inline Functions.** Macros like %C2C() that return values need different usage examples than standalone macros. A heuristic based on name length, absence of output parameters, and CALL SYMPUT patterns correctly classifies most inline functions.

RESULTS AND IMPACT

Metric	Before DoxySAS	With DoxySAS	Improvement
Time per macro (simple)	5 min	30 sec	10× faster
Time per macro (complex)	15 min	1 min	15× faster
50-macro batch	~8 hours	~30 min	16× faster
Full library (500+ macros)	Weeks	1 afternoon	~95% reduction
Documentation consistency	Variable	100% standardized	—
Documentation coverage	~30%	100%	—

Table 2. Quantitative Impact of DoxySAS

Beyond time savings, DoxySAS improved team onboarding (new members have searchable documentation from day one), code review efficiency (reviewers understand macros without reading source line by line), knowledge preservation (documentation survives team turnover), and maintenance confidence (programmers review documented behavior and dependencies before modifying macros).

CONCLUSION

DoxySAS demonstrates that AI can eliminate the documentation burden plaguing SAS programming teams. By combining deterministic code parsing with AI-powered content generation, Doxygen's proven HTML rendering, and AI chatbot integration, we built a pipeline that transforms raw, undocumented macros into a fully searchable, interactive documentation site.

Documentation generation is uniquely well-suited for AI: the input (source code) is structured, the output (Doxygen headers) follows a well-defined format, and the analysis required (understanding and describing code) is exactly what large language models excel at. For organizations maintaining large SAS

macro libraries, DoxySAS offers a path from documentation debt to documentation abundance—and with the chatbot, from passive documentation to active, conversational knowledge access.

REFERENCES

Bellmer, T. (2021). "Introduction to Doxygen." Proceedings of the SAS Global Forum 2021, Paper 1077-2021. Available at: https://communities.sas.com/t5/SAS-Global-Forum-Proceedings/Introduction-to-Doxygen/ta-p/726318

van Heesch, D. (1997). Doxygen: Generate documentation from source code. Available at: https://www.doxygen.nl/

SASjs Macro Core Library. Available at: https://core.sasjs.io/

OpenAI (2024). GPT-4.1 Model Documentation. Available at: https://platform.openai.com/docs/models

ACKNOWLEDGEMENTS

The author thanks the GMTI (Global Macros Tools and Innovation) team at Bristol Myers Squibb for their invaluable feedback and support during the development and deployment of DoxySAS. Special thanks to Lei Zhang and Tatiana Kazakova for their extensive and detailed feedback that shaped the tool, Derek Morgan & Nicole Thorne for their guidance, Karma Tarap, Maragret Wishart & Sydney Hyde(for helping with Open source tools), and Pooja Ghangare & Kamil Cichacki for their adoption and feedback. The author also thanks Tom Bellmer and Allan Bowe whose foundational work on Doxygen for SAS inspired this project.

CONTACT INFORMATION

Your comments and questions are valued and encouraged. Contact the author at:

Saikrishnareddy Yengannagari
Bristol Myers Squibb
saikrishnareddy.yengannagari@bms.com