

PIDE MCP: Connecting AI Agents to Isabelle^{*}

Kevin Kappelmann[Ⓢ]

School of Computer Science, University of Sheffield, United Kingdom
k.kappelmann@sheffield.ac.uk

Abstract. PIDE [7] is Isabelle’s asynchronous model of document-oriented proof processing. Its frontends include Isabelle’s jEdit and VSCode editors. PIDE MCP is a new frontend for AI agents: a customisable MCP server managing headless PIDE sessions. We describe its architecture, propose desiderata for programs connecting agents to proof assistants, and assess PIDE frontend-based approaches against them.

AI agents are increasingly connected to proof assistants, e.g., for autoformalisation. Several connectors have been developed, including LSP-based [5], REPL-based [1], and batch build-based [6] ones. We propose desiderata for such connectors, present a new PIDE frontend-based architecture and an implementation thereof, called PIDE MCP [3], and assess it against the desiderata. Early PIDE MCP users have reported developments in, e.g., cryptography, stochastic portfolio theory, domain theory, formal languages, and protocol verification, spanning several hundred thousand lines.¹ An extended preprint is available [4].

The following desiderata aim to promote applicability, efficiency, effectiveness, and usability of programs connecting AI agents and proof assistants: (a) *Completeness*: the program can access all document-managing operations and all information exposed by the assistant. (b) *Reactive feedback*: the program continuously and asynchronously receives feedback from the assistant. (c) *Editor-agnosticity*: the program is decoupled from editor frontends. (d) *Co-presence*: users can concurrently work on the program’s documents while having access to the assistant’s full interaction possibilities. (e) *Customisability*: users can modularly remove and add new tools that the program exposes to the agent.

Fig. 1 shows the single-sessioned PIDE frontend-based architecture. Requests are translated into session operations through PIDE’s public API. Using PIDE *snapshots*, operations can access available markup without blocking. *Markup* is information that annotates commands, e.g., proof states and timings. Modifying operations are forwarded to the prover, which asynchronously and continuously returns messages with markup, gradually exposing it in later snapshots.

PIDE MCP is a Model Context Protocol (MCP) server extending this architecture to multiple headless PIDE sessions. As a PIDE frontend, it can access all document-managing operations and information exposed by Isabelle (a) and receives feedback reactively (b). Being headless, it runs standalone and is editor-agnostic (c). It synchronises its documents’ contents with disk on every read and

^{*} Supported by Renaissance Philanthropy’s AI for Math Fund (Copilots for Isabelle).

¹ Reported on the Isabelle mailing list, the Isabelle Zulip, and in private communication.

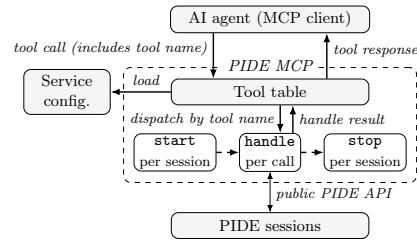
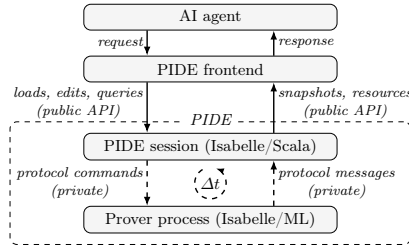


Fig. 1. PIDE frontend-based AI connection **Fig. 2.** Tool handling in PIDE MCP

edit, allowing users to work side by side with PIDE MCP (d). However, as session sharing among PIDE frontends is not yet supported, another frontend on the same files runs a separate session, duplicating computational effort. Customisability (e) is realised via Isabelle’s service mechanism [8] (cf. Fig. 2): tools are registered as services and loaded on startup without changing PIDE MCP’s code. Per-session lifecycle hooks support resource setup and cleanup for tools.

Instead of a headless PIDE session, one can connect to an editor-attached one, as done by Isabelle/Q [2], a jEdit plugin. As a PIDE frontend-based connector, it satisfies desiderata (a) and (b). Being tied to jEdit, it fails editor-agnosticity (c). Agent and human share jEdit’s PIDE session, realising co-presence (d). Customisability (e) is achievable as with PIDE MCP but is currently not implemented. Further viable approaches are LSP-, REPL-, and batch build-based, but only PIDE frontend-based approaches achieve completeness. Our preprint [4] gives details.

Promising future work includes realising general session sharing among PIDE frontends, thereby combining the strengths of PIDE MCP and Isabelle/Q, and evaluating which tools and feedback improve an agent’s efficiency and effectiveness via ablation studies, for which PIDE MCP’s design is well-suited.

References

1. Aniva, L., Sun, C., et al.: Pantograph: A Machine-to-Machine Interaction Interface for Advanced Theorem Proving, High Level Reasoning, and Data Extraction in Lean 4. In: Tools and Algorithms for the Construction and Analysis of Systems (2025)
2. AWS Labs: Isabelle/Q (2026), <https://github.com/awslabs/AutoCorrode>
3. Kappelmann, K.: Isabelle PIDE MCP (2026). <https://doi.org/10.5281/zenodo.22726529>
4. Kappelmann, K.: PIDE MCP: Connecting AI Agents to Isabelle (2026). <https://doi.org/10.5281/zenodo.21519364>
5. Kozyrev, A., Khramov, N., et al.: RocqStar: Leveraging Similarity-driven Retrieval and Agentic Systems for Rocq generation. In: Proceedings of the 25th International Conference on Autonomous Agents and Multiagent Systems (2026)
6. Urban, J.: 130k Lines of Formal Topology in Two Weeks: Simple and Cheap Autoformalization for Everyone? (2026). <https://doi.org/10.48550/arXiv.2601.03298>
7. Wenzel, M.: Asynchronous User Interaction and Tool Integration in Isabelle/PIDE. In: Interactive Theorem Proving (2014)
8. Wenzel, M.: The Isabelle System Manual (2026), <https://isabelle.in.tum.de/doc/system.pdf>