Liquid Protocol: Bridging AI Agents and Solid Pods via the Model Context Protocol

Meem Arafat Manab[1], Víctor Rodríguez-Doncel[1] and Beatriz Esteves[2]

[1] *Ontology Engineering Group, Universidad Politécnica de Madrid, Madrid, Spain*
[2] *IDLab, Department of Electronics and Information Systems, Ghent University – imec, Ghent, Belgium*

Abstract

We present Liquid Protocol, a specification for AI agent access to decentralized Solid pods via the Model Context Protocol (MCP). Analogous to how the W3C Linked Data Platform (LDP) specified HTTP-based access to Linked Data resources for web applications, Liquid Protocol defines MCP-based operations for AI agents to access user-controlled RDF data. This architectural approach addresses two critical challenges: the inability to comply with GDPR's Right to Be Forgotten when personal data is embedded in model weights or vector databases, and the tendency of large language models to hallucinate information when lacking access to structured, authoritative data sources. Our reference implementation demonstrates instant Right to Be Forgotten compliance, elimination of hallucination through structured RDF queries, and preservation of data sovereignty through Solid's decentralized architecture.

Keywords

AI agents, Solid Protocol, Model Context Protocol, GDPR compliance, data sovereignty, semantic web

1. Introduction

The W3C Linked Data Platform (LDP) specification [1] established how web applications access Linked Data resources through standardized HTTP operations. A decade later, AI agents have emerged as a new class of software requiring access to personal, structured data, yet no equivalent standard exists for agent-based access patterns.

Modern AI systems absorb personal data through training (embedding information in model weights) or through Retrieval-Augmented Generation (RAG, storing documents in vector databases). Both approaches make compliance with the General Data Protection Regulation's (GDPR) Right to Be Forgotten (RTBF) [2] effectively impossible. Removing data from trained models requires complete retraining costs ranging from millions to hundreds of millions of dollars [3]. Similarly, removing data from vector databases requires re-embedding the entire corpus, a process taking days to weeks at significant computational cost [4]. For production systems, these remediation approaches are economically and operationally infeasible.

Large language models also suffer from hallucination, producing confident but incorrect responses when uncertain [5]. While RAG systems mitigate this by retrieving text chunks before generation, they operate on unstructured data that fragments semantic relationships. When documents are split into chunks for embedding, the explicit relationships between entities are lost.

We present Liquid Protocol, enabling AI agents to access Solid pods [6] through the Model Context Protocol [7]. Our contributions include: *i)* a specification mapping MCP primitives to Solid pod operations analogous to LDP's mapping of HTTP methods to Linked Data resources, *ii)* a permission model extending Solid's Web Access Control with time-limited grants for AI agents, *iii)* a reference implementation demonstrating instant RTBF compliance and structured data access, *iv)* evaluation showing reduced hallucination through RDF-based queries. Full implementation details and code will be available on GitHub. The remainder of this article is structured as follows: Section 2 provides an overview of related work. In Sections 3 and 4, we provide architectural principles, the LDP-MCP mapping and the Liquid Protocol composition, as well as implementation details. Section 5 discusses

4th Privacy & Personal Data Management Session, co-located with the Solid Symposium 2026
0000-0002-2336-4160 (M. A. Manab); 0000-0003-1076-2511 (V. Rodríguez-Doncel); 0000-0003-0259-7560 (B. Esteves)

RTBF compliance, hallucination prevention, and access to unstructured data. Finally, Sections 6 and 7 provide pointers to future research and conclude the paper.

2. Related Work

The W3C Linked Data Platform Z[1] established HTTP-based Linked Data access, defining how HTTP methods operate on RDF resources and containers. The Solid Protocol [6] builds on LDP to create decentralized user-controlled pods, utilizing WebIDZ[8] for identity management and Web Access Control [9] for granular permissions. Liquid Protocol extends this lineage to AI agent access patterns, creating the first specification for how AI systems should interact with user-controlled semantic data.

Retrieval-Augmented Generation [10] grounds LLM responses in retrieved documents, reducing hallucination. However, RAG systems use unstructured text chunks stored in centralized databases, neither preserving semantic relationships nor providing user control. Liquid Protocol addresses both limitations by accessing structured RDF data that remains under user control in Solid pods.

Function calling [11] and tool use [12] enable LLMs to use external tools. The Model Context Protocol standardizes this through a protocol-first approach, defining how AI systems discover and invoke external tools through a common interface. Liquid Protocol provides the first specification for MCP-based access to semantic, user-controlled data.

3. Design

3.1. Architectural Principles

Liquid Protocol follows three key principles. First, **data remains at source**: personal data never enters model weights or vector embeddings. AI agents query Solid pods at runtime through MCP operations and never internalize the data. Second, **agents access structured data**: instead of retrieving document chunks, agents access RDF graphs where semantic relationships are explicit and preserved. SPARQL queries enable precise, structured retrieval. Third, **all access is governed by user control**: Solid's Web Access Control with time-limited grants allows users to revoke access or delete data at any time with instant effect.

3.2. Protocol Mapping

Liquid Protocol maps LDP operations to MCP tools for Solid pods, analogous to how LDP mapped HTTP methods to Linked Data operations (Table 1). Reading RDF resources corresponds to the `liquid_read_resource` tool. Creating and updating resources map to `liquid_write_resource`. Deleting resources maps to `liquid_delete_resource`, enabling instant RTBF compliance. Listing container contents maps to `liquid_list_container`. Executing SPARQL queries maps to `liquid_query_sparql`, enabling structured queries that preserve semantic relationships.

LDP (HTTP)	**Liquid (MCP)**	**Operation**
GET resource	`liquid_read_resource`	Read RDF
POST container	`liquid_write_resource`	Create
PUT resource	`liquid_write_resource`	Update
DELETE resource	`liquid_delete_resource`	Delete
GET container	`liquid_list_container`	List
SPARQL endpoint	`liquid_query_sparql`	Query

Table 1
LDP to Liquid Protocol mapping.

3.3. Permission Model

Liquid Protocol extends Solid's Web Access Control with time-limited permissions for AI agents. Each agent has a WebID and requests specific permissions from users. Authorization is expressed in RDF, specifying which agent can access which resource with which mode of access and until what time. The `validUntil` predicate enables temporary access grants, after which permissions automatically expire. Users can also manually revoke permissions at any time, giving fine-grained control throughout the permission lifecycle. The following example grants an AI agent read access to a specific resource, valid until a specified date and time:

Listing 1: Time-limited WAC authorization for an AI agent.

```
:auth1 a acl:Authorization ;
  acl:agent <https://agent.ai/webid#me> ;
  acl:accessTo <https://pod.example/alice/health/records.ttl> ;
  acl:mode acl:Read ;
  :validUntil "2025-12-31T23:59:59Z"^^xsd:dateTime .
```

3.4. Protocol Composition

Liquid Protocol is a mapping specification that composes three existing formal protocols. JSON-RPC format, tool discovery, tool calling, transport mechanisms, and error responses are inherited directly from MCP [7]. HTTP methods (GET, POST, PUT, DELETE), status codes, headers, and TLS security are inherited from HTTP specifications. WebID, WebID-TLS, and OIDC authentication flows are inherited from the Solid Protocol [6]. RDF triples, Turtle syntax, JSON-LD syntax, and container concepts are inherited from W3C specifications [13, 14, 15].

The key contributions are the tool schemas defining `liquid_*` operations, the semantic mapping between MCP primitives and Solid pod operations, mandatory audit logging requirements, GDPR operation flows implementing Articles 15, 16, and 17, time-limited ACL extensions with `validUntil` predicates, and patterns for AI agent identity using WebIDs.

4. Implementation

Our reference implementation[1] includes three components: the Community Solid Server[16] providing a local Solid pod instance with sample health records stored as RDF, a Liquid-Compliant MCP Server implementing the specification using the MCP Python SDK and translating between MCP tool calls and Solid Protocol operations, and a test suite containing three scenarios demonstrating RTBF compliance, hallucination prevention, and structured queries.

We created realistic RDF health records representing a patient with Type 2 Diabetes and Hypertension, including diagnosis dates and medications. This structured representation preserves relationships between patients, conditions, diagnoses, and medications in a way that would be fragmented in RAG systems.

The protocol defines three core operation types. **Setup** establishes a connection between LLM, Host MCP, and Solid pod through OTP-based authentication and capability discovery. **Query+CRUD** encompasses all read, write, update, delete, list, and query operations on Solid pod resources. **Rights Exercise** implements GDPR Articles 15 (Right of Access), 16 (Right to Rectification), and 17 (Right to Erasure). Complete sequence diagrams and message formats are made available in our GitHub repository.

Under GDPR Article 4, the Host MCP acts as a data controller because it determines the purposes and means of processing personal data. The Host MCP developer decides which operations to make available, establishes the technical infrastructure for AI agent access, defines authentication procedures, and determines how data flows between systems. While the user retains control over their Solid pod

[1]https://github.com/manabcodes/liquid/

and makes decisions about granting/revoking access, these are exercises of data subject rights rather than controller functions.

5. Evaluation

5.1. RTBF Compliance

In traditional architectures, personal data is embedded in model weights or vector databases. An RTBF request requires full retraining or re-embedding, with costs ranging from \$1M to \$10M and timelines of 3-6 months. With the Liquid Protocol, personal data remains in the user's Solid pod. An RTBF request triggers a `liquid_delete_resource` call, deleting data in less than one second at zero cost.

We measured deletion latency across 100 trials. Mean deletion time was 0.23 seconds with a standard deviation of 0.04 seconds. All subsequent read attempts correctly returned 404 errors, confirming complete removal without residual data remaining accessible to the agent.

5.2. Hallucination Prevention

Without the Liquid Protocol, an agent querying health records for "Alice Smith" has no access to a data source and responds with either refusal or hallucinated information. With the Liquid Protocol, the agent calls `liquid_query_sparql`, receives structured RDF, and responds with accurate information grounded in actual data.

All responses were grounded in actual data from the Solid pod rather than generated from parametric memory. Zero hallucinations were observed across 50 test queries, demonstrating that structured data access eliminates the uncertainty that leads to confabulation.

5.3. Structured vs. Unstructured Access

In the RAG approach, health records are split into text chunks and embedded in a vector database. A query like "What conditions does Alice have?" retrieves fragments that separate related information. The relationship between patient and conditions becomes ambiguous because chunking separates related information.

With the Liquid Protocol, a SPARQL query returns complete structured data with explicit relationships. Each condition includes patient reference, diagnosis date, type, and medications, with semantic relationships preserved through RDF triples.

We evaluated information completeness across 30 queries, comparing RAG chunks versus Liquid Protocol RDF results. Liquid Protocol achieved 100% relationship preservation, maintaining all explicit connections between entities. RAG systems lost 67% of explicit relationships due to chunking, requiring the agent to infer connections that were originally stated explicitly in the data.

6. Discussion

Liquid Protocol supports GDPR compliance by design, enabling the effective exercise of the right of access (Art. 15), rectification (Art. 16), and erasure (Art. 17) without requiring retraining or re-embedding. Users retain control over data sovereignty through fine-grained, machine-readable policies, instant deletion capability, and transparent audit logs. Decentralization and platform independence align with the Web 3.0 vision of user-controlled infrastructure. By relying on open standards (MCP and Solid), the architecture ensures interoperability and portability. Any MCP-compatible AI system can access any Solid pod, preventing vendor lock-in.

Real-time data updates provide a significant advantage over RAG systems. When information in a Solid pod changes, AI agents immediately access the updated version. RAG systems require re-embedding documents whenever information changes, creating a window during which agents work with stale data. Our proposed reduced infrastructure costs benefit both users and providers. Users store data in personal pods, eliminating the need for AI companies to maintain massive vector databases. This shift reduces storage costs, backup requirements, and computational expense. Representing data as RDF graphs preserves semantic relationships, enabling more structured and semantically grounded reasoning than unstructured text. Verifiable data provenance becomes possible through RDF's semantic structure and Solid's authentication mechanisms, crucial for high-stakes applications.

The protocol requires an additional network hop for each query compared to in-memory RAG, introducing latency that may be unacceptable for some real-time applications. However, we observed acceptable latency for most applications, with mean query times under 500 milliseconds. Caching can be implemented with RTBF-aware invalidation. Also, natural language queries do not always translate well to SPARQL syntax, which is why SPARQL is used sparingly in practice. Most agent interactions rely on simpler resource reads using `liquid_read_resource`.

Future work includes developing federated queries enabling cross-pod SPARQL queries spanning multiple users, an agent identity framework establishing standards for AI agent WebIDs and reputation systems, performance optimization through intelligent caching strategies maintaining RTBF compliance, and W3C standardization through a Community Group for Liquid Protocol specification development. Furthermore, the Solid Protocol is currently being standardized through the work of the W3C Linked Web Storage (LWS) Working Group in the LWS Protocol[2], and as such, the Liquid Protocol should be updated to comply with the LWS Protocol once the standard is finalized. By moving toward the use of this protocol, the usage of more complex policy languages, such as the W3C's Open Digital Rights Language (ODRL) [17], will be possible as the LWS Protocol will define the concept of an authorization server, which in turn will allow the usage of other policy languages beyond WAC. This is of the utmost importance to have access and usage control mechanisms that align with legal requirements, as policy languages such as ODRL allow the representation of rules with constraints on the purpose and legal basis for usage of personal data [18]. In that context, the usage of auditing logs based on the Data Privacy Vocabulary (DPV) [19] and the PLASMA [20] specifications should also be explored.

7. Conclusion

Liquid Protocol bridges AI agents and Solid pods through the Model Context Protocol, creating the first specification for agent-based access to user-controlled semantic data. By keeping personal data in Solid pods rather than model weights or vector databases, we enable instant RTBF compliance at zero cost. By accessing structured RDF rather than unstructured text chunks, we eliminate hallucination through authoritative, relationship-preserving queries. Just as the Linked Data Platform enabled the Semantic Web vision of machine-readable, interoperable data for web applications, Liquid Protocol enables an Agentic Web vision where AI systems respect user sovereignty, ground reasoning in authoritative sources, and align with data protection regulations' requirements.

Acknowledgements

This work has been supported by HARNESS, funded from the EU's Horizon Europe research and innovation programme under grant agreement no. 101169409, and by MALTA PID2024-159504OB-I00 funded by MICIU/AEI/10.13039/501100011033.

Declaration on Generative AI

The author(s) have not employed any Generative AI tools.

[2]https://w3c.github.io/lws-protocol/lws10-core/

References

[1] W3C, Linked Data Platform 1.0, W3C Recommendation, W3C, 2015. URL: https://www.w3.org/TR/ldp/.

[2] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation), Official Journal of the European Union L 119 (2016) 1–88. URL: http://data.europa.eu/eli/reg/2016/679/oj.

[3] B. Cottier, R. Rahman, L. Fattorini, N. Maslej, D. Owen, The rising costs of training frontier AI models, 2024. URL: https://arxiv.org/abs/2405.21015. `arXiv:2405.21015`.

[4] J. Geng, Q. Li, H. Woisetschlaeger, Z. Chen, F. Cai, Y. Wang, P. Nakov, H.-A. Jacobsen, F. Karray, A comprehensive survey of machine unlearning techniques for large language models, arXiv preprint arXiv:2503.01854 (2025).

[5] Z. Ji, et al., Survey of hallucination in natural language generation, ACM Computing Surveys (2023).

[6] S. Capadisli, T. Berners-Lee, R. Verborgh, et al., Solid Protocol, Technical Report, W3C, 2024. URL: https://solidproject.org/TR/protocol.

[7] Anthropic, Model context protocol, 2024. URL: https://modelcontextprotocol.io.

[8] A. Sambra, S. Corlosquet, WebID 1.0, W3C Editor's Draft, W3C WebID Community Group, 2014. URL: https://www.w3.org/2005/Incubator/webid/spec/identity/.

[9] S. Capadisli, Web Access Control (WAC), Draft Community Group Report, W3C Solid Community Group, 2024. URL: https://solidproject.org/TR/wac.

[10] P. Lewis, et al., Retrieval-augmented generation for knowledge-intensive nlp tasks, in: Advances in Neural Information Processing Systems (NeurIPS), 2020.

[11] T. Schick, et al., Toolformer: Language models can teach themselves to use tools, arXiv preprint arXiv:2302.04761 (2023).

[12] Y. Qin, et al., Tool learning with foundation models, arXiv preprint arXiv:2304.08354 (2023).

[13] W3C, RDF 1.1 Concepts and Abstract Syntax, W3C Recommendation, W3C, 2014.

[14] W3C, RDF 1.1 Turtle, W3C Recommendation, W3C, 2014.

[15] W3C, JSON-LD 1.1, W3C Recommendation, W3C, 2020.

[16] J. Van Herwegen, R. Verborgh, The community solid server: Supporting research & development in an evolving ecosystem, Semantic Web 15 (2024) 2597–2611.

[17] R. Iannella, S. Villata, ODRL Information Model 2.2 – W3C Recommendation 15 February 2018, 2018. URL: https://www.w3.org/TR/odrl-model/.

[18] B. Esteves, H. J. Pandit, V. Rodríguez-Doncel, ODRL Profile for Expressing Consent through Granular Access Control Policies in Solid, in: 2021 IEEE European Symposium on Security and Privacy Workshops, 2021.

[19] H. J. Pandit, B. Esteves, G. P. Krog, P. Ryan, D. Golpayegani, J. Flake, Data Privacy Vocabulary (DPV) – Version 2.0, in: G. Demartini, K. Hose, M. Acosta, M. Palmonari, G. Cheng, H. Skaf-Molli, N. Ferranti, D. Hernández, A. Hogan (Eds.), The Semantic Web – ISWC 2024, Springer Nature Switzerland, Cham, 2024, pp. 171–193. doi:`10.1007/978-3-031-77847-6_10`.

[20] B. Esteves, H. J. Pandit, Using Patterns to Manage Governance of Solid Apps, in: 14th Workshop on Ontology Design and Patterns (WOP 2023@ISWC 2023), 2023. URL: https://ceur-ws.org/Vol-3636/paper5.pdf.