HTWK Leipzig University of Applied Sciences

Logical Topology

Florian Gudat

Version ea5e943, 2024-06-17

Table of Contents

Version

ea5e943, 2024-06-17

Editor

Florian Gudat

Module

Mastermodul (C533.2 Compulsory module)
https://modulux.htwk-leipzig.de/modulux/modul/6291

Module Supervisor

Prof. Dr.-Ing. Jean-Alexander Müller

Lecturer

Herr Prof. Dr. rer. nat. Andreas Both
Herr M. Sc. Michael Schmeißer

Institute

Leipzig University of Applied Sciences

Faculty

Computer Science and Media

Acronyms

ACL	Access Control List
ACP	Access Control Policy
API	Application Programming Interface
CRUD	Create, Read, Update and Delete
CSS	Community Solid Server
DNS	Domain Name System
DPC	Data Privacy Cockpit
DPV	Data Privacy Vocabulary
DPoP	Demonstration of Proof-of-Possession
ESS	Enterprise Solid Server
GDPR	General Data Protection Regulation
HTTPS	Hypertext Transfer Protocol Secure
HTTP	Hypertext Transfer Protocol
IBM	International Business Machines
IEC	International Electrotechnical Commission
IE	Information Engineering
IP	Internet Protocol
ISO	International Organization for Standardization
LTS	Long-Term Support
N/A	Not Applicable
ODRL	Open Digital Rights Language
OIDC	OpenID Connect
OSI	Open Systems Interconnection
RDF	Resource Description Framework

ROA	Resource-Oriented Architecture
SPARQL	SPARQL Protocol and RDF Query Language
ShEx	Shape Expressions
UI	User Interface
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
WAC	Web Access Control

Namespaces

This enumeration lists the prefixes and the associated namespace. It will be used as the standard syntax for RDF prefixes. When a prefix is followed by a colon symbol, the part after the colon can be appended to the namespace URI, thereby creating a new URI.

acl	http://www.w3.org/ns/auth/acl#
al	*dynamic* (see [Custom Vocabulary])
claim	urn:claim# (see [Custom Vocabulary])
ex	*example*
foaf	http://xmlns.com/foaf/0.1/
http	http://www.w3.org/2011/http#
interop	http://www.w3.org/ns/solid/interop#
ldp	http://www.w3.org/ns/ldp#
pim	http://www.w3.org/ns/pim/space#
rdfs	https://www.w3.org/2000/01/rdf-schema#
solid	http://www.w3.org/ns/solid/terms#
st	http://www.w3.org/ns/shapetrees#

The prefixes and namespaces enumerated above are applicable to diagrams, listings, and inline listings throughout the entirety of the document.

Notation

The diagrams in this document were generated using Asciidoctor Diagram 2.3.1[1] and its bundled PlantUML[2] version.

Unless otherwise specified, all framed diagrams will use the UML 2.5.1[3] standard, constrained by the limitations of PlantUML. As defined in the standard, the following abbreviations will be utilized to identify the type of UML diagram:

cmp component diagram

sd interaction diagram

stm state machine diagram

In addition to the UML abbreviation, the following abbreviations are used to identify non-UML diagrams:

dm data model diagram; The information structure is entirely based on RDF, and will be presented as entity-relationship diagrams in Clive Finkelstein's IE[4] notation, with some additional elements. The text in the double angle brackets will define the entity type (e.g., [Dataset], [Thing], …). The path labels indicate potential routes through the graph structure, while the number within the bracket indicates the branch that has been taken.

wbs work breakdown structure; This diagram is a decompositional diagram[5], intended for use in hierarchical structures, originally designed as a project management tool. In this context, it is used to illustrate any kind of hierarchical structure.

All diagrams and figures presented in this work were created by the author. Any discrepancies have been highlighted in the corresponding figures.

[1] https://docs.asciidoctor.org/diagram-extension/latest/
[2] https://plantuml.com
[3] https://www.omg.org/spec/UML/2.5.1
[4] https://plantuml.com/en/ie-diagram
[5] https://plantuml.com/en/wbs-diagram

Chapter 1. Component Orchestration

The system components consist of three main parts: the client, the proxy, and the [Solid Provider] as the server. The client does not require any concept-specific logic and will be omitted from the system component view. Clients can access the public endpoint without any changes to the API. The [Solid Provider] should also remain unaffected and only be accessed through its HTTP APIs. When accessing the storages that exist in the [Solid Provider], it is important to divide them by ownership. This approach results in two different orchestrations of the system components: one where the captured data is held in trust, and another where the data is owned by the client.

1.1. Client as Data Owner

In this approach, the main entry, such as a proxy module manager or router, delegates the network request of a monitored resource or endpoint to the proxy module in charge. The module verifies that the resource exists in storage. If so, the request module data is appended to a resource container within that storage. Figure 1 illustrates this topology.

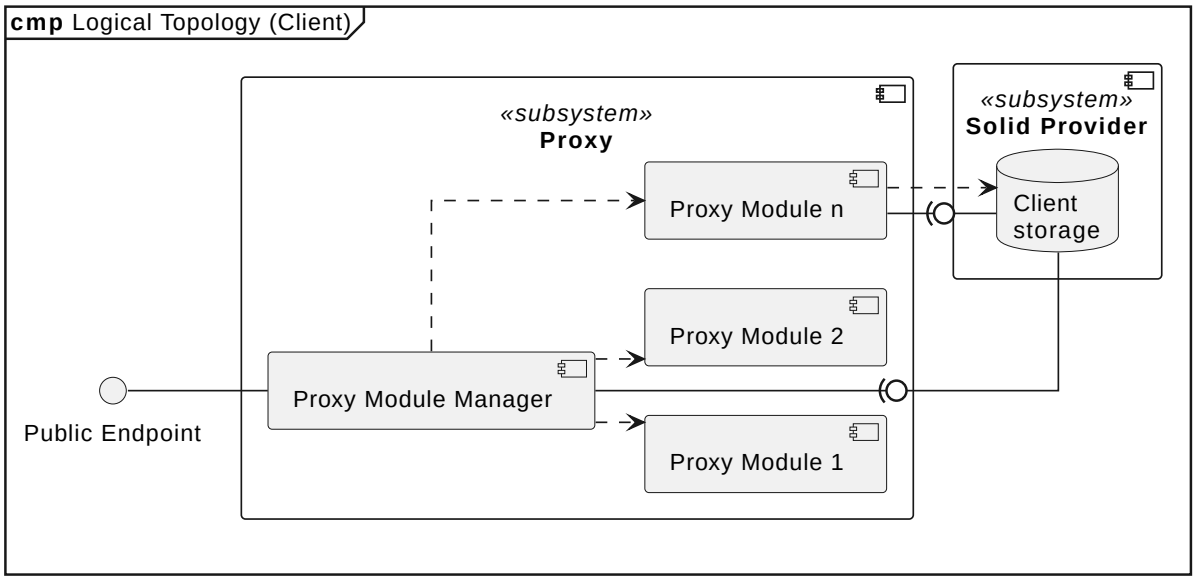

Figure 1. Component Diagram of the Logical Topology (Client)

1.2. Trustee as Data Owner

Similar to the previous approach, the main entry delegates the network request for a monitored resource or endpoint to the responsible proxy module. The module then verifies the resource's existence in storage. If applicable, append the request module data to a resource container in the module's storage. Figure 2 illustrates this topology, with the trustee as the data owner.

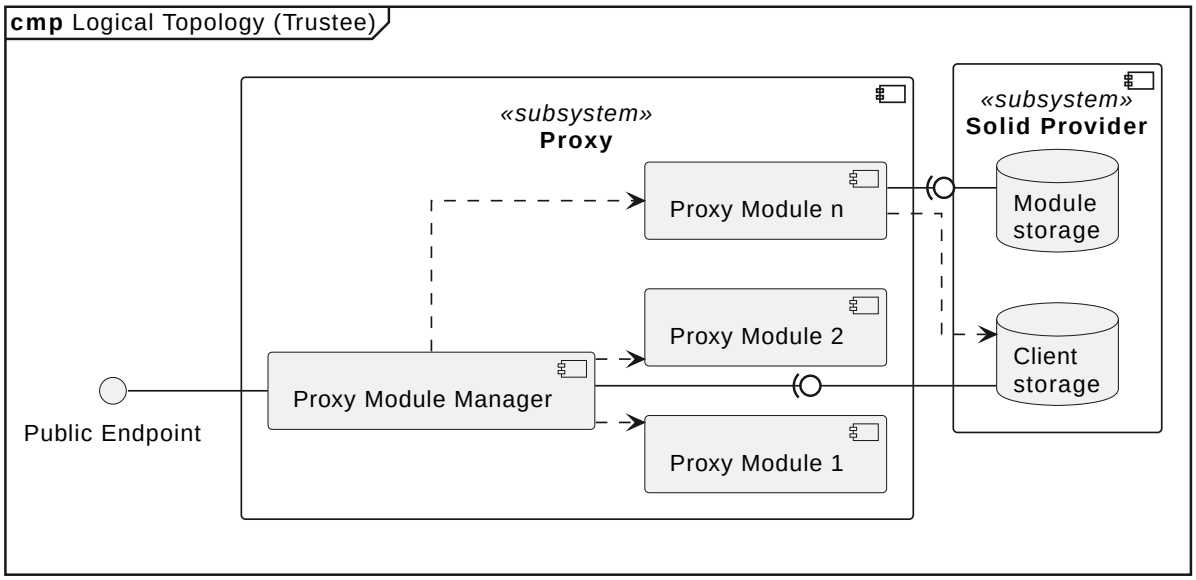

Figure 2. Component Diagram of the Logical Topology (Trustee)

Depending on whether the capturing strategy is permanent or registration-based, both approaches may require agent identity verification. This requirement is indicated by a dashed arrow between the proxy module and the storage in both figures.

The trustee-as-data-owner approach eliminates the need for ownership verification for every requested resource and avoids potential issues with the module's writing permissions to the client's storage. This approach is preferred over the client-as-data-owner approach, where the module relies on enough permissions.

Bibliography

Colophon

Built with Asciidoctor PDF 2.3.17, Asciidoctor Bibtex 0.9.0 and Asciidoctor Diagram 2.3.1 on linux-musl.

Repository	https://github.com/guddii/SEACT/tree/main
Revision	https://github.com/guddii/SEACT/commit/ea5e94359349f1fed086a61c7cfe386089ee4658
Build	https://github.com/guddii/SEACT/actions/runs/9552829495