

Towards Cyber-by-Design Security for Digital Twins: Implementing Zero Trust Architecture

Nizar Choubik
IRT SystemX
91120 Palaiseau, France
nizar.choubik@irt-systemx.fr

Abdelhadi Belfadel
IRT SystemX
91120 Palaiseau, France
abdelhadi.belfadel@irt-systemx.fr

Jérôme Letailleur
IRT SystemX
91120 Palaiseau, France
jerome.letailleur@irt-systemx.fr

Pierre-Marie Bajan
IRT SystemX
91120 Palaiseau, France
pierre-marie.bajan@irt-systemx.fr

Stephen Creff
IRT SystemX
91120 Palaiseau, France
stephen.creff@irt-systemx.fr

Amira Ben Hamida
IRT SystemX
91120 Palaiseau, France
amira.benhamida@irt-systemx.fr

Abstract—Industry 4.0 has positioned digital twins as key enablers for real-time monitoring, simulation, and intelligent decision-making across industrial environments. While their integration enhances operational efficiency, it also expands the attack surface, introducing critical cybersecurity challenges. This paper explores recent research efforts aimed at embedding Zero Trust Architecture into digital twin platforms, promoting a cybersecurity-by-design approach tailored to the needs of connected, dynamic industrial systems. The proof-of-concept is implemented within an Asset Administration Shell-based open-source software infrastructure dedicated to create, deploy and monitor Digital Twins.

Index Terms—Digital twins, Industry 4.0, Cybersecurity, Zero Trust Architecture

I. INTRODUCTION

Industry 4.0 has transformed the factory floor into a tightly coupled cyber-physical ecosystem. Sensors, Programmable-Logic Controllers (PLCs), and cloud gateways emit a continuous stream of operational data, while advanced analytics feed new set-points back to machines in real time. At the center of this loop stand *Digital Twins* (DTs) precise, continuously synchronized software replicas of physical assets, production lines, and even entire plants. Their promise lies in faster commissioning, predictive maintenance, and risk-free “what-if” simulations.

However, the very features that make Digital Twins valuable (such as rich data models, remote control capabilities, or open APIs) also significantly expand the attack surface. A recent review emphasized that insecure data pipelines, weak access control, and twin-to-twin interactions are major security concerns in Industry 4.0 environments [1]. Since DTs often act as bridges between Operational Technology (OT) and IT networks, a breach can propagate from cloud interfaces down to shop-floor devices, with potentially serious consequences.

As a result, *cybersecurity by design* has become a foundational principle in modern DT development. Rather than applying security patches after deployment, it is now essential to embed protection mechanisms from the start. These include:

- continuous authentication of every client (human or service);
- least-privilege authorization based on real-time context (e.g., role, device posture, or scope of request);
- fine-grained auditing to comply with standards like IEC 62443.

To help define and structure such requirements, the Digital Twin Consortium (DTC) introduced the *Capabilities Periodic Table (CPT)* [2]. This visual framework categorizes DT capabilities into six main domains: Data Services, Integration, Intelligence, User Experience (UX), Management, and Trustworthiness. It provides a technology-neutral vocabulary that guides organizations in defining functional and non-functional requirements for secure DT systems.

Among these domains, the *Trustworthiness* category is particularly relevant to cybersecurity. It includes essential capabilities such as privacy, device security, encryption, safety, resilience, and reliability. These aspects are critical for building secure and robust digital twin platforms that not only protect sensitive data but also ensure operational continuity and stakeholder confidence.

Traditional security models based on network perimeter defenses like firewalls, VLANs, or VPNs are no longer sufficient in this context. DTs are accessed by distributed users, cloud applications, and edge devices that do not necessarily reside on a trusted sub-net. To address this, the National Institute of Standards and Technology (NIST) advocates the adoption of a *Zero Trust Architecture* (ZTA), where every access request must be authenticated, authorized, and encrypted regardless of network origin [3].

This paper improves the high-level industrial DT Architecture platform discussed in [4], [5], and contributes to a cybersecurity-by-design and a practical ZTA implementation tailored to a digital twin platform aligned with the IEC 63278 [6] standard that defines the Asset Administration Shell (AAS) digital representation coming from Industry 4.0. This ZTA-based approach is realized with open-source components:

Keycloak as a Policy Decision Point (PDP), a lightweight proxy as Policy Enforcement Point (PEP), and the DT platform back-end services as secured back-end DTs resources. We show how security policies can be preserved under modern Keycloak versions, how the one-realm-per-service pattern improves isolation, and how the resulting architecture supports secure real-time operations.

II. RECENT RESEARCH

Recent studies show a growing interest in using Zero Trust principles to secure DT systems in Industry 4.0. Several research efforts explore how to integrate cybersecurity-by-design using DTs as part of the security architecture rather than just a passive digital representation.

In this regard, Kumar et al. [7] proposed a novel framework called Digital Twins-enabled Zero Touch Network (ZTN), which integrates smart contracts and explainable artificial intelligence (XAI). Their approach uses DTs not only to simulate industrial systems but also to enforce real-time security policies. It continuously verifies trust between devices and uses smart contracts to automate access control decisions in dynamic environments.

Chen et al. [8] focused on applying game theory to improve security decision-making in DT-based networks. Their work highlights how various stakeholders, such as devices, users, and administrators, can optimize their strategies using a reward-penalty model. This model enables dynamic risk-based access control, where DTs act as agents that monitor, report, and adjust security postures based on changing system behavior.

Paul and Rao [9] proposed a Zero Trust model tailored for smart manufacturing systems, emphasizing the role of DTs in enforcing security by design. Their approach treats every component (users, devices, and services) as untrusted by default, requiring strict authentication and authorization. A key contribution of their work is the use of DTs to simulate cyberattacks and perform penetration testing in virtual environments. This allows early identification of vulnerabilities without disrupting operations. They also introduce layered access controls and runtime monitoring to enforce least-privilege policies, helping to secure DT environments while supporting industrial compliance standards like IEC 62443.

Another relevant study by Russo [10] combines industrial demilitarized zones (IDMZ) with Zero Trust concepts. He suggests placing DTs at the center of the security model to bridge communication between operational technology (OT) and IT networks. This hybrid model allows DTs to monitor asset behavior and support the detection of anomalies or unauthorized access.

El-Hajj et al. [1] conducted a systematic review of the role of Digital Twins in industrial cybersecurity, with a strong emphasis on Zero Trust principles. Their study highlights how DTs can support continuous monitoring, identity-based access control, and real-time anomaly detection. Importantly, the authors advocate for embedding security at the design stage of DT systems, ensuring secure data flows, trust boundaries, and

policy enforcement from the start. They also discuss practical challenges like interoperability and data integrity, positioning DTs as active components in enforcing security-by-design in Industry 4.0 environments.

In their work on the secure usage of the AAS, the authors of [11] provide a detailed overview of best practices to harden AAS-based systems against cyber threats. Since the AAS acts as a standardized digital representation of industrial assets, it is often exposed to external interactions, making security a critical requirement. The paper identifies common attack surfaces such as unsecured communication channels, weak identity management, and unsafe access policies. It also outlines mitigation strategies including TLS-based encryption, fine-grained access control, and runtime monitoring. These recommendations are particularly relevant for platforms that, like ours, rely on AAS services to structure and expose DT functionality in compliance with Industry 4.0 standards.

While these contributions offer valuable insights into the theoretical and strategic integration of ZTA with DTs, they often focus on conceptual models, simulation environments, or specific security mechanisms. Few works provide a practical, modular implementation of ZTA tailored to the operational complexity of DT platforms. In particular, the integration of realm-based service isolation, automated policy injection, and standards-compliant interfaces such as those defined by the Industry 4.0 AAS standard is not addressed in existing literature.

Building on these insights, this work presents a practical implementation of Zero Trust Architecture for Digital Twin platforms. It emphasizes modular design, service isolation based on risk levels, and standards-compliant access control using open-source tools. The proposed approach demonstrates how fine-grained access control and automated policy enforcement can be embedded within DT architectures to support security, scalability, and interoperability in industrial environments.

III. ARCHITECTURE OVERVIEW

A. Architectural Design of a Digital Twin Platform

To ensure scalability, interoperability and security, our DT platform adopts a modular and standards-aligned architecture inspired by the reference model introduced in [4]. The platform integrates multiple components for data modeling, simulation, access control, and external system interoperability. Figure 1 presents an overview of the high-level functional decomposition.

The architecture is structured around three key domains:

- **DT Data Model Management and Connectivity with Physical Layer:** Depicted in a green color. This component comprises a robust data modeling layer that supports the creation, synchronization, and execution of digital twin models. These models are continuously updated using real-time data from physical assets through bi-directional integration protocols such as OPC UA and MQTT. Integration with IT systems (e.g., MES/ERP) is also supported, while a dedicated storage layer handles both structured and unstructured data.

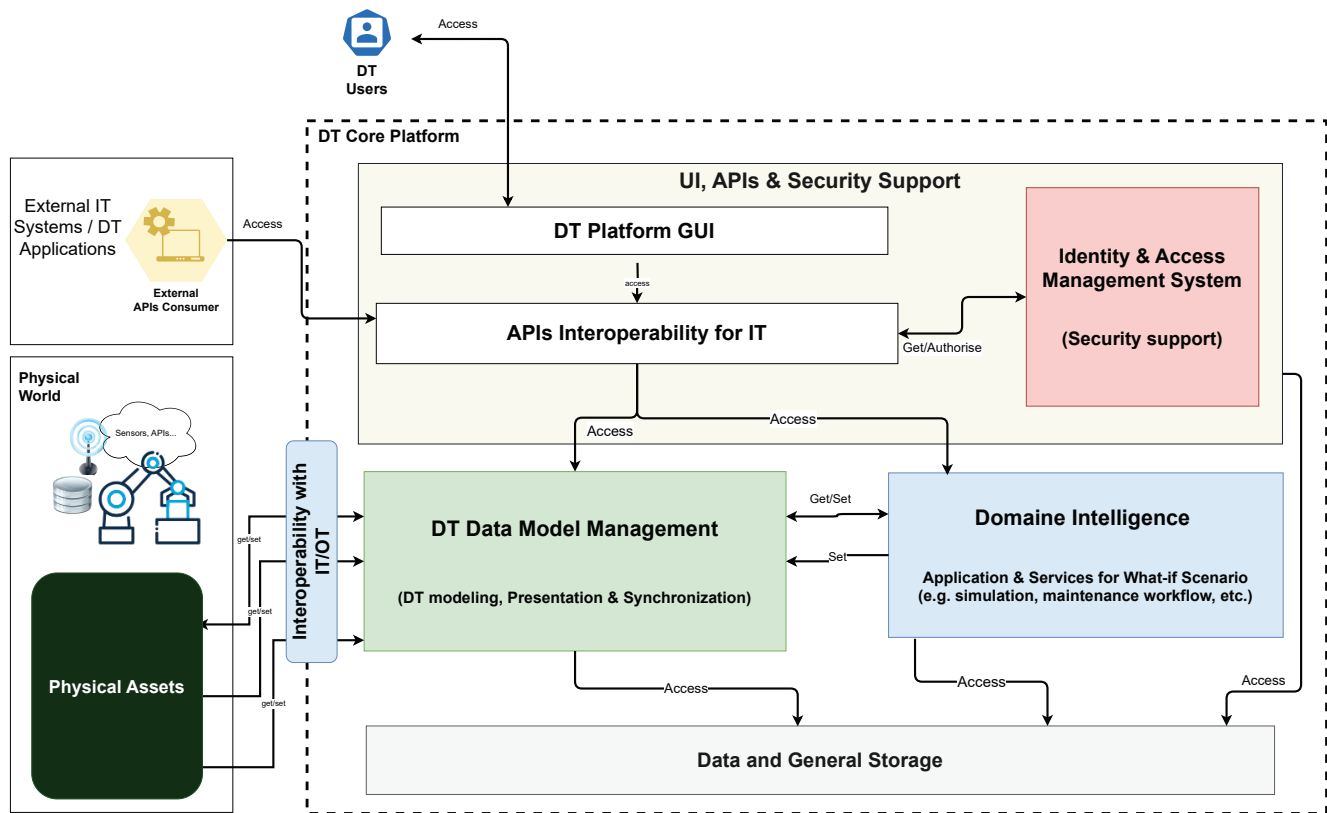


Fig. 1. High-level Digital Twin Platform Architecture adapted from [4].

- **Domain Intelligence, Applications, and What-If Scenario Services:** Depicted in a blue color. Built on top of the data modeling layer, the domain intelligence layer enables advanced analytics and decision support through what-if scenario workflows. This includes embedding expert knowledge, industry-specific applications, and services that allow users to simulate and evaluate different operational strategies or failure conditions, thereby improving planning, performance, and resilience.
- **UI, APIs & Security Support:** Depicted in Figure 1 in a yellow color. It Provides user-facing services such as GUI dashboards, analytics, and API interfaces for external IT systems. It includes an Identity and Access Management (IAM) system to enforce secure access and support Zero Trust principles.

This architecture follows the ISO 23247 reference model and supports the IEC 63278 Asset Administration Shell for standard-compliant representation of assets. Each component maps to one or more capability domains defined in the Digital Twin Consortium's CPT, ensuring alignment with industry best practices. In this regard, the "Security Support" depicted in Figure 1 in red color, that comprises Identity and Access Management system contributes to the Trustworthiness domain by enforcing authentication, access control, and secure policy evaluation. These elements align with Zero Trust principles, which we build upon in this work. In what follows, we

introduce a detailed version of this architecture, incorporating concrete ZTA components such as a Policy Decision Point (PDP), Policy Enforcement Point (PEP), and fine-grained role-based policies for secure digital twin operations.

B. Zero-Trust Foundations

: The latest draft of NIST SP 800-207 [3] defines a *Zero-Trust Architecture* (ZTA) as "a collection of concepts and component relationships designed to reduce uncertainty when enforcing least-privilege, per-request access decisions in information systems" [3]. In plain words, every request must prove *who* is calling, *what* it wants to do and *whether* it is allowed, no matter where the packet originates. A device sitting next to the server on the same VLAN is treated with the same suspicion as a laptop roaming on the public Internet.

: The NIST reference model depicted in Figure 2 splits traffic into two logical layers:

- **Data-plane:** raw application packets exchanged between a subject (user or service) and an enterprise resource;
- **Control-plane:** signals and decisions needed to decide whether the data-plane packet is legitimate.

A *Policy Enforcement Point* (PEP) intercepts every packet on the data-plane, asks the *Policy Decision Point* (PDP) on the control-plane, and forwards it only if the PDP replies *allow*. The PDP may further consult continuous-monitoring systems or threat feeds before reaching a verdict.

In modern industrial deployments that mix on-premise OT equipment, cloud APIs and vendor dashboards, validating every HTTP call in real time reduces lateral-movement opportunities once a single node is breached.

In the following sections, we present the technical architecture adopted for our Zero Trust implementation. In this design, **Keycloak** is responsible for both policy management and decision-making, acting as the **PDP**. A lightweight reverse proxy written in Python serves as the **PEP**, intercepting requests and querying Keycloak for access decisions. Finally, the **enterprise resources** including digital twin services built upon the AAS represent the protected targets of the system.

IV. TECHNICAL ARCHITECTURE FOR SECURING DIGITAL TWINS

A. Keycloak as Authorization Service

Keycloak is an open-source IAM platform developed by Red Hat. It provides a full set of features for authentication, single sign-on (SSO), identity federation, and fine-grained authorization services. One of the main reasons for choosing Keycloak in our industrial digital twin platform is its flexibility and open architecture. It supports standard protocols such as OpenID Connect and SAML, integrates with enterprise directories like LDAP and Active Directory, and is completely free to use. These features make it an ideal choice for building Zero Trust infrastructures in cost-sensitive industrial contexts.

In addition to its IAM capabilities, Keycloak includes a built-in **PDP**, which is a central element in implementing ZTA [3]. It supports different types of policies: role-based, attribute-based, and JavaScript-based policies, giving administrators the flexibility to define dynamic and context-aware access rules. These policies can be managed either manually through the web console or programmatically via its authorization APIs [12], [13].

Figure 3 illustrates how Keycloak implements its Authorization Services stack to perform PDP/PAP functionalities. The main components are as follows:

- 1) **Policy Enforcer.** A lightweight adapter installed on the resource server (e.g., Java EE, Spring Boot, Node.js). It intercepts protected HTTP requests, generates UMA permission tickets, and forwards them to Keycloak for decision-making.
In our implementation, this enforcer role is handled by a custom-built reverse proxy.
- 2) **Authorization Services.** Keycloak exposes REST endpoints compliant with the UMA (User-Managed Access) protocol:
 - `/token` – Clients send permission tickets to this endpoint. If access policies are satisfied, Keycloak returns a valid Requesting Party Token (RPT).
 - `/protection` – This endpoint allows administrators or services to register and manage protected resources, scopes, and policies dynamically.
- 3) **Policy Evaluation Engine.** This component evaluates the incoming requests using a combination of rules—role-

based, attribute-based, and custom JavaScript logic. It also supports plug-in extensions for custom decision strategies.

- 4) **Storage & Audit.** All authorization metadata : resources, scopes, roles and policies are stored persistently. Each access request and evaluation outcome is recorded in audit logs for compliance and forensic analysis [12].

B. Mapping ZTA to Technical Components

The core elements of ZTA are instantiated in our platform using open-source components as follows:

- **Policy Decision Point.** The PDP role is fulfilled by **Keycloak**, using its built-in *Authorization Services* [12]. It provides full support for UMA 2.0, allows for role and attribute-based access rules, and supports custom JavaScript policies. Administrators can manage policies dynamically through a documented REST API.
- **Policy Enforcement Point.** A reverse proxy developed in Flask acts as the PEP. It listens on a dedicated port in “catch-all” mode, intercepting all incoming traffic. Its responsibilities include validating access tokens, triggering the UMA flow with Keycloak, and forwarding only those requests that are explicitly authorized. This ensures that no service is reachable unless traffic passes through this enforcement layer.
- **Enterprise Resources.** The digital twin services, such as AAS registries, repositories, discovery endpoints or MQTT bridges, are deliberately not exposed to the network directly. They are bound only to local interfaces and are accessible exclusively through the PEP. This guarantees strict traffic mediation and eliminates any bypass path, fully aligning with ZTA principles [15].

C. Per-Request Control Flow

Regarding the decision cycle, Figure 4 depicts the control sequences following these steps:

- 1) A browser or machine client issues an HTTP request; the target URL actually points to the PEP.
- 2) If no valid token is present, the PEP redirects the user to Keycloak for login (standard OpenID Connect dance).
- 3) Keycloak authenticates (password, MFA, certificate, ...) and returns a short-lived access token plus a refresh token.
- 4) For each protected URL the PEP asks Keycloak’s “protection API” to evaluate the requested resource, HTTP method and scopes.
- 5) The PDP runs the policy engine and sends back *allow* or *deny*.
- 6) Happy path: the PEP proxies the call to the protected service.
- 7) If denied the user immediately receives HTTP 403; no DT endpoint ever sees the packet.

D. Manage Policies Injection and CI/CD

In our earlier implementation, we had already developed a Python-based deployment script to automate the configuration

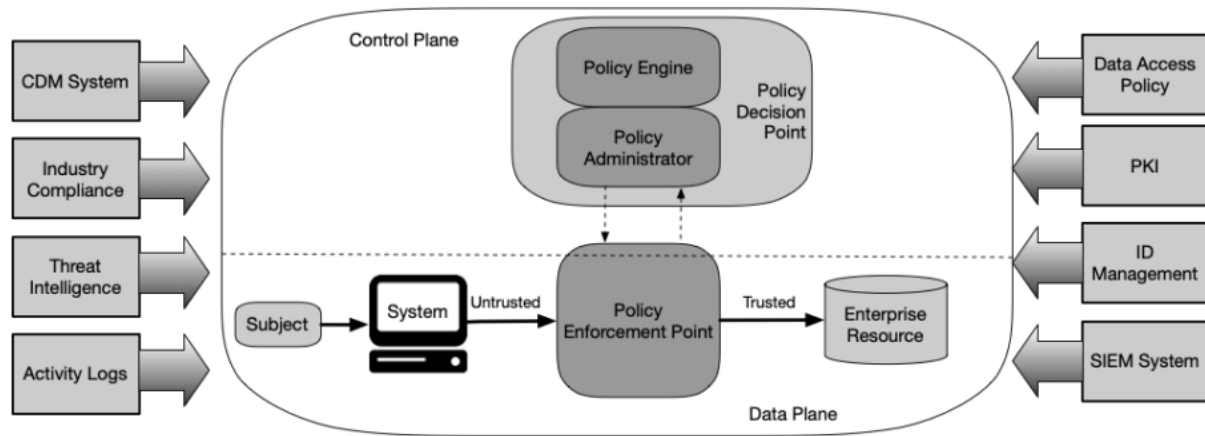


Fig. 2. NIST Zero-Trust reference model [3].

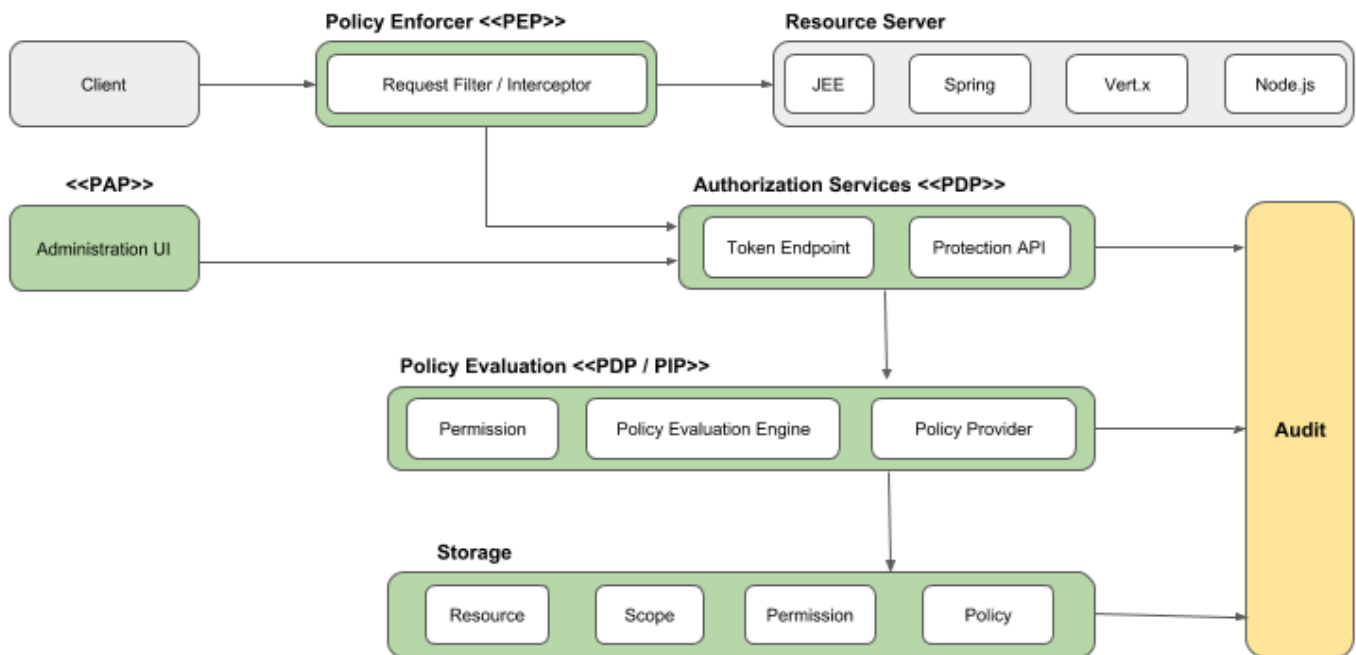


Fig. 3. Keycloak Authorization-Services stack (green = Keycloak component, grey = external caller) [14]

of our Keycloak realm. This script handled the creation of users, assignment of roles, registration of clients, and all other essential components needed for a secure digital twin platform. However, one critical part was still missing: the definition of security policies under the `authorizationSettings` section.

We initially tried to complete this by extending the realm export with custom inline JavaScript-based policies directly embedded in the JSON configuration. This seemed convenient and worked well with earlier versions of Keycloak. Unfortunately, starting from Keycloak version 18 and above, inline JavaScript is no longer supported in realm JSON imports due

to stricter security policies.

According to the official Keycloak Developer Guide, script-based policies must now be externalized. This means each JavaScript policy must be saved as a separate `.js` file and packaged into a JAR file. The JAR is then deployed as a custom provider, and scripting support must be explicitly enabled through the environment variable `"KC_FEATURES=scripts"` [13].

To comply with this constraint, we adapted our deployment process as follows:

- 1) We collected all JavaScript fragments used for authorization logic and placed them inside the folder

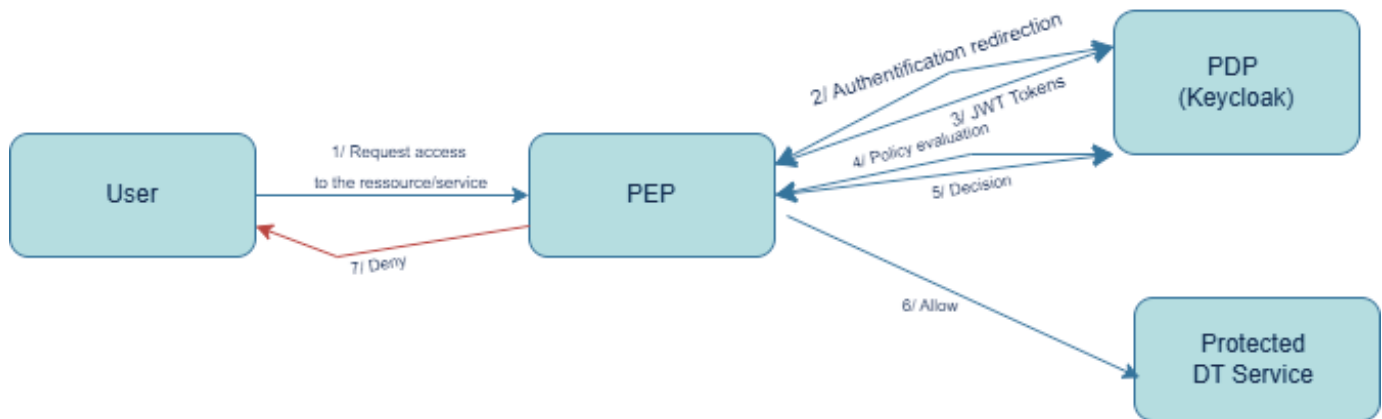


Fig. 4. Per-request decision cycle enforced by the Flask PEP and Keycloak PDP.

META-INF/keycloak-scripts/authorization/.

- 2) These scripts were then packaged into a single JAR file named `kc-scripts.jar` using a standard Java build tool (e.g., `jar` or `Maven`).
- 3) The JAR was copied to Keycloak's provider directory.
- 4) Finally, in our realm JSON, each policy was updated to reference the script by name using the syntax `"type": "script-<filename>"` instead of embedding the JavaScript code directly.

With this setup, the full realm configuration including users, roles, clients, and policies can now be deployed automatically and consistently, even on Keycloak versions greater than or equal to 18. This approach preserves the flexibility and power of script-based policies while remaining compatible with the platform's latest security model.

V. USE CASE: SETTING A ZTA APPROACH WITH REALMS AND RBAC STRATEGY FOR AN AAS-BASED DT PLATFORM

Realms in Keycloak.: A *realm* is an isolated identity domain: it possesses its own users, roles, groups, clients and policies. Two realms cannot read each other's data unless an administrator explicitly sets up federation or cross-realm brokering. The concept is handy for multitenant installations or, as in our case, to separate DT services with different risk levels.

One service = one realm.: Figure 5 depicts our Digital Twin high-level architecture platform. Each white building block (DT Data Component, What-if Scenario Component, Dashboarding, etc.) is protected by **its own realm and its own PEP**. This isolation ensures that if one DT service is compromised, the impact is confined to that realm; other services still require separate authentication and authorization flows.

To keep the design lightweight and auditable, we rely exclusively on a **Role-Based Access Control (RBAC)** model. In this context, roles are sufficient to distinguish between

user profiles (e.g., read-only dashboards vs. full-access maintenance tools). Every policy simply verifies whether a user's token includes the expected role, making access decisions easy to understand and manage in operational environments.

A. Roles and Policies for an AAS-based Architecture Component

The DT Data Manager component is based on an AAS architecture. It is developed with Basyx SDK¹ and managed in this context in a single realm named `'BaSyxSecured'`. Some role examples are provided, defining specific access privileges across the AAS architecture environment. These roles enable access control over components such as AAS registries or submodels. It includes:

- **admin:** Grants full administrative access both to the Keycloak realm and to all the services. This role is typically assigned to platform administrators or superusers.
- **basyx.reader:** Allows read-only access to all AAS endpoints, typically via HTTP GET requests. This role is suitable for dashboards, monitoring systems, or users who need to observe system state without making modifications.
- **basyx.writer:** Grants permissions to create, update, and delete AAS elements. This role is essential for engineering applications that configure or deploy new digital twins.
- **basyx.file-sme-reader:** Provides read access to file-based submodel elements (SMEs), which are used to store structured documents or configuration files.

In our context, Access control decisions are driven by Keycloak's **Authorization Services**, where roles are checked through policies. Two concrete examples illustrate how such policies are defined and evaluated:

Admin Access Policy: A role-based policy that uses the **UNANIMOUS** strategy, which means that all conditions must be met for access to be granted. In this case, the request is allowed only if the user's token contains both the `admin` role and the `default-roles-basyx` role set. These are usually assigned to a trusted system

¹<https://github.com/eclipse-basyx/basyx-java-server-sdk>

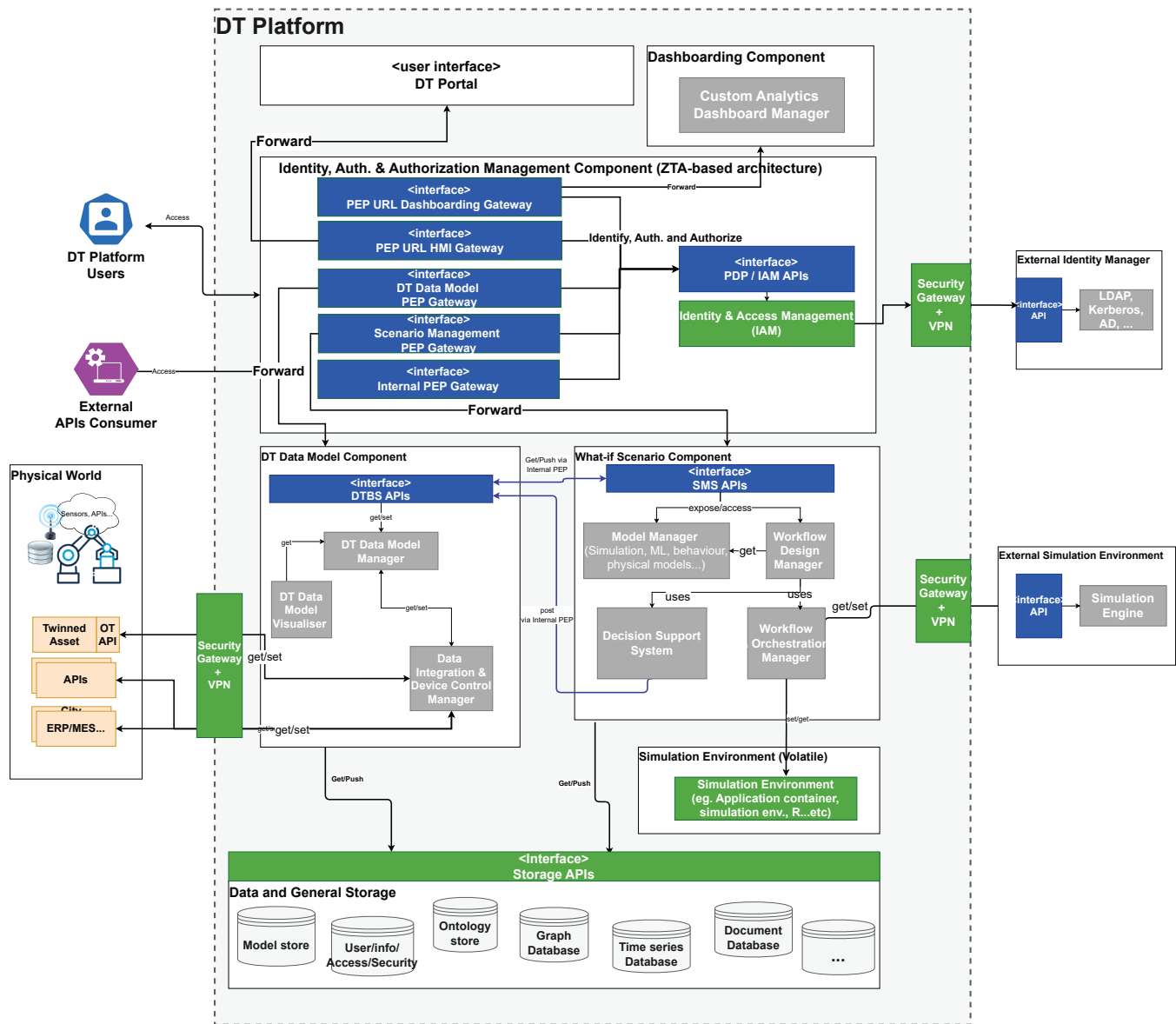


Fig. 5. High-level Digital Twins platform architecture with a complete security layer. Each component (white) has its own Realm; Each has its own PEP and Keycloak client as depicted inside the Security component.

administrator who can perform all operations without restriction.

Read-Only Policy: A simpler role-based policy that grants access if the user token includes the `basyx.reader` role. It is typically attached to a permission definition that restricts HTTP methods to GET, ensuring that the user cannot perform any write or delete actions. This policy is suitable for external applications, dashboards, or operators that need to monitor digital twins but not alter them.

Permission Matrix for ZTA Settings These policies are then mapped to a set of defined permissions, which protect critical DT data endpoints and services. The protected URLs include the AAS Registry, Submodel Repositories, Discovery

Service, and others used by a Web UI or API clients. Table I shows how these policies translate into effective access control based on the caller's role and token content.

This proposed role-policy-permission model enables fine-grained access control tailored to the different actors in a digital twin ecosystem. By using Keycloak's declarative authorization layer, new roles and policies can be added or adjusted without modifying application code, supporting a clean separation of concerns between identity management and DT resource logic.

Figure 6 illustrates the security policy configuration automatically pushed inside Keycloak, which acts as the PDP in this architecture. Here, access to various AAS resources is governed by two RBAC-based policies. These policies evaluate

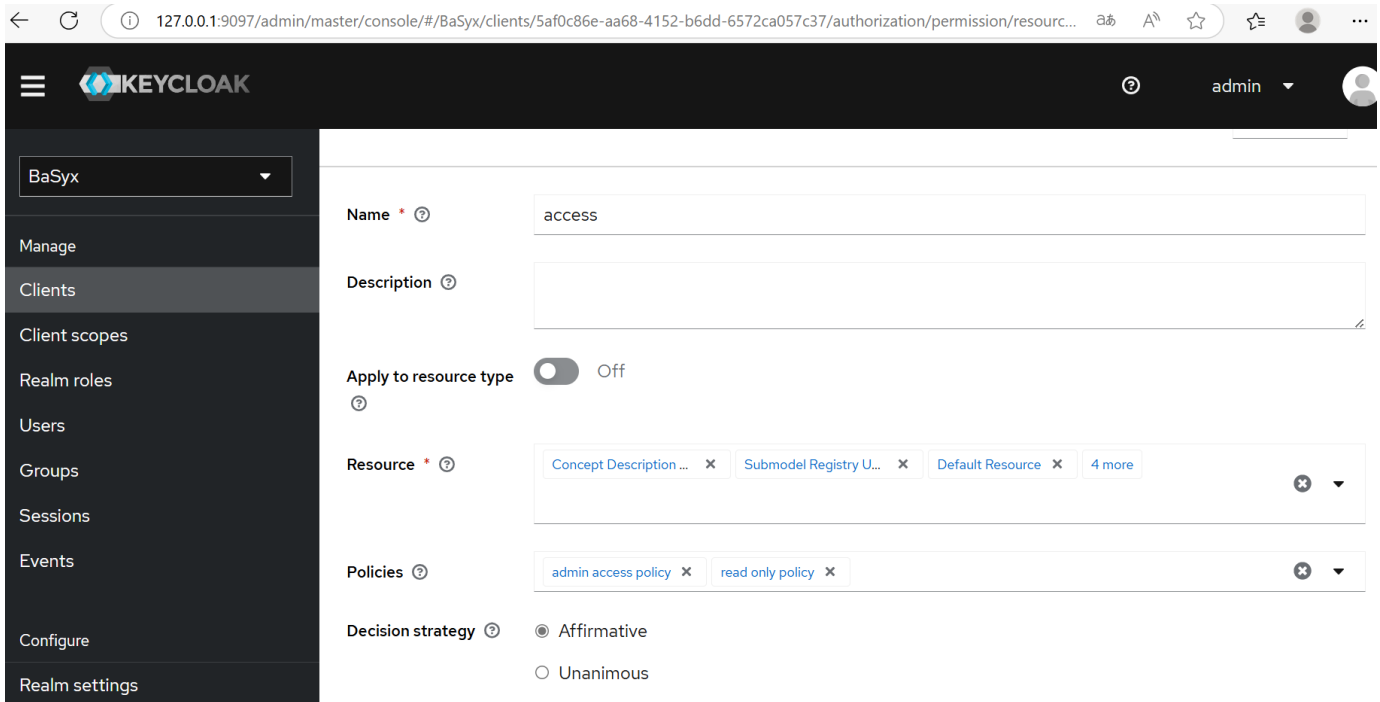


Fig. 6. Generated PDP Configuration inside Keycloak

TABLE I
EXAMPLE FOR PRIVILEGE TIERS AFTER POLICY EVALUATION.

Caller Role	HTTP Verbs Allowed	Typical Use Case
admin	GET, POST, PUT, DELETE	Full system management
basyx.reader	GET only	Visual dashboards, monitoring
No valid token or role	— (401 / 403 error)	Rejected or anonymous access

whether the requesting user holds the appropriate role before granting access, enforcing strict role-driven control over the digital twin environment.

VI. ARCHITECTURAL IMPLICATIONS AND CHALLENGES

The proposed ZTA architecture for DT platforms builds upon established security principles and open-source tools. Its novelty lies in the contextual adaptation of ZTA to the DT domain and platform as a whole, particularly through its practical integration with industrial standards such as the AAS. This enables fine-grained access control at the property level, realm-based service generation and isolation for DT services, and automated policy injection—features that are not directly supported by generic ZTA implementations. These adaptations address the unique challenges of DT platforms, including heterogeneous IT/OT integration, real-time data governance, and the involvement of multiple stakeholders with varying access requirements and DT data security and governance.

Despite these strengths, several practical considerations must be acknowledged. Deploying ZTA in an industrial DT context is inherently complex due to the diversity of systems,

departments, and legacy technologies involved. A coherent and organization-wide governance framework for identity management, data access, and policy enforcement should be defined early in the development process to prevent fragmentation and ensure consistent security practices across the platform.

The strategy of isolating DT services into distinct realms enhances security by separating services based on risk levels. However, we notice that this approach introduces significant administrative overhead. Managing multiple realms, each with its own clients, roles, and policies, requires advanced tooling and operational expertise, which may not be readily available in all organizations.

From a technical perspective, the proposed architecture relies heavily on Keycloak for identity and access management. While Keycloak offers a practical and widely adopted open-source solution, its central role introduces a potential single point of dependency. Any limitations, outages, or version-specific constraints in Keycloak could impact the overall system. Future work should explore fallback mechanisms, such as federated identity providers or alternative policy engines, to enhance resilience and reduce vendor lock-in.

While this architecture is tailored to the DT context, its adaptation introduces complexities that requires further analysis. DT platforms demand real-time, fine-grained access control across heterogeneous IT/OT systems, and the mapping of identity and policy constructs to various DT components such as services, DT data models and interfaces, is non-trivial. These challenges highlight the importance of pragmatic design choices that align with the reality and complexity of industrial DT environments.

VII. CONCLUSION AND PERSPECTIVES

This paper demonstrates how Zero Trust Architecture principles can be operationalized in a DT context, advancing a cybersecurity-by-design methodology suited for connected and dynamic industrial systems. We proposed and implemented the architectural components and deployment steps adapted to industrial environments. This approach demonstrates that assigning a dedicated realm and Policy Enforcement Point to each core Digital Twin service provides a modular and effective means of applying the NIST ZTA principles. This design ensures strong security boundaries by isolating services by default, minimizing potential impact in the event of a compromise, and maintaining strict access control.

By leveraging the latest stable release of Keycloak, we ensure continued support, security, and compatibility. Additionally, advanced authorization logic is implemented through a minimal set of JavaScript-based policies, encapsulated in a single external JAR, aligning with modern deployment best practices.

As a perspective, we aim to extend the proposed architecture to encompass all components outlined in the high-level system architecture. The current implementation, presented as a proof of concept, can be further enhanced by integrating advanced ZTA capabilities, such as context-aware access control, to strengthen cybersecurity assurance and improve adaptability in dynamic industrial environments. In addition, the absence of formal security validation, including attack simulations, performance benchmarking, and scalability assessments, limits the ability to fully evaluate the architecture's effectiveness in mitigating digital twin threats. Addressing these gaps will be a key focus of future work to ensure the robustness and operational maturity of the proposed solution.

ACKNOWLEDGMENTS

This work has been supported by the French government under the “France 2030” program, as part of the SystemX Technological Research Institute, Industrial Digital Twin Program JN1.

REFERENCES

- [1] M. El-Hajj, T. Itäpelto, and T. Gebremariam, “Systematic literature review: Digital twins’ role in enhancing security for industry 4.0 applications,” *Security and Privacy*, vol. 7, no. 3, 2024. [Online]. Available: <https://onlinelibrary.wiley.com/doi/abs/10.1002/spy2.396>
- [2] Digital Twin Consortium, “Capabilities periodic table,” 2024. [Online]. Available: <https://www.digitaltwinconsortium.org/initiatives/capabilities-periodic-table/>
- [3] S. Rose, O. Borchert, S. Mitchell, and S. Connelly, “Zero trust architecture,” NIST Special Publication 800-207, Tech. Rep., Oct. 2022. [Online]. Available: <https://doi.org/10.6028/NIST.SP.800-207>
- [4] A. Belfadel, S. Creff, and A. B. Hamida, “Advancing industrial digital twins: Towards an open platform aligned with standards,” in *IFIP International Conference on Product Lifecycle Management (PLM)*. Springer, 2024.
- [5] T. Lacheroy, A. Belfadel, and J.-Y. Didier, “XRTwin4Industry: A Comprehensive Framework for Standardisation and Interoperability in XR-Enabled Industrial Digital Twins,” in *2025 IEEE International Conference on Pervasive Computing and Communications Workshops and other Affiliated Events (PerCom Workshops)*. Los Alamitos, CA, USA: IEEE Computer Society, Mar. 2025, pp. 146–151. [Online]. Available: <https://doi.ieeecomputersociety.org/10.1109/PerComWorkshops65533.2025.00056>
- [6] International Electrotechnical Commission, *Asset administration shell for industrial applications – Part 1: Asset administration shell structure*, International Electrotechnical Commission (IEC) Std. IEC 63 278-1:2023, 2023, available at the IEC Webstore. [Online]. Available: <https://webstore.iec.ch/en/publication/65628>
- [7] R. Kumar *et al.*, “Digital twins-enabled zero touch network: A smart contract and explainable ai integrated cybersecurity framework,” *Future Generation Computer Systems*, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S0167739X24000608>
- [8] H. Feng *et al.*, “Game theory in network security for digital twins in industry,” *Digital Communications and Networks*, 2024. [Online]. Available: <https://www.sciencedirect.com/science/article/pii/S2352864823000172>
- [9] B. Paul and M. Rao, “Zero-trust model for smart manufacturing industry,” *Applied Sciences*, vol. 13, no. 1, 2022. [Online]. Available: <https://www.mdpi.com/2076-3417/13/1/221>
- [10] S. Russo, “Industrial demilitarized zone and zero trust cybersecurity models for industrial control systems,” <https://amslaurea.unibo.it/id/eprint/26737/1/TesiSilvioRusso.pdf>, 2023, master’s Thesis, University of Bologna.
- [11] J. Hörber, M. Eisenhauer, L. Reiner *et al.*, “Secure usage of asset administration shells: An overview and analysis of best practices,” Fraunhofer IESE, Tech. Rep., 2021. [Online]. Available: <https://d-nb.info/1246875705/34>
- [12] Keycloak Project, “Authorization services architecture,” 2025, keycloak v24 documentation. [Online]. Available: https://www.keycloak.org/docs/latest/authorization_services/
- [13] —, (2025) Server developer guide – script providers. Keycloak v24 documentation - Accessed: 2025-07-03. [Online]. Available: https://www.keycloak.org/docs/latest/server_development/#script-providers
- [14] —, *Keycloak Authorization Services Guide*, 2025, accessed: 2025-07-03. [Online]. Available: https://www.keycloak.org/docs/latest/authorization_services/index.html
- [15] Eclipse BaSyx, “BaSyx component deployment and security guide,” 2024. [Online]. Available: <https://eclipse.dev/basyx/>