D2.6 Security and trust management for CECC - Final

Document Summary Information

Project Identifier	HORIZON-CL4-2022-DATA-01. Project 101093129		
Project name	Agile and Cognitive Cloud-edge Continuum management		
Acronym	AC^3		
Start Date	January 1, 2023	**End Date**	December 31, 2025
Project URL	www.ac3-project.eu		
Deliverable	D2.6 Security and trust management for CECC - Final		
Work Package	WP2		
Contractual due date	M24: 31st December, 2024	**Actual submission date**	13th January, 2025
Type	R (Report, Document)	**Dissemination Level**	PU – Public
Lead Beneficiary	CSG		
Responsible Author	John Beredimas (CSG)		
Contributors	Ayoub Mokhtari(EUR), Sofiane Messaoudi (EUR), Souvik Sengupta (ION), Athanasios Kordelas (CSG), John Beredimas (CSG), Dimitrios Amaxilatis (SPA), Asimakis Lykourgiotis (CSG)		
Peer reviewer(s)	Abdelhak Kadouma (FIN), Amadou BA (IBM)		

Funded by the European Union

AC^3 project has received funding from European Union's Horizon Europe research and innovation programme under Grand Agreement No 101093129.

Revision history (including peer reviewing & quality control)

Version	Issue Date	% Complete	Changes	Contributor(s)
V0.1	17/10/2024	5%	Initial Deliverable Structure (ToC)	Asimakis Lykourgiotis (CSG) John Beredimas (CSG)
V0.2	07/11/2024	10%	Final Deliverable Structure (ToC) API security initial contributions	John Beredimas (CSG) Athanasios Kordelas (CSG)
V0.3	15/11/2024	40%	Initial Contributions	Athanasios Kordelas (CSG) Sofiane Messaoudi (EUR) Ayoub Mokhtari (EUR) Dimitrios Amaxilatis (SPA) Souvik Sengupta (ION)
V0.4	18/11/2024	50%	Executive Summary and Conclusions	John Beredimas (CSG) Asimakis Lykourgiotis (CSG)
V0.5	18/11/2024	60%	Formatting fixes	John Beredimas (CSG)
V0.6	22/11/2024	70%	Formatting fixes Internal review (responsible author)	Asimakis Lykourgiotis (CSG) John Beredimas (CSG)
V0.7	09/12/2024	80%	Address peer review comments	Athanasios Kordelas (CSG) Sofiane Messaoudi (EUR) Ayoub Mokhtari (EUR) Dimitrios Amaxilatis (SPA) Souvik Sengupta (ION)
V0.8	10/12/2024	90%	Acronyms, responsible author review	John Beredimas (CSG)
V0.9	17/12/2024	95%	Address Technical Manager comments	John Beredimas (CSG) Souvik Sengupta (ION)
V1.0	20/12/2024	96%	Address comments by the Project Coordinator	John Beredimas (CSG)
V1.1	09/01/2025	98%	Address comments by the Project Coordinator	John Beredimas (CSG) Sofiane Messaoudi (EUR)
V1.2	10/01/2025	100%	Document clean-up for submission	John Beredimas (CSG)

Disclaimer

While the information contained in the documents is believed to be accurate, the authors(s) or any other participant in the AC^3 consortium make no warranty of any kind with regard to this material including, but not limited to the implied warranties of merchantability and fitness for a particular purpose.

Neither the AC^3 consortium nor any of its members, their officers, employees or agents shall be responsible or liable in negligence or otherwise howsoever in respect of any inaccuracy or omission herein.

Without derogating from the generality of the foregoing neither the AC^3 Consortium nor any of its members, their officers, employees or agents shall be liable for any direct or indirect or consequential loss or damage caused by or arising from any information advice or inaccuracy or omission herein.

Copyright message

Table of Contents

List of Figures

List of Tables

Glossary of terms and acronyms used

Abbreviation / Term	Description
2FA	Two Factor Authentication
AC^3	Agile and Cognitive Cloud Edge Continuum Management
AES	Advanced Encryption Standard
AFL	Adaptation and Federation Layer
AMQP	Advanced Message Queuing Protocol
ARM	Application and Resource Management
API	Application Programming Interface
CA	Capped Adjustment
CPU	Central Processing Unit
CPX	NetScaler container-based form factor
CECC	Cloud Edge Computing Continuum
CECCM	Cloud Edge Computing Continuum Manager
EDC	Eclipse Dataspace Components
FHS	Federation Hosting Services
FHSOp	FHS Operator
HMAC	Hash-based Message Authentication Code
HTTP	HyperText Transfer Protocol
HTTPS	HTTP Secure
IdP	Identity Provider
IEEE	Institute of Electrical and Electronics Engineers
JSON	JavaScript Object Notation
JWT	JSON Web Tokens
KPI	Key Performance Indicator
LCM	Life-Cycle Management
LMS	Local Management System
mTLS	Mutual TLS
NAC	Network Access Control

NB	Northbound
NBI	Northbound Interface
OSR	Ontology & and Semantic aware Reasoner
PAP	Policy Administration Point
PDP	Policy Decision Point
PEP	Policy Enforcement Point
QoE	Quality of Experience
RBAC	Role-Based Access Control
RSA	Rivest–Shamir–Adleman algorithm
RTO	Recovery Time Objectives
S3	Simple Storage Service
SIIF	(IEEE) Standard for Intercloud Interoperability and Federation
SLA	Service Level Agreement
SLO	Service Level Objectives
SSL	Secure Sockets Layer
STS	Security Token Service
TRL	Technical Readiness Level
WP	Work Package
ZTNA	Zero Trust Network Access

1 Executive Summary

This document, "D2.6 Security and trust management for CECC - Final" is a crucial output of the AC^3 (Agile and Cognitive Cloud Edge Continuum Management) project, funded under Horizon Europe's Research and Innovation Action program. It delves into the intricate realm of security and trust management within the Cloud Edge Computing Continuum (CECC) and its interactions with the application developer and the infrastructure, aligning with task T2.4's objectives entitled "Security and Trust Management for CECC". It should be noted that task T2.4 is the only task that addresses security and trust aspects within AC^3.

The integration of cloud, edge, and far-edge computing platforms within the CECC landscape has transformed the digital ecosystem. This convergence enhances performance, scalability, and cost-efficiency while enabling real-time responsiveness across applications. Key benefits include reduced latency, improved application responsiveness, optimized resource allocation, enhanced bandwidth efficiency, and reinforced data privacy and security. The integration of these platforms creates a dynamic ecosystem that optimizes performance, resource management, and data privacy, setting the stage for operational excellence in the digital era.

AC^3 project involves the interaction of different stakeholders to run micro-service-based applications on top of a CECC infrastructure, where we can mention application developers, CECC Manager (CECCM) providers, and infrastructure providers. Establishing a zero-trust security principle among the stakeholders while establishing trust when handling the requested Service Level Agreement (SLA) of the application is a must-have for any complex ecosystem like that of AC^3. The complexity of the AC^3 model was initially captured in deliverable D2.1 [1], detailing all the components involved in deploying and running micro-service-based applications, covering the functional blocks as well as the communication interfaces among these blocks. However, the envisioned architecture does not detail how these interfaces are secured and how trust for SLA management can be established. This deliverable addresses this gap by proposing solutions for security and trust in AC^3.

In this deliverable, we build upon the initial high-level framework for zero-trust security and trust management within the AC^3 project, originally presented in deliverable D2.5 [2]. It is important to note that the term "trust" in this deliverable is used as a key approach for security architecture, but it is also an approach to managing SLA established between the application developer and CECCM and between CECCM and the different infrastructure providers. The aim of the first usage of trust is to secure all the communication interfaces, both internal and external, to the CECCM, leveraging the AC^3 architecture as originally proposed in D2.1 and updated based on the work of Work Package (WP) 3 and WP4. The second usage of the word trust focuses on building a trust model to allow CECCM to select the infrastructure provider to run a micro-service. The trust model relies on Blockchain and Smart Contracts to derive the reputation of each infrastructure provider. Our objective is to safeguard data and ensure trust within CECCM and its interactions (i.e., Interfaces) with the application developer and the infrastructure. We have developed a robust data management strategy, emphasizing data security throughout its lifecycle, from retrieval to storage and monitoring. Additionally, we have fortified security measures within the CECCM, ensuring operational security while adhering to strict SLA benchmarks. Trust is a paramount concern, and stakeholders can rely on the integrity and reliability of the CECC ecosystem.

Section-3 initially provides an updated AC^3 security architecture, considering the updated high level AC^3 architecture, which is detailed in D2.2. It is split in four sub-sections

1. Section 3.1 briefly outlines the three-planes model of AC^3, User, Management and Infrastructure, and then examines both inter-plane security, as well as intra-plane, for securing the communications between the CECCM microservices.
2. Section 3.2 delves deeper into Application Programming Interface (API) security, outlining how AC^3 builds on top of the Kubernetes Gateway Ingress API to provide for encryption, validation, authentication, authorization and policies to ensure utmost security in API interactions.

3. Section 3.3 focuses on data, considering access, security, and integrity through the entire lifecycle of data management.
4. Section 3.4 considers the microservice migration algorithms introduced in WP3 and the security implications thereof.

Trust is a cornerstone in AC^3, as highlighted in Section 4. The Trust Model is expounded upon, with a specific focus on SLA Management using Smart Contracts. The detailed discussion encompasses SLA and Smart Contracts definition, Blockchain integration, a new Trust Manager component, algorithmic and mathematical details as well as experimental validation. Additionally, Section 4 addresses Trust in Data Management, emphasizing security measures of data both in transit and at rest. These sections reflect our unwavering commitment to building and maintaining trust in the AC^3 ecosystem.

In conclusion, this deliverable builds on top of the initial report on security and trust management introduced with D2.5 [2], leveraging insights and outcomes from WP3 and WP4 as well as further evolving the security architecture and capabilities. The exhaustive content, drawn from various contributors and meticulously detailed in this report, encapsulates the collective effort invested in shaping the AC^3 security landscape.

2 Introduction

This section establishes the context of the work detailed in this deliverable. It will outline the main purpose of the deliverable, its core objectives, and how it connects to the broader project framework and associated deliverables. Additionally, this section will delineate the anticipated outcomes and their alignment with the commitments outlined in the Grant Agreement. It concludes by providing the structural organization of the deliverable.

2.1 Purpose and Objectives

The diverse actors in the AC^3 ecosystem require mechanisms to ensure the security and protection of all transactions and communications. Furthermore, there is a need to provide the CECCM with SLA management mechanisms to build trust knowledge regarding the infrastructure providers and hence ease the selection of one provider over the other when deploying microservice components.

The initial deliverable on security and trust management [2] introduced and detailed three new entities: Security Gateway, Identity Provider (IdP) and Authorization Server and Security Policies Administrator. It also designed a novel architecture that proposed a third-tier Trust Manager entity that would collect feedback from application developers and Key Performance Indicators (KPIs) from CECCM to detect SLA violations based on a Blockchain and Smart Contract approach.

This deliverable provides a significant update on the AC^3 security and trust. It revisits and expands all aspects of AC^3 security previously introduced, including security architecture, API security, security of Data. It considers microservices migration, introduced in deliverable D3.1 [3], and the respective security architecture and mechanisms. It significantly expands on the SLA management trust model, both by revisiting the respective architecture as well as introducing a new mathematical model and algorithm backed by experimental results. Last, but not least, it recalls D3.3 on data management [4] and provides the respective update for trust of data.

2.2 Mapping AC^3 Outputs

The purpose of this section is to map AC^3 Grant Agreement commitments, both within the formal Deliverable and Task description, against the project's respective outputs and work performed.

Table 1: Adherence to AC^3 GA Deliverable & Tasks Descriptions

AC^3 GA Component Title	AC^3 GA Component Outline	Respective Document Chapter(s)	Justification
DELIVERABLE			
D2.6 Security and Trust management for CECC - Final *"Final version including updates based on T2.2, WP3 and WP4"*			
TASKS			
Task T2.4: Security and trust	**Task T2.4**: This task devises robust protocols and mechanisms to establish trust with the	Section 3, Section 4	Secure the intra and extra CECCM components communications and establish the trust in AC^3 Framework.

management for CECC	candidates to the resource federation before granting access, guarantee communication, resource, and data security within the federation, and revoke access from federation members if security breaches are observed. Further, mechanisms to verify and guarantee signed SLA between the different stakeholders involved in the federated infrastructure. On the other hand, this task will implement protocols and enablers to secure the communication channel and protect the CECCM from external attacks as the latter is exposing interfaces to the public (application developers)		
Task T2.2: Reference architecture for CECC	**T2.2:** This task defines a detailed specification of the CECC architecture, key components, and features of the CECCM, and the required interfaces and protocols to perform federation and interact with the different infrastructures in a secure and trusted manner.	Section 3.1	The updated AC^3 architecture has driven the corresponding updates of the Security Architecture
Task 3.3: Service migration of stateful microservices	**T3.3:** This task will provide the necessary algorithms to overcome the challenges related to the migration of microservices, particularly those needing storage and volume (i.e., stateful).	Section 3.4	The AC^3 security architecture has been revised to accommodate for secure migration of stateful microservices
Task 3.4: Applications	**T3.4:** This task covers one of the key innovations of	Section 3.3, Section 4.6	The Security and Trust of data management have been updated

data management as a PaaS	the project, which is the integration of data management as PaaS within the CECCM. Therefore, this task will provide all the necessary components to achieve this objective.		in consideration of the respective work performed under WP3

2.3 Advancements with respect to deliverable D2.5

This deliverable presents a compelling update on the AC^3 security and trust management, building upon our previous work in D2.5 [2]. The updates introduced in this deliverable accommodate for both the high-level architecture changes introduced by WP3 and WP4, as well as advancements related to security and trust management. Specifically, the following contributions are advanced from D2.5:

- **Section 3**
  - **Section 3.1**: the high-level architecture of the CECCM has been updated, as a result of the advances and work in WP2, WP3 and WP4. Towards this end, the security architecture has also been updated to consider the respective architecture and component changes of CECCM.
  - **Section 3.2**: API Security previously considered the various security capabilities and security features of NetScaler. In this deliverable we follow a more practical approach. We examine in detail the exact placement of NetScaler in the AC^3 architecture and interfaces. We also consider advances in NetScaler's cloud-native capabilities, which enable the use of the Kubernetes Gateway API [5], a new Kubernetes API object that offers for a more flexible and stronger security posture, compared to the pre-existing Ingress API object.
  - **Section 3.3**: Security of data management is revisited to consider the outcomes of WP3 and deliverable D3.3 [4]. Specifically, two new sub-sections have been introduced related to security of In Use Data and the role of digital contracts between data sources and data consumers.
  - **Section 3.4**: this is a new section to consider security for microservices migration, which was introduced in deliverable D3.1 [3].
- **Section 4**
  - **Section 4.1**: The initial trust model is significantly expanded through the introduction of a new component, namely the Trust Manager. A novel trust architecture that leverages this component is introduced, building upon the Federation Hosting Services (FHS) framework of the IEEE Standard for Intercloud Interoperability and Federation (SIIF) [6].
  - **Section 4.2** to **Section 4.5**: these subsections expand upon the original smart contract and blockchain approach for SLA management. They introduce a detailed mathematical model and algorithm, as well as experimental validation of results.
  - **Section 4.6**: this section recalls the WP3 introduction of the Piveau catalogue [7] and the IONOS S3 Extension for Eclipse Dataspace Components (EDC) [8] to walk through the updated architectural framework, workflow and technical innovations for trust of data management.

2.4 Deliverable Overview Structure

In this section, a description of the Deliverable's Structure is provided, as follows:

- Section 3 updates the AC^3 security by revisiting the security architecture to align with the latest changes of the high-level architecture, modernizes API security to align with the latest Kubernetes specifications, considers changes to data management security so as to align with the WP3 updates and introduces a new section for secure migration of stateful microservices.
- Section 4 expands upon the initial trust model and architecture, via the introduction of a new Trust Manager Component that is integrated with the IEEE SIIF. It provides a detailed algorithmic and mathematical modelling of said component, as well as experimental validation. Finally, it updates Trust of Data Management, leveraging the respective components introduced in D3.3 [4].

3 AC³ Security

The **"AC³ Security"** segment tries to build upon foundational architectures addressed in preceding deliverables, thereby refining and contemporizing the security model to incorporate recent advancements delineated in Work Packages WP3 and WP4. This segment is systematically structured to furnish a comprehensive exposition of the security frameworks and mechanisms employed to preserve data sanctity, ensure API integrity, and shield communications within the CECC ecosystem.

Through the refined security architecture, leveraging advanced cryptographic protocols, containerization strategies, and real-time monitoring infrastructures, the AC³ security framework aspires to present a resilient, scalable solution requisite for safeguarding CECC infrastructures. This section elucidates not only the technical advancements introduced but also highlights the indispensable role of security in nurturing trust and ensuring reliability across the AC³ ecosystem.

3.1 Security Architecture

3.1.1 High-level architecture

In this section, we present an updated high-level security architecture for AC³, based on the final version of the overall AC³ architecture. The key updates to the security architecture, compared to the previous one proposed in the D2.5 deliverable [2], are focused on internal components within the CECCM. Specifically, the following updates are present in the overall final architecture

- the Service Catalogue and Data Catalogue are now a single “Catalogues” component
- a number of user plane components, namely “Catalogues,” “Ontology & Semantic Aware Reasoner” (OSR), KPI collection and exposure, and Northbound API Engine, are now part of the “Application Gateway”.

The CECCM internal and external communication security will be detailed in the following subsections. External communications involve interactions between the CECCM and external entities such as application developers and the Northbound (NB) Interfaces (NBIs) of the different infrastructures Local Management System (LMS). Internal communications consist of the interactions between the CECCM components. From our perspective, securing both north/south communications between users and the CECCM or between CECC federated infrastructure and the CECCM, as well as east/west communications among services within the CECCM can significantly reduce the vulnerabilities in the CECCM architecture. A leveraged version of the AC³ architecture featuring security is illustrated in Figure 1.

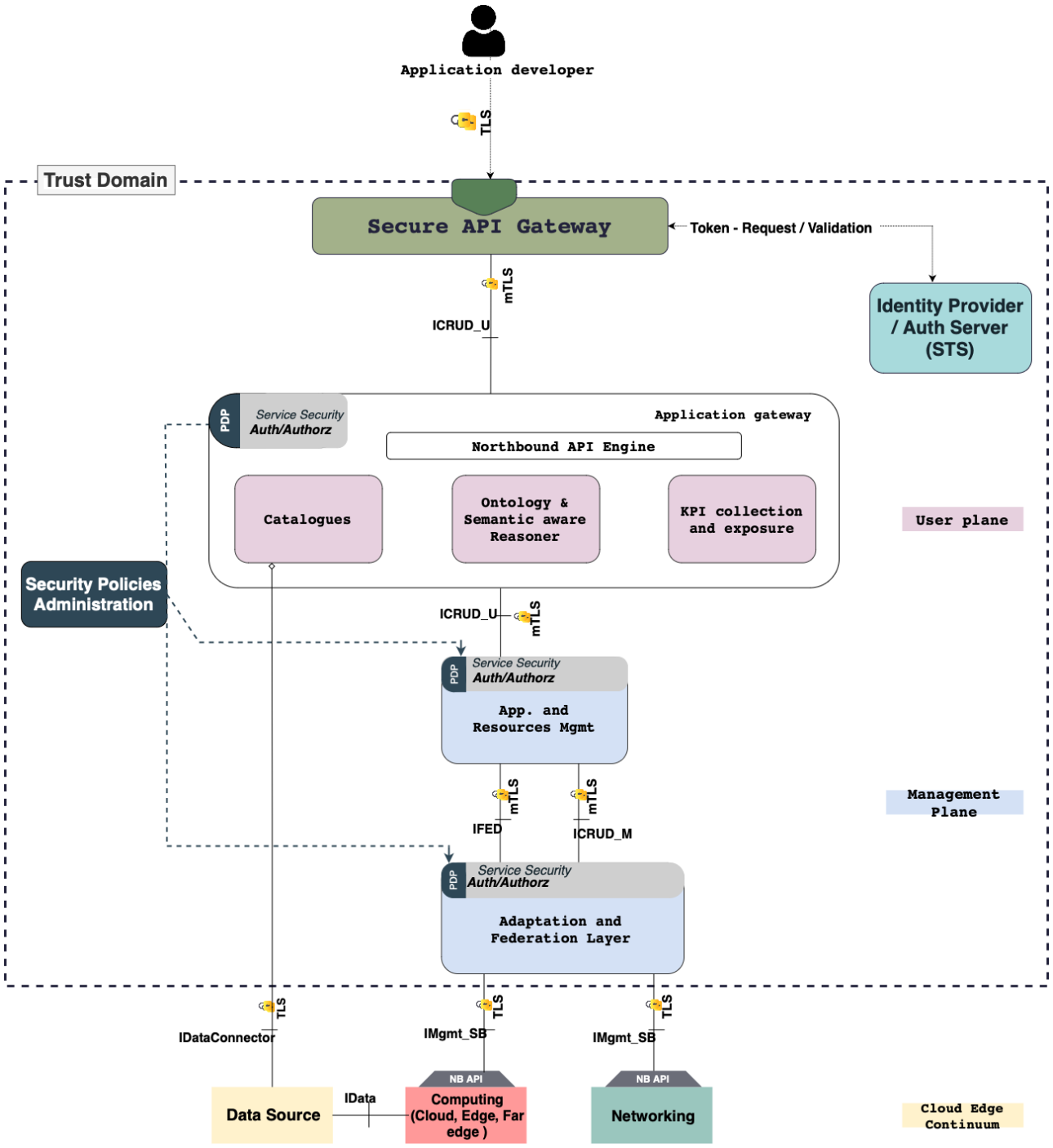


Figure 1. Security Architecture Overview

3.1.2 Secure User-CECCM Communication

The Secure API Gateway depicted in Figure 1 acts as a policy enforcement point (PEP), authenticating, authorizing, and intercepting all end-user requests going to the Application Gateway Northbound API to enforce access-control policies, and then it dispatches the requests with the user's context to the Application Gateway

Northbound API implementation. The Secure Gateway's role is to ensure that only CECCM users with trusted credentials can access the APIs and that only those requests are directed to the upstream services.

For end-user authentication, commonly utilized approaches include strong two-factor authentication (2FA) or certificate-based authentication, which safeguards an API at the CECCM entry point. Access to other services is conditional on the customer's possession of a valid credential issued by a trusted entity such as an IdP. The alternative approach involves employing OAuth-2.0–based access delegation. OAuth 2.0, an authorization framework facilitating delegated access control, is the preferred method for safeguarding APIs when one system intends to access an API on behalf of another system or user. In this context, the secure gateway's responsibility is to authenticate the OAuth 2.0 security tokens accompanying each API request. These tokens serve as a representation of both the third-party application and the user who granted access to the third-party application to interact with an API on their behalf.

In addition to identifying the requester during the authentication process, the Secure Gateway can enforce authorization by applying access control policies. More detailed access control policies are then implemented at the service level by the respective service itself. Irrespective of the chosen authentication and authorization approach, the primary function of the IdP/Auth Server depicted in Figure 1 remains unchanged, producing tokens or certificates to the Secure Gateway for user authentication and authorization.

After verifying the integrity of the connection, the secure gateway forwards the requests with user context to the Application Gateway API to be then redirected to the appropriate CECCM service, e.g., Application and Resource Management service. The user context includes fundamental details about the end user, and the client context provides information about the client application. This information can be utilized by the CECCM services for service-level access control. For instance, consider a scenario in which application developer <A> wants to consume data of another deployed application <App1>, the application developer <A> context will be used to limit his service access to only consuming the <App1> data and to prevent access to <App1> Life-Cycle Management (LCM). To transmit the context to CECCM services, there are a couple of options: either pass the user context in an HTTP header or create a JSON Web Token (JWT) containing the user data. While the first option is straightforward, it raises trust concerns when one service transmits the user context in an HTTP header to another service, as the second service lacks assurance that the user context remains unaltered. Opting for the second approach with JWT provides confidence that a man-in-the-middle cannot modify its content without detection, as the issuer of the JWT signs it. Therefore, the second approach is considered more secure. It's worth noting that the communication between the Secure API Gateway and the Application Gateway API (if not implemented as a single component) necessitates mutual Transport Layer Security (mTLS) authentication to ensure channel security.

Consider a scenario where a developer seeks to deploy an application within the infrastructure managed by the CECCM. As outlined in Deliverable 2.1 [1], the deployment process entails the developer furnishing the application's definition to the CECCM. In addition to this, for initial deployment requests, the developer is required to submit its credentials to the CECCM through the secure gateway.

Subsequently, upon receiving these credentials, the secure gateway interacts with the IdP to either generate or retrieve the user certificate. This certificate serves as an authentication mechanism for subsequent interactions. Once obtained, the secure gateway initiates a request to the IdP to acquire a JWT token containing the user context, encompassing roles and privileges.

The secure gateway then scrutinizes the user's authorization level in comparison to the request at hand. If there is a match, the secure gateway proceeds to forward the request, along with the user context, to the user plane for enforcement. In cases where the user's authorization level does not align with the request, the secure gateway rejects the request. This intricate process ensures the secure and authenticated deployment of applications within the CECCM infrastructure. These sequences are illustrated in Figure 2.

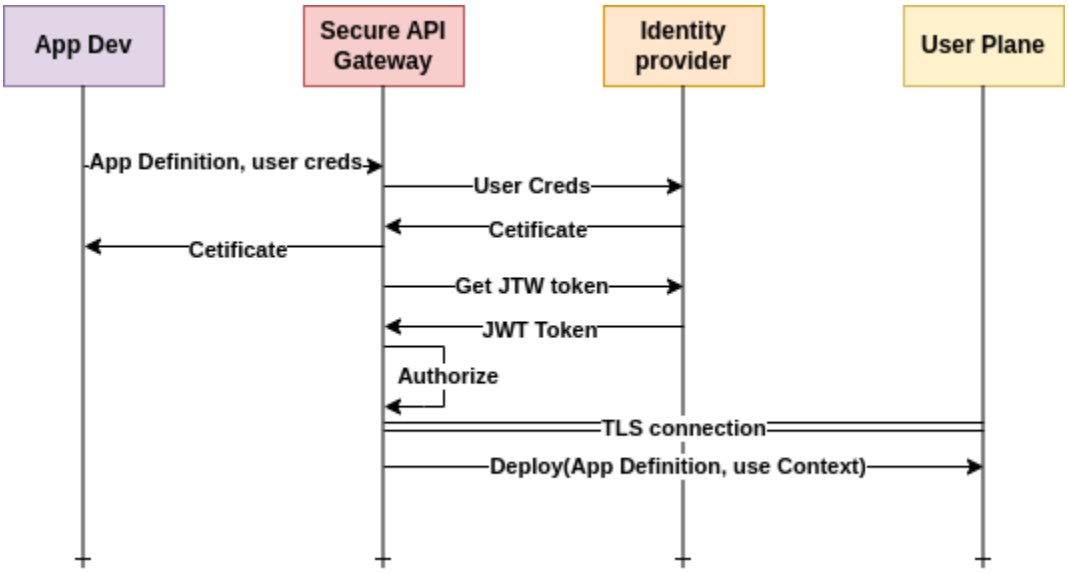


Figure 2. Secure Communication Workflow between the Application Developer and the CECCM

3.1.3 Secure CECCM Service-Service Communication

Similar to user-CECCM communications, ensuring authentication and authorization between services is essential to minimize vulnerabilities and security threats. In general, security models designed to safeguard service-to-service communication need to account for communication channels traversing trust boundaries and the nature of the communication itself, whether it's synchronous or asynchronous. In our specific case, details about the communication method are not yet available. Regarding trust domains, we are considering one for the CECCM services and another for the federated infrastructure for now.

Authentication: For authenticating inter-CECCM components communications, three commonly used approaches are generally employed: network access control (NAC) vs. zero-trust network access (ZTNA), mTLS, or JWTs.

NAC entails no explicit security enforcement in service-to-service communication. Instead, NAC relies on network-level security, where the assurance lies in preventing attackers from intercepting communications between services. A NAC approach would require deploying the CECCM within the same infrastructure and network, potentially leading to constraints on implementation. In addition, NAC assumes that all services in the same network can be trusted, which is not necessarily true, considering the sophisticated attacks frequently employed against modern infrastructure. Hence, this approach may not be the most suitable for our case. In contrast, the ZTNA paradigm offers an alternative viewpoint to NAC. ZTNA assumes a consistently hostile and untrusted network environment, rejecting any assumptions trusting the network. Every request within this approach must pass through authentication and authorization at each CECCM service (component) before being accepted for further processing.

One method for authenticating CECCM inter-service communication is mTLS, which stands as a widely adopted approach for securing service-to-service interactions. Each service within the deployment is equipped with a public/private key pair, utilizing this pair for authentication when communicating with recipient services through mTLS, ensuring mutual identification between communicating services.

Another approach for securing service-to-service communications is JWT. In contrast to mTLS, JWT operates at the application layer rather than the transport layer. A JWT serves as a container capable of transporting a set of claims from one location to another. These claims can encompass a variety of information, such as end-user attributes, end-user entitlements (defining what the user can do), or any data that the calling CECCM service wishes to convey to the recipient service. The JWT encapsulates these claims and is signed by the issuer of the JWT, which can be the Security Token Service (STS) or the calling CECCM service itself.

In our scenario, to capitalize on the strengths of the two previous mechanisms, we have opted for a hybrid approach. This involves combining mTLS for encryption and authentication with JWT for transmitting essential information between services, such as user details or authorization levels.

With mTLS, communication is encrypted, ensuring the confidentiality and integrity of the transmitted data. Mutual authentication is achieved as both the calling service and recipient service authenticate each other through their presented certificates.

To convey authorization-related information between the communicating services, we utilize JWT. When a calling service authenticates itself using mTLS, it can include a JWT in the request headers or as part of the payload. This JWT encapsulates various information, such as roles, permissions, or other pertinent claims. Upon receiving the JWT, the recipient service can verify its signature using the public key associated with the issuer, ensuring the integrity of the token and the authenticity of the claims.

Authorization: When considering authorization in a typical multi-service CECCM, service-level authorization is necessary to provide each service with a higher degree of control over enforcing access-control policies according to its specific requirements. The role of the Policy Decision Point (PDP) integrated within each CECCM component (see Figure 1) acts as a store for policies. These policies are centrally defined at the Policy Administration Point (PAP) and locally evaluated (at each CECCM component level). To receive policy updates from the centralized PAP, each CECCM service acts as an event consumer and subscribes to the relevant PAP event. Upon receiving an event, the service retrieves the corresponding policy from the PAP and updates its embedded PDP accordingly to be used in CECCM service-level authentication and authorization.

The precise definition of the centralized PAP definition depends on the CECCM implementation. In the illustrative example depicted in Figure 3, where the CECCM is implemented using a microservices approach, we can leverage the service mesh pattern for enforcing security at each microservice while decoupling the security from the microservice logic. In this context, the control plane provides the centrally defined security policies acting as PAP. Said security policies include, but are not necessarily limited to, access control and authorization, encryption and mTLS certificate management, routing rules and observability (logging, tracing, monitoring) policies.

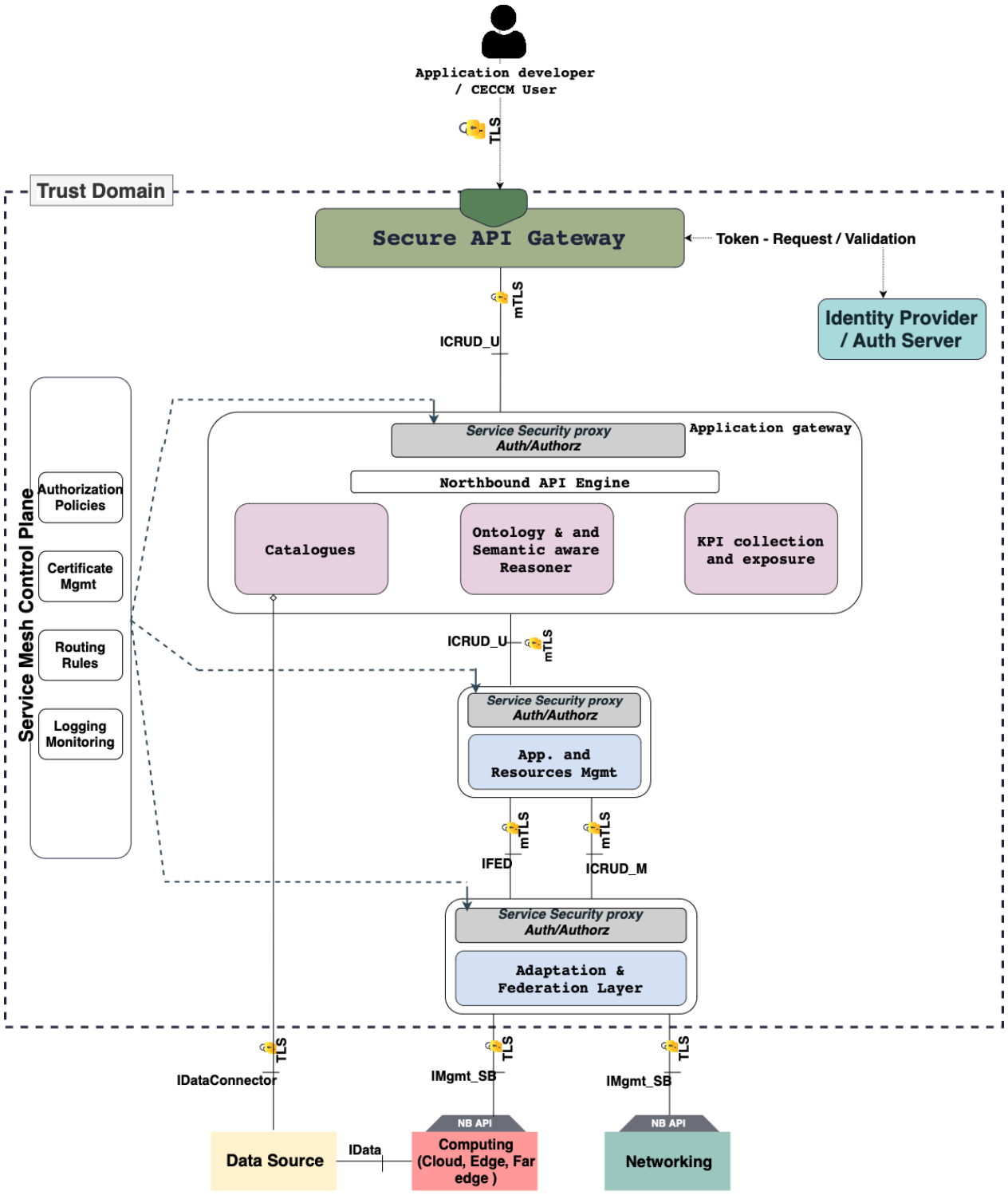


Figure 3. Security Architecture Instantiation in the Case of CECCM Microservices based Implementation

Let's consider an illustrative scenario depicted in Figure 4 involving the application gateway and the Application & Resource Mgmt component. In this instance, the application gateway seeks access to computing resources from the Application & Resource Mgmt service.

To initiate the process, the application gateway submits a request to the Application & Resource Mgmt, presenting its certificate issued by the IdP to substantiate the request. Within its authentication layer, the Application & Resource Mgmt service validates the sender's identity, confirms the request's alignment with the specified authorization level, and, upon successful verification, fulfills the request. The Application & Resource Mgmt service responds by providing both the requested computing resources and its own certificate to the application gateway.

Upon receiving the requested computing resources and the Application & Resource Mgmt certificate, the application gateway undergoes an authentication confirmation process, leading to the establishment of an mTLS session between the two components. With this authentication in place, the application gateway gains the privilege of making subsequent requests without undergoing repeated authentication. It is crucial to note that the authorization process is reiterated for each request, ensuring the ongoing security of the communication.

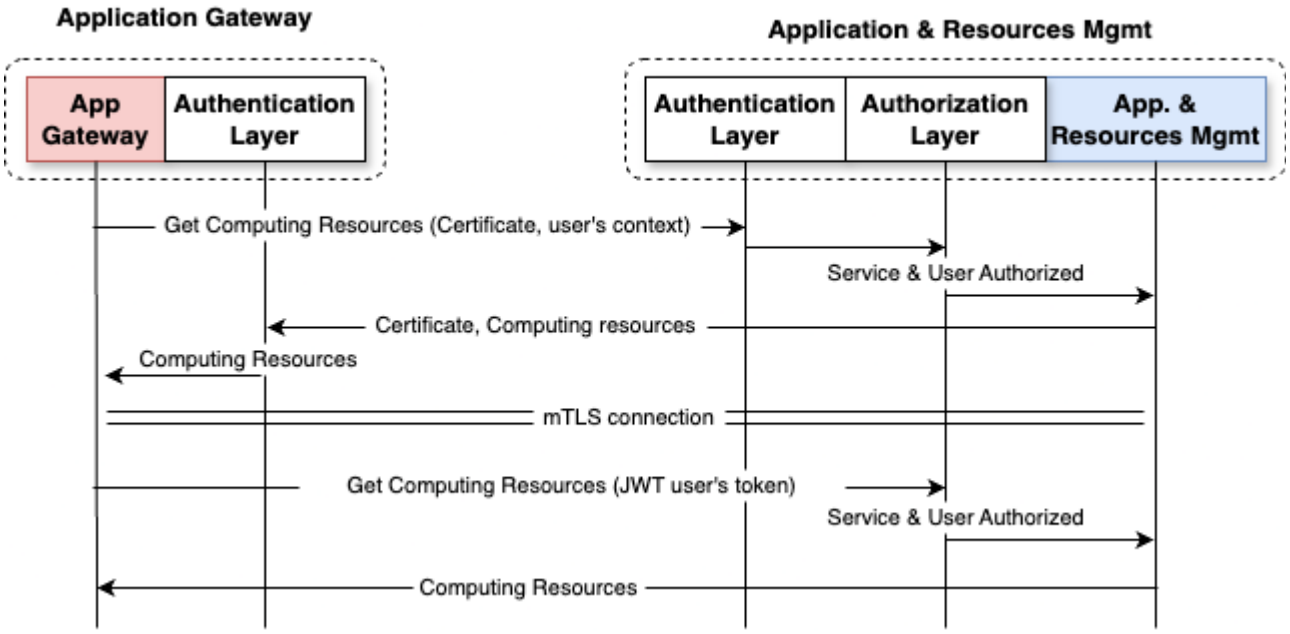


Figure 4. Secure Communication Workflow between CECCM components

3.1.4 Secure CECCM-Federated Infrastructure Communication

In a typical deployment of multiservice software, it is common to encounter scenarios involving multiple trust domains. By "trust domain," we refer to a collection of services that place trust in a single IdP. From a security standpoint, when one service communicates with another within the same trust domain, both services may trust the same IdP or certificate authority. Leveraging this mutual trust, the recipient service can verify a security token received from a calling service.

In our specific case, there are instances where the CECCM needs to establish connections with the NB APIs of LMS within the federated infrastructure. In such a case, the different infrastructures LMSs need to provide certificates delivered by a trusted public certificate authority to ensure the authenticity of each infrastructure. In addition, the PDP or the security layer of the CECCM adaptation and federation component can encrypt a secret key using the LMS public key and send it to the infrastructure LMS for further communication tunnel encryption. This process is illustrated in Figure 3.

As mentioned previously, the security of communications and agreements among the federation actors is out of the scope of AC^3. However, communications between the CECCM and the various LMSs need to be secured to at least ensure the authenticity of the LMS and the confidentiality of the information exchanged. The following workflow diagram (see Figure 5) describes an example where the Adaptation Agent initiates a secure connection with an LMS for an application onboarding scenario. In order to authenticate the LMS and trust its certificate, we assume that CECCM trusts the public certificate authority that signed the LMS certificate. This certificate not only authenticates the LMS, but also uses its public key for session key encryption. Once a secure session is established, the adaptation agent can interact with the LMS's NBI endpoint.

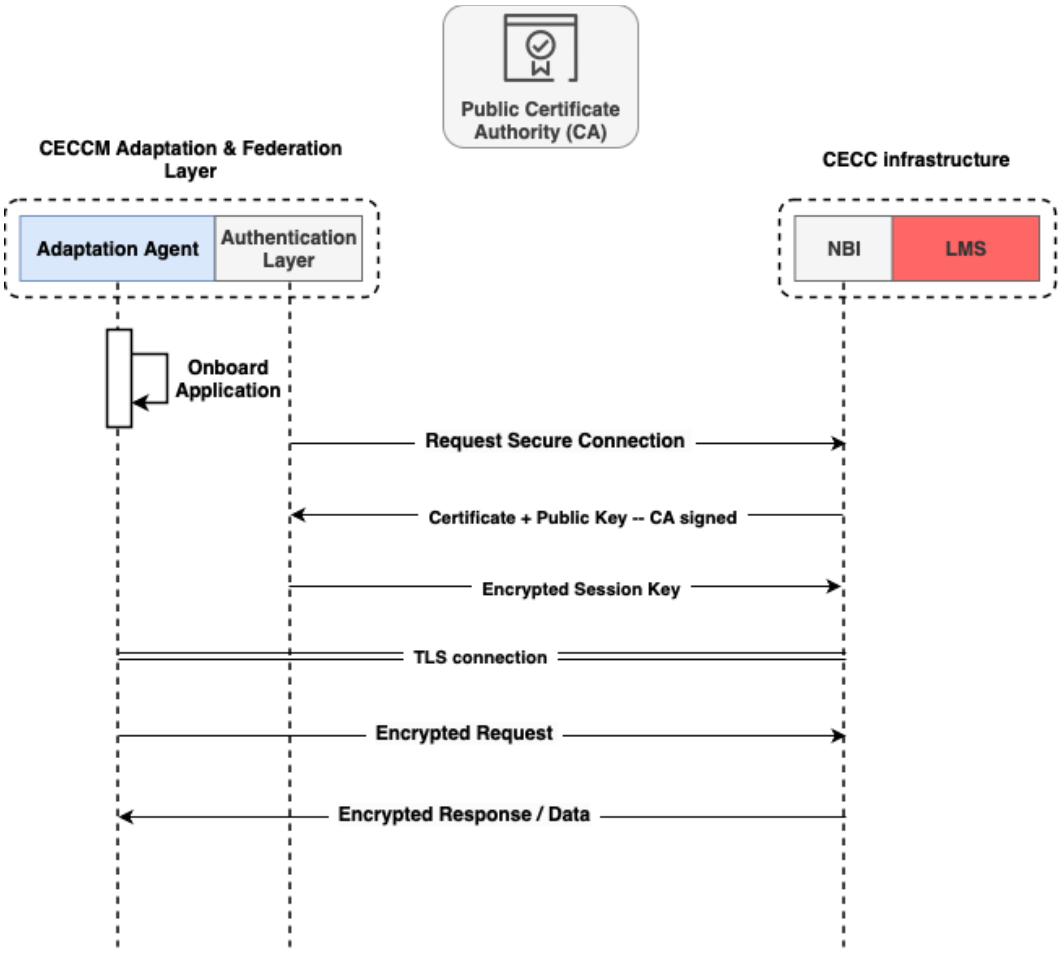


Figure 5. Secure Communication Workflow between CECCM and CECC infrastrcuture

3.2 API Security

In D2.5 [2] we explore the capabilities of NetScaler CPX regarding API security and its potential contributions to the CECCM. In this deliverable, we follow a more practical approach that focuses on the application of CPX within the AC^3 CECCM environment. We describe the specific points in the security architecture where CPX can be integrated, the API security functions it provides, and the key features that deliver value.

3.2.1 AC^3 API Security

The AC^3 API Security Framework builds on foundational principles to create a secure and adaptable environment tailored to its needs. Specific components within AC^3 leverage these features, as represented in Figure 1, to ensure robust protection and seamless API interactions.

NetScaler CPX is well-positioned to play a pivotal role in enabling the aforementioned security capabilities. Specifically, Figure 6 illustrates the CPX placement in the CECCM.

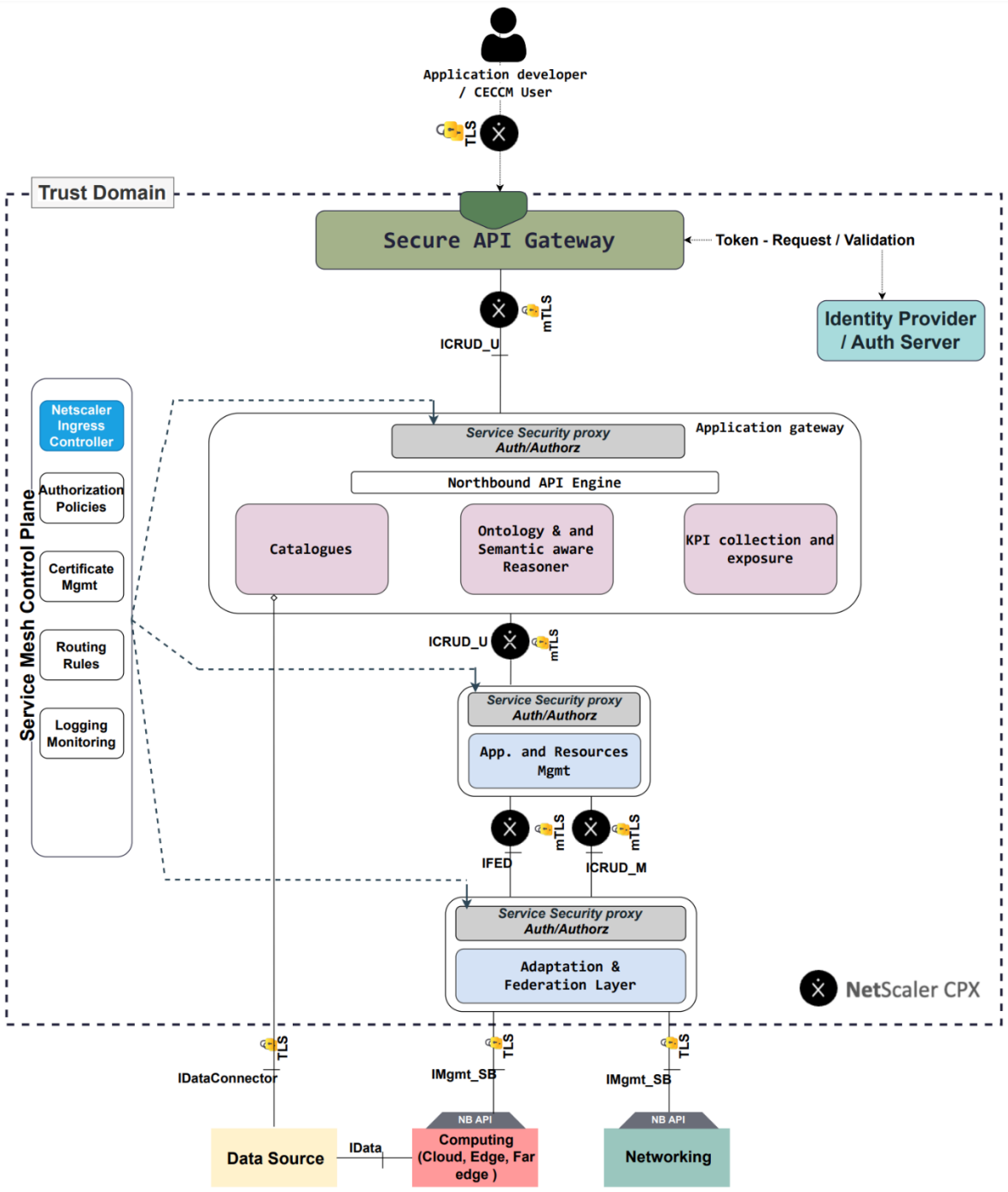


Figure 6. NetScaler CPX Placement in the CECCM

There are two main components illustrated in the diagram above:

- **NetScaler CPX**, which is used to intercept and manage both north-south traffic, between the NBI clients (application developers, API clients, etc.) and the CECCM as well as east-west service-to-service traffic between the CECCM component.

- **NetScaler Ingress Controller,** which integrates with Kubernetes fabric, if applicable, to allow for a desired state deployment and configuration of the CPX.

3.2.1.1 North-south traffic security

North-south commonly refers to traffic entering the system from external users. Acting as a bridge between external users and internal services, CPX can ensure robust API security by integrating with the Secure API Gateway and the IdP for token validation and request routing. Key CPX capabilities to protect the Secure Gateway and provide a frontline of defense against malicious attackers of the NBI include the Web Application Firewall (WAF), API Validation, OpenID Connect (OIDC) with OAuth2 and Geo-location filtering. WAF is a firewall that safeguards the gateway by filtering incoming HTTP requests and blocking common threats such as SQL injections and cross-site scripting attacks. API Validation checks incoming requests against established schemas and access policies to prevent unauthorized data and actions from reaching sensitive resources. OIDC**/**OAuth2 authentication protocols are crucial for managing access at the user level. OIDC provides identity verification for users, while OAuth2 handles authorization, offering fine-grained access control across the API's components. Last, but not least, geolocation identification provides a rudimentary but powerful protection against attacks originated by malicious parties in unexpected geo(s).

3.2.1.2 East-west traffic security

East-west commonly refers to traffic between microservices. Securing CECCM service-to-service interactions employs a combination of encryption and traffic management measures. mTLS establishes the identity of the client and server involved in internal service-to-service communications, thus securing interactions within the system. Transport Security (TS) refers to the protection of information against interception during transmission. Finally, Rate Limiting (RL) and rate shaping can throttle either the number of API requests or throughput consumed to reduce the risk of overloading resources and the respective threat of denial-of-service attacks.

3.2.2 Deep Dive into NetScaler Capabilities

NetScaler CPX provides a complete, vendor-agnostic solution for API security. It integrates seamlessly into distributed and microservices architectures, and its capabilities cover many key fields, ensuring secure, efficient, and scalable application delivery.

3.2.2.1 API Policies & Validation

API validation ensures secure and efficient management of network traffic while preventing invalid or malicious requests from reaching backend services. These capabilities are enhanced with vendor-agnostic configurations, focusing on dynamic traffic management, declarative rules, and compliance with validation standards. **Schema-Based Validation** provides built-in support for OpenAPI specifications and ensures compliance with methods, numbers, and required parameters. Query strings, endpoint whitelisting, and method-specific rules are validated to prevent malformed or malicious requests. **Dynamic Rules** supports adaptive traffic management through declarative rules, reducing misbehavior caused by invalid requests. Includes thresholds for traffic shaping and rate limiting to prevent service overloads. Automated responses like "429 Too Many Requests" enhance system reliability. **Role Definitions** and Access Personas allow granular, role-based access control for administrators, developers, and operators, enabling secure policy definition and modification. **Mutual TLS (mTLS)** ensures trusted interactions by providing identification during client and server communication. Backend services get more protection through secure mTLS protocols. **Threat Detection and Mitigation** where Web Application Firewalls (WAF) detect and mitigate vulnerabilities, including SQL injection and cross-site scripting (XSS). **Bot Management** protects against automated threats by analyzing and blocking suspicious bot traffic. Bot Management stops automated attacks by analyzing and blocking suspicious bot traffic.

3.2.2.2 Microservices Authorization

In a distributed system, robust authorization mechanisms are necessary for protecting internal and external links of any platform. Important aspects include Identity-Based Access Controls to secure the protection of sensitive endpoints from unauthorized access. Integration with diverse methods of authentication allows their flexible use in existing architectures. **Scoped Permissions** offers fine-grained access restrictions suitable for microservices, reducing the risk of unauthorized service interactions. **Token-Based Authentication s**upports secure token protocols like OAuth2, JWT, and OpenID Connect (OIDC). Multi-factor authentication (MFA) provides extra stages of verification to improve security. **Audit and Compliance** tracks all authentication and authorization events for regulatory compliance. Easily integrates with centralized Security Information and Event Management (SIEM) systems for detailed monitoring and reviews.

3.2.2.3 Traffic Management and Advanced Security

NetScaler CPX enhances performance and security through traffic management and proactive threat protection. **Rate Limiting and Traffic Shaping are** configurable tools that manage high volumes of traffic and mitigate spikes or Distributed Denial of Service (DDoS) attacks. It also ensures continuity of service by enforcing dynamic rate limits. **TLS Offloading s**ecures both internal and external communications with advanced encryption while offloading cryptographic processing to optimize backend performance. **Dynamic Filtering** and **DDoS Mitigation** provide real-time filtering mechanisms for emerging cyber threats and Inbuilt DDoS protection ensures robust security of applications.

3.2.2.4 Application Security & Monitoring

NetScaler CPX provides robust features for protecting sensitive data and ensuring high application reliability. Data Protection implements advanced encryption and configurable WAF rules to safeguard sensitive client and business data. **Proactive Monitoring** enables detailed logging and traffic analysis for real-time threat detection. Integrates with monitoring tools to streamline incident response. Finally, Compliance Reporting monitors API traffic and security events for audit and compliance purposes, by following industry standards.

3.2.3 Gateway API

The Gateway API [5] represents a modern, cloud-native evolution of traffic management, offering rich capabilities that extend the NetScaler offering significantly. This is a new contribution to Netscaler, in the context of AC^3 project, which increases the Technical Readiness Level (TRL) of NetScaler by adopting the most recent, vendor-agnostic Kubernetes-native standards to answer the needs of modern applications. By integrating Gateway API features, NetScaler will address dynamic environments with improved interoperability, scalability, and security. Gateway API introduces the following key capabilities:

- Declarative Security Policies: Simplified, intent-based definitions for traffic and API management.
- Scalable and Flexible Architecture: Custom resources add extensibility in order to ensure adaptivity to dynamic requirements.
- Stronger Security Postures: Stronger authentication and authorization, rate-limiting, and data encryption processes.
- Vendor Independence: Better interoperability in multi-cloud and hybrid ecosystems, thus reducing vendor lock-in.

The Gateway API extends traditional Kubernetes Ingress by solving a number of limitations and adding new functionality. The **Feature Scope** of Ingress has a narrow scope, mostly for basic traffic routing, as it also lacks configuration for rich use cases, whereas the Gateway API has a wide scope, covering comprehensive traffic

control, security, and extensibility to diverse routing, traffic policy, and advanced use cases. In terms of **Granularity,** Ingress does not have fine-grained role-based configuration and application of policies, while the Gateway API provides them hence permitting customized access control. For **Scalability**, Ingress is suitable for small deployments as it lacks extensibility for large-scale complex environments. On the other hand, Gateway API is designed for scalability and extensibility to support dynamic traffic loads and future-proof with custom resource definitions (CRDs). Regarding **Security Features**, ingress provides Basic TLS and limited security configurations, while Gateway API includes mTLS, token-based authentication (JWT, OAuth2), and out-of-the-box WAF features. Finally, Ingress does not provide good Interoperability as it is tightly coupled with specific Kubernetes implementations while the Gateway API is designed to be vendor-neutral, which allows for easy integrations across an enormous variety of platforms and tooling, thus increasing the flexibility.

Adopting the Gateway API for the AC^3 project has transformative benefits for both the project and its consumers. It enhances security through features like mTLS, RBAC, and rate-limiting, ensuring robust APIs for interactions between consumers. Operations become easier to manage with declarative configuration and centralized policy enforcement, which reduces overhead while maintaining consistent application of security and traffic policies across environments. The vendor-neutral design frees consumers from reliance on proprietary solutions, giving them the flexibility to choose among compatible services. This also prevents lock-in, enabling organizations to migrate to other providers or integrate new tools with ease. Built-in logging and monitoring features improve compliance by simplifying adherence to regulatory standards, providing users with confidence in their data protection practices. By incorporating the Gateway API, the AC^3 project aligns with modern Kubernetes standards, offering improved security, flexibility, and scalability while further advancing NetScaler's technical readiness.

3.3 Security of Data Management

Developing a secure application involves implementing numerous protective measures, with the utmost significance placed on those that ensure the security of the application's data. These security measures related to data are also the most challenging to put into practice. When it comes to safeguarding application data, two distinct categories (refer to Figure 7) necessitate protection: data in motion and data at rest. While in use data is safeguarded by the security measures implemented in the AC^3 applications. **Data in motion** refers to data that is actively being transported from one place to another, whether it is over the internet or within a private network. Ensuring the security of data during its journey from one network to another or during the transfer from a local storage device to a cloud storage device is referred to as data protection in transit. Regardless of where data is on the move, it's essential to implement effective data protection measures for data in transit, as it is typically considered less secure while in the process of being transferred. **Data at rest** refers to data that is currently not in motion, meaning it is not actively being transferred between devices or networks. This includes data stored on devices like hard drives, laptops, flash drives, or data that is archived or stored in some other way. Data protection at rest focuses on securing this dormant data, regardless of the device or network where it resides. Although data at rest is sometimes perceived as less vulnerable compared to data in transit, attackers often consider it a more attractive target than data in motion. The level of risk associated with data in transit or data at rest is contingent on the security measures in place to protect data in either state. **Data in use** refers to any data that is actively being processed, accessed, or manipulated by computer systems and applications. Unlike data at rest, which is stored on physical or digital media, or data in motion, which is data being transferred from one location to another, data in use is the information currently being handled by an application's CPU and memory. This can include operations such as calculations, transformations, and the temporary staging of data during execution processes.

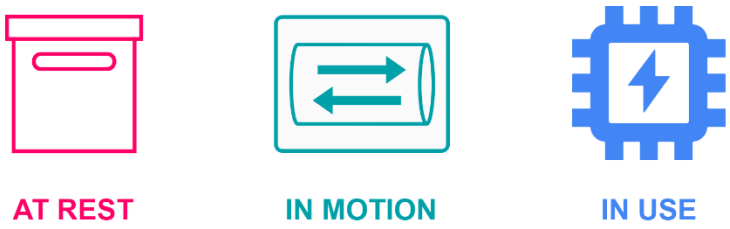


Figure 7. The three (3) States of Data

3.3.1 In Motion Data Security

Ensuring the security of data in motion—data actively moving from one location to another within the network—is a critical component of AC^3's data management strategy. As data transits from data sources to application environments, it is potentially vulnerable to interception, manipulation, or loss. To mitigate these risks, AC^3 implements robust security protocols for both COLD data, which does not require real-time processing, and HOT data, which is associated with streaming and requires immediate handling. This section explores the measures in place to secure data in motion and the role of developers in maintaining data integrity throughout this process.

For COLD data, AC^3 utilizes HTTPS endpoints to securely transfer data from the source to the application environment. HTTPS ensures that data is encrypted during its transit, safeguarding the confidentiality and integrity of the data by protecting it from eavesdropping, tampering, and forgery. This encryption protocol is crucial for preventing unauthorized access and maintaining trust in the data's authenticity and security as it moves through potentially insecure networks.

In the case of HOT data, which needs to be processed in real-time or near-real-time, AC^3 employs secure message exchange protocols designed for high-performance and secure data streaming. These protocols, HTTPS, Advanced Message Queuing Protocol (AMQP)/Secure Socket Layer (SSL) or others) ensure that data remains encrypted during transmission and is only accessible to authorized systems and personnel. The use of such protocols is essential in environments where data must be quickly and securely moved to keep up with real-time processing demands without sacrificing security.

Data Connectors act as a secure gateway at both ends of the data transmission path: initiating secure data transfers from the source and ensuring the secure reception of data at the consumer's end. They encapsulate data with security protocols (HTTPS, AMQP/SSL, or others) that protect it from unauthorized access, tampering, and interception during transit. By doing so, the Connector provides a continuous shield of protection that spans the entire journey of the data from source to destination. The Connectors leverage state-of-the-art security tools and protocols to maintain the confidentiality, integrity, and availability of data in motion, depending on their use case. Some of the key technologies and methods used include:

1. TLS/SSL Encryption: For all data in transit, the Connector uses TLS and SSL encryption protocols. These protocols create a secure channel over the internet, ensuring that the data sent between the data source and consumer is encrypted and inaccessible to eavesdroppers.
2. OAuth2 for Authorization: The Connector implements OAuth2 to provide secure delegated access. This is crucial when data needs to be accessed by third-party applications, ensuring that only authorized applications can access or transmit data.
3. JWT: For securely transmitting information between parties as a JSON object, JWT is used extensively. It ensures that the data can be trusted because it is digitally signed using a secret with the Hash-based Message Authentication Code (HMAC) algorithm or a public/private key pair using the Rivest-Shamir-Adleman (RSA) algorithm.

The Connector is not merely a passive component but an active participant in the data security framework of the AC^3 project. Its implementation of comprehensive, multi-layered security measures ensures that all data in motion is robustly protected against a wide range of cyber threats. Moreover, its placement at both the sending and receiving ends of data transfers assures that security is not a one-sided affair but a holistic strategy encompassing every aspect of data movement.

Once data is transferred into the AC^3 environment and reaches the developer's application, maintaining security over these data exchanges remains paramount. Developers are responsible for implementing secure interfaces within their applications to handle incoming and outgoing data securely. This includes maintaining the encryption standards provided by AC^3 and ensuring that any data sent from the application to other components or back to the data source is protected similarly.

3.3.2 At Rest Data Security

Data at rest within AC^3 typically consists of data temporarily stored after being transferred from the data source and before processing, as well as data output by applications waiting to be either further processed or ultimately disposed of. To safeguard data at rest, AC^3 implements several robust security technologies and protocols depending on the LMS where the application is executed, including:

- **Encryption:** All data stored within the AC^3 system, whether awaiting processing or pending disposal, is encrypted using strong encryption standards such as AES (Advanced Encryption Standard) with key management practices that comply with industry security standards. This ensures that data if intercepted, remains unreadable and secure.
- **Access Controls:** The AC^3 system enforces strict access controls and authentication mechanisms. Access to stored data is tightly controlled, with permissions granted only to authenticated and authorized users. This minimizes the risk of data leakage or unauthorized data manipulation.
- **Secure Storage Locations:** Data is stored in secure storage locations that are regularly audited for vulnerabilities and compliance with security policies. These storage locations are equipped with physical and logical security measures to protect against a range of threats.
- **Data Disposal:** Once data is no longer needed, or after it has been processed, it is securely disposed of according to predefined retention policies that comply with legal and regulatory requirements defined by the provider of the data. Secure deletion practices ensure that disposed data cannot be recovered or misused.

While the AC^3 project provides secure storage and encryption facilities, developers are also responsible for implementing additional security measures within their applications. This includes ensuring that any data written to or read from the AC^3 storage locations is handled securely within the application and that all data exchanges are conducted through secure interfaces. Developers must also adhere to best practices in software security to prevent vulnerabilities that could compromise data at rest.

3.3.3 In Use Data Security

Ensuring data security during its active use is crucial for maintaining confidentiality, integrity, and availability. In the context of the AC^3 project, data in use is protected through stringent measures designed to isolate and secure it from external threats during application execution. This section details the approaches and responsibilities associated with securing in-use data.

To safeguard data while it is actively processed by applications, AC^3 employs a combination of virtualization and containerization technologies. Applications are created in isolated environments, ensuring that the data in use cannot be accessed or tampered with by unauthorized entities. Meanwhile, containerization packages an

application with all its dependencies, creating a consistent running environment across various computing platforms. This isolation is crucial for protecting sensitive data from lateral movement within the network or from potential leaks across different application layers. Containerization, in particular, enhances security by enforcing strict access controls and resource limitations. Each container operates as an independent unit with a minimal attack surface, reducing the risk of unauthorized data access. The encapsulation of applications in containers ensures that they run in a dedicated space, separated from the host system and other containers. This isolation prevents malicious processes from affecting or accessing the data processed by other applications.

While AC^3 provides a secure infrastructure for application execution, the actual implementation of the application's business logic, including specific data security measures during its use, remains the responsibility of the application developer. It is the developer's duty to ensure that their applications adhere to best security practices, such as implementing secure coding techniques, validating inputs to prevent injection attacks, and using encryption to protect data at the application layer. This responsibility is critical because while AC^3 can secure the environment and the data at the infrastructure level, the security of the data, as it is processed directly by the application, depends largely on the security measures embedded within the application itself.

Figure 8 shows an overview of how the protocols and security elements presented above are used by the AC^3 Data Management Application Addons to ensure that data remains secure at all points in the lifecycle of an AC^3 application.

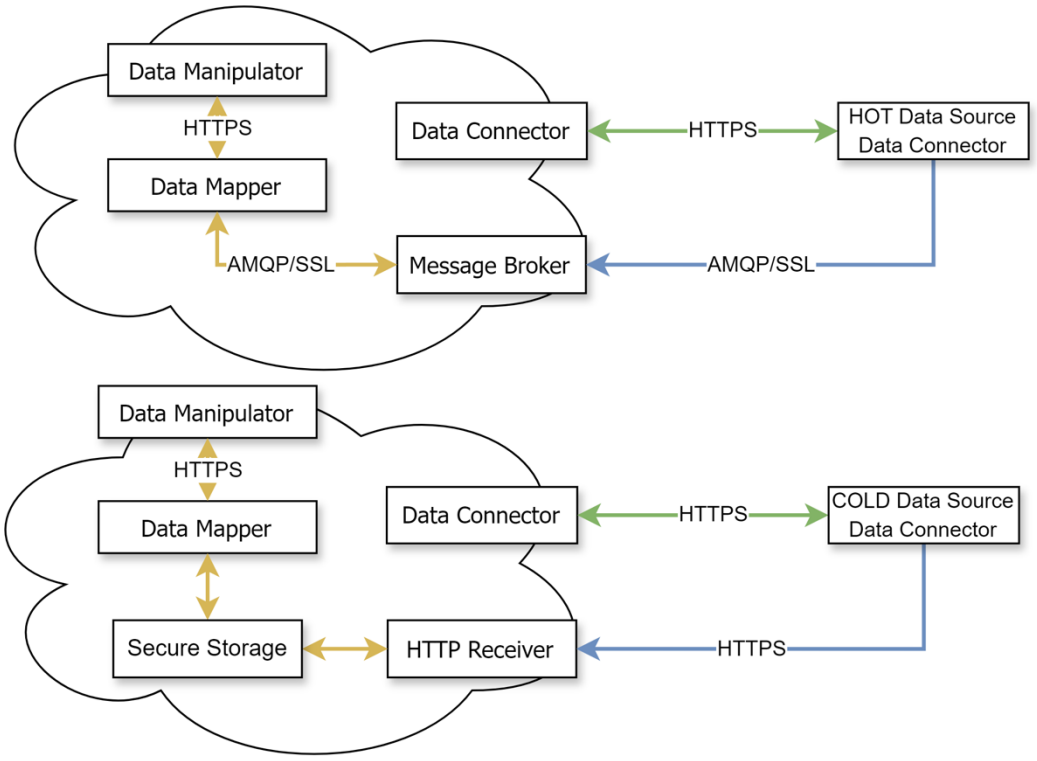


Figure 8. Data Security elements in HOT and COLD data cases.

3.3.4 The role of the digital contracts between data sources and data consumers

In AC^3, the framework for data exchange between data providers and consumers is meticulously governed by digital contracts. These contracts are essential in defining the terms and conditions under which data is provided and consumed, ensuring that all parties adhere to agreed-upon rules and standards. Digital contracts serve as

formal agreements that specify the modalities of data exchange between the data sources (providers) and the applications (consumers). These contracts are digitally signed agreements that outline various aspects of the data exchange, including data types, volumes, frequency of data delivery, quality standards, and security requirements. By setting these parameters, digital contracts ensure that both providers and consumers have a clear understanding of their obligations and the expectations from both sides. Some of the features they can provide include the following:

- **Access and Usage Rights**: The contract specifies who can access the data and for what purposes. It includes limitations and rights concerning data usage, helping to prevent misuse and ensure compliance with privacy laws and regulations.
- **Data Security and Compliance**: Security requirements are a crucial component of digital contracts, dictating the security measures that must be in place to protect the data during exchange and when at rest. These specifications help in aligning with industry standards and regulatory requirements.
- **Quality and Reliability**: Digital contracts also define the standards for data quality and reliability that must be met by the providers, ensuring that the consumers receive data that is accurate and fit for their applications.
- **Costs and Penalties**: In some cases, the contracts may include cost details for data access and penalties for non-compliance with the terms of the contract, providing a financial framework for the data exchange.

It is important to note that while AC^3 facilitates the infrastructure for secure data exchange, the actual contracts are established directly between the developers (consumers) and the data providers. AC^3 does not govern the content of these contracts; rather, it provides the technological and security framework within which these agreements operate. This arrangement allows developers and data providers to independently negotiate terms that best suit their needs and requirements, without interference from the AC^3 system administration. The responsibility for adhering to the terms of the digital contract lies with the data providers and the developers. Data providers must ensure that they supply data that adheres to the agreed specifications and security protocols. On the other hand, developers are responsible for using the data in accordance with the contract, including respecting any limitations on data use and ensuring that any further data handling complies with the specified security measures.

Digital contracts are a cornerstone of data governance in AC^3, providing a structured and secure framework for data exchanges. These contracts not only define the operational and legal boundaries of data use but also enhance trust among parties by ensuring data is exchanged under mutually agreed-upon terms. By facilitating independent contractual relationships within a secure technological environment, AC^3 supports robust, compliant, and efficient data utilization across its platform.

3.4 Security of Micro Service Migration

One of the goals of the AC^3 project is to simplify the lifecycle management of containerized microservices applications. One key solution for ensuring a seamless application lifecycle is service migration when necessary. In AC^3, the stateful migration process is detailed in D3.1 [3]. The approach involves creating a checkpoint of the application to be migrated on the source node once a migration decision is triggered. This checkpoint, encapsulating the application's state in the form of a container image, is then stored in an accessible image registry. From there, it can be retrieved by the destination node to complete the migration process. The intermediate image, stored in the registry, includes the application's data, which may vary depending on the application. It could contain sensitive information, such as user data and microservices components. Since this image is stored in an image registry, unauthorized access by untrusted entities could have harmful consequences.

To mitigate this risk, the migration process described in D3.1 requires an additional layer of security to protect the data during storage and transfer.

As stated in the previous sections, all communication between entities—whether internal CECCM components or interactions between the CECCM and external entities—is encrypted using TLS. This ensures the security of image transfers during migration. However, once the image is successfully pushed to the registry, access control measures must be implemented to restrict access exclusively to the destination server authorized to pull the image. Consequently, securing stateful microservices migration primarily depends on safeguarding access to the image registry.

To achieve this, we opted to provision a central image registry, as illustrated in Figure 9, which serves as a temporary storage for images prior to their migration to the destination server. As described in section 3.1.4, authentication within the infrastructure—primarily between the LMSs and CECCM components, such as the adaptation agents—is managed via an IdP backed by a trusted certification authority. To centralize authentication and authorization, we delegated access control for the central image registry to the IdP. This design allows the IdP to function as the authentication server for the image registry, ensuring secure and streamlined access management.

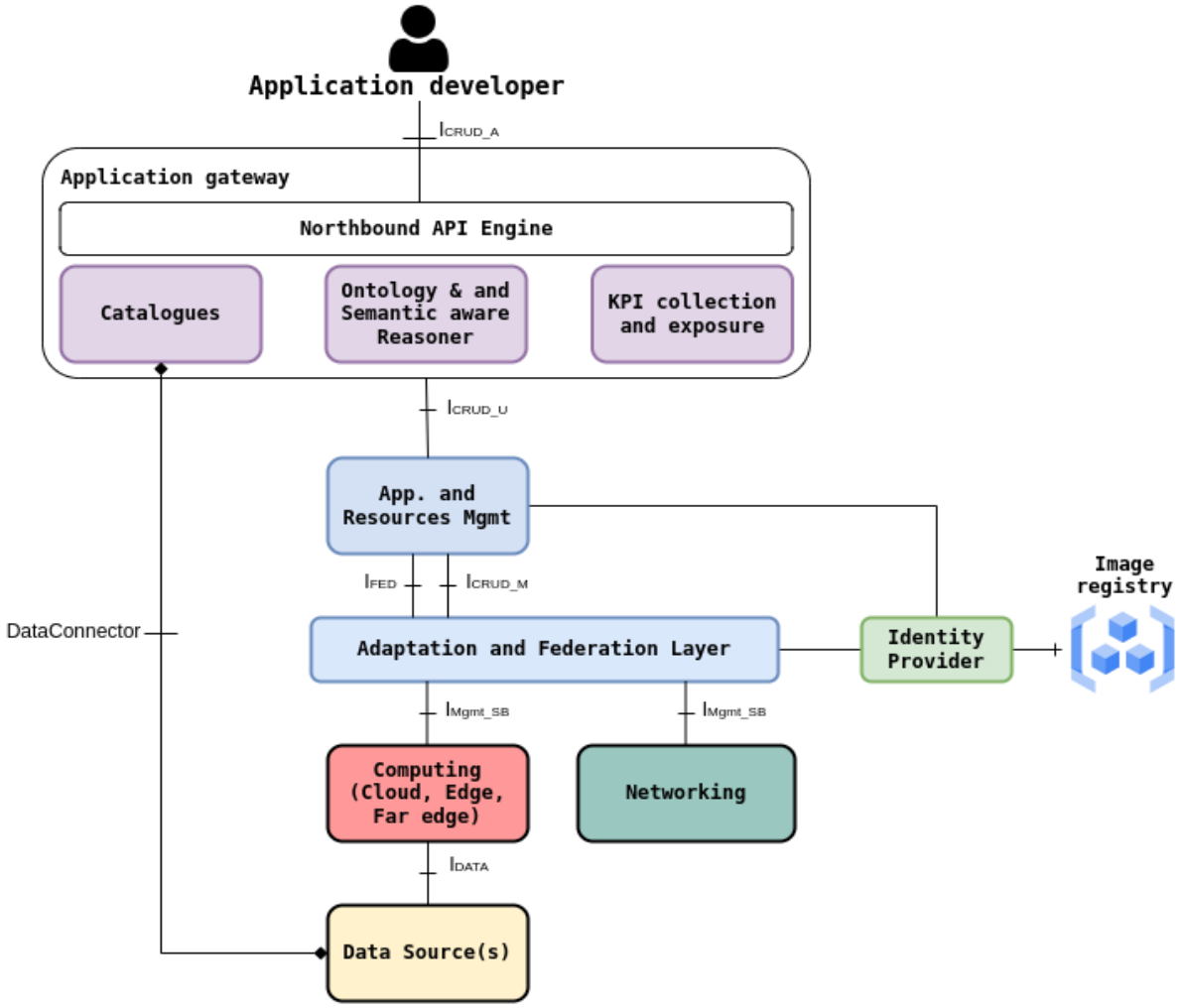


Figure 9. Central Image registry serving for storing application states during the migration

When a migration decision is made, the LCM determines which service to migrate and its destination node. It then sends the request to the decision enforcement component. The decision enforcement instructs the source node, through the adaptation gateway, to upload the application data to the provisioned image registry. Once the upload is complete, the decision enforcement generates credentials for the destination node and directs the IdP to grant it read access to the image registry. These credentials are then communicated to the local system by the decision enforcement, enabling it to access the registry. The LMS authenticates itself to the image registry through the IdP, retrieves the image, and initiates its use. This workflow is illustrated in Figure 10.

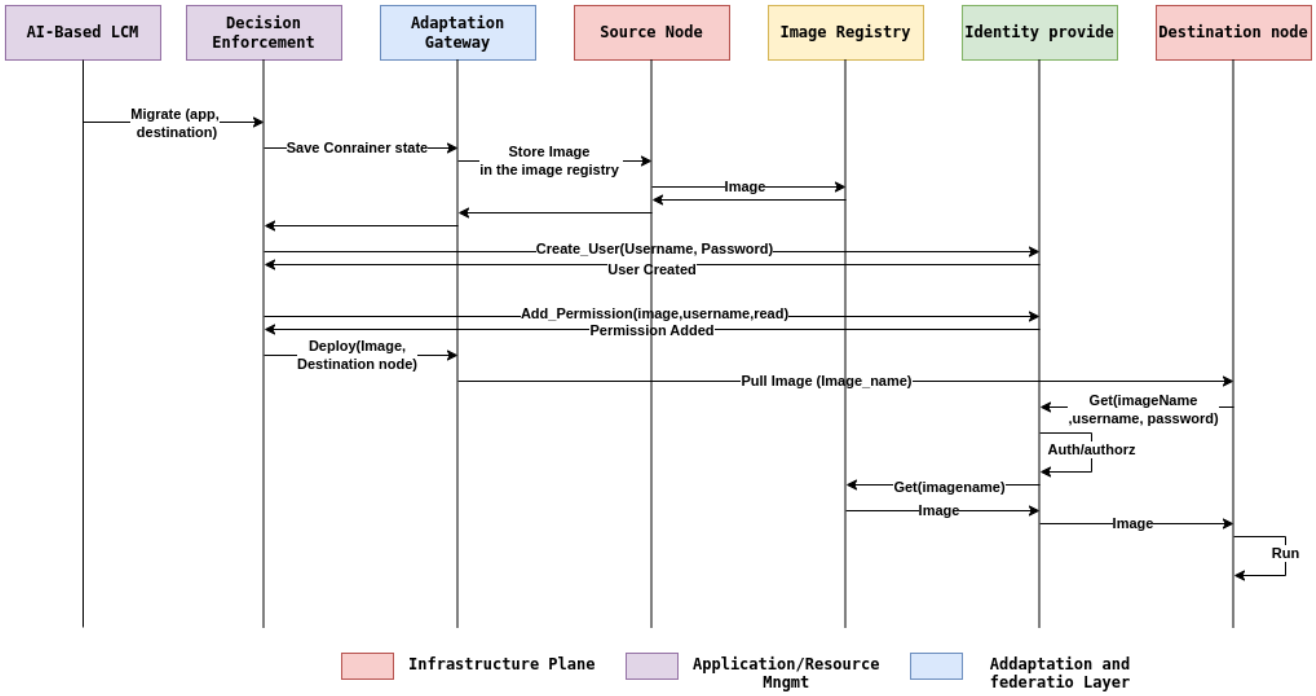


Figure 10 - Securing Migration workflow

4 AC³ Trust

Trust is the foundation of collaboration within federated systems like CECC, where multiple stakeholders, such as infrastructure providers, application developers, and service orchestrators, must work together seamlessly. In this context, trust refers to the confidence in the ability of stakeholders to fulfill their roles and obligations, particularly in meeting SLAs and ensuring reliable operations. A trust profile is a quantified representation of this confidence based on historical performance, SLA compliance, and other relevant metrics, which guide decision-making within the federation.

To foster trust among the diverse stakeholders that provide CECC infrastructure, AC³ introduced enhancements to the overall architecture by incorporating trust management functionalities in D2.5 [2]. These functionalities aim to establish and maintain trust profiles for the various infrastructure providers participating in the federation. These profiles quantify the level of trust assigned to each provider based on their ability to support and validate the SLA agreements established between the CECCM, the infrastructure providers, and application developers.

As outlined in D2.1 [1], application developers specify application components, including details about container images, microservice configurations, and, critically, the SLA. SLAs typically define performance expectations for microservices, including key parameters such as service availability, link capacity, and latency requirements between microservices. The CECCM leverages this SLA to guide the deployment of microservice components across the CECC infrastructure.

The deployment process involves selecting infrastructure providers from the federation to host specific microservices. Traditionally, this selection is handled by the Resource Broker, a CECCM component, which evaluates providers based on available resources and associated costs. In AC³, however, we extend this decision-making process by incorporating the infrastructure providers' reputation as an additional criterion. This reputation reflects the providers' historical performance in meeting SLA requirements, thereby enhancing trust in the federation's operation.

One of the key innovations of the AC³ project is the separation of resource provisioning from application management. The CECCM owner can deploy and manage applications on the CECC infrastructure without owning the underlying physical resources. This separation is made possible through the use of a federation of resources, where interoperability and collaboration are governed by the IEEE Standard for SIIF [6].

To support this, AC³ adopts the FHS reference model in this deliverable, which integrates a Trust Manager component. The Trust Manager operates as an external entity within the CECCM ecosystem, interfacing with federation members (e.g., application developers, CECCM, and infrastructure providers) via standardized APIs. This framework ensures robust trust management and seamless communication among all parties, reinforcing the reliability and scalability of the CECC infrastructure.

Indeed, the architecture proposed in this deliverable introduces significant enhancements compared to the one presented in the previous deliverable on trust and security in the CECCM [2]. While the earlier architecture primarily focused on foundational components of the Trust Manager and infrastructure monitoring, the current model incorporates advanced features to address scalability, robustness, and trust management within the CEC environment. Specifically, it integrates the Trust Manager component into the FHS model of the IEEE SIIF standard. These advancements collectively enhance the system's reliability and trustworthiness, aligning with the objectives outlined for this phase of the project.

4.1 AC³ Trust architecture

To build and maintain the reputation of infrastructure providers, we propose a novel trust architecture, as shown in Figure 11, seamlessly integrated into the FHS framework, as defined by the IEEE SIIF. This architecture

complements the existing AC³ framework and introduces a specialized component, the Trust Manager, as an independent entity hosted by the Federation Manager (refer to D2.1). The Trust Manager plays a key role in deriving and managing the reputation of infrastructure providers, leveraging the FHS infrastructure to ensure interoperability and scalability.

The proposed architecture expands upon existing AC³ components—such as KPI collection/exposure for monitoring and the Application and Federation Layer (which includes the Resource Broker, among others)—by integrating the following modules within the Trust Manager:

1. **KPI Monitoring Module:** This module collects and aggregates monitoring data related to SLA performance from all key stakeholders, including the CECCM, infrastructure providers, and application developers. By analyzing this data, the module can detect SLA violations and identify which component of the federated infrastructure is the responsible entity for said violations. This multi-provider monitoring strategy enhances the AC³ architecture's flexibility in deploying microservice components across multiple infrastructure providers while ensuring comprehensive insights into SLA adherence across the entire federation.
2. **SLA Management Module:** This module uses collected KPI data to detect SLA violations automatically. To achieve this, Smart Contracts formalize SLAs and dynamically monitor compliance. Automating SLA enforcement provides real-time insights into SLA violations, fostering a reliable trust framework.
3. **Feedback Module:** This module collects input from end users (i.e., application developers) regarding their service experiences (e.g. service load time). Feedback is gathered periodically or upon application termination and is used to evaluate the quality of service delivered by infrastructure providers.
4. **Trust Management Module:** This core module integrates outputs from the SLA Management and Feedback Modules to calculate and update the reputation of infrastructure providers.

Reputation scores are securely recorded on a blockchain, ensuring transparency and tamper-proof management. These scores are made available to the Resource Broker, enabling informed resource provider selection based on trustworthiness in addition to resource availability and cost.

To ensure interoperability and scalability between the CECCM and the external Trust Manager, the Trust Manager is integrated into the FHS framework of the IEEE SIIF. Within the FHS model, the Trust Manager operates alongside key components of the IEEE SIIF framework, facilitating seamless federation-wide communication and robust trust management. The FHS integrates the following key functionalities:

1. **Authentication and Security:** Based on the AC³ security strategy in D2.5 [2], member authentication is achieved using the OpenID protocol, with Authorization Endpoints (AuthZ) ensuring secure access to FHS and the Trust Manager. Tokens and user roles are managed within the Roles & Attributes Catalog for fine-grained access control.
2. **Federation Database:** The FHS maintains a central repository for federation members and their services (i.e., Member/Service Catalog), providing authentication, authorization, and service details.
3. **Open API-Based User Feedback and KPI Monitoring Collection:** The Trust Manager consolidates feedback and monitoring data from the Application Developer, CECCM KPI Monitoring, and Infrastructure Provider KPI Monitoring modules. These inputs are processed to detect SLA violations and refine the trustworthiness of providers. Data is collected via the FHS API Server.
4. **Open API-Based Trust Score Provision:** The Trust Manager interacts with the Adaptation and Federation Layer (i.e., CECCM) to update trust scores and influence decisions made by the Resource Broker. This ensures that trust scores serve as critical inputs during resource provider selection, balancing performance, cost, and reliability. This interaction is enabled through the FHS Operator (FHSOp) Server API.

The integration of the AC^3 trust architecture into the FHS model ensures robust trust management across the federation. By combining SLA monitoring, user feedback, and blockchain-based reputation management, the architecture empowers the Resource Broker to make well-informed decisions while fostering a trustworthy and reliable CECC ecosystem. Indeed, using the IEEE SIIF standard ensures compatibility and scalability. Furthermore, the FHS-based Open API interfaces facilitate communication among federation members and ensure smooth data exchange. This enables the Trust Manager Module to connect with these endpoints to expose reputation scores and retrieve data necessary for trust computation.

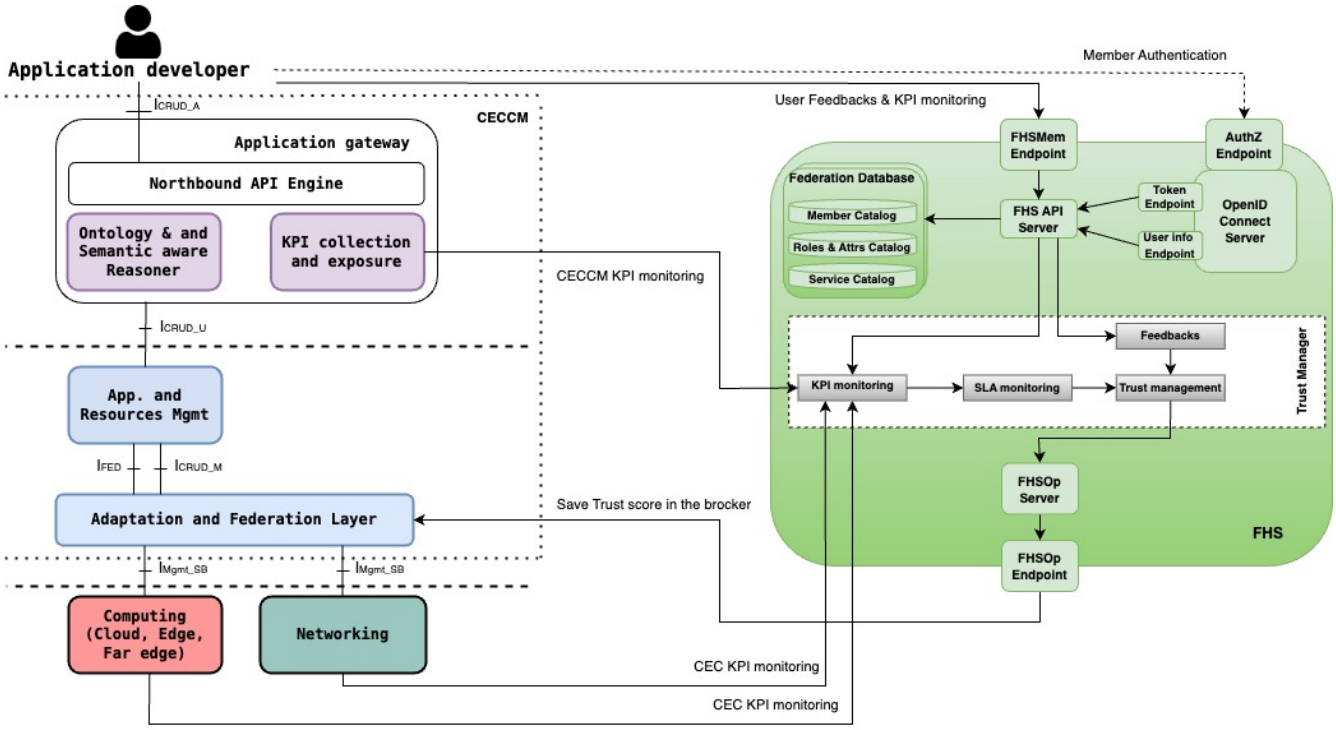


Figure 11. AC^3 Trust Architecture Leveraging the IEEE SIIF FHS Model

4.2 SLA Management

In the AC^3 project, effective management and monitoring of SLAs are critical for building the reputation of stakeholders involved in deploying microservice-based applications over the CECC infrastructure. To this end, AC^3 envisions leveraging Smart Contracts to streamline SLA management. This approach automates the SLA management process, enabling real-time detection of SLA violations. The following sections introduce key principles and delve deeper into the SLA management process.

4.2.1 SLA Definition

Within the AC^3 context, SLAs meticulously define parameters governing the expected performance of micro-services. These parameters encompass important aspects such as service availability, throughput, and latency between micro-services. Notably, this information is obtained during the application development phase when interacting with the OSR module of the CECCM. The SLA acts as a crucial document that establishes a comprehensive framework, ensuring the reliability, availability, and performance of the service. It serves as a foundation for collaboration between the service and its underlying infrastructure throw the CECCM.

The SLA initiation process involves specifying key service metrics to evaluate performance, including maximum response time, optimal uptime, and availability. Additionally, the SLA sets clear expectations for throughput

requirements, providing a solid foundation for assessing the overall success of the service. Scalability is addressed through explicit limits and guidelines, detailing how the infrastructure should dynamically scale to handle increased demand throughout the AC^3 project.

To enhance service reliability, the SLA defines acceptable error rates and redundancy requirements. This strategic approach aims to mitigate the impact of hardware failures or disruptions, ultimately enhancing the overall stability and resilience of the service. Security considerations are integral, with detailed provisions for access controls, authentication protocols, and encryption standards aligning with best practices for safeguarding sensitive information within the AC^3 project.

Provisions for data integrity and recovery are outlined, specifying the frequency of backups, Recovery Time Objectives (RTO), and procedures to restore service in the event of a failure. These measures minimize downtime and ensure consistent availability of data and services. Monitoring and reporting mechanisms are integral components, identifying tools and methodologies for monitoring infrastructure performance coupled with reporting requirements to keep stakeholders informed.

Clear communication is emphasized in the SLA, specifying communication channels and notification procedures. This ensures that stakeholders are promptly informed of any service outages, maintenance activities, or critical events, fostering transparency and accountability throughout the AC^3 project.

In summary, the SLA serves as a comprehensive technical agreement, aligning service and infrastructure components within the AC^3 project. By addressing key metrics, scalability, reliability, security, backup and recovery, monitoring, incident response, compliance, and communication, the SLA establishes a robust foundation for delivering a reliable, secure, and high-performing solution.

4.2.2 SLA Structure

In a manner akin to any contractual agreement, SLA is a meticulously organized collection of elements, encompassing both mandatory and optional components. These elements delineate various aspects, including the duration of validity, the involved contracting parties, the type of services provided, guarantees related to objectives, penalties, clauses for suspension or termination, and compensation. Figure 12 shows the SLA structure, including the following components:

- **Period of Validity:** The timeframe of an SLA is determined by the initiation and conclusion dates of the network resource lifecycle. This period can be altered in accordance with stipulated termination conditions.
- **Parties Involved**: This section outlines the entities engaged in the contractual agreement, namely the three signatory parties – Application Developer, CECCM, and CECC Infrastructure Providers. Additionally, it introduces the trusted fourth party, the trust monitoring system. The latter is instrumental in validating the proper functioning of the desired service, assessing its availability, or determining if it falls below the agreed-upon performance level stated in the SLA. This segment provides details about the parties, including their names and contact information.
- **Template:** An integral component of the SLA, the template defines parameters ensuring the realism and achievability of the offered QoS. This vital section encompasses five key elements: service type, parameters, guarantees, billing, and termination conditions.
- **Service Types:** It facilitates the diversity of services offered without interference between the services. Each service's priority variation necessitates distinct SLAs, considering different services and operational modes.
- **Parameters:** This section defines variables utilized in other parts of the contract, detailing specific elements such as metrics and monitoring types. Metrics are identified by an identifier, a description (e.g., latency, throughput, reliability), and a corresponding unit (e.g., ms, bps, %). Monitoring specifies the

nature of the evaluated metric, including aspects such as average, maximum, minimum, evaluation window, and frequency of value collection.

- **Guarantees:** This segment encompasses three key elements – requirements, terms, and penalties.
- **Requirements:** Essential specifications for the service's optimal functionality are presented here. Specifications may be mandatory or optional, detailing preliminary technical elements to be fulfilled.
- **Terms of Guarantees:** Detailing the Service Level Objectives (SLO) of the contract, this part elaborates guarantees through terms (e.g., reliability, latency, bandwidth, throughput) and objectives using operators like "Or" and "And."
- **Penalties:** In the event of a degradation in the end-to-end network performance (SLA violation), CECCM is obligated to compensate the application developers. Simultaneously, infrastructure providers involved in the SLA violation due to resource degradation must compensate the CECCM. The amount of reimbursement is mutually agreed upon and described in the penalties section.
- **Billing:** Billing generally follows two models – "pay package" or "pay as you go." In this scenario, the "pay as you go" model is adopted, where the service price depends on operating modes and penalties.

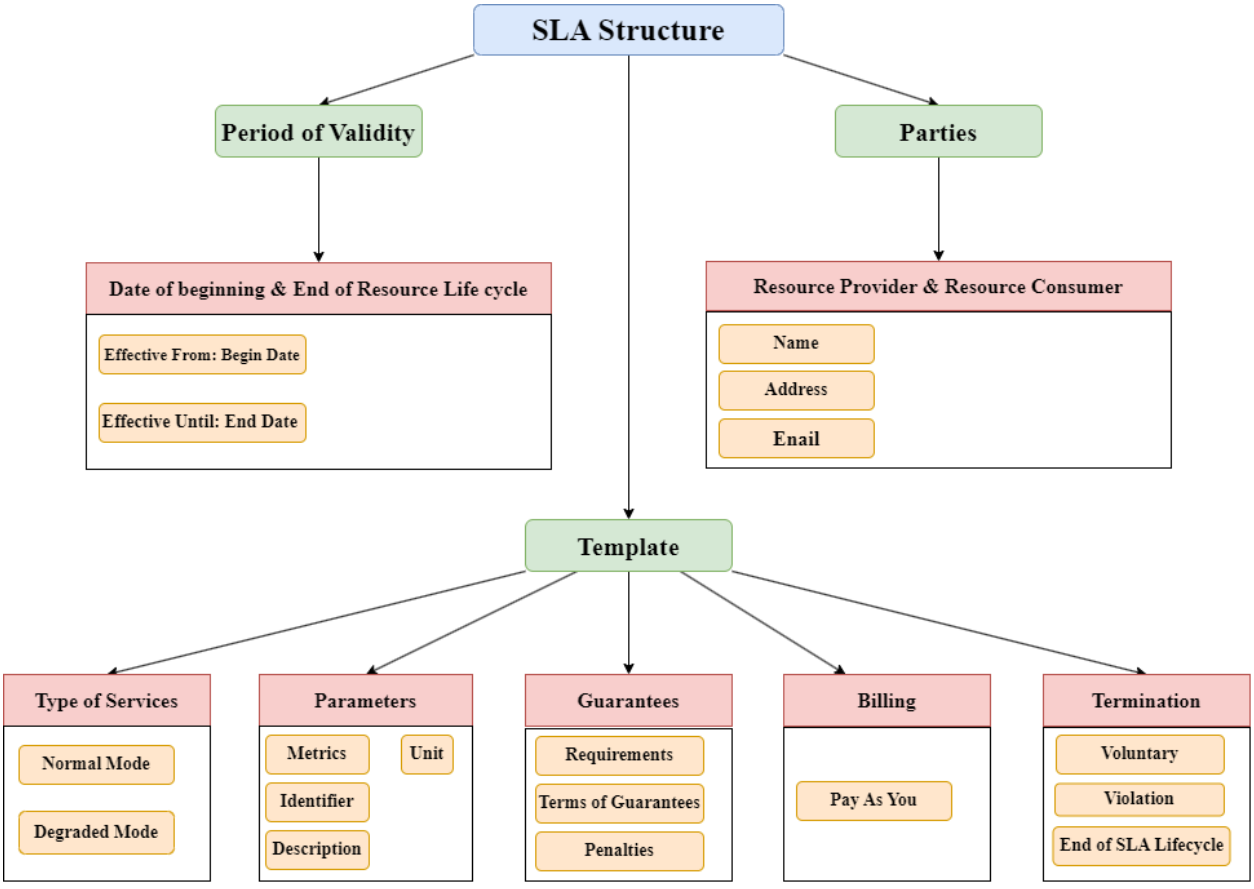


Figure 12: Service Level Agreement Structure

4.3 Trust Computation Mathematical Model

In this section, we present our mathematical model for calculating the trust score in CEC, focusing on the relationship between clients (i.e., application developers) and infrastructures (i.e., CEC infrastructure providers). For simplicity, we have not included the CECCM component in the equations, as it follows the same

representation as infrastructure KPI monitoring. However, in the AC^3 model, CECCM is incorporated to enhance the replication and reliability of the trust score. The model is designed to evaluate the performance of various KPIs over time. Our initial approach to trust computation was straightforward: we compared the actual performance metric to its target value as defined in the SLA. The outcome was then adjusted based on whether a higher or lower value was preferable for trust, as illustrated in Equation (1).

$$T_{c/i} = \frac{1}{n}\sum_{i=1}^{n} \min\left(1, \left(\frac{\text{actual}}{\text{target}}\right)^{\text{d}}\right) \quad (1)$$

Where:

- *n* is the number of KPIs
- *d = 1* if higher values are better (KPI *maximization* like throughput),
- *d = −1* if lower values are better (KPI *minimization* like latency).

This method was easy to implement and provided a trust score between 0 and 1. However, it did not account for how long a performance metric was off target or how significant the deviation was over time. As a result, this simple approach could sometimes give a misleading picture of trust.

To address deviations from expected values and ensure robust, reliable trust evaluation, the following mathematical model is introduced. This model includes the final trust calculation, the aggregation of Quality of Experience (QoE) (i.e., client feedback) and SLA metrics, normalization of individual KPIs, and adjustments to the trust score.

The final trust score, *T (t)*, is calculated by averaging the current trust score with the previous trust score from time *t−1*. This method ensures that the trust score reflects both past and present performance. The current performance is represented by the sum of the capped adjustment (CA) and the aggregated score (A), both of which are limited to a maximum value of 1 to prevent the trust score from exceeding its upper bound. By averaging the current and previous trust scores, the algorithm retains the memory of prior performance, ensuring that short-term variations—whether positive or negative do not significantly impact the overall trust score. This helps prevent sudden changes in response to brief fluctuations in performance, thereby stabilizing the trust score over time (see Equation (2)).

$$T(t) = \frac{\min(1, A + CA) + T(t-1)}{2} \quad (2)$$

Where:

- *CA*: is the Capped Adjustment and accounts for deviations in performance,
- *A*: is the Aggregated Client/Infrastructure trust scores.

The aforementioned equation has a number of complex factors, namely *A* and *CA*, which need to be broken down and simplified into simpler components.

1. **Aggregated Client/Infrastructure:** The infrastructure-side trust score and the client-side trust score are combined to create the aggregated trust score, Aggregated Client/Infrastructure trust score (*A*), which is weighted. The weights for the client and infrastructure, respectively, are w_c and w_i, which indicate how important each score is to the system as a whole. This weighted total ensures that both client and

infrastructure performance are included, representing their respective contributions to the trust computation. Depending on the needs of the system, it is possible to highlight one side over the other by adjusting the weights. This aggregate balances infrastructure information, such as server uptime, with client experience metrics, like response time, to provide a comprehensive view of the system's overall performance (See Equation (3)):

$$A = w_{\mathrm{c}} \cdot T_{\mathrm{c}} + w_{\mathrm{i}} \cdot T_{\mathrm{i}} \quad (3)$$

Where:

- w_c is the weight for the Client trust score,
- w_i is the weight for the Infrastructure trust score,
- T_c is the trust score for the client,
- T_i is the trust score for the infrastructure.

The trust scores T_i and T_c can be further refined and normalized.

2. **Normalized Trust:** In order to facilitate fair comparisons between various performance measurements, also known as KPIs, the normalized trust computation ensures that they are scaled to a common range. Equation (4) shows how well the system is functioning in comparison to expectations by dividing the actual performance of a metric by its target performance. The ratio will be greater than one if the system performs better than expected, and less than one if it performs worse than expected. The formula incorporates an exponent *d*, which modifies the ratio based on whether higher values (like throughput) or lower values (like latency) are desirable, ensuring that all measurements are correctly interpreted. Because it enables measurements with disparate units and ranges to be compared on the same scale, this normalization is essential. To prevent systems from being overrewarded for exceeding their goals, the normalized values are *capped at 1*. Finally, by averaging across all KPIs, the system's total trust score reflects performance across multiple dimensions, rather than relying on a single indicator.

$$T_{\mathrm{c/i}} = \frac{1}{n} \sum_{i=1}^{n} \min\left(1, \left(\frac{\mathrm{actual_i}}{\mathrm{target_i}}\right)^{d}\right) \quad (4)$$

Where:

- *n*: is the number of KPIs,
- $actual_i$: is the actual value of performance metric *i,*
- $target_i$: is the target value for performance metric *i* as specified in the SLA,
- *d*: adjusts the calculation based on whether higher or lower values are better:
  - *d = 1* if higher values are better (e.g., throughput),
  - *d = −1* if lower values are better (e.g., latency).

This version of the equation considers all *n* KPIs by summing the normalized trust values for each KPI and dividing by *n*, the total number of KPIs. This ensures that the overall trust score is averaged across all metrics.

3. **Capped Adjustment:** *Capped Adjustment (CA)* is a mechanism designed to ensure that adjustments to the trust score, based on performance deviations, remain within acceptable bounds. The variable r_i determines the type of adjustment (robustness or penalization), depending on how much the actual performance deviates from the target. This adjustment is then averaged over all KPIs to represent the

overall performance of the system. The performance of the system as a whole is then determined by averaging this adjustment across all KPIs. Because the *CA* is capped between *-1* and *1*, even large performance variances will not cause the trust score to fluctuate dramatically. This is necessary to keep the trust score stable and prevent any single performance issue, no matter how significant, from overly affecting the score. By capping the adjustment, the methodology ensures that trust is earned or lost gradually, reflecting consistent performance rather than sporadic variations. (Equation (5)):

$$CA = \max\left(-1, \min\left(1, w_{\text{adj}} \cdot \frac{1}{n}\sum_{i=1}^{n}\left(\left|\frac{\text{actual}_i - \text{target}_i}{\text{target}_i}\right| \cdot r_i\right)\right)\right) \quad (5)$$

In cases where the performance falls short of the target, a negative adjustment is applied, considering the duration and frequency of violations (Equation (6)):

$$\text{CA} = \max\left(-1, \min\left(1, w_{\text{adj}} \cdot \frac{1}{2n}\sum_{i=1}^{n}\left(\left|\frac{\text{actual}_i - \text{target}_i}{\text{target}_i}\right| \cdot r_i - \left(\alpha\frac{\text{duration}_i}{\text{total\_period}} + \beta\frac{\text{numberViolation}_i}{\text{total\_transactions}}\right)\right)\right)\right) \quad (6)$$

Where:

- w_{adj}: is the weight for the adjustment of trust score,
- r_i: is an adjustment type variable determining whether the adjustment is a penalization ($r = -1$) or robustness ($r = 1$),
- α and β: are constants such that $\alpha + \beta = 1$,
- $duration_i$: is the duration of time the actual value deviates from the target,
- *total_period*: is the total period over which the performance is measured *t−(t−1)*,
- $numberViolation_i$: is the number of violations during the period,
- *total_transactions*: is the total number of transactions during the period.

We selected this trust calculation approach as it offers a thorough and impartial means of assessing system performance while taking both recent and historical behavior into consideration. Since trust is inherently dynamic, it is important to employ a methodology that accounts for both current performance and past trends. By averaging the trust score at time *t* with the prior value at time *t − 1*, we ensure that the trust score evolves smoothly over time without being significantly affected by short-term volatility. The foundation of this trust calculation method is based on research into QoE measures and SLAs, both of which are essential for ensuring system reliability and customer satisfaction. In these domains, trust must be adaptable, capable of adjusting to changing workloads, issues, and performance fluctuations. To enhance this adaptability, our method incorporates capped adjustments to skillfully handle deviations, preventing extreme values from negatively impacting the trust score. To ensure comparability across various measurements, which may have different ranges or scales, we employ normalized KPIs. By combining normalization with weighted ratings for client and infrastructure trust, we can prioritize different aspects of system performance based on the environment's specific requirements. In order to maintain robustness and avoid outliers or single occurrences from affecting the trust score, the capped adjustment strategy was developed, which limits the adjustment values between *-1* and *1*. We make sure that trust accounts for issues that happen, as well as their *frequency* and *duration,* by implementing both positive and negative adjustments depending on performance deviations and the duration of those deviations.

4.4 Trust Computation Algorithm

The trust score at time *t* is determined by the following algorithm. It considers KPIs like throughput and latency.

This algorithm applies weighted aggregations, normalizes measures, and adjusts values that deviate from the target. Stability over time is ensured by averaging the current and past trust scores, reflecting the performance and dependability of the system.

Algorithm: Calculate Trust Score

```
Input: Previous trust score T(t−1), list of actual KPI values, list of target KPI
       values, list of durations, number of violations, weights for each KPI, types
       of KPI (either "higher_better" or "lower_better"), α, β
Output: Newly calculated trust score T(t)

Step 1: Calculate normalized trust values for each KPI
Initialize an empty list: normalized_T
for each actual value, target value, and KPI type
    if KPI type is "higher_better"; then
        Set direction factor d = 1
    else
        Set direction factor d = −1
    Compare the actual value to the target value, adjusting based on direction factor d
    Append the normalized result to normalized_T
end

Step 2: Compute the aggregated trust score for client and infrastructure
Compute the aggregated trust score by averaging the values in normalized_T

Step 3: Determine the adjustment type and calculate capped adjustments
Initialize an empty list: capped_adjustments
for each KPI
    if (The KPI is "higher_better" and actual value is >= to the target value
        OR the KPI is "lower_better" and actual value is <= to the target value)
    then
        Set r_i = 1 (indicating robustness)
        Calculate the capped adjustment using the robustness formula (Equation 6)
    else
        Set r_i = −1 (indicating penalization)
        Calculate the capped adjustment using the penalization formula (Equation 7),
        Which takes into account duration and the number of violations
    Append the capped adjustment to capped_adjustments
end

Step 4: Combine the capped adjustments
Compute the combined capped adjustment by averaging all the values in capped_adjustments


Step 5: Calculate the final trust score
Compute the final trust score by averaging the combined capped adjustment, the aggregated
    trust score, and the previous trust score T(t−1)

Return the final trust score T(t)
```

4.5 Performance Evaluation

4.5.1 Experimental Setup

The proposed experimental setup is depicted in Figure 13, which outlines the architecture of blockchain-enabled cloud application deployment. We used this architecture to mimic a federation of resource providers and show how our approach leverages blockchain and smart contract technologies integrated into the trust management component of the AC^3 trust model to ensure secure, transparent, and reliable data transactions, thereby enhancing trust in the CEC environment. The setup utilizes Kubernetes servers (i.e., pods) to represent the CECC infrastructure in AC^3, for which the trust score is calculated using our trust computation model.

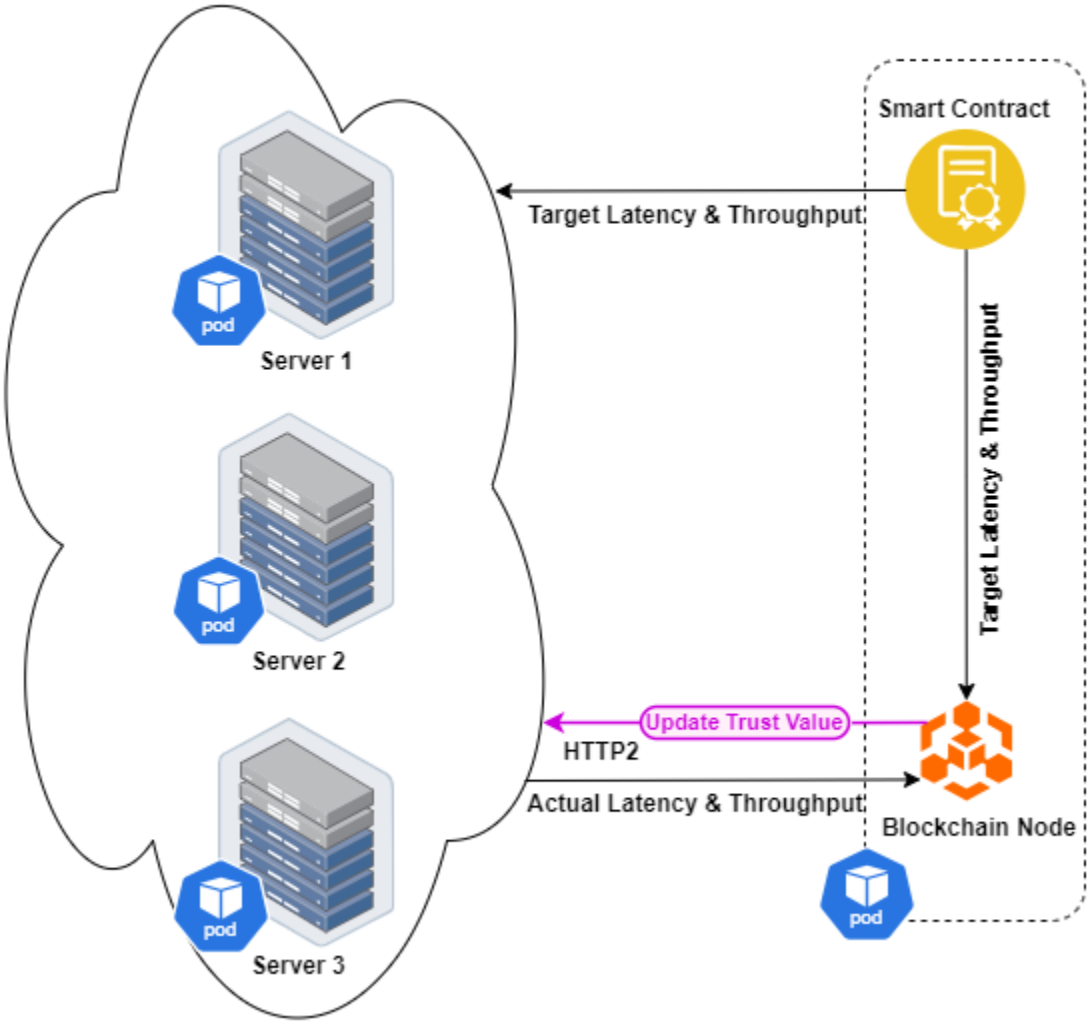


Figure 13: Experimental Setup

The core infrastructure consists of three Kubernetes-managed servers that orchestrate containerized applications and enable seamless communication between components. These servers interact with a blockchain node, responsible for managing blockchain transactions and ensuring data integrity via a Proof of Work (PoW) consensus mechanism.

- Servers (1, 2, 3): Manage containerized applications and facilitate communication across the network, ensuring high availability and scalability,
- Smart Contract: Smart Contract: To automatically enforce SLAs, a smart contract is used. By modifying resource allocation or starting tasks across the edge and cloud as needed, the contract guarantees that SLAs are met. It is activated based on real-time performance measurements, such as latency or throughput.
- Blockchain Node: Handles blockchain transactions, securing and validating them using HTTP/2 for efficient communication.
- Security: Communication between servers and the blockchain node is encrypted using TLS/SSL, ensuring secure data transmission.

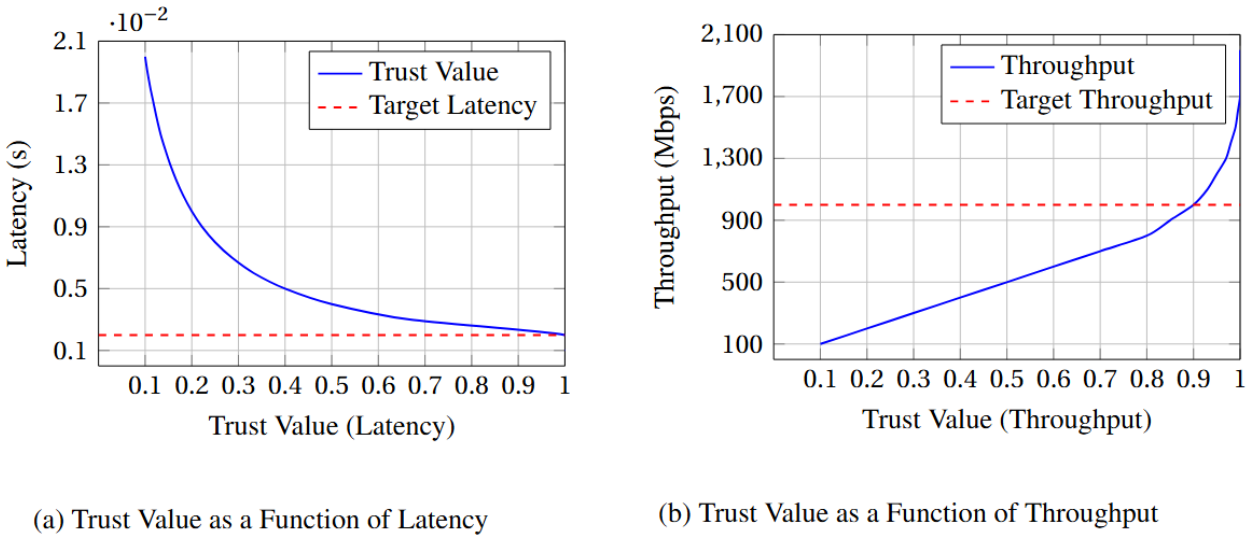
This architecture is a simplified version of the AC^3 overall trust framework, used to verify the proposed algorithm for cloud-edge environments' autonomous SLA enforcement and trust management. To show how the algorithm functions in real-world situations, smart contract and blockchain integration are crucial for transaction security, SLA enforcement automation, and performance data validation. In Table 2, we summarize the different components of our architecture, along with the technologies/protocols used for each and the main role of each component in the overall architecture. This setup ensures scalability, security, and trustworthiness in cloud-edge environments.

Table 2. Technical details of the proposed setup.

Component	**Technology/Protocol**	**Role**
Server 1	Kubernetes	Manages containerized applications and communicates with other servers
Server 2	Kubernetes	Manages containerized applications and communicates with other servers
Server 3	Kubernetes	Manages containerized applications and communicates with other servers
Blockchain Node	HTTP/2	Handles blockchain transactions
Consensus Mechanism (PoW)	Not Applicable	Ensures security and consensus in the blockchain
Security Protocol	TLS/SSL	Secures communication between servers and the blockchain node

4.5.2 Results

In the performance evaluation, our primary focus is on monitoring SLA, with particular emphasis on the relationship between trust, latency, and throughput, using the following model weights: $\alpha = 0.9$, $\beta = 0.1$, $w_l = 0.4$, $w_c = 0.4$, and $w_{adj} = 0.2$.



(a) Trust Value as a Function of Latency

(b) Trust Value as a Function of Throughput

Figure 14: Trust Value Analysis for Latency and Throughput.

Figure 14 presents the trust value as a function of latency and throughput. The two subplots (Figure 14a and Figure 14b) provide insights into how trust correlates with these performance metrics under different network conditions. Each time, we focus on a single KPI to closely observe the behavior of the trust metric. By separating the impacts of latency and throughput on trust, we gain a clearer understanding of the underlying dynamics and how variations in these KPIs affect overall trust in the network. For better visualization and interpretation, we have plotted the target value for each KPI in red color (*Target_Value (Latency) = 0.2 × 10^{-2}* seconds and *Target_Value (Throughput) = 1000 Mbps*) to enable comparison of the trust value variation against the desired target.

- Trust vs. Latency: In subplot 14a, the trust value decreases exponentially as latency increases, following the previously defined algorithm (with d = -1). At very low latency levels (e.g., 0.1 \times 10^{-2} seconds), the trust value is close to its maximum, as latency remains near the defined target. However, as latency rises, the trust value declines rapidly, indicating that even minor increases in latency can cause substantial drops in trust.
- Trust vs. Throughput: In subplot 14b, the trust value increases exponentially as throughput rises, according to the previously defined algorithm (with d = 1). At very high throughput levels (e.g., 900–1000 Mbps), the trust value approaches its maximum, as throughput nears the defined target. However, as throughput declines, the trust value drops rapidly, indicating that even slight decreases in the number of packets sent can lead to substantial reductions in trust.

In summary, Trust is highly sensitive to Latency/Throughput, with small increases/decreases in Latency/Throughput causing sharp declines in trust. This underlines the importance of optimizing for low/high Latency/Throughput in scenarios where trust is paramount. Indeed, as seen in the previous subplot (14a), the behavior of the actual KPI value affects the final trust value.

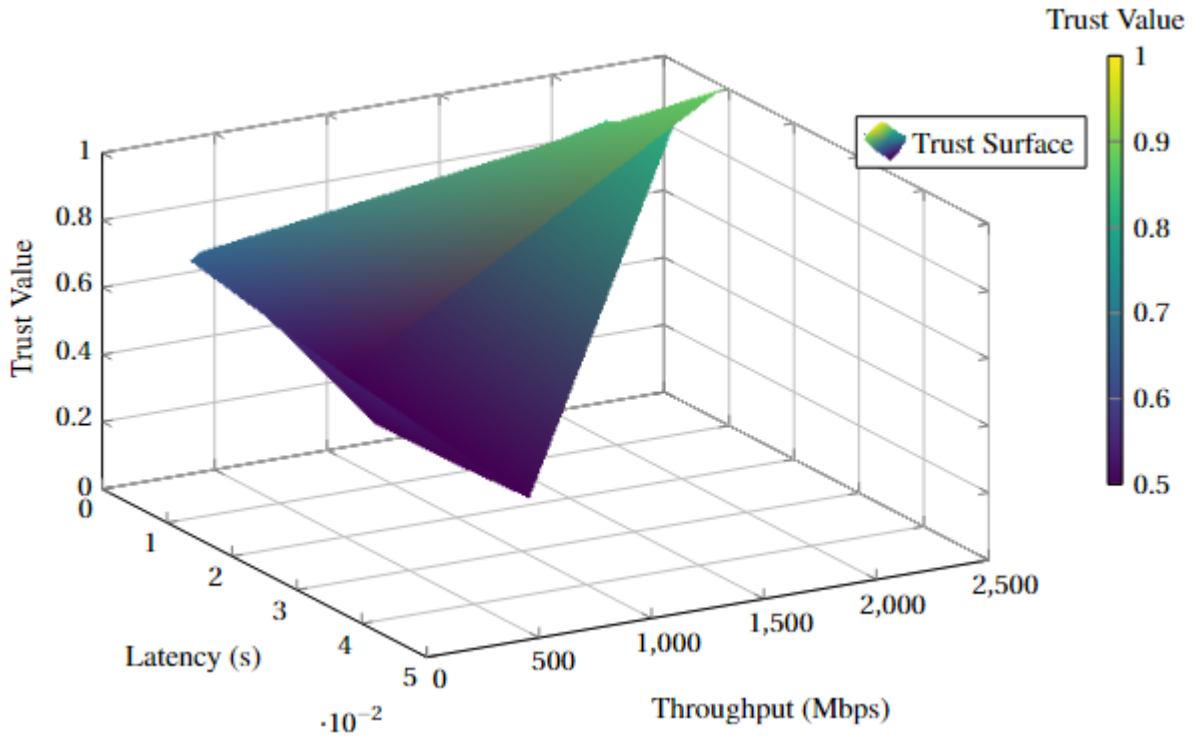


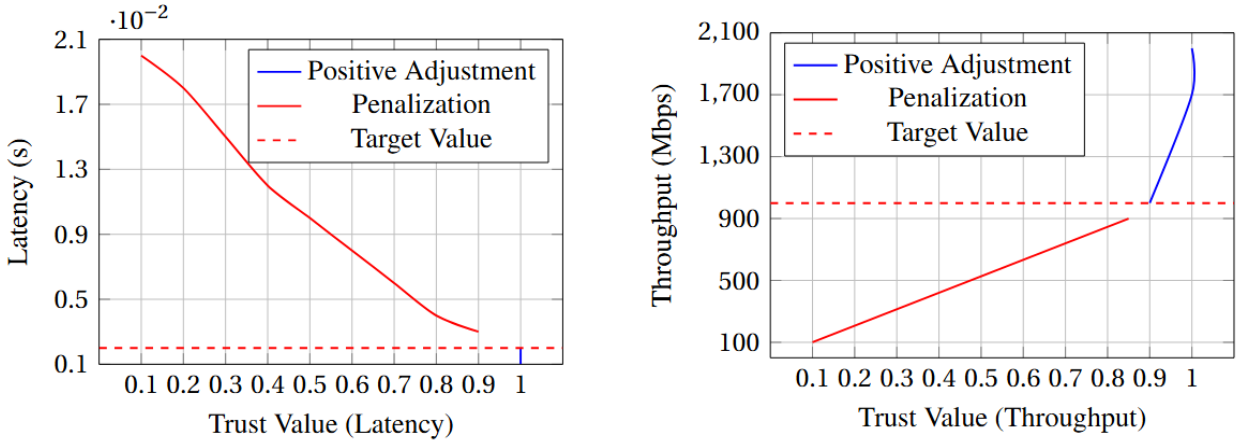
Figure 15: 3D Plot of Trust Value as a Function of Latency and Throughput.

The 3D plot in Figure 15 provides a detailed view of how latency and throughput influence together on the Final Trust Value.

- **Good Latency, Good Throughput**: In scenarios where both latency and throughput are normal, the trust value is maximized. This is represented by the higher regions of the surface in the plot. These regions demonstrate that when a network can deliver data quickly (low latency) and handle a large amount of data efficiently (high throughput), trust in the network's performance is at its peak. This scenario is ideal for high-performance applications where both speed and data volume are critical.
- **Bad Latency, Good Throughput**: Even when throughput is high, if latency is poor, the trust value drops significantly. This scenario is depicted in the plot where, despite good throughput, the trust surface declines as latency increases. It suggests that high throughput alone is not sufficient to maintain trust if the latency becomes excessive. Applications that require real-time data transmission will suffer in such a scenario, as delays in data delivery can undermine the overall performance, leading to reduced trust.
- **Good Latency, Bad Throughput**: Conversely, in scenarios where latency is low (favorable) but throughput is poor, the trust value remains suboptimal. The plot demonstrates that even when data is transmitted quickly if the network cannot handle a sufficient volume of data (low throughput), trust does not reach its maximum potential. This suggests that low latency alone cannot fully compensate for insufficient throughput, particularly in bandwidth-intensive applications that require high data transfer rates.
- **Bad Latency, Bad Throughput:** The worst-case scenario occurs when both latency and throughput are poor, resulting in the lowest trust values. In these regions of the plot, high latency (slow data transmission) combined with low throughput (insufficient data handling capacity) leads to a significant decline in trust. Such conditions are highly detrimental to network performance and can severely impact user experience and the reliability of network-dependent applications.

The interaction between latency and throughput is critical in determining the trust value of a network. The 3D

plot in Figure 15 illustrates that both metrics must be optimized to achieve high trust levels. Scenarios where either latency or throughput is compromised result in diminished trust, highlighting the importance of balanced network performance. In essence, achieving good latency and throughput simultaneously is key to maintaining high trust in network operations, while failure in either aspect can lead to significant trust degradation.



(a) Positive Adjustment and Penalization for Latency

(b) Positive Adjustment and Penalization for Throughput

Figure 16: Impact of the Adjustment on Trust Value Across different scenarios.

Figure 16 explains in depth how changes in latency and throughput under various network events influence trust values. To isolate the influence on the trust metric, the figure is divided into two subplots, each centered around a single KPI. Thanks to this targeted strategy, we can gain a deeper understanding of how specific network parameters affect overall trust. The relationship between latency and trust value is seen in subfigure 16a, especially when low latency is positively adjusted for. The blue curve indicates that when latency rises, the trust value falls. The red dashed line on the target delay represents the threshold that, in an ideal world, should not be crossed in order to preserve optimal levels of trust. The trust loss that occurs when latency exceeds this level is depicted using a penalization curve.

The graph highlights how delay has a big effect on trust. Trust declines substantially when latency rises above the goal threshold, underscoring the need for quick data delivery to preserve user confidence. The need for stringent latency management is highlighted by the fact that even slight latency increases can have a noticeable negative impact on confidence. The trust curve's steep slope emphasizes how important it is to maintain latency within tolerable bounds.

The relationship between throughput and trust value is examined in subfigure 16b, with adjustments and penalizations based on throughput levels. The red dashed line represents the ideal throughput level, while the blue line demonstrates that trust rises as throughput grows. When throughput falls short of this goal, trust deteriorates, as shown by the penalization curve.

To summarize, Figure 16 shows that throughput and latency are essential for preserving network performance confidence. To maintain consumer trust, network operators must prioritize latency management while also guaranteeing sufficient throughput, as the delay has a more noticeable impact.

The number and duration of violation metrics come from each server's real-time performance monitoring data. These metrics reflect the frequency of performance violations and the duration of each violation for each server.

Number of Violations: is recorded each time a server's performance deviates from the necessary level (for instance, due to increased latency or decreased throughput). A set period of time is used for recording all of the infractions for every server (e.g., one day or one hour, depending on the monitoring interval).

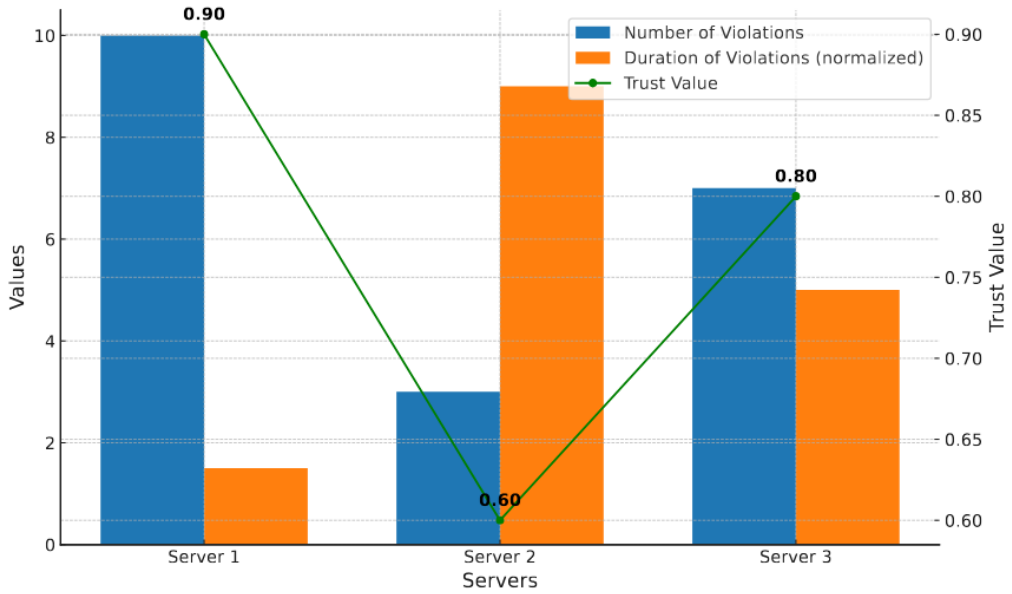


Figure 17: Impact of Violation Metrics on Trust Value Across Servers

Duration of Violations: is the total time the system operates in penalization mode. It is determined by summing the durations of all individual violations. This indicates the duration of time the server operated below optimal performance during the observation period.

The effect of violation metrics, specifically the duration and quantity of violations, on the trust value, is illustrated across three servers in Figure 17. The orange bars represent the normalized duration of these violations, while the blue bars show the total number of violations. The resulting trust value for each server is depicted as a green line. We choose $\alpha = 0.9$ and $\beta = 0.1$ in the trust calculation model to represent the notion that the duration of a violation affects trust more than the total number of violations. However, they can be set as needed.

- Server 1: maintains a high trust value of *0.90* even with the highest number of breaches. This is primarily due to the shorter duration of violations, which highlights that disturbances in trust are more sensitive to their duration than to their frequency. The rapid resolution of issues allows the network to sustain user trust.
- Server 2: despite having the fewest violations, holds the lowest trust value of *0.60*. This is because the violations persisted for longer periods, significantly undermining trust. This scenario underscores that prolonged issues, even if infrequent, severely damage user confidence and stress the need for swift recovery times in network performance.
- Server 3: exhibits a moderate level of both the number and duration of violations, resulting in a moderate trust rating of *0.80*. This balance between duration and frequency leads to an intermediate trust value, indicating that both factors must be managed to maintain consistent user confidence.

This is to say that longer infractions affect users' confidence more because they generate systemic disruptions over an extended period of time, which causes a loss of trust. Even with fewer violations, if they persist longer, the consequences will be more severe than a number of little problems that may not be apparent or that may be resolved right away. Extended system failures provide the impression that the system is unreliable, which

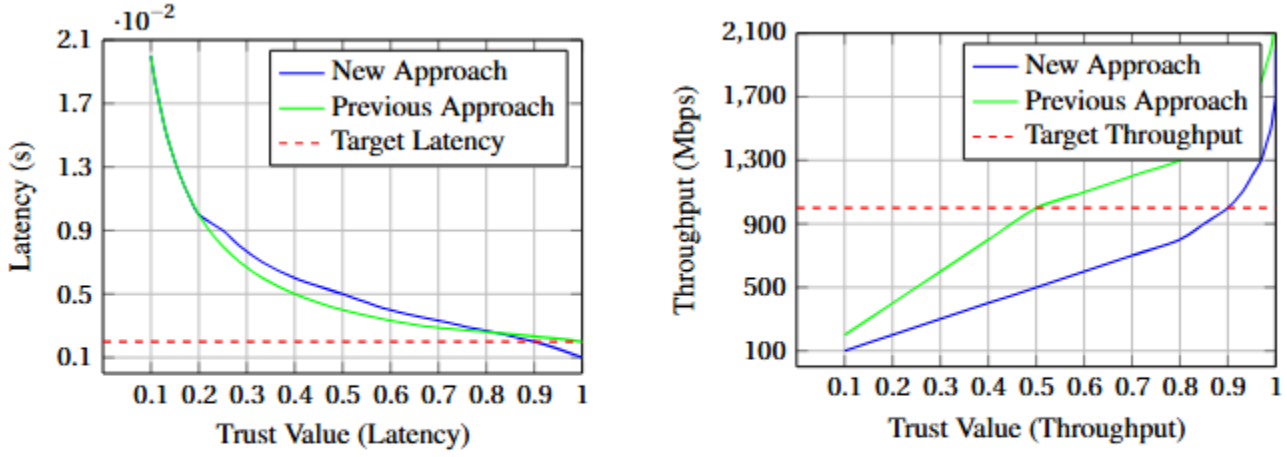
negatively affects user confidence and overall system stability.

In conclusion, Figure 17 demonstrates that the duration of violations has a more significant impact on trust value than their quantity based on the weight set in the model. Even when issues occur frequently, they should be addressed quickly to maintain higher levels of trust; prolonged disruptions have a notably negative effect. This analysis suggests that reducing the duration of violations should be prioritized, in addition to minimizing their frequency, to sustain user trust.

The investigation demonstrates that throughput, latency, and the frequency and duration of network disruptions all impact user trust. While maintaining adequate latency and throughput is crucial, minimizing the duration of outages is especially important for retaining trust. Efforts should focus on balancing these key factors and promptly addressing issues to prevent long-term effects on trust, thereby preserving user confidence.

4.5.2.1 Evaluating Trust Metrics: Simple Approach vs. Final Approach

In this section, we compare the trust metrics derived from a simple straightforward approach (Equation (1)) with our final approach depicted in Equation (2). The analysis focuses on the variations in trust values with respect to latency and throughput under both methodologies. Our goal is to evaluate how each approach impacts the trust value across different network conditions, particularly with respect to latency and throughput.



(a) Trust Value Comparison: Original vs New (Latency)

(b) Trust Value Comparison: Original vs New (Throughput)

Figure 18. Trust Value Analysis: Simple vs. Final Approach for Latency and Throughput

Figure 18 illustrates the trust value comparison between the simple and final approaches as functions of both latency and throughput. The two subplots (Figure 18a and Figure 18b) provide a direct comparison of how each methodology affects trust based on variations in these performance metrics. The target values are plotted in red for comparison purposes, allowing us to observe how closely each method aligns with desired trust outcomes.

- Comparison: Trust vs. Latency: In subplot 18a, we can see the simple and final trust values as functions of latency. While both approaches show a decrease in trust as latency increases, the final approach seems to provide a steeper decline, highlighting a more sensitive response to increased latency when compared to the original method. The red dashed line marks the target latency, demonstrating how the two approaches deviate from the ideal trust value as latency rises.

- Comparison: Trust vs. Throughput: In subplot 18b, both the simple and final approaches display an exponential increase in trust as throughput rises. However, the final approach exhibits a slightly steeper increase, indicating that it rewards high throughput more aggressively than the simple method. The red dashed line marks the target throughput, and the comparison shows that the final approach achieves closer alignment with the ideal trust value at high throughput levels.

In summary, the comparison between the simple (straightforward) and final (complex) approaches reveals that the latter is more sensitive to variations in both latency and throughput, particularly when performance deviates from target values. This suggests that our final method may provide a more nuanced reflection of network performance, making it better suited for scenarios where trust must respond rapidly to changing conditions.

In this study, we present a blockchain-based trust management system tailored specifically for CEC environments. Our approach incorporates a robust adjustment mechanism that significantly enhances the precision and consistency of trust evaluations. This adjustment process is crucial, as it enables our model to distinguish between scenarios that initially appear similar but differ in trustworthiness upon closer examination. By focusing on these nuances, our system can accurately assess which scenario is more or less trustworthy, resulting in a more comprehensive and practical trust evaluation. While the results are promising, there is room for improvement.

4.6 Trust of Data Management

Establishing trust in data management is indispensable for facilitating collaboration, ensuring secure interactions, and preserving data integrity among various stakeholders. Building upon the foundational ***Trust of Data Management*** architecture defined in D2.5 and following the proposed data management strategies for application in CECC [4], this section advances the framework by incorporating state-of-the-art tools, such as the newly modified Piveau catalogue [7] for semantic data description and discovery, alongside the EDC-S3 extension from IONOS for securing data exchange [8]. These instruments are harmonized with overarching principles of blockchain-based trust computation, Gaia-X compliance, and smart contract automation, thereby fortifying AC^3's dedication to scalable and decentralized trust mechanisms.

4.6.1 Architectural Framework: For Trust in Data Management

The AC^3 trust framework effectively integrates the modified Piveau catalogue and EDC-S3 within a multi-tiered architectural structure that ensures the enforcement of security, compliance with SLAs, and data integrity throughout the data lifecycle. Each plane within this architecture plays a critical role in fostering a decentralization and transparency-oriented trust environment.

4.6.1.1 User Plane: Semantic Discovery and Access Control

At the core of semantic data discovery and metadata management lies the Piveau catalogue, seamlessly integrated into the AC^3 application gateway. This enhances the trust model by introducing key functionalities:

- Semantic Enrichment: The Piveau catalogue employs advanced ontological frameworks and linked data methodologies to facilitate semantic interoperability across diverse datasets, thus ensuring precise and meaningful data discovery.
- Provenance and Policy Enforcement: For each dataset, comprehensive metadata—including provenance information, SLA parameters, and access control policies—is meticulously curated to warrant that data access is solely the prerogative of trusted and authorized entities.

4.6.1.2 Cloud-Edge Continuum Plane: Decentralized and Secure Data Exchange

The infrastructure plane makes strategic use of EDC-S3 to ensure decentralized and secure data exchange across federated resources:

- Policy-Driven Data Sharing: Through EDC-S3, data sharing is regulated by policies defined within the Piveau catalogue [9], ensuring compliance with trust agreements and SLA stipulations.
- Interoperability and Federation: Adhering to Gaia-X guidelines, the architecture supports seamless trust-based interactions across various infrastructure providers, with end-to-end encryption via mTLS and data integrity verification via JWTs.

4.6.2 Trust Workflow

The trust management lifecycle in the CECC framework begins with the crucial phase of Data Registration, where data providers catalogue their datasets within the Piveau catalogue. This process involves the meticulous inclusion of metadata that details the provenance of the data—tracking its origin, ownership credentials, the SLA requirements, and carefully defined access policies. Figure 19 represents the class diagram of the different AC^3 functional components and shows how they work together.

This metadata not only establishes the transparency and quality of the datasets but also acts as the foundational element for building federated trust. By associating this metadata with Gaia-X Federation Services, the system establishes a reinforced infrastructure of federated trust that spans multiple organizations, ensuring that data sharing is secure, compliant, and transparent across this interconnected network.

Following registration, the Service Discovery and Access phase allows developers to seek out datasets and services through the Piveau catalogue. This sophisticated platform enables users to apply various filters, such as those based on trust scores, compliance with SLA terms, and semantic compatibility with user needs. These features streamline the process of locating the most relevant datasets and services. Accessing these resources is facilitated through authenticated APIs, which are rigorously validated by the EDC-S3 framework. This ensures that all data interactions are secure and align with predefined access policies, thereby safeguarding the integrity of the data.

The next stage, Data Exchange and Monitoring, highlights the transfer and ongoing oversight of data within the system. Secure data exchanges are orchestrated through the EDC-S3, which ensures adherence to SLA protocols, while maintaining a meticulous audit trail of all transactions. With real-time monitoring frameworks in place, the system is capable of promptly detecting and addressing any SLA violations or anomalies that may arise. This proactive monitoring mechanism serves as a bulwark against potential compliance issues, ensuring that trust is maintained throughout the data handling process.

Completing the lifecycle is the Trust Feedback and Update phase, where trust levels are dynamically adjusted based on ongoing evaluations of SLA adherence and consumer feedback. This adaptive feedback loop is managed by smart contracts within the blockchain infrastructure, which automatically updates trust scores to reflect the current health of data interactions within the ecosystem. The use of blockchain technology not only ensures transparency and immutability of the trust metrics but also fosters a self-regulating environment where trust continuously evolves in response to direct, real-world metrics and interactions. This holistic approach to trust management ensures that the CECC framework remains resilient and reliable in its mission to secure and streamline distributed computing environments.

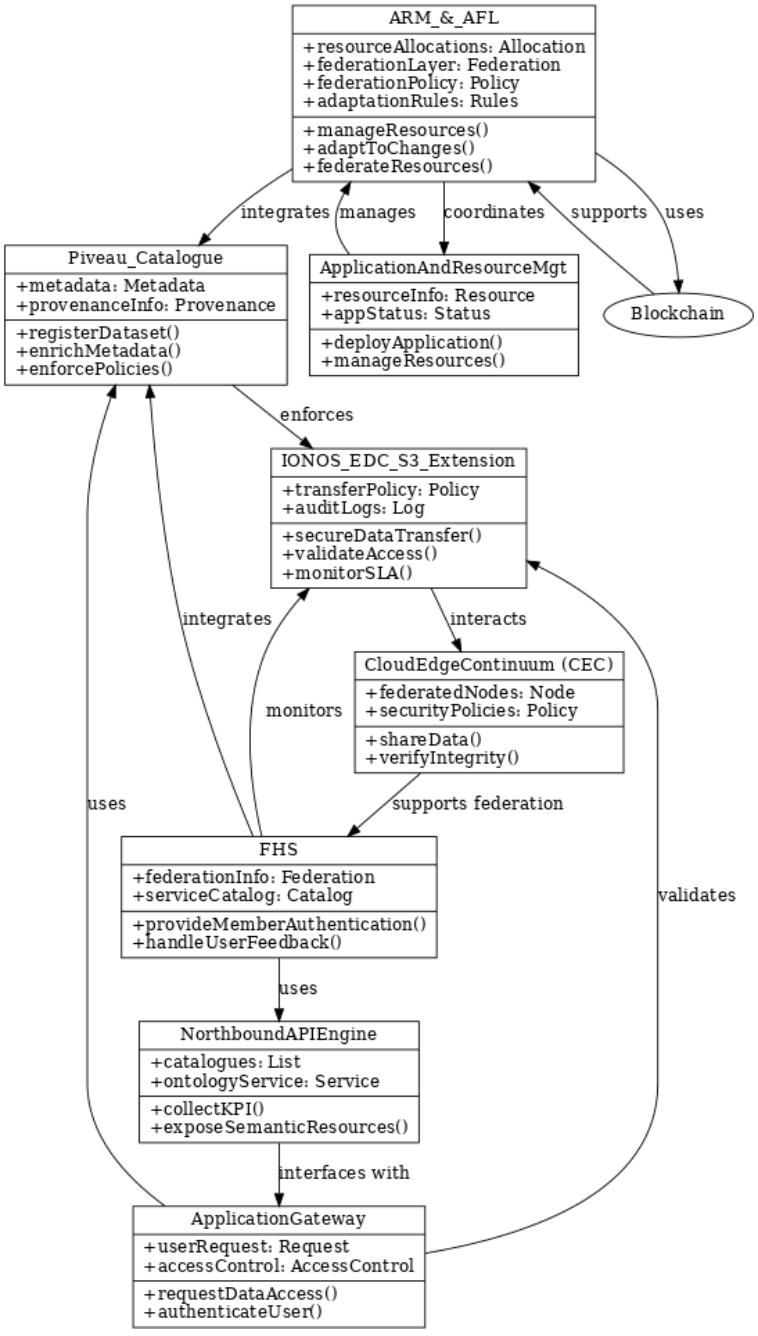


Figure 19. Class diagram for the Trust workflow in AC^3

4.6.3 Technical Innovations

- Hybrid Security Model: The architecture employs a dual-layer security approach—a combination of mTLS for transport layer security and JWT for application layer data integrity—thus ensuring robust protection across inter-component communication channels.
- Semantic Reasoning: By harnessing advanced semantic techniques, the Piveau catalogue enhances trust management through intent-aware SLA policies and contextually driven service discovery.

- Blockchain-Driven Trust: Through the integration of blockchain technology, the framework ensures transparent, immutable, and decentralized trust management. Smart contracts automate SLA validation, simplifying operational complexities and minimizing human error.

The AC^3 trust framework, enhanced with the Piveau catalogue and EDC-S3, subscribes to the principles outlined in section 4.1 to deliver a comprehensive and scalable trust architecture. By incorporating semantic metadata management, secure federated interactions, and decentralized trust computation, the framework adeptly addresses the multifaceted challenges of trust in federated Cloud-Edge environments.

5 Conclusions

In conclusion, this document has comprehensively detailed the critical aspect of security within the AC^3 project. It delivers the final version of the security and trust architecture as envisioned by AC^3. The document has successfully aligned the security architecture with the evolved high-level design, emphasizing a zero-trust model for enhanced protection. By leveraging the Kubernetes Gateway Ingress API, AC^3 has adopted a vendor-agnostic approach to API security, ensuring both flexibility and interoperability while maintaining robust protection. The emphasis on data security throughout its lifecycle, featuring mTLS for secure communication, OAuth for authorization, and JWT for secure data transmission, underscores the project's commitment to safeguarding sensitive information. Furthermore, a key innovation within AC^3 is the introduction of a central image registry that enhances security during microservice migration by providing a secure storage location for container images. Access controls, managed by an IdP, further ensure that only authorized destination nodes can retrieve the images, protecting sensitive data during transfer. Additionally, the current deliverable provided a clear framework for integrating trust management within the AC^3 ecosystem. The Trust Manager component, central to this architecture, leverages Smart Contracts to automate and monitor SLAs, ensuring real-time detection of violations and enhancing transparency. The Trust Computation model, validated through performance evaluations, offers a comprehensive method for assessing the trustworthiness of infrastructure providers by combining historical performance data and user feedback. This approach bolsters the reliability and performance of deployed applications and facilitates informed decision-making for stakeholders. The integration with IEEE SIIF's FHS framework further ensures scalability and interoperability, promoting a collaborative and secure federated environment.

As highlighted in section 2.2, these security and trust measures directly contribute to the overall project goal of providing an agile and cognitive cloud-edge continuum management solution. The business impact of these security measures and trustworthy CECC architecture is substantial, as they enhance trust, reliability, and operational efficiency for application deployment and management within the CECC, delivering substantial business benefits and fostering user confidence.

6 References

[1] AC3 Project, "D2.1 1st release of the CECC framework and CECCM," 31 August 2023. [Online]. Available: https://ac3-project.eu/wp-content/uploads/2024/02/AC3_D2.1.pdf

[2] AC3 Project, "D2.5 Report on security and trust management for CECC," 28 February 2023. [Online]. Available: https://ac3-project.eu/wp-content/uploads/2024/07/AC3_D2.5_v1.6_version_to_be_submitted.pdf

[3] AC3 Project, "D3.1 - Report on the application LCM in the CEC - Initial," 30 June 2024. [Online]. Available: https://ac3-project.eu/wp-content/uploads/2024/11/AC3_D3.1_v7.pdf

[4] AC3 Project, "D.3.3 Report on data management for applications in CECC - Initial," 28 June 2024. [Online]. Available: https://ac3-project.eu/wp-content/uploads/2024/11/AC3_D3.3_20240222_v1.0.pdf

[5] Kubernetes, "Gateway API," [Online]. Available: https://gateway-api.sigs.k8s.io/

[6] R. a. L. C. Bohn, "The IEEE 2302-2021 Standard on Intercloud Interoperability and Federation (SIIF), IEEE Cloud Continuum," 08 March 2022. [Online]. Available: https://doi.org/10.1109/IEEESTD.2022.9732072

[7] Fraunhofer FOKUS, "Piveau - Data catalogues done right," [Online]. Available: https://doc.piveau.io/general/introduction/

[8] IONOS, "IONOS S3 Extension for Eclipse Dataspace Connector," [Online]. Available: https://github.com/Digital-Ecosystems/edc-ionos-s3

[9] Fraunhofer FOKUS, "Piveau documentation," [Online]. Available: https://doc.piveau.io/general/features/