



D4.1 Data management, Intelligence and Security Layers (IT-1)

Document Identification			
Status	Final	Due Date	30/09/2023
Version	1.0	Submission Date	29/09/2023

Related WP	WP4	Document Reference	D4.1
Related Deliverable(s)	-	Dissemination Level (*)	PU - Public
Lead Participant	XLAB	Lead Author	Hrvoje Ratkajec (XLAB)
Contributors	CeADAR, BSC, ZSCALE, NKUA, ENG	Reviewers	Gabriele Giammatteo, (ENG)
			Izabela Zrazinska, (WSE)

Keywords:

ICOS, Data Management Layer, Intelligence Layer, Security Layer

This document is issued within the frame and for the purpose of the ICOS project. This project has received funding from the European Union's Horizon Europe Framework Programme under Grant Agreement No. 101070177. The opinions expressed and arguments employed herein do not necessarily reflect the official views of the European Commission.

The dissemination of this document reflects only the author's view and the European Commission is not responsible for any use that may be made of the information it contains. This deliverable is subject to final acceptance by the European Commission.

This document and its content are the property of the ICOS Consortium. The content of all or parts of this document can be used and distributed provided that the ICOS project and the document are properly referenced.

Each ICOS Partner may use this document in conformity with the ICOS Consortium Grant Agreement provisions.

(*) Dissemination level: **(PU)** Public, fully open, e.g. web (Deliverables flagged as public will be automatically published in CORDIS project's page). **(SEN)** Sensitive, limited under the conditions of the Grant Agreement. **(Classified EU-R)** EU RESTRICTED under the Commission Decision No2015/444. **(Classified EU-C)** EU CONFIDENTIAL under the Commission Decision No2015/444. **(Classified EU-S)** EU SECRET under the Commission Decision No2015/444.

Document Information

List of Contributors	
Name	Partner
Jaydeep Samanta	CeADAR
Sebastian Andrés Cajas Ordoñez	CeADAR
Andrés L. Suárez-Cetrulo	CeADAR
Ricardo Simon Carbajo	CeADAR
Hrvoje Ratkajec	XLAB
Matevž Eržen	XLAB
Anastasios Giannopoulos	NKUA
Panagiotis Gkonis	NKUA
Panagiotis Trakadas	NKUA
Gabriele Giammatteo	ENG
Maria Antonietta Di Girolamo	ENG
Alex Barceló	BSC
Anna Queralt	BSC
Ivan Paez	ZSCALE

Document History			
Version	Date	Change editors	Changes
0.1	27/06/2023	XLAB	The first draft version of ToC
0.2	06/07/2023	XLAB	Updated ToC
0.3	21/08/2021	CeADAR	The first version of Section 3
0.4	24/08/2023	XLAB, CeADAR, NKUA, ENG, ZScale, BSC	Contributions from partners to sections 2, 3 and 4
0.5	06/09/2023	XLAB	Review of contribution and feedback to partners from corrections
0.6	12/09/2023	XLAB	Implemented corrections from partners and version ready for the internal review
0.7	22/09/2023	XLAB	Integrated feedback from the internal review
0.8	28/09/2023	XLAB	Version for the final quality check
1.0	29/09/2023	ATOS	Final version for submission

Quality Control		
Role	Who (Partner short name)	Approval Date
Deliverable leader	XLAB	28/09/2023
Quality manager	ATOS	29/09/2023
Project Coordinator	ATOS	29/09/2023

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	2 of 66
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status:
			Final

Table of Contents

Document Information	2
Table of Contents	3
List of Tables.....	5
List of Figures	6
List of Acronyms.....	7
Executive Summary	8
1 Introduction	9
1.1 Purpose of the document.....	9
1.2 Relation to other project work.....	9
1.3 Structure of the document.....	10
1.4 Glossary adopted in this document	10
2 Data management	12
2.1 Functional description.....	13
2.1.1 Fitting into the ICOS architecture.....	13
2.2 Technical description	14
2.2.1 Prototype architecture and the description of components	15
2.2.2 Technical specifications.....	16
2.3 Delivery and usage.....	17
2.3.1 Package and deployment information	17
2.3.2 Licensing and download information	17
2.4 Limitations and future work.....	18
3 Intelligence Layer.....	19
3.1 Intelligence Layer Coordination module	19
3.1.1 Functional description	19
3.1.2 Technical description.....	23
3.2 AI Analytics	25
3.2.1 Intelligent energy-aware task offloading	26
3.2.2 Load time series forecasting	29
3.2.3 Anomaly detection.....	30
3.2.4 Metrics forecasting	31
3.3 Trustworthy AI.....	32
3.3.1 Fitting into the ICOS architecture.....	33
3.3.2 Approach and design of the components of the Trustworthy AI module.....	34
3.4 Delivery and usage.....	38
3.4.1 Package information	38

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	3 of 66
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status:
			Final

3.4.2	Installation instructions.....	39
3.4.3	Licensing information.....	40
3.4.4	Download.....	40
3.5	Limitations and future work.....	41
4	Security Layer Module.....	42
4.1	Security Layer Coordination Module.....	42
4.1.1	Functional description	42
4.1.2	Technical description.....	43
4.2	Identity and Access Management	45
4.2.1	Functional description	45
4.2.2	Technical description.....	46
4.3	Security Scan.....	49
4.3.1	Functional description	49
4.3.2	Technical description.....	50
4.4	Delivery and usage.....	51
4.4.1	Package information	51
4.4.2	Installation instructions.....	51
4.4.3	Licensing information.....	52
4.4.4	Download.....	52
4.5	Limitations and future work.....	53
5	Conclusions	54
6	References	56
	Annexes.....	58
	Annex I: Testing the Intelligence Layer Coordination API.....	59
	Annex II: Validation of AI Analytics models.....	63
	Load time series forecasting.....	63
	Load anomaly detector.....	65
	Annex III: Intelligence Layer Coordination API Development Guide.....	66

List of Tables

<i>Table 1: ICOS concepts and artefacts</i>	10
<i>Table 2: Requirements starting to be met with the IT-1 of the Data Management [1]</i>	12
<i>Table 3: Data management licensing and download</i>	17
<i>Table 4: Requirements starting to be met with the IT-1 of the Intelligence Layer Coordination module [1]</i>	21
<i>Table 5: Intelligence Layer Coordination API definition</i>	25
<i>Table 6: Requirements starting to be met with the IT-1 of the AI-Analytics module [1]</i>	26
<i>Table 7: AI Analytics API endpoints</i>	32
<i>Table 8: Requirements involving Trustworthy AI [1]</i>	33
<i>Table 9: Intelligence Layer Coordination API dependencies</i>	38
<i>Table 10: Intelligence Layer Functionalities Layer in the ICOS Alfa and Beta releases [2]</i>	41
<i>Table 11: Requirements starting to be met with the IT-1 of the Security Layer Coordination module [1]</i>	42
<i>Table 12: Security Layer Coordination API definition</i>	45
<i>Table 13: Requirements starting to be met with the IT-1 of the Identity and Access Management module [1]</i>	46
<i>Table 14: Requirements starting to be met with the Alfa version of the Security Scan module [1]</i>	49
<i>Table 15: Modified Security Layer functionalities in the ICOS Alfa and Beta releases</i>	53
<i>Table 16: Data Management, Intelligence and Security Layers IT-1 parts as ICOS assets</i>	55
<i>Table 17: Architecture of the LSTM network model used for the load timeseries forecasting.</i>	63

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	5 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

List of Figures

Figure 1: Data management in ICOS	13
Figure 2: Data exchange in the Data management layer	15
Figure 3: Data Management prototype architecture	16
Figure 4: High-level architecture of ICOS Intelligence Layer API	20
Figure 5: Result of remote workshop for the Intelligence Layer organised by CeADAR	21
Figure 6: Intelligence Layer Coordination module in ICOS	22
Figure 7: Architecture of ICOS interacting regarding the Intelligence Layer Coordination API	22
Figure 8: Content of the Intelligence Layer Coordination module	23
Figure 9: Intelligence Layer Coordination module API flow	24
Figure 10: ICOS system architecture of the DECOFFEE scheme enabling both vertical and horizontal actions towards latency-aware and energy-efficient distributed task offloading policy	27
Figure 11: The decentralized DRL model (based on Q-learning) running on a single ICOS agent towards implementing the overall DECOFFEE algorithmic scheme	28
Figure 12: LSTM architecture for load time series forecasting in rolled and unrolled forms.	30
Figure 13: Up-to-date ICOS architecture concerning the Trustworthy AI module.	33
Figure 14: Importance of features (y-axis) leading to outputs in test set forecasts (x-axis)	35
Figure 15: Example monitoring system that raises alerts when the mean absolute error exceeds a threshold	35
Figure 16: Periodic model training and weight aggregation over dispersed nodes with the help of Flower	36
Figure 17: Strategy/server initialization in Flower	37
Figure 18: MAE Loss for different Server - Client optimizers	37
Figure 19: Loss versus round	38
Figure 20: Security Layer Coordination module in ICOS	43
Figure 21: Prototype architecture and data workflow of the Security Layer Coordination module	44
Figure 22: OIDC generic authentication flow	47
Figure 23: Identity and Access Management internal architecture	47
Figure 24: High-level Wazuh's architecture	50
Figure 25: API definition for train_cpu_utilisation in Swagger	59
Figure 26: POST request with default (above) and user-defined (below) parameters.	59
Figure 27: Response received upon successful execution of request	60
Figure 28: Example of curl POST request to train a model	60
Figure 29: Predict_cpu_utilisation API with input and output parameters description.	61
Figure 30: POST request with default (left) and user-defined (right) input parameters	61
Figure 31: Example of curl POST request to inference with a model	62
Figure 32: Example model inference response received upon successful execution of request	62
Figure 33: Mean squared error (MSE) loss curve as a function of the training epochs for different values of learning rate α (panel a) and lookback window length W (panel b).	64
Figure 34: Actual and predicted load (Gbps) for each IoT area/cell-specific LSTM model. Validation data refers to 1-week total load values used for the testing set.	64
Figure 35: Percentage anomaly detection accuracy across different supervised classification models	65

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	6 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

List of Acronyms

Abbreviation / acronym	Description
AI	Artificial Intelligence
API	Application Programming Interface
CPU	Central Processing Unit
DNN	Deep Neural Network
DQN	Deep Q-Network
DRL	Deep Reinforcement Learning
DDRL	Decentralised Deep Reinforcement Learning
Dx.y	Deliverable number y belonging to WP x
FL	Federated Learning
GDPR	General Data Protection Regulation
GUI	Graphical User Interface
HTTP	Hypertext Transfer Protocol
IAM	Identity and Access Management
IDE	Integrated Development Environment
IT-1	First iteration
IT-2	Second iteration
JSON	JavaScript Object Notation
JWT	JSON Web Token
LSTM	Long Short-Term Memory
ML	Machine Learning
MLOps	Machine Learning Operations
MSE	Mean Squared Error
NDN	Named Data Networking
OIDC	OpenID Connect
PDP	Policy Decision Point
PEP	Policy Enforce Point
PIP	Policy Information Point
REST	Representational state transfer
SL	Supervised Learning
TLS	Transport Layer Security
mTLS	mutual TLS
UI	User Interface
WP	Work Package
WPx	Work Package number x

Executive Summary

The document is the first implementation deliverable of the ICOS components Data Management, Intelligence and Security Layers, presenting the following aspects: a), the functionalities, b) the prototype architecture and how it fits into the general ICOS architecture, c) the technical description and specifications, d) delivery (package, installation, download) and licensing information, and any limitations and future work to be done for the next releases (ICOS Beta and Final releases).

In the first iteration (IT-1), Data management software has been mainly integrated with the Intelligence Layer and with the Distributed & Parallel Execution module which is part of the Runtime Manager component of the Meta-kernel layer (WP3). A first integration with the Job Manager (also a part of the Runtime Manager component) to facilitate its deployment has also been performed. The two main software solutions used are Zenoh and dataClay. The layer also defined REST APIs for communication with other ICOS components.

The Intelligence Layer in IT-1 includes the Intelligence Layer Coordination module, AI analytics and Trustworthy AI modules. The Intelligence Layer Coordination module provides a unified interface for interacting with the Intelligence Layer functionalities and communicating with other ICOS components. It communicates with the AI analytics module which is responsible for building and deploying machine learning models (4 models are developed in IT-1). It will also communicate with the Trustworthy AI module that, in IT-1, provides two strategies to ensure the trustworthiness and privacy of models: 1) Federated learning and privacy and 2) Explainable AI and monitoring.

The Security Layer in IT-1 includes the Security Layer Coordination module, Identity and Access Management module and the Security Scan module. Security Layer Coordination module provides a unified interface for interacting with Security Layer and with other ICOS components as well as additional logic and security functionalities needed for seamless interactions between Security Layer's modules. The Identity and Access Management module manages user's identities, credentials, roles, and tokens to identify and authorise operations in the ICOS system. The Security Scan module actively checks for security issues within deployed ICOS modules and components, detects issues, and recommends mitigations and/or recovery processes.

All presented software will be used in the first iteration of ICOS (ICOS Alfa release) and will be expanded with new functionalities in future releases (ICOS Beta release, second iteration (IT-2)) according to updates to ICOS requirements and architecture.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	8 of 66				
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final

1 Introduction

1.1 Purpose of the document

The purpose of the document is to report on the design and implementation efforts done in WP4 in tasks T4.1 (Data Management), T4.2 (Intelligence Layer Module), T4.3 (Security Layer Module) and T4.4 (APIs and Interfaces) regarding the development of ICOS components Data Management, Intelligence and Security Layers in the period M06-M13. It also delivers the first version (IT-1) of the Data Management, Intelligence Layer and Security Layer software.

In this regard, the document presents the following aspects for each layer:

- ▶ the functionalities,
- ▶ the prototype architecture and how it fits into the general ICOS architecture,
- ▶ the technical description and specifications,
- ▶ delivery (package, installation, download) and licencing information, and
- ▶ any limitations and future work to be done for the next releases (ICOS Beta and Final releases).

In this iteration, the work done in T4.4 is directly included in the functionalities of each ICOS component, presented in individual sections below.

1.2 Relation to other project work

The first implementation deliverable D4.1 ICOS components Data Management, Intelligence and Security Layers, the software and document relate to:

- ▶ The tasks "T2.2 – Compute continuum requirements definition" and "T2.3 – AI, data management and trust/security requirements" concerning the analysis of the ICOS-related technologies, the definition of the project's use cases, and the elicitation of the system requirements relevant for the above mentioned ICOS components,
- ▶ The deliverable "D2.1 - ICOS ecosystem: Technologies, requirements, and state of the art (IT-1)" [1] that summarizes the work conducted since the beginning of the project in the task "T2.1 – Ecosystem identification: Baseline technologies" concerning the description of use cases, elicitation of requirements and the analysis of the State of the Art of baseline technologies, which were also used in the IT-1,
- ▶ The task T2.4 Architectural Design and the deliverable "D2.2 ICOS architectural design (IT-1)" [2] that defines the main architectural components of the ICOS system, among them Data Management, Intelligence, and Security Layers, and describes their functionalities, properties, and interactions.

Additionally, the work reported in this document will be the basis for:

- ▶ The first integrated ICOS Alfa release, to be delivered in M15,
- ▶ The second integrated ICOS Beta release, to be delivered in M22 and
- ▶ The second iteration of Data Management, Intelligence and Security Layers (IT-2) which is planned to be delivered in M30.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	9 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

1.3 Structure of the document

This document is structured in 7 major sections:

- ▶ **Section 1 – Introduction** presents the objectives of the document and introduces its content, structure, relationships with the other project’s documents and the glossary,
- ▶ **Section 2 – Data Management** provides the functional and technical description, delivery (package, installation, download) and licencing information as well as any limitations and future work for the first iteration of the Data Management component,
- ▶ **Section 3 – Intelligence Layer** provides functional and technical description, delivery (package, installation, download) and licencing information as well as any limitations and future work to be done for the first iteration of the Intelligence Layer Coordination module. It also provides the functional description of the first models of the AI Analytics module and the approach and design used in developing the Trustworthy AI module,
- ▶ **Section 4 – Security Layer** provides the functional and technical description, delivery (package, installation, download) and licencing information as well as any limitations and future work for the Security Layer modules developed in IT-1 (Security Layer Coordination module, Identity and Access module and Security Scan module),
- ▶ **Section 5 – Conclusion** summarises the content of the document and provides final considerations on how the technical work will continue in the project.
- ▶ **Section 6 – References:** list references used.
- ▶ **Section 7 – Annexes:** provides various annexes with more detailed explanations to Section 3 Intelligence Layer.

1.4 Glossary adopted in this document

The following table provides definitions of ICOS concepts and artefacts (as identified in D2.1 [1] and D2.2 [2]) that are discussed in the following sections of this deliverable.

Table 1: ICOS concepts and artefacts

Cloud Continuum
A group of resources (CPU (Central Processing Unit), memory, network, storage, intelligence) managed seamlessly end-to-end that may span across different administrative/technology domains in multi-operator and multi-tenant settings.
ICOS Agent
The basic ICOS software component that needs to be installed on each node (infrastructure node) capable of executing external software and/or providing data and metadata.
ICOS Controller
The ICOS component/software that is responsible for peering with ICOS Agents for providing control and lifecycle
ICOS Instance
A subset of the CC participating in the execution of a multi-component Application of a certain topology, resource requirements and constraints.
ICOS Node
Any resource, physical or virtual, that is running either the ICOS Software (can be either an ICOS Controller or an ICOS Agent).

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	10 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

ICOS Shell
A client distribution that is used to access the ICOS. The ICOS Shell runs externally to the ICOS Instance.
ICOS User
The ICOS user is the actor that uses ICOS to deploy his/her applications over suitable infrastructure, exploiting the ICOS management and optimization capabilities to deploy applications fuelling novel business opportunities.
ICOS Application
User application deployed on an ICOS instance.

2 Data management

The Data Management component is responsible for optimising access to data from the rest of ICOS layers, to optimise the performance of these components in the execution of their responsibilities.

It is a transversal component in the ICOS architecture, aimed at supporting the various data needs in the rest of the layers, making the distribution and heterogeneity of the infrastructure transparent to them, and providing efficient data access at the same time.

This section is structured as follows: the functional description of the Data Management is provided in subsection 2.1, which also contextualises the component within the ICOS architecture, subsection 2.2 describes the technical details of the solution, whereas the details for installation and licensing can be found in 2.3. Finally, subsection 2.4 explains the limitations and future work.

Table 2: Requirements starting to be met with the IT-1 of the Data Management [1]

Requirement ID	Requirement Name	Description
CM_FR_10	Data Management	ICOS must be able to maintain the data sources topology as well as data source types (metadata) for proper application data assignment.
CM_FR_11	Data Access	ICOS must be able to provide different data access methods, including selective access as well as streaming access, according to flexible high level data access application program interfaces
DRT_FR_01	Data distribution across the continuum	ICOS SHOULD take advantage of all available devices to place data. Flexibility to adapt to different configurations (cloud-only, edge-only, cloud-edge)
DRT_FR_02	Transparent data access	ICOS must be able to provide location and format transparent data access methods through flexible high level data access application program interfaces (API)
DRT_FR_03	Minimization of data transfers	ICOS MUST avoid unnecessary data movements to increase performance, reduce network congestion, and favour privacy by exploiting near-data processing
DRT_FR_04	Support for distributed/parallel execution	ICOS SHOULD provide the integration of data management with the execution runtime to support efficient scheduling and execution of the required tasks.
DRT_FR_05	Failure recovery mechanism/management	ICOS MUST provide the capability to restart the failing transfers of the data in case of the failures (e.g., losing the connectivity)
DRT_FR_06	Secure data exchange	ICOS SHOULD support data preserving communication techniques between modules (interfaces) for secure data exchange.

Requirement ID	Requirement Name	Description
DRT_FR_07	Data Recompileation	Control Nodes must be able to recollect data in a scheduled or periodic way from the nodes it orchestrates

2.1 Functional description

At a high level, the main functionalities of the Data Management component are the following:

- ▶ Data distribution across the continuum, taking advantage of the entire available infrastructure,
- ▶ Smart data placement and dynamic adaptation to changes in the infrastructure during operation: devices joining or leaving, reorganisations, etc,
- ▶ Seamless access to data in ICOS, regardless of the location (device or cloud) or nature of resource (in-motion or at-rest), by providing an integrated data platform spanning the whole continuum,
- ▶ Minimization of data transfers to improve performance and trust, by exploiting near-data processing in various types of devices.

2.1.1 Fitting into the ICOS architecture

The Data Management component provides support mainly to those components that need to access and process large amounts of data, namely the Intelligence Layer and the Distributed & Parallel Execution module within the Meta-Kernel Layer. These are the components that can benefit the most from the Data Management functionalities and optimizations, as they require efficient execution of complex data processing tasks across distributed data. Additionally, the minimization of data transfers using the near-data processing capabilities provided by the Data Management component not only improves performance but also favours privacy, as data does not need to leave its original location to be processed. See Figure 1.

The Data Management component also provide secure communications functionality ICOS nodes, and to the Job Management within the Meta-Kernel Layer to deploy the component across the nodes belonging to an ICOS instance.

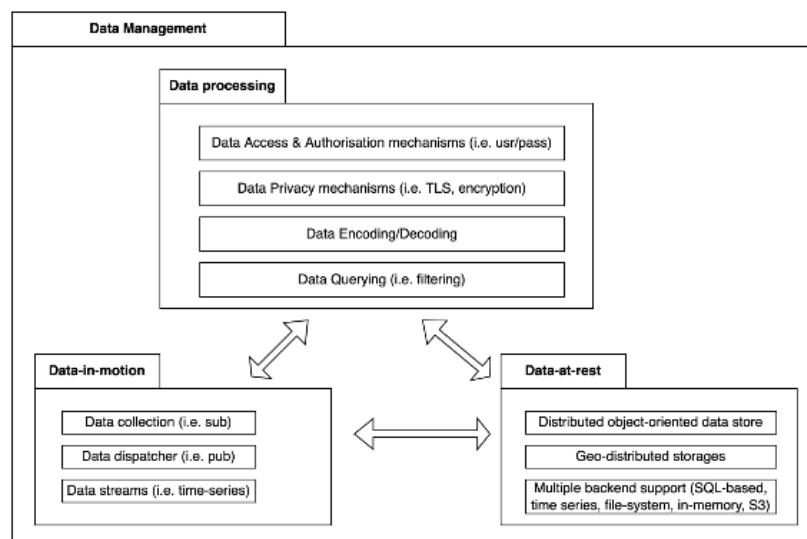


Figure 1: Data management in ICOS

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	13 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

2.2 Technical description

The Data Management component can be seen as a (hyper)distributed data store that allows ICOS components to access data regardless of its location in the continuum. In addition, it aims at optimising data access by executing part of the code within the data store, thus minimising data movements within and across nodes.

Access transparency is not only meant for the location of the data but also for its interface. More precisely, the data used by a component, (which may be implemented using Machine Learning or Deep Learning libraries such as PyTorch or Tensorflow to process ICOS data), can be seamlessly accessed by the component, regardless of its location within the ICOS infrastructure, as if it was in a single node.

Finally, the Data Management component exposes a specific API that allows it to communicate with the Distributed & Parallel Execution module within the Distributed Meta-Kernel Layer, which coordinates the execution of processing tasks. This enables this module to control where the data is and schedule task execution (i.e. pub/sub/query/process) accordingly. During this process it is also creating replicas when necessary in order to parallelize the processing on big datasets.

Due to its code execution and other performance optimizations, a registration step is required prior to the execution of the components that access Data Management, so that it is aware of the data structure, the methods, and the relationships that other components will use to access and process data. This is provided using regular Python program that defines the classes and methods that are required by each component.

Once this is done, the component will be able to connect to the Data Management by indicating the necessary connection information (host, port, username and password). Data can then be pushed/pulled to/from the Data Management using a simple object-oriented API.

Additionally, the Data Management component supports the encapsulation of the data exchange for the business logic. Initially, a request could come from a client (message producer), and once processed by the business logic, the response could be sent to a service (message consumer) interested in it. Typically, a message can either be a command, which specifies an operation to be invoked by the receiver, or an event, which signals to the receiver that something of interest happened to the sender. A sender can use inbound adapters to receive messages from the channel and outbound adapters to send response messages to changes, as shown in Figure 2.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	14 of 66				
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final

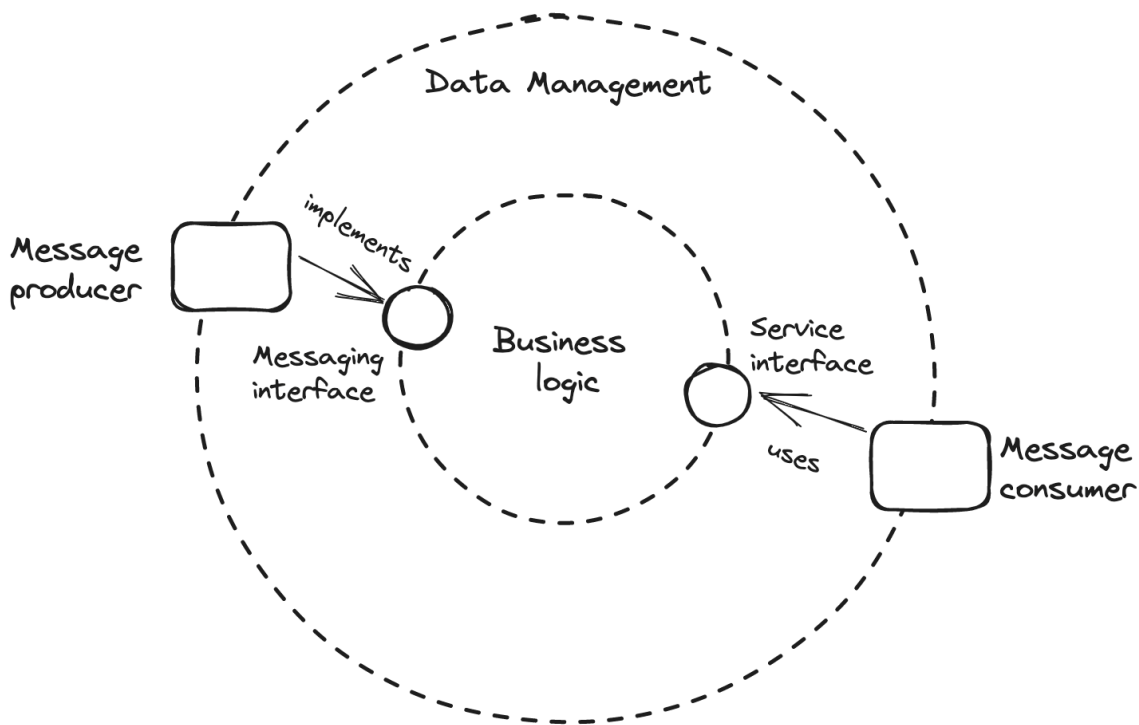


Figure 2: Data exchange in the Data management layer

The Data Management components supports different communication patterns, such as:

- ▶ Point-to-point: in this messaging pattern the communication is delivered to exactly one consumer instance, but it requires knowing the endpoint and port of the message consumer,
- ▶ Publish-subscribe: this messaging pattern decouples the subjects from observers via asynchronous messaging. Typically, the publisher/applications declares a topic which is named data and data names will directly be used in network packet forwarding; consumer applications request desired data by its name, so communications are based on the Named Data Networking (NDN) approach and are consumer-driven,
- ▶ Push-Pull: this messaging pattern is often referred to as fan-out/fan-in where the messages are delivered to a pool of subscribers in a round-robin fashion and each message is delivered to each worker. Another benefit is that it enables the processing of multiple messages within a predefined batch or unit of work.

2.2.1 Prototype architecture and the description of components

Data Management has two main subcomponents: a Metadata Service, which is a distributed service holding all the information about the data stored and its location, and one or more Backends that store the data in a distributed manner and are also responsible for the execution of the methods related to the data they hold. In this way, data transfers to/from ICOS components are minimised.

In addition, Backends hold the data in-use already instantiated in memory (as long as it fits, which will depend on the capacity of the devices at hand, as well as on the actual size of the data). Thus, data transfers between physical storage devices and the execution environment are also avoided, further reducing execution time.

The Metadata Service and the Backends can live in the same or in a different set of nodes, depending on the characteristics of each particular infrastructure. The number of backends is also customizable according to the infrastructure available for the ICOS instance deployed.

Components communicate with Data Management through a client module that interacts with the Metadata service and the Backends. See Figure 3.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	15 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

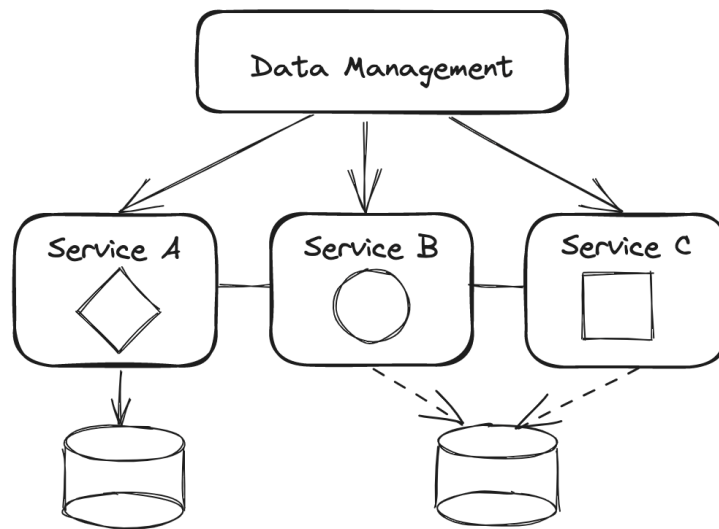


Figure 3: Data Management prototype architecture

As mentioned earlier, the Data Management component can be seen as a (hyper) distributed data store, which brings many benefits but at the same time it adds complexity to the ICOS architecture. In a monolithic application, the data generally resides in a single data store, but in a distributed system such as the ICOS platform, it is spread across multiple services, where a service can have its own data repository or have a common data store for multiple services. Thus, some use cases might require stitching data together from multiple sources. The Data Management API can offer a higher-level API that queries multiple services and composes their responses. This relieves the client from knowing which services to query and reduces the number of requests it needs to perform to get the data it needs.

2.2.2 Technical specifications

Two main software solutions are used in the implementation of the Data Management layer. These frameworks provide a foundational backbone that is being extended to satisfy the requirements for data management in ICOS:

- ▶ Zenoh[8] – a pub/sub/query protocol,
- ▶ dataClay[9] – a distributed active object store.

The Data Management layer provides a high-level interface to the ICOS components. This ensures that the components can be developed in a portable way. In order to achieve that, the Data Management layer is able to provide a unified hub for three distinct flows. Each component can choose the interface that is more appropriate for its needs:

- ▶ An API REST usable from all ICOS components, regardless of their programming language. This ensures interoperability between components with a unified data store. Will be used by those components that are not implemented in Python, or those that do not require data processing (only get/put).
- ▶ A native Python object interface with active methods, which increases performance and programmability. This is a key feature for supporting Python libraries such as PyTorch. For instance, this makes it possible for the Intelligence Layer to use and manage remote and distributed data structures without requiring explicit data transformations.
- ▶ An efficient inter-component communication. This comes with support for different communication patterns as mentioned above: point-to-point, publish-subscribe and push-pull. This one can be combined with any of the previous options, when streaming or real time data is involved.

The communication channels used by the Data Management layer make use of cryptographic protocols (TLS (Transport Layer Security) and/or mTLS (mutual TLS)).

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	16 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

2.3 Delivery and usage

The following subsections present package information on the Data Management layer, with information on where to download and set them up, as well as relevant information regarding licensing.

2.3.1 Package and deployment information

All the software components – both the existing ones (Zenoh, dataClay) and the future ones being developed in the ICOS project – are already containerized. At the moment, it is entirely possible to run all parts of the Data Management layer through the manual orchestration of containers (with docker-compose or any other container orchestration mechanism).

The general containerized deployment can be installed with the following commands:

```
$ docker image pull eclipse/zenoh
$ docker image pull ghcr.io/bsc-dom/dataclay
$ docker image pull redis

$ docker run ghcr.io/bsc-dom/dataclay dataclay.metadata
$ docker run redis
$ docker run eclipse/zenoh
$ docker run ghcr.io/bsc-dom/dataclay dataclay.backend
```

In this example, we can see the deployment of the metadata (which can be distributed) and depends on Redis. Depending on the final topology, multiple backends and Zenoh routers can be deployed.

For future releases (ICOS Beta Release, IT-2), we envision a unified procedure for the global ICOS onboarding process. The procedure for adding a new device to an ICOS infrastructure will trigger the deployment process of the Data Management layer. It is an ICOS objective to define and implement this onboarding process; during this process, we will develop the necessary manifests that will describe and orchestrate the different Data Management components.

2.3.2 Licensing and download information

The Data Management uses and derives from the two open-source software projects explained in section 2.2.2 and shown in Table 3.

Table 3: Data management licensing and download

Project	Source code repository	Licence
dataClay	https://github.com/bsc-dom/dataclay	3-Clause BSD License
Eclipse Zenoh	https://github.com/eclipse-zenoh/zenoh/	Apache License 2.0 & Eclipse Licence 2.0

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	17 of 66	
Reference:	D4.1	Dissemination:	PU	
	Version:	1.0	Status:	Final

2.4 Limitations and future work

In the IT-1 release, the Data Management component has been mainly integrated with the Intelligence Layer and with the Distributed & Parallel Execution module. A first integration with the Job Manager to facilitate its deployment has also been performed. At the moment, integration with the Security Layer has been discussed but not yet implemented.

Additionally, REST APIs for those components implemented in languages other than Python need to be developed.

Providing support to federated learning, as part of the Intelligence Layer, is also planned for IT-2 release.

Also, in IT-2, Data Management will support strategies for data replication, data consistency, and data partitioning that can be tailored according to the ICOS applications needs. When dealing with decentralised data storages, multiple trade-offs should be taken into account, such as:

- ▶ **Data replication:** one reason for replicating data is to increase availability. If some data is stored exclusively on a single process, and that process goes down, the data will not be accessible anymore.
- ▶ **Data consistency:** Implementing replication is challenging because it requires keeping replicas consistent with one another, even in the face of failures. Thus, ICOS data stored should support a consistency model to guarantee to clients the data appears to be stored on a single process.
- ▶ **Data partitioning:** when an application's data keeps growing in size, its volume will eventually become so large that it cannot fit on a single machine. Thus, it needs to be split into partitions. One alternative is to create partitions on demand, also referred to as dynamic partitioning. ICOS will explore ways of doing that, referred to in the literature as range partitioning and hash partitioning.

Finally, for IT-2 the Data Management will provide additional functionalities to support not only other ICOS components but also applications running on top of ICOS.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)			Page:	18 of 66
Reference:	D4.1	Dissemination:	PU	Version:	1.0
				Status:	Final

3 Intelligence Layer

The Intelligence Layer enhances the security and functionality of the Security and Meta-kernel layers. It helps to ensure that analytical and machine learning models are used safely, effectively, and ethically across data-intensive Edge/Cloud spectrum applications.

The layer is coordinated by the Intelligence Layer Coordination module, which provides a unified interface for interacting with the Intelligence Layer functionalities and communicating with other ICOS components. It also coordinates the internal Intelligence Layer modules to fulfil incoming requests from Meta-kernel and user layers (ICOS Shell), providing and requesting services. This is achieved through an API presented in subsection 3.1.

Part of the work performed in Subsection 3.1 is about enabling further modules to be integrated into the API later on, such as the Data Processing module covered in D2.1 [1]. Data processing pipelines are run as part of the API to prepare training and inference data for the model.

This section is structured as follows: we present work performed to develop 1) the Intelligence Layer Coordination API (subsection 3.1), 2) models, part of the AI Analytics module, developed for this iteration and ICOS Alfa release (subsection 3.2), and then 3) strategies to ensure trustworthy AI in the current API in the future (subsection 3.3). Finally, we document licensing and instructions to download and set up the software in subsection 3.4, and limitations and future work in subsection 3.5.

3.1 Intelligence Layer Coordination module

The Intelligence Layer Coordination module is responsible for coordinating the use of machine learning models across the Cloud Continuum. This includes implementing policies for sharing, updating, and using models. It also maintains metadata about the models in its model registry, such as their descriptions, algorithms, and testing performance metrics.

The Intelligence Layer Coordination module also provides graphical interfaces for monitoring and managing models and APIs for Machine Learning Operations (MLOps) and storage. These interfaces make it easy for users to interact with the Intelligence Layer and manage their models.

Models callable from this API are part of the Analytics module and thus are described in subsection 3.2.

3.1.1 Functional description

Users can integrate with this API mainly using command line interfaces or Python scripts from an Integrated Development Environment (IDE) or software like Jupyter Notebooks. This API also includes a *Swagger UI* as Graphical User Interface (GUI) for documentation and testing purposes. The software works using the Python library BentoML [13]. A diagram of the workflow followed can be seen below in Figure 4.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	19 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

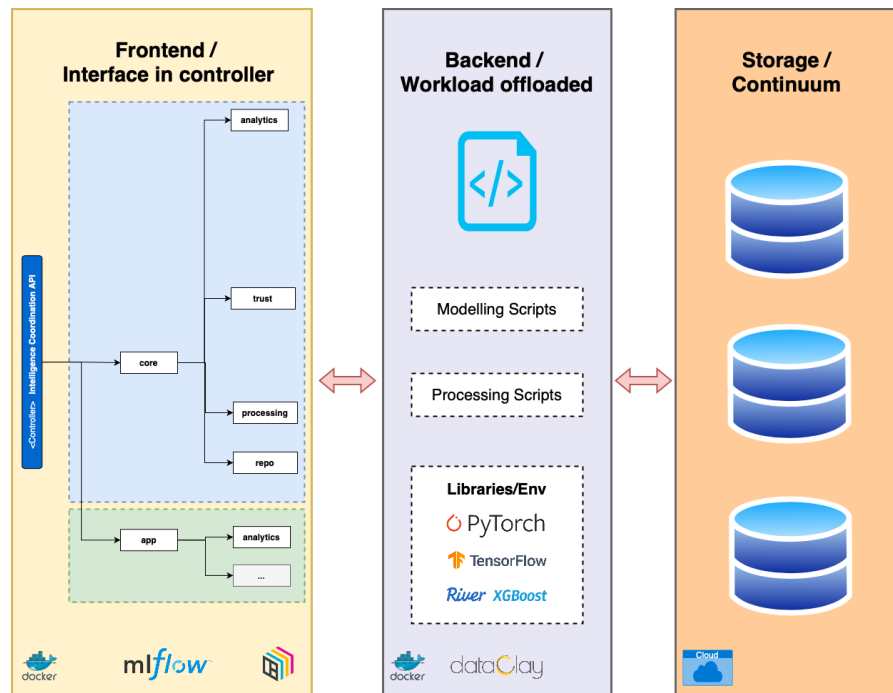


Figure 4: High-level architecture of ICOS Intelligence Layer API

The ICOS applications in the Security, Meta-kernel, and user layers interact with the left-hand side API <https://swagger.io/tools/swagger-ui/> (see Figure 4 above). The API then invokes the backend components internally. These components will be described later in this section and mapped to their respective Intelligence Layer modules.

The API runs Machine Learning (ML) services from multiple frameworks as a service and runs on a server (an ICOS Controller by default) that orchestrates AI workloads.

The goal of this API is twofold:

- ▶ First, models can be pre-built and added to the API as specified in a developer guide mentioned in Annex III: Intelligence Layer Coordination API Development Guide. The API outputs model predictions or information about a new model trained in this scenario. This is performed for easy integration of ML models with automated functions of the operating system developed in ICOS.
- ▶ Second, part of this API is targeted to extend ML libraries to make them available to a technical user in order to save storage resources in devices with access to the API. In this context, the API returns a framework environment to allow users an easy plug-and-play with the environment already available in the API.

In the API IT-1 version for the ICOS Alfa release, four main models are supported regarding task offloading: CPU utilisation prediction, energy consumption forecasting, load time series forecasting and load-balancing related anomalies. The priorities regarding the models to be implemented for the ICOS Alfa release were agreed upon among partners in a session held on the 8th of June, 2023. A screenshot of the result of this session is present in Figure 5.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	20 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

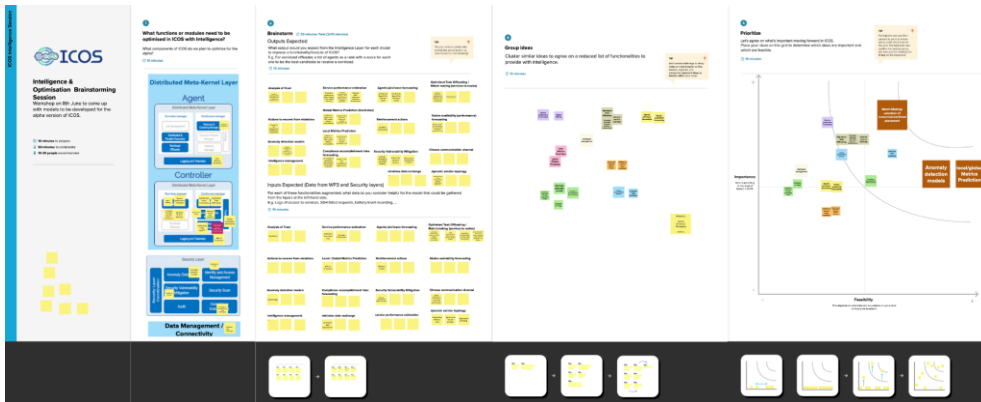


Figure 5: Result of remote workshop for the Intelligence Layer organised by CeADAR

With this API, its models and workload distribution, three different requirements from D2.1 [1] are already met: CM_FR_15, CM_FR18 and CM_FR_20. These are shown in Table 4 below.

Table 4: Requirements starting to be met with the IT-1 of the Intelligence Layer Coordination module [1]

Requirement ID	Requirement Name	Description
CM_FR_15	ML Scalability and maintenance	ICOS SHOULD perform model training detached from ICOS nodes. By training the models on a separate platform with more computing power, it is possible to produce models with higher accuracy and reduce the time required to train them. Once the models are trained, they can be deployed on ICOS nodes to make predictions and carry out specific tasks.
CM_FR_18	MLOps frameworks	ICOS MUST allow the storage, retrieval and modification of AI models available to be used by clients and AI-as-a-Service (AIaaS) providers from ICOS.
CM_FR_20	Continuous learning	The ICOS SHOULD support algorithms for continuous learning, adapting to drifts and shifts and avoiding catastrophic forgetting.

3.1.1.1 Fitting into the ICOS architecture

The API, developed as part of the Intelligence Layer Coordination module, powers different parts of ICOS with intelligence which is centralised in the Intelligence layer and mainly reachable through this API. See Figure 6 below.

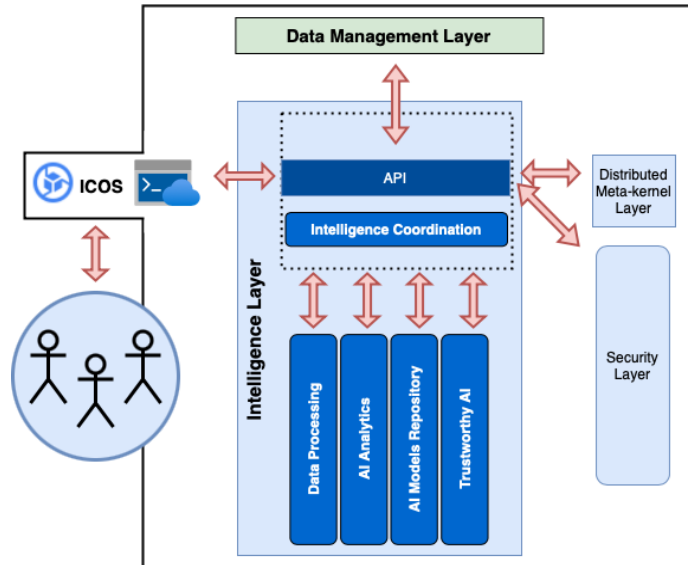


Figure 6: Intelligence Layer Coordination module in ICOS

The API sits in an ICOS Controller with the model registry. The Intelligence Layer Coordination API assumes that this server is part of the Controller in the current deliverable and for the ICOS Alfa release. In upcoming deliverables, the server will be able to be located in ICOS agents for further workload distribution; this would involve privacy measures for how the controller-based models learn, such as Federated Learning (FL). These strategies are covered later in this document and will be implemented in upcoming deliverables. The architecture of ICOS concerning the Intelligence Layer Coordination API is as follows, see Figure 7.

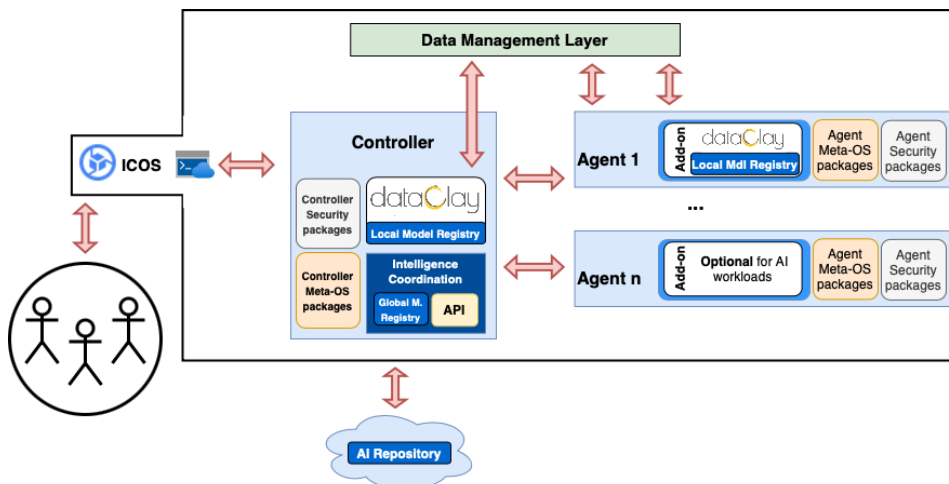


Figure 7: Architecture of ICOS interacting regarding the Intelligence Layer Coordination API

Meta-kernel modules, applications running in the network topology and users are able to reach the Intelligence Layer API as long as they can reach the controller.

3.1.2 Technical description

The software has been developed using Python 3.10. The Intelligence Layer Coordination module API supports asynchronous calls to ML models and has native integration with MLFlow [10], Grafana [11] and Prometheus [12] to export model usage statistics as a system service. This enables further optimisation of AI offloading if required in future releases. The Intelligence Layer Coordination module API is AI-framework agnostic, and both the API and models deployed, even when offloaded, run in Docker containers.

3.1.2.1 Prototype architecture

The structure of the Intelligence Layer Coordination module is illustrated in Figure 8 below.

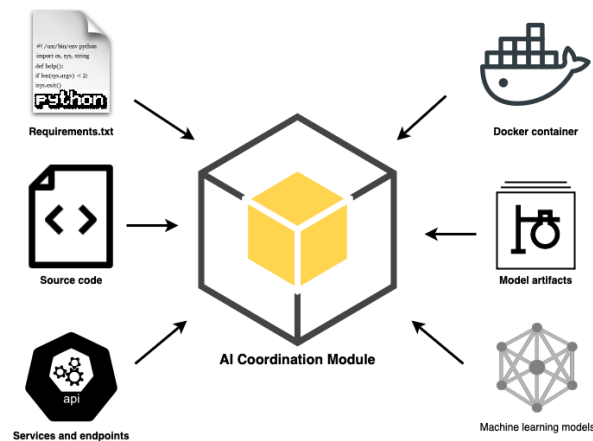


Figure 8: Content of the Intelligence Layer Coordination module

It has a model registry in the server where the API is executed, which contains models trained in the server's controller or offloaded by this. Strategies for federated learning in subsection 3.2 enable local model registries at workload offloading nodes. However, this part is out of scope for the current IT-1 and ICOS Alfa release.

3.1.2.2 Description of components

The current version of the API supports the following Intelligence Layer's modules:

- ▶ **Data Processing:** This module, which is not explicit in the API but part of the functionality offered, provides libraries and frameworks for processing and transformation at scale. It also coordinates data access, storage, and preprocessing using the Data Management module,
- ▶ **AI Analytics:** This module aggregates libraries and algorithms for training and forecasting using state-of-the-art methods. Its goal in this version is to provide forecasts that could be used to optimise the behaviour of the Meta-kernel. More detail is provided in subsection 3.2.

There are two Intelligence Layer modules non yet fully supported:

- ▶ **AI Models Repository** is partially supported by the creation of model registries. Full support for this module creating an online repository is expected for the Final ICOS release in M36, as stated in the project's proposal.
- ▶ **Trustworthy AI** is planned for the ICOS Beta release. The approach that will be followed in implementing this module for the ICOS Beta release is covered in subsection 3.3.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	23 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

This API consists of several pieces that ensure efficient and effective data handling for machine learning purposes. The flow followed by running internal functions in the Intelligence Layer Coordination module is illustrated in Figure 9 below.

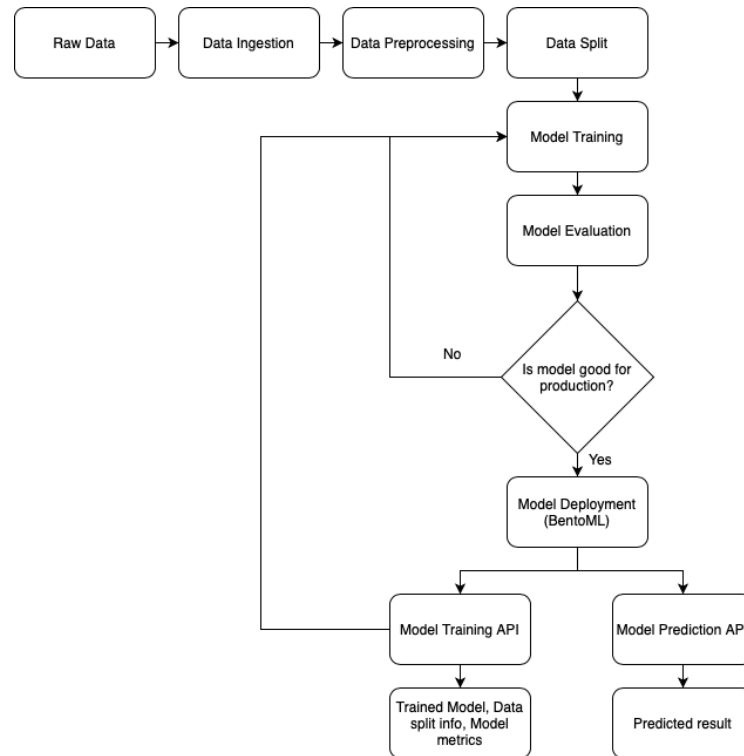


Figure 9: Intelligence Layer Coordination module API flow

First, raw data is loaded from the Data Management layer with the Data Preprocessing module. Then Preprocessing, splits and model training are performed. As part of model training, the model is tested to understand its performance in historical data. If the model performance is over a threshold, this is deployed and available in the API’s model registry for inferencing.

Data ingestion relies on data provided by the Data Management layer. The API has a component in charge of obtaining the relevant source of data (for a given model) from the Data Management and triggers preprocessing and model training. Once a model is trained, it is available in the model registry to be inspected in BentoML for AI experimentation purposes. Models deployed are available for inference in the IT-1 of the Intelligence Layer Coordination module API and for model reuse at the training stage and transfer learning in future releases. While this flow is not explicit in the API IT-1, models integrated follow it.

3.1.2.3 Technical specifications

In this version of the API, we have used open-source Python libraries such as BentoML [13], Pandas[14], statsmodels[15] and scikit-learn[16] to 1) create the API with a model registry (2) and workload distribution, (3) preprocess data and (4) integrate an initial set of models.

The software developed is compatible with the most popular ML-related frameworks: Pytorch[17], Tensorflow[18] and Scikit-learn among others. The API developed in its current version makes it possible to run them as a service in low-end devices and distribute the workload across the network topology.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	24 of 66
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

The definition of the API can be found in Table 5 below.

Table 5: Intelligence Layer Coordination API definition

Partner	CeADAR
Product	Intelligence Layer Coordination module
ICOS layer	Intelligence Layer module
Version	IT-1 (ICOS Alfa version)
Release Date	September 30th, 2023
LICENSE	This product is under a Software License by CeADAR Ireland (see Section 3.4)
API-Endpoint	http://host:3000
Environment	Docker-container
Dependencies/ libraries	Python 3.10, NumPy, Pandas, BentoML, xgboost, TensorFlow, scikit-learn, statsmodels
Detailed description:	
<p>The Intelligence Layer Coordination module facilitates optimisation, predictive analytics, and applying machine learning models across the Computing Continuum. It entails implementing policies for utilising, sharing, and updating models. This acts as an interface and provides coordination between the Meta-kernel and user layers providing and requesting services. This component helps to coordinate with other Intelligence layers of the same domain in the continuum, providing ICOS with the ability to learn collaboratively.</p>	

Internal components of the API have been first tested through the debugger in the IDE Pycharm with a set of unit tests. Later, functions exposed in the API were tested using both the *Swagger UI* and *curl* (command line). Testing for each of the functions exposed is shown in Annex I: Testing the Intelligence Layer Coordination API.

3.2 AI Analytics

The AI Analytics module is a toolkit for building and deploying machine learning models. It provides a variety of algorithms and libraries, including advanced machine learning algorithms to learn from data streams incrementally, lowering computational costs. The module is designed to be efficient and scalable, making it ideal for use in large-scale projects.

This first version of the module aggregates four methods:

1. Intelligent energy-aware task offloading,
2. Load time series forecasting,
3. Forecasting upcoming load anomalies (related to the first one) and
4. CPU utilisation forecasting to improve workload distribution.

These methods are covered in subsections 3.2.1, 3.2.2, 3.2.3, and 3.2.4, respectively. Methods 2 to 3 are also associated with trained, tested, and deployed ML models in the current version (IT-1), whereas the implementation of the Decentralised Deep Reinforcement Learning (DDRL) model for method 1 is expected for later versions, as it has multiple dependencies and prerequisites from other modules/models (see subsection 3.3.1). The validation of methods 2 to 3 for IT-1 is presented in Annex II: Validation of AI Analytics models.

The techniques implemented and integrated with the Intelligence's Layer Coordination API for this method are a step forward to meeting some of the requirements defined regarding the AI Analytics module. The requirements that started to be satisfied are covered in Table 6 below.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	25 of 66	
Reference:	D4.1	Dissemination:	PU	
	Version:	1.0	Status:	Final

Table 6: Requirements starting to be met with the IT-1 of the AI-Analytics module [1]

Requirement ID	Requirement Name	Description
CM_FR_17	AI for resource management	ICOS MUST classify infrastructure that facilitates the resource management on the network and implements various resource management algorithms for the continuum, as well as clustering the various resources in the ecosystem to optimize those available.
CM_FR_20	Continuous learning	ICOS SHOULD support algorithms for continuous learning, adapting to drifts and shifts and avoiding catastrophic forgetting.
SST_FR_04	Secure & robust model training	ICOS SHOULD provide the ability to train models in a federated learning fashion to ensure privacy and trust. Privacy can be guaranteed by training data on the edge. Robustness is assured by assessing the data against anomaly detection.
SST_FR_09	Anomaly detection	ICOS MUST provide a mechanism for the detection of anomalies (e.g. any kind of abnormal situations, including potential security threats, that is recorded in application, system or network logs) in the applications/services on the cloud/network/edge provider.

3.2.1 Intelligent energy-aware task offloading

The AI Analytics module presents, among others, intelligent capabilities in proactive and smart decisions about task computations. The decisions are expected to be supported in distributed locations, whereas the main objectives to drive the task offloading decisions should ensure both a low task drop ratio and high energy efficiency.

This advanced functionality will be implemented using state-of-the-art decentralised reinforcement learning techniques, exploiting proactive information about the ICOS agents' load traffic and energy consumption status. To this end, this version of the AI Analytics module describes the objectives and the concepts underlying the intelligent task offloading functionality, as well as provides two of the required learning algorithms for (i) predicting the anomalies in the load patterns and (ii) forecasting the future load values of each ICOS agent. The following subsections outline the conceptual and algorithmic framework associated with the intelligent task offloading model.

Overview and suitability

This subsection describes an intelligent scheme for Decentralised Computation Offloading with Energy Efficiency-aware (DECOFFEE) vertical and horizontal actions across cloud/edge layers of the ICOS. The algorithmic implementations of the DECOFFEE model are based on the principles of Deep Reinforcement Learning (DRL), exploiting also Supervised Learning (SL) model outputs, and can be used as a functionality within the ICOS Intelligence Layer. The overall framework is basically a

decentralised computation offloading scheme that targets to dispatch the tasks efficiently in terms of latency and energy efficiency and also allows both vertical and horizontal offloading decisions. The suitability of DECOFFEE to the ICOS principles lies in two aspects: i) it is a decentralised scheme enabling each ICOS agent to effectively handle its computation task in a distributed manner, and ii) it allows both vertical and horizontal decisions within and across the ICOS continuum, which is in line with the ICOS objectives to go beyond the constantly vertical management and the monolithic centralised orchestration.

DECOFFEE system architecture

In this subsection, the system model for implementing the DECOFFEE scheme is presented. As shown in Figure 10, the system consists of three layers: the IoT layer, the Edge layer, and the Cloud layer. The IoT layer generates tasks that can be executed by the ICOS agents located in the Edge layer. The Cloud layer is used as a backup option for offloading tasks that have high computational requirements and relaxed deadlines. We make the following assumptions about the system model:

- ▶ The tasks are indivisible and must be executed as a whole by a single agent. This assumption can be cancelled in future work to allow divisible tasks.
- ▶ The agents are DRL agents that have been pre-trained with an objective function that jointly optimises latency, drop ratio, and energy consumption. The training process is out of the scope of this subsection and will be discussed in detail in the next versions.
- ▶ The agents can observe the local state of their own tasks (which are the task characteristics collected by the respective geographical IoT area that each agent covers), as well as some predictions about the load and energy consumption of other agents in the same cluster. These predictions are provided by the ICOS controllers, which collect and exchange data among the agents periodically.
- ▶ Based on the observed state (local task features and predictions about the other agents), each agent can make a three-way decision: to execute the task locally, to offload the task horizontally to another agent in the Edge layer, or to offload the task vertically to the Cloud layer. The decision is made by the DRL model of each agent, which aims to achieve a good balance between low latency, low drop ratio, and high energy efficiency (or low energy consumption). The DRL models attempt to jointly guarantee this threefold objective under computational capacity constraints, latency constraints, and consumption limitations, which are competitive factors to the objectives.

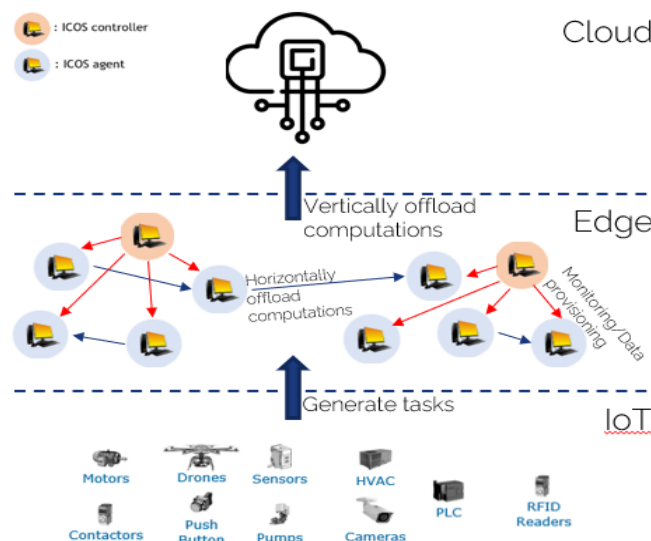


Figure 10: ICOS system architecture of the DECOFFEE scheme enabling both vertical and horizontal actions towards latency-aware and energy-efficient distributed task offloading policy

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	27 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

In the current document, we omit the mathematical formulations of the system model (communication latencies, system notations, state-action-reward formulas, optimisation problem formulation, etc.) and focus on providing an overview of the concept. The architectural and structural overview of the target neural network model will be given in the next subsections.

Structure and prerequisites towards DECOFFEE implementation

In this subsection, the Deep Neural Network (DNN) structure of the single-agent DECOFFEE model, which is based on the Deep Q-Network (DQN) algorithm, is described. The DQN is a Q-learning method that uses a neural network to approximate the Q-function[3]. The DQN is trained offline with a set of equations that model the delays and energy consumptions of different actions (possible actions from a given state are the available offloading decisions). The Q-value of a given action quantifies the “quality” (the ‘Q’ letter) of performing the particular action a from the given state s , thus, the DQN estimates the two-variable function $Q(s,a)$.

Figure 11 shows that the inputs of the DQN are the state vector, which basically describes the environment status at a given time slot. In the DECOFFEE case, the environment is described through the characteristics of the task arrived by the agent, the available space for computation of the other agents, and the predictions about the load and energy consumption of the other agents. All of these metrics are included in the state vector due to their direct impact on the final decision and the objective function, aiming to select those actions that produce low task drop ratio and high energy efficiency at the whole system level.

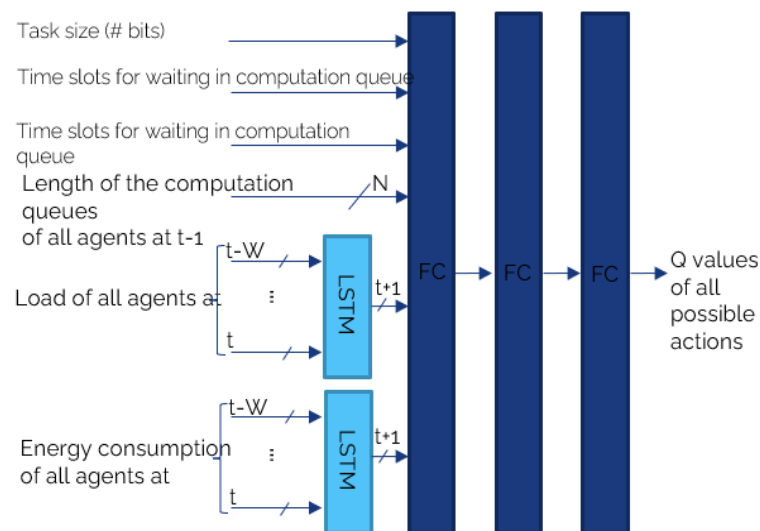


Figure 11: The decentralized DRL model (based on Q-learning) running on a single ICOS agent towards implementing the overall DECOFFEE algorithmic scheme

Different equations are employed to account for the local computations, the round trips with the cloud, the vertical/horizontal transmissions, and the waiting times in the queues. The input of the DQN is a vector that contains the following features: task size, waiting time in computation/transmission queues, length of the computation queues of all agents, and future load/consumption estimations of other agents. The output of the DQN is a scalar value that represents the Q-value of each action. The action with the highest Q-value is chosen by the agent. The training phase of the single-agent DQN aims to properly adjust the DNN weights, to ensure that, when an agent infers the model, it will suggest actions/decisions that guarantee successful task completion and optimal energetic resource usage.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	28 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

To train the DQN, at least three datasets are required: one for the task traffic at the edge layer, one for the energy consumption of different types of tasks, and one for the task characteristics, specifically:

- ▶ The task traffic dataset consists of *load time series* from various servers that generate tasks for the edge layer.
- ▶ The energy consumption dataset consists of *energy consumption time series* from various servers that execute different types of tasks.
- ▶ The task characteristics dataset consists of a *list of tasks with their features*, such as size, arrival time, deadline, processing duration, etc.

All of these datasets are prerequisites to simulate different scenarios and evaluate the performance of the DECOFFEE model. In the current version of the DECOFFEE model, the first dataset has been analysed (see subsections 3.2.2 and 3.2.3) to provide the first critical input of the target DECOFFEE model. The overall idea to implement the DECOFFEE model relies on decomposing it into several components for flexibility purposes, especially into:

- ▶ **Component 1:** Load time series forecasting (subsection 3.2.2) or Prediction of load anomalies (subsection 3.2.3) to feed the input of the DECOFFEE model.
- ▶ **Component 2:** Energy consumption time series forecasting (in later versions) to act as the second input of the DECOFFEE model. The energy consumption dataset will be created by NKUA using the local Kubernetes cluster-based premises to track the course of the server consumption (in Watts) during the execution of diverse computational tasks. To this end, tools like Scaphandre and Prometheus will be developed.
- ▶ **Component 3:** Traces with task features will be given by an open dataset of the Google cluster, with the rest of the required DECOFFEE inputs being available by this dataset (arrival time, processing time, task deadline, etc). This component is basically the training of the overall DECOFFEE model.

3.2.2 Load time series forecasting

In this subsection, the principles underlying the development of load forecasting models are outlined. Key motivations to provide estimations about the upcoming load values (e.g. of the ICOS agent or any other computational node) in the ICOS AI Analytics module are (i) to allow for timely knowledge about overload, thereby enabling proactive actions (e.g. increasing the computational capacity of nodes or offloading tasks properly) and (ii) to offer the DECOFFEE model knowledge about the computational effort of the active ICOS agents, thus facilitating the offloading decisions.

Neural network models were deployed to ensure accurate estimations of the traffic series, given their proven ability to fit complex functions. Load forecasting models were trained based on a 5-areas dataset, including the traffic of the network derived by IoT devices. Below, the dataset and the model structure for predicting the area-specific traffic series are detailed.

Dataset and model structure

To train the load forecasting model, we used a Long Short-Term Memory (LSTM) network, which is, in general, appropriate to resolve temporal dependencies that are met in time series data. The training was based on the dataset extracted in [4]. This dataset contains labelled data for traffic series (data rate in bytes per 15-minute period) in 5G cellular networks for 6 months (1 sample per 15 minutes), generated according to 5G specifications. Considering a topology of 5 partially overlapped 5G cells with varying:

- ▶ Network area information (e.g., area identifiers of a group of associated devices: ID 1, 2, 3, 4, and 5),
- ▶ Subscription categories (e.g., the policy of a group of subscribed devices: platinum, gold and silver) and
- ▶ Personal equipment devices (e.g., device type information of a device: IoT device, vehicle, cell phone, smartwatch and tablet).

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	29 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

The dataset includes dynamic traffic patterns, accounting for spatial-temporal variations during simulations, such as handovers, User Equipment (UE), mobility/velocity, subscription prioritisation, area adjacency, and device-specific characteristics. The total load of each IoT area was computed as the summed data rate across its associated subscribers and devices. To build the LSTM model for predicting the total cell load, we split the whole dataset into training and testing sets. The training set included the load measures during the first 5 months plus the first 3 weeks of the 6th month, whereas the last week of the 6th month comprised the testing set.

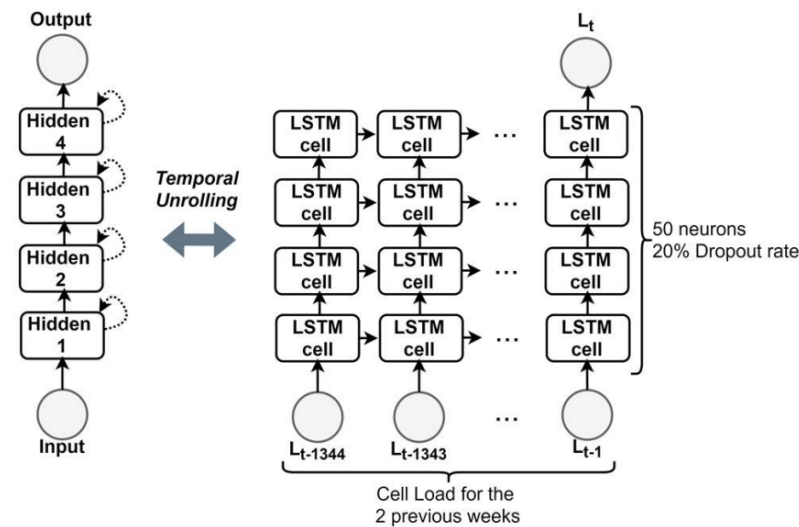


Figure 12: LSTM architecture for load time series forecasting in rolled and unrolled forms.

Figure 12 depicts the structure of the LSTM used to predict the load of each IoT area. The feature vector for predicting one target sample at time t consists of the load values of the previous 2 weeks (or in general, W points, where W is the lookback window). Four hidden layers were stacked, each one having 50 hidden LSTM units and 20% dropout rate for eliminating overfitting. The loss function used to update the neural network weights during backpropagation was the Mean Squared Error (MSE). Hence, whenever a batch of 64 training samples passed the LSTM layers, the MSE was computed as the sum of squared differences between the actual and predicted load values.

The validation of the model is presented in Annex II: Validation of AI Analytics models.

3.2.3 Anomaly detection

In general, the purpose of an anomaly detection algorithm is the proper discrimination of the data samples in ‘normal’ or ‘abnormal/outlier’ classes. In this subsection, we also built an anomaly detection model to predict the traffic anomalies in each of the five IoT areas. Anomalies are included in the traffic time series of the data and are defined as unexpectedly high bursts of the load, relative to the average load course. Thus, time points corresponding to anomalies have been labelled accordingly, whereas the rest of the time points have been coded complementary (e.g. ‘0’ for no anomaly and ‘1’ for anomaly).

The validation of the model is presented in Annex II: Validation of AI Analytics models.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	30 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

3.2.4 Metrics forecasting

The AI Analytics module can predict system performance metrics towards the enhancement of workload distribution. In this version of the module, the AI Analytics module can train and use models to predict CPU utilisation one minute ahead. This functionality has been implemented using ARIMA as a baseline method, as well as state-of-the-art adaptive learning algorithms for continuous learning due to the dynamic CPU changes and task shifts in different nodes over time. The techniques used, decisions made, testing, and specifications are described in the following subsections.

In this version, the Intelligence Layer Coordination API offers, among others, a mechanism for estimating the CPU consumption of the ICOS agents. Its API service encompasses a well-structured mechanism made to define a reference to the saved model in the AI analytics module, which in this case, is a model to predict CPU utilisation one step in the future for a given sample of prior CPU usage values. It is a cornerstone of the system's proactive resource allocation and management techniques.

Technical implementation

The baseline model is based on ARIMA with a one-step-ahead approach, with a parameterizable train-test split, set by default at 80% for training and 20% for testing. API Swagger also contains the XGBoost as an option. The first model uses incremental learning to adapt to the latest CPU workload dynamics. Each one-minute-long timestamp reflects a single time unit. In the context of an autoregressive task, the dataset is partitioned into windows of different sizes, obtaining the best results for a window size of 12. Thus, 12 minutes of historical data per testing example. While other models were investigated, incremental models stand out due to their ability to adapt to the intrinsic non-stationarities of the workloads run in the server used for data collection. This data-gathering procedure was carried out on an *Orange Pi 5* [5] edge device for a month. A testing dataset using the last 6,000 data samples of such month was used for model evaluation using a purely incremental *Prequential Evaluation* [6], being this a common testing mechanism in continuous machine learning. The initial remainder of the month was used to pre-train the models before the prequential (or test-then-train) evaluation.

This predictive model tested ARIMA, an incrementalised ARIMA, online incremental learning models, deep learning and classical machine learning models. Incremental learning models obtained the lowest errors overall. Thus, these can be up-to-date, running indefinitely and learning from continuous data streams. Annex II: Validation of AI Analytics models describes the results obtained during the validation phase.

Description of components

The *psutil* library was used to collect system utilisation data, such as CPU utilisation, memory, disk, and sensor information. This approach took a month on an Orange Pi 5 device, with 80% of the time dedicated to pre-training and 20% to a test-then-train split (prequential assessment). Each timestamp's data was continuously saved for further study. In our study, incremental models outperformed all others and became the standard approach. The dataset was separated into 20% for testing and 80% for model pre-training to design and validate the model. This method guaranteed that performance evaluations were in line with current data, keeping to the principles of incremental learning. SMAPE was preferred over MAPE for error evaluation during testing because of its sensitivity to tiny values and resilience to normalisation difficulties [7]. MAE was also considered viable due to its lack of denominator and resistance to normalization effects.

Technical specifications:

The model deployed is an incremental setting of ARIMA using *statsmodels*. Its use entails the use of the *Swagger API* or *curl* through the *bentoml* service, as previously shown in subsection 3.1. Training and testing pipelines may be fully parameterised using this API, which is a crucial component of the design. See the endpoints in

Table 7 below.

As explained before, users may easily configure the appropriate parameters using the *Swagger UI*. Backend pipelines will be run after parameter selection using the button "Execute" or by using *curl*.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	31 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

Table 7: AI Analytics API endpoints

HTTP request (ICOS Shell)	description	parameter	ICOS component	responsible	component name	component port	component endpoint
POST /analyticx_predict_cpu_utilisation/	Predicts the CPU utilisation one step in the future	App descriptor	Analytics module	CeADAR	CPU usage predict API	3000	POST /core/analyticx_predict_cpu_utilisation/
POST /analyticx_train_cpu_utilisation/	Trains a model to predict CPU utilisation	App descriptor	Analytics module	CeADAR	Train CPU usage API	3000	POST /core/analyticx_train_cpu_utilisation/
POST /analyticxs_learnone_utilisation/	Trains a model in a streaming fashion using its lags and corresponding label	App descriptor	Analytics module	CeADAR	Train CPU usage API	3000	POST /core/analyticxs_learnone_utilisation/

Model results are shown in the same interfaces where the commands are run. In this model, a single scalar value is to be shown, representing the one-step-ahead predicted with the CPU utilisation model.

3.3 Trustworthy AI

The Trustworthy AI module is essential for ensuring the responsible use of AI in ICOS. It aims to provide a tool for developing models while respecting privacy and trustworthiness policies. By adhering to these requirements, AI models in ICOS can be developed to provide reliable results while protecting user data and promoting ethical standards. This can be done by:

- ▶ Providing explainable AI algorithms that give meaningful insights into the output of models to the different layers in ICOS and monitoring these algorithms over time.
- ▶ Training models in a federated learning fashion to protect data privacy in datasets that contain user-specific data.

The source code of this module is out of scope for the IT-1 (see subsection 3.5). In this subsection, we aim to summarise the approach that will be followed in implementing this module for the ICOS Beta release as well as define strategies to be followed and work to be performed in future releases (IT-2, ICOS Final release) to ensure AI trustworthiness. The sole definition of these is already an asset in ICOS as the Intelligence Layer is being designed for this release considering the future addition of these pieces.

The list of requirements from D2.1 [1] expected to be met by implementing this module is shown in Table 8.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)			Page:	32 of 66
Reference:	D4.1	Dissemination:	PU	Version:	1.0
				Status:	Final

Table 8: Requirements involving Trustworthy AI [1]

Requirement ID	Requirement Name	Description
CM_FR_16	Federated Learning benchmarking	ICOS MUST be able to train models over open-source federated learning frameworks. Models trained on ICOS nodes (the edge or on the cloud) will be able to share knowledge when applicable to learn collaboratively.
SST_FR_04	Secure & robust model training	ICOS SHOULD provide the ability to train models in a federated learning fashion to ensure privacy and trust. Privacy can be guaranteed by training data on the edge. Robustness is assured by assessing the data against anomaly detection.
SST_FR_05	Providing trustable models	ICOS SHOULD use models ethically unbiased. By using these models, ICOS strives to preserve trust in the outputs of the ICOS system and provide a fair and equitable experience for all.

3.3.1 Fitting into the ICOS architecture

Figure 13 depicts the integration of the Federated Learning, Explainable AI algorithms, and the monitoring toolset, which will be integrated into Intelligence Layer Coordination’s API in the ICOS Beta release as part of the Trustworthy AI module, with the other layers and components in ICOS. The Trustworthy AI works within the Intelligence Layer to ensure that the data is correctly stratified for tackling bias in datasets and that algorithms are used to analyse data using explainable AI.

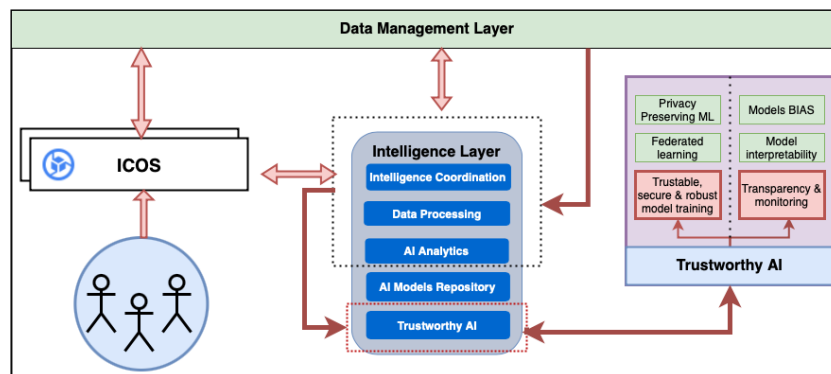


Figure 13: Up-to-date ICOS architecture concerning the Trustworthy AI module.

An ICOS user or a Meta-Kernel or Security Layer application interacts with the data management system and the Intelligence Layer Coordination API. The Intelligence Layer Coordination has a specific service for trustworthiness under its core and user sections.

Finally, to ensure trustworthiness, it is necessary to define trusted storage endpoints overseen by the Data Management Layer. This is achieved, first of all, by going through secure communications channels (e.g. by using TLS and/or mTLS cryptographic protocols, already supported by the Data

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	33 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

Management components). In addition to that, the Data Management Layer must perform data validation, to check for format correctness. Data is then loaded by the Data Processing module and sent to the Trustworthy AI module through the Intelligence Layer Coordination module for safe and resilient model training, ensuring no data leakage during model training.

3.3.2 Approach and design of the components of the Trustworthy AI module

The Trustworthy AI module is organised into two main parts:

- ▶ **Transparency and monitoring:** Providing model explainability through a series of AI interpretability algorithms and model monitoring. This seeks to deploy methods for understanding how AI models make decisions and that decisions are taken consistently over time. This can be helpful for transparency purposes, identifying bias and building trustworthiness, as well as debugging models at the model training stage, as models are able to display what leads to a decision or forecast. This is covered in subsection 3.3.2.1.
- ▶ **Federated learning and privacy:** Ensuring trustworthy, secure, and robust model training through federated learning techniques and privacy-preserving AI. This allows models to be trained on data that remains on users' devices, protecting user privacy by preventing sensitive data from being shared with a central server. This is covered in subsection 3.3.2.2.

Both explainable AI methods and the ability to train models using federated learning can preserve privacy and protect data used for training the models by avoiding direct querying of local models, including user-specific data. This gives the different ICOS layers confidence in the models' output.

The implementation is planned for the ICOS Beta release while technical description and specifications of this module will be provided in D4.2.

3.3.2.1 Explainable AI and Monitoring

Using explainable AI algorithms for model interpretability and monitoring their performance is vital in ICOS for both ensuring transparency and relying on the models provided by the Intelligence Layer.

- ▶ A toolset of explainable AI techniques to be integrated into the API will be deployed to identify and mitigate biases, enhancing the accuracy and fairness of AI systems by giving insight into how models make judgments. Its goal is also to spot biases or flaws in the model architecture or underlying training datasets.
- ▶ Similarly, a toolset to monitor incoming data in the Data Preprocessing module and for the continuous evaluation of models deployed will be integrated into the API to ensure that models can be trusted in the event of potential data shifts. Its goal is the automated identification of changes and anomalies in the feature space and model performance.

Model explainability

In the first group of visualisations, the Trustworthy AI module inspects AI models via interactive AI dashboards that show the inner workings of the models available in the Intelligence Layer Coordination model registries. Figure 14 below shows an example visualisation of features' importance leading to different forecasts over a given algorithm in an evaluation test set.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)			Page:	34 of 66
Reference:	D4.1	Dissemination:	PU	Version:	1.0
				Status:	Final

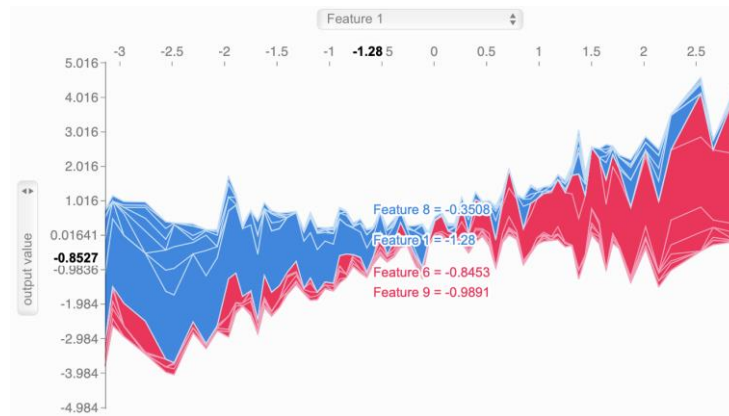


Figure 14: Importance of features (y-axis) leading to outputs in test set forecasts (x-axis)

The Trustworthy AI module provides a collection of these visualisations, giving technical users of the system the ability to understand the model behaviour and identify bias. Seeking to meet Article 13 of the AI Act[19], the aim is to make models transparent enough to allow their output interpretability and understand their limitations.

Model monitoring

Models in ICOS are expected to be designed and developed in conjunction with the Trustworthy AI module so that these can be overseen and monitored while deployed. Thus, the second group of techniques to be included in this part of the Trustworthy AI module is to identify drifts, shifts, and anomalies in the feature space of an AI model. Signs of anomalies, dysfunctions, and unexpected performance are raised as part of this module. Figure 15 below shows a model being monitored, detecting model or concept drifts and raising alarms when the model underperforms. Changes in the data received by a model may impact its performance.

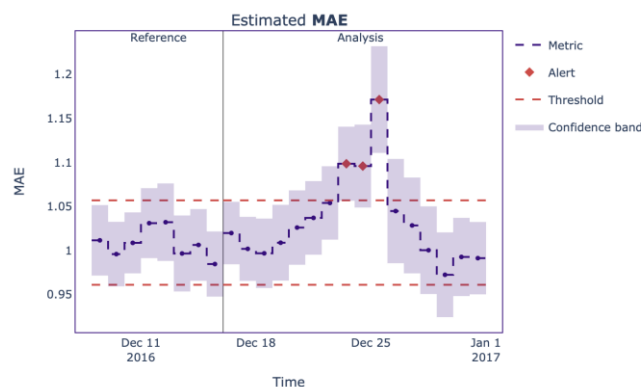


Figure 15: Example monitoring system that raises alerts when the mean absolute error exceeds a threshold

This continuous monitoring process aims to ensure the ongoing safety and effectiveness of the developed models. This monitoring system collects, reports, and analyses incoming data from the Data Processing module throughout its lifetime, allowing for the continuous re-evaluation of the models.

This module uses testing and validation set error results, saved as metadata in the Intelligence Layer Coordination API as part of the IT-1 to understand deviation from the expected error.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	35 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

3.3.2.2 Federated learning and privacy

The ICOS platform should be able to support Federated Learning (FL) among the participating nodes in order to speed up training times on the one hand, and on the other hand to satisfy latency requirements and ensure privacy protection.

Once a process is initiated that requires effective ML model training (e.g., resource optimization), the appropriate ICOS Controller defines the set of ICOS participating nodes in FL model training. During the training round (multiple training rounds may be required), the server and the participating client nodes do not exchange any data. The centralised server, where periodic model updates take place, can be either a local ICOS Controller or a cloud server. This approach can be applicable in a wide variety of ICOS use cases, where centralised data collection would require multiple communication rounds with all involved data generation end-points, that can be scattered over wide geographical locations.

In this context, ML model training takes place in each participating ICOS node, via the locally available dataset. A basic assumption is that the appropriate model per case has been made available to the ICOS nodes at a previous step via the AI Models Repository module of the Intelligence Layer (common ML model training per node). Periodic ML model training takes place, where at periodic time intervals, the generated ML model parameters per node are sent to the central server for model aggregation, as it was previously mentioned. Hence, generated data remains localised, thus ensuring a privacy-oriented approach. Training rounds are updated according to the evaluation criteria that have been set (e.g., MAE, MSE, etc.) or according to requests made by applications.

An open-source FL platform that will be used in the context of ICOS is the Flower framework (<https://flower.dev/docs/framework/index.html>), which allows flexible initialization and training of multiple ML models over various client nodes. An instance of this framework is depicted in Figure 16, where, in this case, only the updated weights of the trained neural network are sent to the central node for model updates.

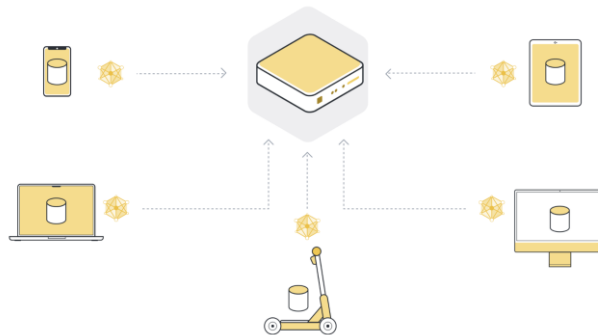


Figure 16: Periodic model training and weight aggregation over dispersed nodes with the help of Flower

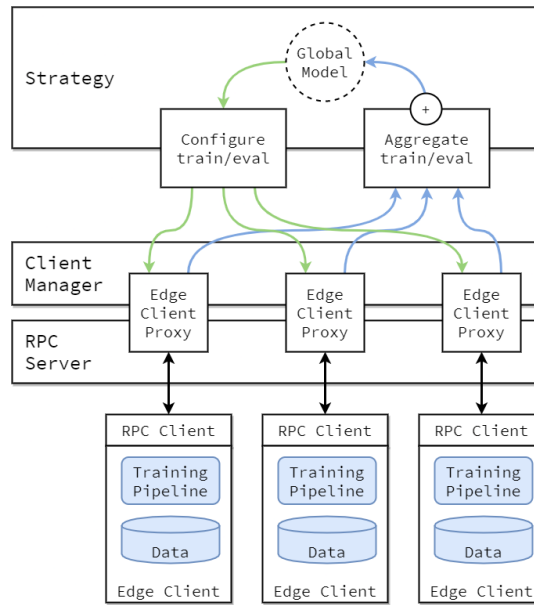


Figure 17: Strategy/server initialization in Flower

Flower allows flexible model training, dynamic configuration and update of client nodes, support of different strategies per case, data loading, local model training, and evaluation according to predefined criteria that can be set by the user. An overall view of the internal architecture is depicted in Figure 17, where model configuration, aggregation, and retraining are made feasible via RPCs to the involved clients.

In this context, an example showing 6 local training curves and their FL version is shown in Figure 18. To this end, we have used a testing set with 50 testing samples per FL agent to illustrate the collaborative performance across samples from all local agents. In this Figure, an indicative example is presented regarding MAE estimation, with the help of six participating clients and four techniques (FedSGD, FedAvg, FedAvgM, FedProx).

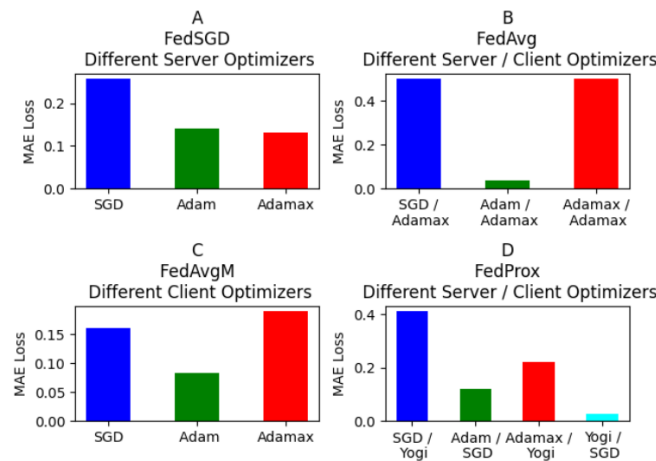


Figure 18: MAE Loss for different Server - Client optimizers

In Figure 19, the loss versus training round is depicted.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	37 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

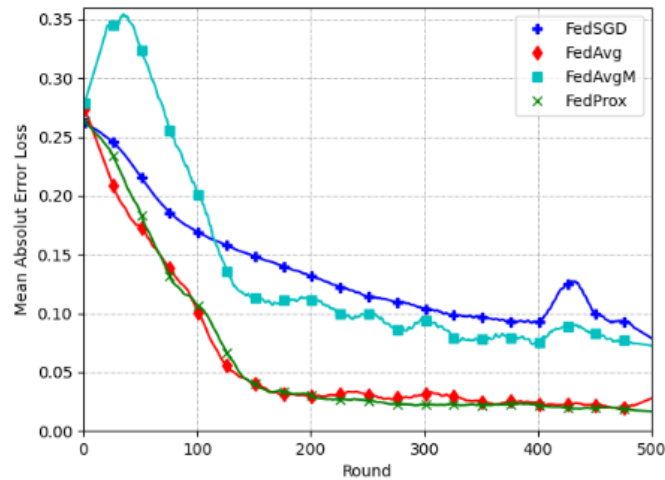


Figure 19: Loss versus round

3.4 Delivery and usage

Deliverable D4.1 for the Intelligence Layer counts with four models callable from an API as described in previous subsections. The API source code is made available through the CeADAR Gitlab production repository.

For its usage, a development guide has been provided to the main partners developing AI models in the Intelligence Layer API. This guide is part of the CeADAR Gitlab production repository made available with the API source code and is also provided in Annex III: Intelligence Layer Coordination API Development Guide.

The following subsections detail information on the API provided and their models, with instructions on how to download and set them up, as well as relevant information regarding licensing.

3.4.1 Package information

Table 9 below represents the dependencies of the Intelligence Layer Coordination API for a successful deployment and service.

Table 9: Intelligence Layer Coordination API dependencies

Package name	Version	Package info
bentoml	1.0.20	https://github.com/bentoml/BentoML
numpy	>=1.15.0,<1.24.3	https://numpy.org/doc/stable/
pandas	>=1.4.3	https://pandas.pydata.org/docs/
seaborn	>=0.11.2,<=0.12.2	https://github.com/mwaskom/seaborn

Package name	Version	Package info
matplotlib	3.7.1	https://matplotlib.org/3.7.1/index.html
scikit-learn	1.2.2	https://scikit-learn.org/stable/whats_new/v1.2.html
notebook	6.5.1	https://docs.jupyter.org/en/latest/
xgboost	1.7.6	https://xgboost.readthedocs.io/en/stable/
tqdm	4.65.0	https://tqdm.github.io/
pydantic	1.10.10	https://docs.pydantic.dev/latest/
scipy	1.10.1	https://docs.scipy.org/doc/scipy/
tensorflow	>=2.4	https://devdocs.io/tensorflow~2.4/

3.4.2 Installation instructions

The ICOS Intelligence Layer Coordination API is built using a Bentoml Docker file that is meant to be run on a Linux machine, either x86 or arm64. It contains the necessary packages and tools for running the API server. To use the Bentoml Docker, we need to install the Bentoml package, import the bento, and then containerize it.

Below we list generic steps to deploy the docker container for the API. A full list of steps will be provided together with D5.1 as an annex.

- ▶ To install bentoml package, run the following command in the shell:

```
pip install bentoml==1.0.20
```

The above command will install bentoml and create a folder named “bentoml” in the home directory of the device.

- ▶ To import the bento, run the following command in the shell:

```
bentoml import BENTO_PATH
```

The above command will import the saved bento and install it in the “bentoml” directory created in step-1.

- ▶ To build the docker image, run the following command in the shell:

```
bentoml containerize [OPTIONS] BENTO:TAG
```

The above command will create a docker image that will contain all the packages listed in section 3.4.1 installed.

- ▶ To run the docker and launch the API service, run the following command:

```
docker run -it -rm -p 3000:3000 docker_name:latest serve
```

The above command would enable the ICOS Intelligence Layer Coordination API server to be launched at <http://0.0.0.0:3000>, which could be accessed using any browser.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	39 of 66	
Reference:	D4.1	Dissemination:	PU	
	Version:	1.0	Status:	Final

3.4.3 Licensing information

Intelligence Layer API

The ICOS Intelligence Layer API is governed by specific licensing terms and conditions outlined below. The Software is accessible via the Gitlab repository covered in the next section. This API is the proprietary intellectual property of CeADAR Ireland (University College Dublin). All rights, including but not limited to the source code, design, documentation, and related materials, are exclusively held by CeADAR Ireland (University College Dublin).

Any utilisation of said API beyond the purview of the Intelligence Coordination Operating System (ICOS) or after the conclusion of the 36th month of the project necessitates the establishment of a formal agreement with University College Dublin.

Licensees are granted a non-exclusive, worldwide, royalty-free license to utilise, modify, distribute, reproduce, and sublicense the Software, subject to certain conditions. These conditions include retaining the original copyright notice and this licensing information in any copies or substantial portions of the Software. Usage for unlawful, unethical, or harmful purposes is prohibited, and the Software is provided on an "as is" basis without warranties.

Licensees express their understanding and acceptance of the terms delineated in this licensing information by engaging with, accessing, or contributing to the Software. CeADAR Ireland (University College Dublin) retains the right to modify or update these terms at its discretion.

For inquiries concerning licensing, permissions, or related matters, please contact:

- ▶ **Dr Ricardo Simon Carbajo:** Director of Innovation and Development's Group in CeADAR Ireland (UCD): E-mail: ricardo.simoncarbajo@ucd.ie
- ▶ **Dr Andrés L. Suárez-Cetrulo:** Data Science Architect of Innovation and Development's Group in CeADAR Ireland (UCD): E-mail: andres.suarez-cetrulo@ucd.ie

AI Analytics module

AI models part of the AI Analytics module work as plugins and thus each of the models developed may have different licensing requirements.

- ▶ **CPU metrics model:** This model is governed by the same licensing terms and conditions as the CeADAR API.
- ▶ **Load balancing models** (DECOFFEE, anomaly detection and load forecasting): Licensing of these models will be governed by Apache 2.0.
- ▶ **Trustworthy AI module** Different parts of the Trustworthy AI module work as plugins and thus each of them may have different licensing requirements.
- ▶ **Explainable AI and monitoring:** This part is governed by the same licensing terms and conditions as the CeADAR API.
- ▶ **Federated Learning and Privacy:** Licensing of this part will be governed by Apache 2.0.

3.4.4 Download

Intelligence Layer API

The source code of the API can be downloaded or pulled from the CeADAR production Gitlab repository: gitlab.com/CeADARIreland/UCD/IDG/PRODUCTION/ICOS-AI-Coordination-API

The repository can be opened on invitation. To request access to this repository, please contact:

- ▶ **Dr Andrés L. Suárez-Cetrulo:** Data Science Architect of Innovation and Development's Group in CeADAR Ireland andres.suarez-cetrulo@ucd.ie
- ▶ **Sebastian Cajas Ordoñez:** AI Scientist at Innovation and Development's Group in CeADAR Ireland (UCD): sebastian.cajasordonez@ucd.ie
- ▶ **Jaydeep Samanta:** Data Scientist at Innovation and Development's Group in CeADAR Ireland (UCD): jaydeep.samanta@ucd.ie

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	40 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

3.5 Limitations and future work

Table 10 below illustrates the functionalities of the Intelligence Layer for ICOS Alfa and Beta releases according to deliverable D2.2 [2]. For the ICOS Alfa release, the scope of this piece is basic model management, training and API. This has been achieved in this section 1) using a model registry as part of the API, 2) allowing model training as part of the API, pushing these models then to the registry, and 3) developing the API covered in Section 3.1.

Table 10: Intelligence Layer Functionalities Layer in the ICOS Alfa and Beta releases [2]

Layer	Alfa Release	Beta Release
	M15	M22
Intelligence	Basic Models Management (SUC_AI_1) Basic models training (SUC_AI_2, SUC_AI_3) Intelligence Coordination APIs	Automated models training deployment and execution (SUC_AI_2, SUC_AI_5) Nodes Classification Models (SUC_AI_7) Anomalies Detection Models (SUC_AI_8) Policies violation prediction Models (SUC_AI_9)

While the development of this first API sets a ground base to work towards the ICOS Beta release, work on the models expected for the ICOS Alfa will need to start before M15 to ensure proper data collection and taking requirements for which model outputs and formats will be expected by the Meta-kernel and Security layers, and to align on what data will be available and how often. For instance, for real-time data collection, Zenoh still needs to be explored as a technology.

From the Intelligence Layer's perspective, in the ICOS Alfa version, the API is expected to run in an ICOS controller, as models after training need to be moved back to the model registry. Moving models from external devices to the ICOS Controller's registry could imply a breach in the General Data Protection Regulation (GDPR) rules and AI trustworthiness that will be later addressed in the ICOS Beta release using the strategies to ensure trustworthiness covered in Section 3.3.

Finally, the current deliverable is focused on the Meta-kernel and Security layers, as this has occurred before data collection is planned for use cases. Different pieces of work will be ongoing from M13 onwards to provide the users layer with a mechanism to run models using the environment provided as part of the Intelligence Layer Coordination API.

4 Security Layer Module

The Security Layer is responsible for guaranteeing the security of ICOS users, resources, and applications at all times.

The layer is coordinated by the Security Layer Coordination module that provides a unified interface for interacting with Security Layer modules and communicating with other ICOS components in the Meta-kernel Layer and Intelligence Layer. The exception is the Identity and Access Management module which is directly accessible by other ICOS modules (as presented in D2.2 [2]). The Coordination module also manages data flows inside the layer and provides additional logic needed for seamless interactions between the Security Layer's modules.

This section is structured as follows: we present work performed to develop i) the Security Layer Coordination API in subsection 4.1, ii) the Identity and Access Management module in section 4.2, and iii) the Security scan in subsection 4.3. Finally, we document licensing and instructions to download and set up the software in subsection 4.4, and limitations and future work in subsection 4.5.

4.1 Security Layer Coordination Module

This section provides the functional and technical description of the Security Layer Coordination Module and its API.

4.1.1 Functional description

The Security Layer Coordination module API acts as a reverse proxy service in case of inbound traffic and as a proxy in case of outbound communication. Through this, it provides additional scheduling and security functionalities (time scheduling and CRON jobs).

When ICOS applications in the Intelligence layer and Meta-kernel layer make requests to the Security Layer modules, these requests are forwarded to the modules through the Coordination module API. This helps increase the security of the modules.

When Security Layer modules (Security Vulnerability Mitigation, Compliance Enforcement, Security Scan and Audit) want to communicate with each other or with other ICOS components in Intelligence Layer or Meta-kernel layer, they do it through the Security Layer Coordination module API.

By acting as the proxy for the Security Layer modules, the Security Layer Coordination module contributes to meet the following ICOS functional requirements from the D2.1 [1] (see Table 11 below):

Table 11: Requirements starting to be met with the IT-1 of the Security Layer Coordination module [1]

Requirement ID	Requirement Name	Description
SST_FR_06	Secure infrastructure and code	ICOS SHOULD assess security of infrastructure (e.g., running Critical Security Controls - CIS benchmarks) and system and application code (e.g. running vulnerability scanning of docker images)
SST_FR_11	Compliance detection	ICOS MUST provide a mechanism for the detection of compliance problems regarding controls of specific standards and/or specific policies and rules.

Requirement ID	Requirement Name	Description
SST_FR_12	Compliance enforcement	ICOS MUST provide a system to trigger infrastructure changes to ensure standard and/or policy compliance

4.1.1.1 Fitting into the ICOS architecture

Security Layer Coordination module is the central component of the Security Layer. Its API communicates with the Meta-Kernel Layer and the Intelligence Layer Coordination module API to forward requests directed at the following Security Layer modules: Compliance Enforcement, Security Vulnerability Mitigation, Security Scan, and Audit. See Figure 20 below.

The Security Layer Coordination module is located in the ICOS Controller.

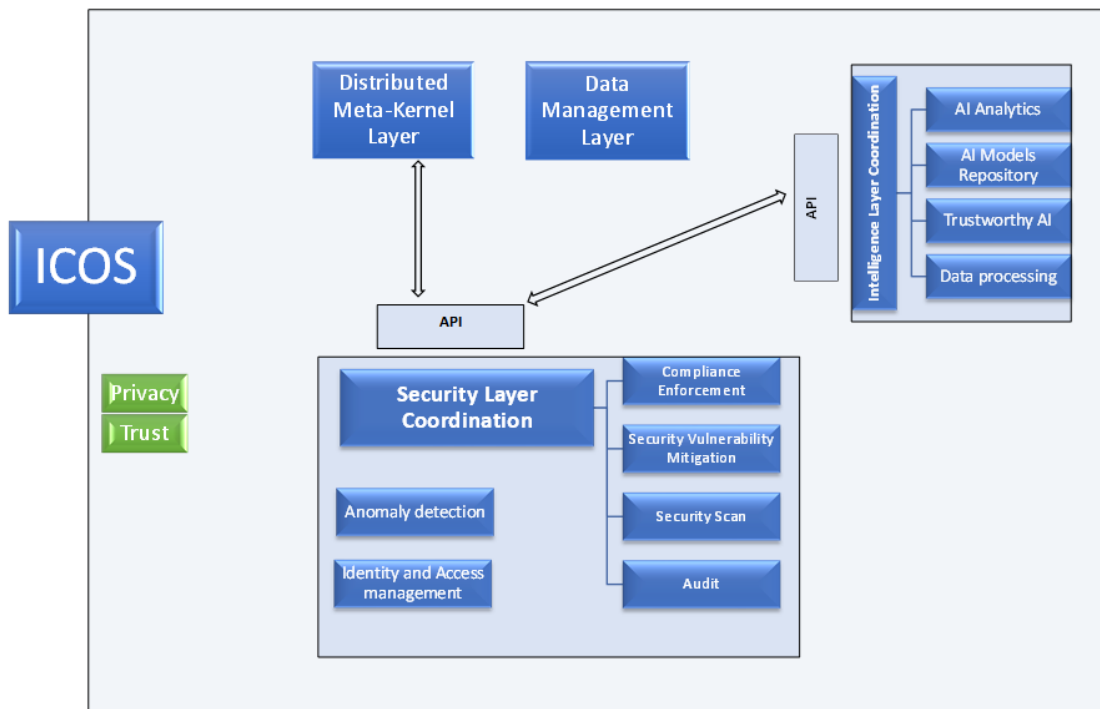


Figure 20: Security Layer Coordination module in ICOS

4.1.2 Technical description

Security Layer Coordination module consists of multiple backend services joined together in a single Docker container. The current iteration implements two APIs, providing proxy and reverse proxy functionalities and additional scheduling services such as time scheduling and CRON jobs. These are implemented using Python and libraries such as Flask and Redis.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	43 of 66
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status:
			Final

4.1.2.1 Prototype architecture

The Figure 21 presents the prototype architecture and data workflow of the Security Layer Coordination module.

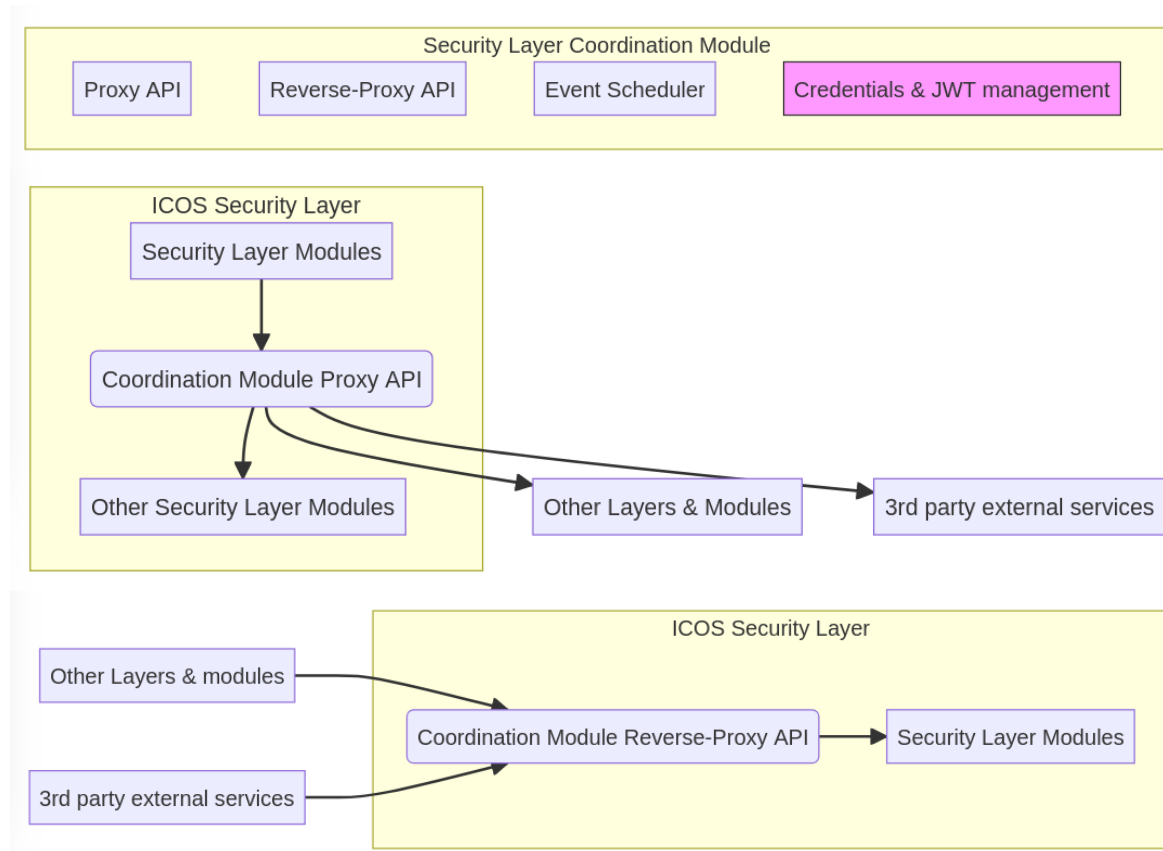


Figure 21: Prototype architecture and data workflow of the Security Layer Coordination module

4.1.2.2 Description of components

- ▶ **Proxy API (IT-1):** Works as a gateway for the outbound communication, when Security Layer modules request information from other components that may or may not be part of the Security Layer or ICOS ecosystem in general.
- ▶ **Reverse-Proxy API (T-1):** Serves as a gateway for the Security Layer's incoming traffic/request.
- ▶ **Event scheduler (IT-1):** Implements scheduling capabilities for executing delayed or periodic tasks. When events are scheduled through proxy or reverse-proxy APIs, this component then calls relevant Security Layer modules based on the provided arguments.
- ▶ **Credentials & JWT management (ICOS Beta release/IT-2):** An additional component that will be implemented at a later stage and which is in charge of handling API/service credentials and issuing JWT tokens for communication with modules inside the Security Layer with the aim to achieve Zero-trust architecture.

4.1.2.3 Technical specifications

The definition of the API can be found in Table 12 below.

Table 12: Security Layer Coordination API definition

Partner	XLAB
Product	Security Layer Coordination module
ICOS layer	Security Layer module
Version	IT-1 (ICOS Alfa version)
Release Date	September 30th, 2023
LICENSE	Open source under the Apache 2.0 license
API-Endpoint	http://host:8080/api/in/v3 http://host:8080/api/out/v3
Environment	Docker-container
Dependencies/ libraries	Flask, Redis, Redis Queue
Detailed description:	
<p>The Intelligence Layer Coordination module facilitates secure communication between the Security Layer modules and other ICOS components in the Meta-Kernel and Intelligence Layer requesting services and as well as between Security Layer modules.</p> <p>It also manages data flows inside the layer and provides additional logic needed for seamless interactions between the Security Layer's modules, offering functionalities such as task scheduling and CRON jobs.</p>	

4.2 Identity and Access Management

The Identity and Access Management (IAM) module is the ICOS architectural component that is responsible for ensuring that the right people have access to the right resources in the system. In the ICOS context, the *people* are the users that need to interact with the ICOS metaOS, so they are the Application Integrators and the Infrastructure Providers (as identified in the D2.1 [1]). The *resources* that need to be protected are the ICOS services (components and modules) that expose the system functionalities (e.g. Job Manager, Monitoring, Intelligence Layer).

4.2.1 Functional description

The IAM component provides three main functionalities to the ICOS metaOS:

- ▶ the management of the Application Integrators and Infrastructure Providers identities and their permissions,
- ▶ the authentication of the users and,
- ▶ the authorization of the requests to ICOS services within the system.

These functionalities implement three ICOS functional requirements listed in Table 13.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	45 of 66
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status:
			Final

Table 13: Requirements starting to be met with the IT-1 of the Identity and Access Management module [1]

Requirement ID	Requirement Name	Description
SST_FR_07	SecureAPI	ICOS MUST provide API supporting AuthT/AuthZ and Audit capabilities
SST_FR_08	SecureLIB	ICOS SHOULD provide libraries supporting Authentication /Authorization to 3rd parties
OP_FR_09	Granularity of the access	ICOS SHOULD provide the possibility to define different roles in accessing to ICOS

4.2.1.1 Fitting into the ICOS architecture

The IAM is part of the Security Layer and runs in the ICOS Controller providing its functionalities to all ICOS services in the Controller that are intended to receive user requests. As explained in section (Description of the components - 4.2.2.2), these ICOS services will need to be integrated with the IAM component by calling its APIs to request and validate authentication and authorization tokens.

4.2.2 Technical description

The strongest requirement for the IAM service is the security, given the fact that the module manages the user's identities, credentials, roles, and tokens to identify and authorise operations in the system. For this reason, we decided to base its design and the implementation on standard, widely adopted and open-source solutions. This choice ensures to use highly transparent, verified and trusted software that has fewer probabilities to contain vulnerabilities that could lead to attacks.

After reviewing multiple options, we decided to base the IAM module on the **OpenID Connect** (OIDC) protocol [20]. It is a protocol for managing authentication and authorization flows based on the OAuth 2.0 protocol [21]. It has several benefits like:

- ▶ it is modern (published in 2014), at the state-of-the-art and widely supported by existing systems and applications,
- ▶ it is based on lightweight and modern web technologies like REST, JWT and JSON,
- ▶ it supports multiple authentication methods and authorization flows that can adapt to different use cases and security requirements (e.g. username/password, single-use codes, federated identities, dynamic clients),
- ▶ management of authentication and authorization is centralized, and services never directly access the user's credentials (improving the overall security of the system),
- ▶ it can be used to achieve Single-Sign-On across multiple services in the system,
- ▶ it has several extensions (some still in draft mode) that can be useful in the project (e.g., dynamic client registration, federation).

At the core of the OIDC protocol there is the **OIDC Provider**. The OIDC Provider is a server that authenticates the users and generates **tokens**. Tokens are strings in JWT [22] format that are signed (and optionally encrypted) by the OIDC Provider. Tokens contain multiple information (called "claims") such as: identity of the user (name, email, username), the issuing authority (the OIDC Provider that generated the token), expiration time, audience, user's roles.

In very general terms, services that need to authorise the incoming requests expect a token to be included in the request. They verify that the token is valid, read the data contained in the token and finally decide, based on the data contained in the token, whether to allow or deny the request. If the token is not available in the request, the service will first redirect the user to the OIDC Provider to perform the authentication.

Figure 22 below depicts the authentication and authorization needs of users that interact with ICOS, concretely, the generic “ICOS Service X” using the ICOS Shell. This is the most basic and important requirement and, therefore, will be the one implemented in the first ICOS Alfa release. However, the IAM component will be extended in the next ICOS releases to implement other flows that will cover service-to-service and Controller-to-Controller authentication and authorization needs.

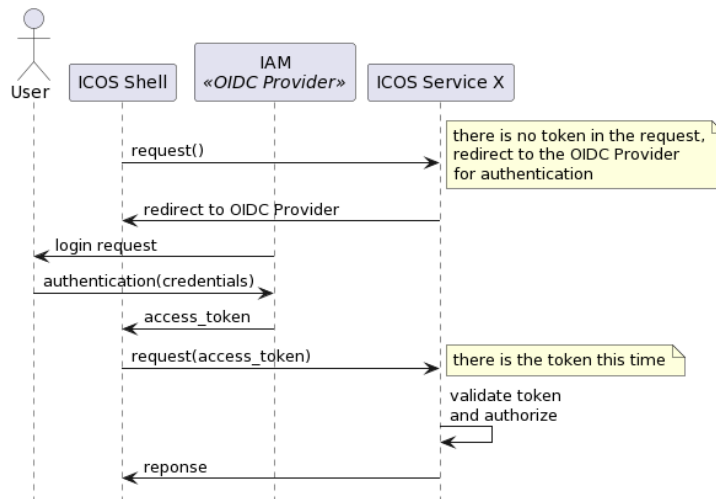


Figure 22: OIDC generic authentication flow

4.2.2.1 Prototype architecture

The IAM component is internally composed of multiple logical components as depicted in Figure 23. They are described in detail in section 4.2.2.2.

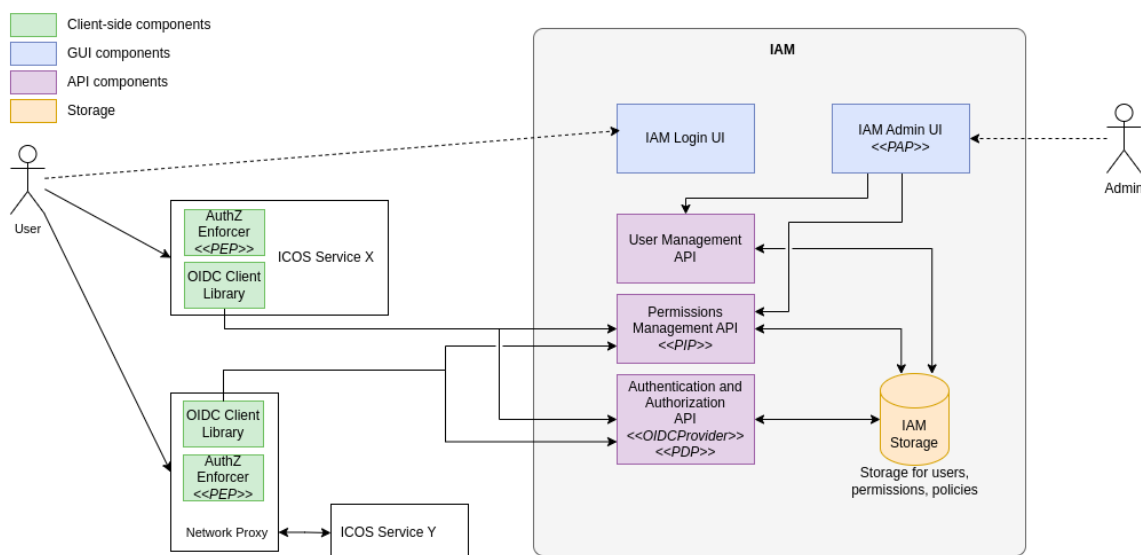


Figure 23: Identity and Access Management internal architecture

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	47 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

4.2.2.2 Description of components

The list of internal logical components of the IAM architectural component are:

- ▶ **IAM Storage:** is a database that stores all the data managed by the IAM component. In particular user identities, permissions, and authorization policies,
- ▶ **Authentication and Authorization API:** exposes an API to issue and validate tokens. It implements the role of OIDC Provider and Policy Decision Point (PDP),
- ▶ **Permissions Management API:** exposes operations to manage (create, view, update, delete) permissions. It implements the role of a Policy Information Point (PIP),
- ▶ **User Management API:** exposes operations to manage (create, view, update, delete) users,
- ▶ **IAM Admin UI:** is a graphical user interface to let the administrators manage the users, the permissions, and the configuration of the IAM instance,
- ▶ **IAM Login UI:** is a graphical user interface dedicated to the authentication of the users. It is the interface to which users are redirected when they need to log in using their credentials.

Beside the components that constitute the IAM module, there are two additional components that are intended to be used client-side in the ICOS services to interact with the IAM APIs:

- ▶ **OIDC Client Library:** is a client-side library that interact with the IAM APIs for initiating authentication flows and validate tokens,
- ▶ **AuthZ Enforcer:** is a client-side library that interacts with the IAM APIs to evaluate user's permissions and enforce authorization policies (e.g., allowing or denying requests). It plays the role of a Policy Enforce Point (PEP).

The **integration of the client-side components** can be implemented in two ways (as depicted in Figure 23):

- ▶ integrate them directly in the ICOS service as a **library/plugin**. It usually requires an extension/adaptation of the component's logic to include the new authentication and authorization logic,
- ▶ use a **network proxy** that capture all the incoming requests, perform the authentication and the authorisation and forward the request to the service. It does not need any change to the ICOS service source code, but it requires a more complex configuration at deployment time.

Choosing between the two options will be a design decision of the different ICOS services that will need to implement the integration. The IAM component will support both of them.

4.2.2.3 Technical specifications

From a technological point of view, following the principle of relying on stable, trusted and secure software, we decided to base the IAM module on the following software projects:

- ▶ **Keycloak [23]:** will be used to implement the authentication and authorization logic, the APIs, and the GUIs. Keycloak is an open-source Identity and Access Management solution that supports the OIDC and OAuth 2.0 protocols. It is released under the Apache 2.0 licence,
- ▶ **PostgreSQL [24]:** will be used to store IAM data. PostgreSQL is an open-source relational database. It is released under its own licence (PostgreSQL License),
- ▶ **Envoy [25]:** can optionally be used as the network proxy to make the integration of authentication and authorization logic into the ICOS services transparent. Envoy is an open-source network proxy known to be reliable, fast, and easily programmable. It is released under the Apache 2.0 licence.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	48 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

4.3 Security Scan

The Security Scan module actively checks for security issues within deployed ICOS modules and components, detects issues, and recommends mitigations and/or recovery processes (with the help from the Security Vulnerability Mitigation module).

In IT-1, the Security Scan module integrates the Wazuh security platform to provide basic functionalities and in later iterations (IT-2) it will also integrate other tools to provide full functionalities, as described in D2.2 [2].

4.3.1 Functional description

Wazuh [26] is an open-source security monitoring tool for threat detection, integrity monitoring, incident response, and basic compliance monitoring. It can be deployed on-premises or in hybrid and cloud environments.

In general, Wazuh provides the following functionalities:

- ▶ Malware and intrusion detection,
- ▶ Log data analysis,
- ▶ File integrity monitoring,
- ▶ Vulnerability detection,
- ▶ Configuration assessment.

Wazuh is composed of a Wazuh server and Wazuh agents. The Wazuh agents are located on the ICOS Agents and communicate information about the detected anomalies to the server, located on the ICOS Controller. The agents can run on many different platforms, such as Windows, Linux, Mac OS X, AIX, Solaris, and HP-UX. Wazuh includes several modules that each support their respective detection capability. For each of the modules, specific rules are defined that include internal metrics and thresholds to trigger events or alerts. When an alert is produced based on some detected event(s), additional actions can be triggered to notify a user or another component about it.

The Security Scan module contributes to meeting the following ICOS functional requirements from the D2.1 [1] (see Table 14 below):

Table 14: Requirements starting to be met with the Alfa version of the Security Scan module [1]

Requirement ID	Requirement Name	Description
SST_FR_06	Secure infrastructure and code	ICOS SHOULD assess security of infrastructure (e.g., running Critical Security Controls - CIS benchmarks) and system and application code (e.g. running vulnerability scanning of docker images)

4.3.1.1 Fitting into the ICOS architecture

The Security Scan module collaborates with other modules from the Security Layer through the Security Layer Coordination module. It detects issues and then can recommend mitigation actions, recovery processes and/or actions for regulatory compliance with the help of the Security Vulnerability Mitigation module or Compliance Enforcement module.

In IT-1, Wazuh agents located on the ICOS Agent nodes collect all the required logs and configurations, pre-process them and forward them to the Wazuh server located on the ICOS Controller node. The latter processes the data, checks for the security issues and then sends the alerts, along with recommended

actions, through the Security Layer Coordination module to the Security Vulnerability Mitigation or Compliance Enforcement module.

4.3.2 Technical description

Wazuh is composed of a Wazuh server and multiple Wazuh agents. The agents are deployed on the individual monitored ICOS nodes and communicate information about the detected anomalies to the server. In a cloud environment, the agents are deployed on the virtual machines inside the monitored cloud infrastructure, independent of the cloud provider. Wazuh server should be installed on a dedicated (virtual) machine, ideally in the same network segment as the agents or otherwise made available by the network routing rules.

We are currently planning to use the latest version of the tool (v4.5), but this might change at a later date due to currently unforeseen dependencies and compatibility problems.

4.3.2.1 Prototype architecture

The basic architecture of Wazuh is depicted in Figure 24. Looking at it from a high-level perspective, it consists of Wazuh Agents and Wazuh Server. The Wazuh agent (installed on endpoints) with different interfaces (modules) is able to detect different metrics on the host. The Wazuh server consists of worker nodes (Wazuh cluster), a Kibana [27] Server that provides a web user interface for overview of all logs and relevant events, and an ElasticSearch [28] database server that stores the logs and detected events coming from the agents.

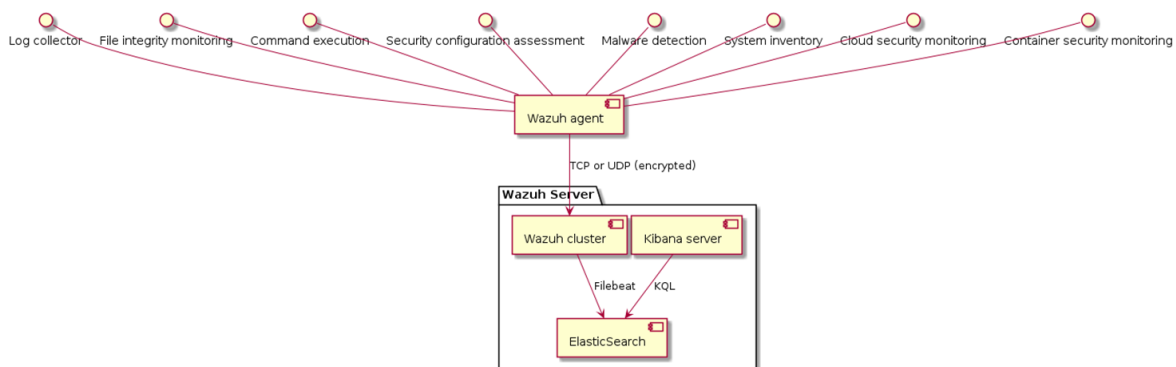


Figure 24: High-level Wazuh's architecture

4.3.2.2 Description of components

Wazuh agents communicate with the Wazuh server using Rsyslog. Wazuh server pushes the alerts and recommended actions to the Security Layer Coordination module.

4.3.2.3 Technical specifications

The Wazuh server includes the Wazuh manager component along with the ELK (ElasticSearch, Logstash, Kibana) stack for gathering, storing, and displaying data. Custom integrations are possible to send alerts from Wazuh to any external component. The Wazuh agents communicate with the server using Rsyslog.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	50 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

4.4 Delivery and usage

The following subsections present package information on the Security Layer Coordination module, Identity and Access Management module and Security Scan module, with instructions how to download and set them up, as well as relevant information regarding licensing.

4.4.1 Package information

Security Layer Coordination module

Security layer coordination module is packaged as a Docker container and can be installed and used as such.

Identity and Access Management module

The Identity and Access Management module is packaged and distributed using Docker containers. In addition, a Helm chart is provided for the deployment and the configuration of all the components of the IAM module in a simplified and reproducible way.

Security Scan module

The Security scan module consists of one to three Docker containers that work together to provide needed functionalities. The exact installation depends on the specific use-case.

4.4.2 Installation instructions

Security Layer Coordination module

Pre-build module will be distributed via container registry. Users should follow the following, general commands to install and run:

```
$ docker image pull <ICOS_REGISTRY>/security-layer-coordination-module
$ docker run security-layer-coordination-module
```

Identity and Access Management module

Pre-build module will be distributed via container registry. For the two main components that constitute the IAM module, Keycloak and PostgreSQL, the official Docker images will be used. They can be installed and executed using the following commands:

```
$ docker image pull postgres
$ docker image pull quay.io/keycloak/keycloak
$ docker run postgres
$ docker run quay.io/keycloak/keycloak
```

The IAM component can be installed as a whole using the Helm chart distributed through the project's repository. It will install its internal components and dependencies with a default working configuration. It can be deployed using the command:

```
$ helm install icos/security-iam
```

For customization options, refer to <https://production.eng.it/gitlab/icos/security/iam>.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	51 of 66				
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final

Security Scan module

Wazuh can be installed and ran in multiple ways; one of which is also containerized Docker version. Required images are available on Wazuh's official Docker Hub and can be installed and ran using the following commands:

```
$ docker image pull wazuh/wazuh-manager
$ docker image pull wazuh/wazuh-indexer
$ docker image pull wazuh/wazuh-dashboard

$ docker run wazuh-manager
$ docker run wazuh-indexer
$ docker run wazuh-dashboard
```

Users can install and run all of the available images or just some of them - depending on their needs/requirements.

4.4.3 Licensing information

Security Layer Coordination module

The Security Layer Coordination module is an open-source software under the Apache 2.0 licence.

Identity and Access Management module

The IAM module is open-source software released under the Apache 2.0 licence. All the dependencies are open-source software released under compatible licences.

Security Scan module

The Wazuh is an open source, licensed with a modified GNU GPLv2 licence.

4.4.4 Download

Security Layer Coordination module

The source code of the Security Layer Coordination API can be downloaded or pulled from the project's development Gitlab repository: <https://production.eng.it/gitlab/icos/security/coordination-module.it>

The repository can be opened on invitation. To request access to this repository, please contact: **Mr. Gabriele Giammatteo** (gabriele.giammatteo@eng.it).

Identity and Access Management module

The source code of the Identity and Access Management module can be downloaded or pulled from the project's development Gitlab repository:

<https://production.eng.it/gitlab/icos/security/iam>

The repository can be opened on invitation. To request access to this repository, please contact: **Mr. Gabriele Giammatteo** (gabriele.giammatteo@eng.it).

Security Scan module

The source code of Wazuh can be downloaded or pulled from the official Wazuh GitHub repository: <https://github.com/wazuh/wazuh>

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	52 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

4.5 Limitations and future work

Table 15 below illustrates the functionalities of the Security Layer for ICOS Alfa and Beta releases. A change has occurred from the same list presented in deliverable D2.2 [2], concretely Secure Connections and Trust between nodes (SUC_SC_9, SUC_SC_10) was implemented as part of the Data management IT-1 using Zenoh communication protocol and also the first version of Security Assessment (SUC_SC_8) was implemented using the Security Scan module.

For ICOS Beta release, we also plan to implement first versions of the Audit, Compliance Enforcement and Anomaly detection modules as well as of Trust and Privacy functionalities.

Table 15: Modified Security Layer functionalities in the ICOS Alfa and Beta releases

Layer	ICOS Alfa Release	ICOS Beta Release
	M15	M22
Security	User AuthN/AuthZ, Users Management (SUC_SC_1, SUC_SC_2) Secure Connections and Trust between Nodes (SUC_SC_9, SUC_SC_10) Security Assessment (SUC_SC_8) Security Coordination APIs	Service-to-Service and Controller-to-Controller AuthN/AuthZ (SUC_SC_1, SUC_SC_2) Compliance Analysis (SUC_SC_5) Audit Analysis (SUC_SC_4) Security Assessment (SUC_SC_8)

Security Layer Coordination module

In the current iteration (IT-1), Security Layer Coordination Module provides proxy and reverse proxy functionalities for other Security Layer modules as well as scheduling service for executing delayed or periodic tasks.

In the next releases (ICOS Beta release, IT-2), we plan to implement fully functional connection with the other ICOS components (Meta-kernel, Intelligence Layer) and Security Layer modules as well as additional credential checking and JWT management service that will be used to establish zero trust architecture inside the Security Layer as a whole.

Identity and Access Management module

In the ICOS Security Layer module IT-1 release, the IAM component is implemented and configured to support only the user authentication and authorization through the integration with the ICOS Shell component, since all user's calls should pass through the ICOS Shell. In the next releases (ICOS Beta, IT-2), additional authentication and authorization flows will be considered to be covered for service-to-service and controller-to-controller calls.

We are also investigating the usage of solutions at platform level that can make the introduction of network proxies transparent (e.g. Istio [29], Cilium[30]).

Security Scan module

In the current iteration (IT-1), Security Scan only incorporates the Wazuh tool. In the next iterations (ICOS Beta release, IT-2), we plan to implement other tools, like Vulnerability Assessment Tools (VAT) and Advanced Mitigation strategy – PMEM [31], to provide additional scanning capabilities.

5 Conclusions

The document presented the functional and technical aspects of the first version (IT-1) of Data Management, Intelligence Layer and Security Layer software.

The Data Management layer can be seen as a (hyper)distributed data store that allows ICOS components to access data regardless of its location in the continuum. In addition, it aims at optimising data access by executing part of the code within the data store, thus minimising data movements within and across nodes. In IT-1, it has been mainly integrated with the Intelligence Layer and with the Distributed & Parallel Execution module which is part of the Runtime Manager component of the Meta-kernel layer. A first integration with the Job Manager (also a part of the Runtime Manager component) to facilitate its deployment has also been performed. The two main software solutions used in the Data Management layer IT-1 are Zenoh and dataClay. The layer also defined REST APIs for communication with other ICOS components.

The first iteration of Data management layer will be used for the ICOS Alfa release (task T5.3 and deliverable D5.1). For ICOS Beta release (D5.2) and future IT-2, the Data Management will provide support for federated learning and additional functionalities to support not only other ICOS components but also applications running on top of ICOS.

The Intelligence Layer in IT-1 includes the Intelligence Layer Coordination module, AI analytics and Trustworthy AI modules. The Intelligence Layer Coordination module provides a unified interface (API) for interacting with the Intelligence Layer functionalities and communicating with other ICOS components. This module communicates with the AI analytics module which is responsible for building and deploying machine learning models (4 models are developed in IT-1). The Intelligence Layer Coordination module also communicates with the Trustworthy AI module that, in IT-1, provides two strategies to ensure the trustworthiness and privacy of models: 1) Federated learning and privacy and 2) Explainable AI and monitoring. Intelligence Layer Coordination Module and AI analytics module will be used for the ICOS Alfa release (task T5.3 and deliverable D5.1), with the first training of the models on real data from the use case. For the ICOS Beta release (D5.2), strategies for ensuring trustworthiness through the Trustworthy AI module as well as additional functionalities for the interaction of users with models through the Intelligence Layer Coordination Module will also be implemented.

The Security Layer in IT-1 includes the Security Layer Coordination module, Identity and Access Management module and the Security Scan module. Security Layer Coordination module provides a unified interface for interacting with Security Layer and with other ICOS components as well as additional logic and security functionalities needed for seamless interactions between Security Layer's modules. The Identity and Access Management module manages user's identities, credentials, roles, and tokens to identify and authorise operations in the ICOS system. The Security Scan module actively checks for security issues within deployed ICOS modules and components, detects issues, and recommends mitigations and/or recovery processes. All three modules will be used for the ICOS Alfa release (task T5.3 and deliverable D5.1). Additional functionalities are considered to be implemented in all three modules for the ICOS Beta release (D5.2): a) for the Security Layer Coordination module credential checking and JWT management service, b) for the Identity and Access Management additional authentication and authorization flows for service-to-service and controller-to-controller calls and c) for the Security Scan additional scanning capabilities.

The Table 16 below presents which parts of Data Management, Intelligence Layer and Security Layer IT-1 can be considered as ICOS assets.

ICOS components described in this document will continue to be developed in the respective work package and the updated versions will be the basis for the second integrated ICOS Beta release (M22) and second iteration of Data Management, Intelligence and Security Layers (IT-2) as presented in deliverable D4.2 (M30).

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)			Page:	54 of 66
Reference:	D4.1	Dissemination:	PU	Version:	1.0
				Status:	Final

Table 16: Data Management, Intelligence and Security Layers IT-1 parts as ICOS assets

ICOS COMPONENT	ICOS ASSETS
Data Management module	<ol style="list-style-type: none"> 1. <u>Metadata Service</u>, which is a distributed service holding all the information about the data stored and its location, 2. <u>Backends</u> that store the data in a distributed manner and are also responsible for the execution of the methods related to the data they hold. 3. <u>A REST API</u> via the Zenoh-rest plugin, which is automatically started on port 8000 and ready to answer HTTP request. 4. <u>Storage manager plugin</u>, which provides the ability to store values associated with a set of keys, allowing other nodes to pub, sub, and query the most recent values associated with these keys. It relies on dynamically loaded backends, leveraging third party technologies i.e. databases.
Intelligence Layer module	<ol style="list-style-type: none"> 1. <u>Intelligence Layer Coordination module API</u>: This supports async calls to model building and predictions, has its own model registry, and allows the reuse of previously trained models in an ICOS controller. 2. <u>Intelligent energy-aware task offloading model</u>: This model is based on Deep Reinforcement Learning (DRL) running on each ICOS agent. ICOS controller(s) are used to provide global data to each ICOS agent. The objective of the model is to suggest decisions of task offloading, allowing local executions of the tasks, horizontal offloading to another agent or vertical offloading to the Cloud. Decisions guarantee energy efficiency and low latency of the task completion. 3. <u>Load time series forecasting model</u>: This model is used to forecast the upcoming load of each ICOS agent. Three variants of this model are also provided, including different moments of the future predictions. This model adds incremental learning capabilities to the API. 4. <u>Forecasting upcoming load anomalies (related to the first one) model</u>: This model is used to classify the load instances as ‘normal’ or ‘anomaly’ and can be used by each ICOS agent to estimate its type of traffic. This model adds incremental learning capabilities to the API. 5. <u>CPU utilisation forecasting to improve workload distribution model</u>: This model adds continuous and incremental learning capabilities to the API to adapt to the dynamic changes experienced by different metrics in ICOS controllers and agents.
Security Layer module	<ol style="list-style-type: none"> 1. <u>Security Layer Coordination module API</u>: provides a unified interface for interacting with Security Layer modules and communicating with other ICOS components. It also manages data flows inside the Security Layer and provides additional logic and security functionalities needed for seamless interactions between Security Layer’s modules. 2. <u>Identity and Access Management module</u>: manages user’s identities, credentials, roles, and tokens to identify and authorise operations in the ICOS system. 3. <u>Security Scan module</u>: actively checks for security issues within deployed ICOS modules and components, detects issues, and recommends mitigations and/or recovery processes.

6. References

- [1] ICOS. *D2.1 - ICOS ecosystem: Technologies, requirements, and state of the art (IT-1)*. Francesco D’Andria. 2023
- [2] ICOS. *D2.2 - ICOS architectural design (IT-1)*. Gabriele Giammatteo. 2023
- [3] Giannopoulos, A., Spantideas, S., Kapsalis, N., Karkazis, P., & Trakadas, P. (2021). Deep reinforcement learning for energy-efficient multi-channel transmissions in 5G cognitive hetnets: Centralized, decentralized and transfer learning based solutions. *IEEE Access*, 9, 129358-129374.
- [4] Sevgican, S., Turan, M., Gökarslan, K., Yilmaz, H. B., & Tugcu, T. (2020). Intelligent network data analytics function in 5G cellular networks using machine learning. *Journal of Communications and Networks*, 22(3), 269-280.
- [5] Orange Pi 5. <http://www.orangepi.org/html/hardWare/computerAndMicrocontrollers/details/Orange-Pi-5.html>, retrieved 2023-09-12.
- [6] João Gama, Raquel Sebastião & Pedro Pereira Rodrigues (2012). *On evaluating stream learning algorithms*, [On evaluating stream learning algorithms | SpringerLink](#), retrieved 2023-09-12.
- [7] Goodwin, P. and Lawton, R. (1999). On the asymmetry of the symmetric MAPE. *International journal of forecasting*, 15(4), 405-408.
- [8] Zenoh, <https://zenoh.io/>, retrieved 2023-09-20.
- [9] dataClay, <https://dataclay.bsc.es/>, retrieved 2023-09-20.
- [10] MLFlow, <https://mlflow.org/>, retrieved 2023-09-20.
- [11] Grafana, <https://grafana.com/>, retrieved 2023-09-20.
- [12] Prometheus, <https://prometheus.io/>, retrieved 2023-09-20.
- [13] BentoML, <https://docs.bentoml.org/en/latest/index.html>, retrieved 2023-09-20.
- [14] Pandas, <https://pandas.pydata.org/>, retrieved 2023-09-20.
- [15] Statsmodels, <https://www.statsmodels.org/stable/index.html>, retrieved 2023-09-20.
- [16] Scikit-learn, <https://scikit-learn.org/stable/index.html>, retrieved 2023-09-20.
- [17] PyTorch, <https://pytorch.org/>, retrieved 2023-09-20.
- [18] TensorFlow, <https://www.tensorflow.org/>, retrieved 2023-09-20.
- [19] The AI Act, [The Act | The Artificial Intelligence Act](#), retrieved 2023-09-12.
- [20] OpenID Connect protocol, <https://openid.net/>, retrieved 2023-09-12.
- [21] OAuth 2.0 protocol, <https://oauth.net/2/>, retrieved 2023-09-12.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	56 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

- [22] JSON Web Token, <https://jwt.io/>, retrieved 2023-09-12.
- [23] Keycloak, <https://www.keycloak.org/>, retrieved 2023-09-12.
- [24] PostgreSQL, <https://www.postgresql.org/>, retrieved 2023-09-12.
- [25] Envoy, <https://www.envoyproxy.io/>, retrieved 2023-09-12.
- [26] Wazuh, <https://wazuh.com/>, retrieved 2023-09-12.
- [27] Kibana, <https://www.elastic.co/kibana/>, retrieved 2023-09-25.
- [28] Elastic Search, <https://www.elastic.co/>, retrieved 2023-09-25.
- [29] Istio, <https://istio.io/>, retrieved 2023-09-12.
- [30] Cilium, <https://cilium.io/>, retrieved 2023-09-12.
- [31] Advanced Mitigation strategy – PMEM, <https://github.com/H2020-FISHY/pmem>, retrieved 2023-09-25.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	57 of 66				
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final

Annexes

This section contains annexes to this deliverable.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)			Page:	58 of 66		
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final

Annex I: Testing the Intelligence Layer Coordination API

Testing over the train_cpu_utilisation function:

To launch the *Swagger UI*, the URL http://0.0.0.0:3000/#/core/analytics_train_cpu_utilisation can be typed in the browser on the machine where the component has been installed and that launches the train_cpu_utilisation API on the *Swagger UI* as illustrated in Figure 25 below.

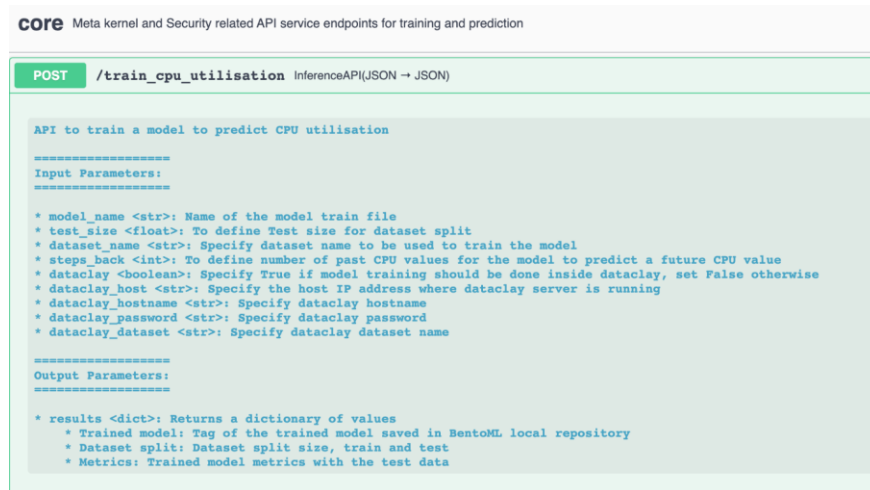


Figure 25: API definition for train_cpu_utilisation in Swagger

The API input and output parameters are described inside a text window below the POST API method. The user can call the API with the default parameters displayed in the 'Request body' or modify the input parameters by simply clicking the "Try it out" icon on the *Swagger UI*, as shown in Figure 26. Once the input fields have been modified, the user can click on 'Execute' to send the POST request, which will internally call the train model function with the arguments provided.

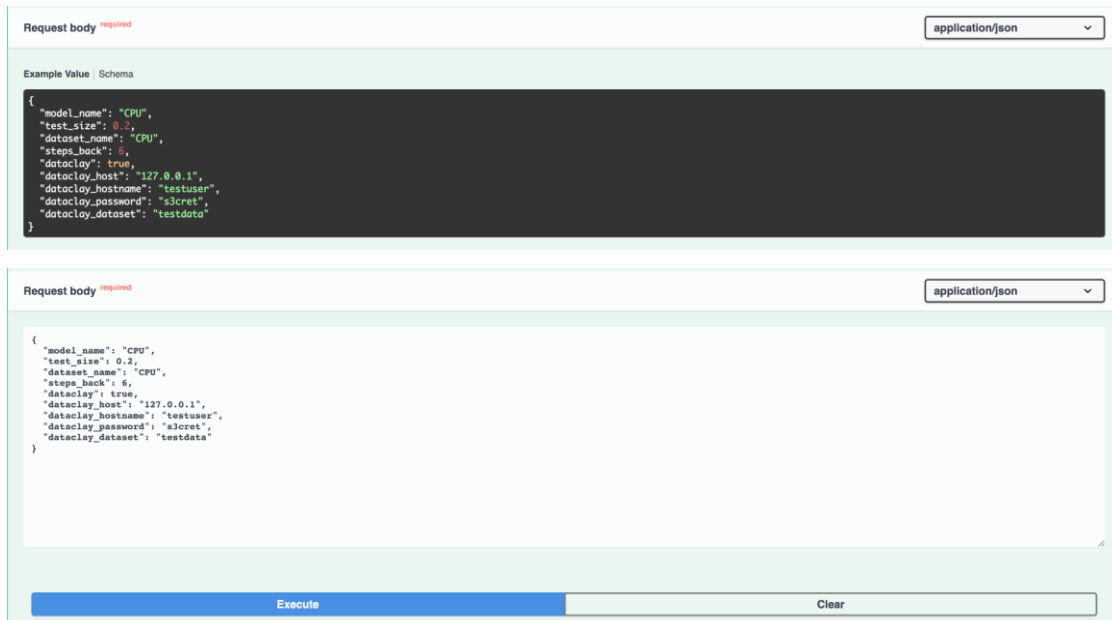


Figure 26: POST request with default (above) and user-defined (below) parameters.

This API will complete and deliver the response, as Figure 27 below shows.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	59 of 66
Reference:	D4.1	Dissemination:	PU
	Version:	1.0	Status: Final

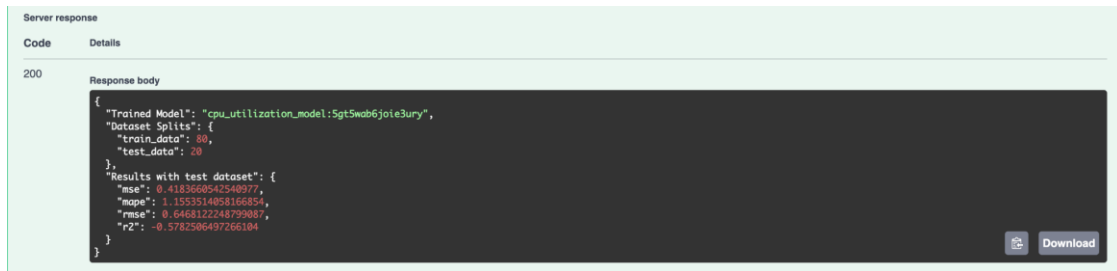


Figure 27: Response received upon successful execution of request

Upon successful completion, the train API will return the trained model tag name; data splits metadata and model metrics. This model tag can be used to identify the trained model, and the same model can be used to predict.

The second approach to test the functions exposed in the API is through the use of curl as illustrated in Figure 28 below.

```

curl -X 'POST' \
'http://0.0.0.0:3000/train_cpu_utilisation' \
-H 'accept: application/json' \
-H 'Content-Type: application/json' \
-d '{
  "model_name": "CPU",
  "test_size": 0.2,
  "dataset_name": "CPU",
  "steps_back": 6,
  "dataclay": true,
  "dataclay_host": "127.0.0.1",
  "dataclay_hostname": "testuser",
  "dataclay_password": "s3cret",
  "dataclay_dataset": "testdata"
}'

```

Figure 28: Example of curl POST request to train a model

Testing the predict_cpu_utilisation function:

The Predict model API can be launched in *Swagger UI* using the URL http://0.0.0.0:3000/#/core/analytics_predict_cpu_utilisation in the browser where the component has been installed. This API request body is a POST method with model tag-name, steps-size and input data as input parameters and predicted value as output parameter, as shown in Figure 29 below.

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	60 of 66
Reference:	D4.1	Dissemination:	PU
		Version:	1.0
		Status:	Final

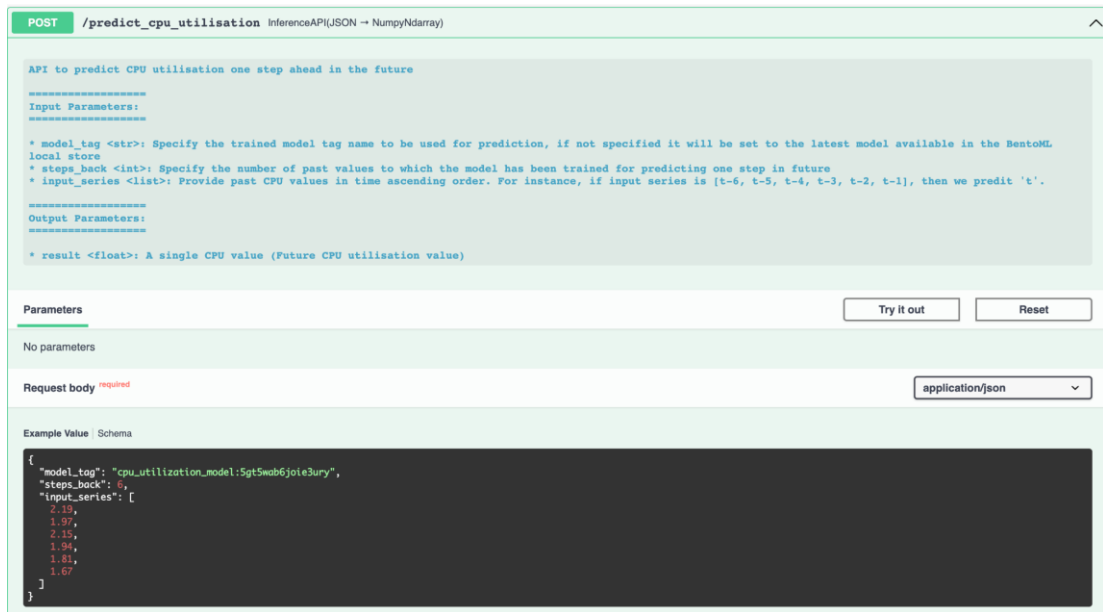


Figure 29: Predict_cpu_utilisation API with input and output parameters description.

Users can either use an already trained model saved inside the device or can first train a custom model using a training function (e.g. **train_cpu_utilisation**) for later inferencing. The model tag provided with the POST request is the one required for inference. Figures Figure 30, Figure 31 and Figure 32 below, as previously shown for model training, cover: i) input parameters required for model inference (with self-explanatory meaning), ii) an example of a curl request for inferencing with a model previously trained and available in the registry, and iii) an example result of a model forecast respectively.



Figure 30: POST request with default (left) and user-defined (right) input parameters

Model tag, as an input in Figure XX1, is given as an output when first training the model. Previously trained models are available in the model registry.


```
curl -X 'POST' \
  'http://0.0.0.0:3000/predict_cpu_utilisation' \
  -H 'accept: application/json' \
  -H 'Content-Type: application/json' \
  -d '{
    "model_tag": "cpu_utilization_model:5gt5wab6joie3ury",
    "steps_back": 6,
    "input_series": [
      2.19,
      1.97,
      2.15,
      1.94,
      1.81,
      1.67
    ]
  }'
```

Figure 31: Example of curl POST request to inference with a model

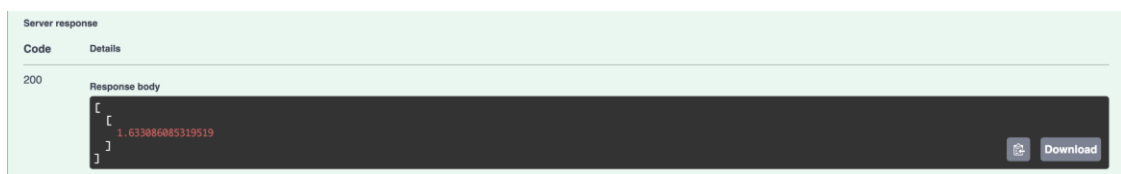


Figure 32: Example model inference response received upon successful execution of request

In the figures above, the user tests the model specified in “model_tag” that receives the last six CPU-utilisation overall percentages (at the minute level; thus, the last six minutes) in a device to predict the utilisation one minute ahead (see Figure 32).

Annex II: Validation of AI Analytics models

Load time series forecasting

The learning hyperparameters of the LSTM models were firstly tuned towards stabilising optimally the model performance. Based on the similarities in the spatiotemporal fluctuations of the load traffic between all the neighbouring IoT areas, the learning rate (a) and the lookback window length (W) hyperparameters were fine-tuned for one single area, and the resulting optimal values were inherited by the rest of the LSTM models. The configuration of the LSTM model used for the training process is summarised in Table 17. Initial simulations proved that 20 epochs are sufficient for low-valued convergence ($MSE10^{-3}$) of the loss function.

Table 17: Architecture of the LSTM network model used for the load timeseries forecasting.

PARAMETER	Value
LSTM regression model	Sequential (TensorFlow Keras 2.3)
Input Neurons	$24 \times 4 = 96$
Output Neurons	1
Number of hidden layers	4
Hidden units per layer	50
Dropout Rate	20%
Optimizer	Adam
Learning rate	0.01
Training set samples	$(6 \times 30 \times 24 \times 4) - (8 \times 24 \times 4) = 16512$
Testing set samples	$1 \times 7 \times 24 \times 4 = 672$
Feature scaling	MinMaxScaler (sklearn library)
Training epochs	20
Batch size	64
Loss function	Mean Squared Error

Observing that different values of the learning rate significantly affect the LSTM performance, different models were derived with varying values ($a=0.1$, $a=0.01$, $a=0.001$). Figure 33 depicts the MSE loss convergence as a function of the training epochs, with the value of $a = 0.01$ showing the optimal (e.g. fastest and minimum) MSE loss. Moreover, the lookback window length, which considerably influence the dimensionality, complexity, and memory requirements of the LSTM models (directly proportional to the input layer size), was varied for three training setups (2 weeks or $W = 1344$, 1 week or $W = 672$, 1 day or $W = 96$ samples). To quantify the optimal value of W , Figure 34 shows the MSE loss curves, revealing the optimal window length for $W = 96$ samples.

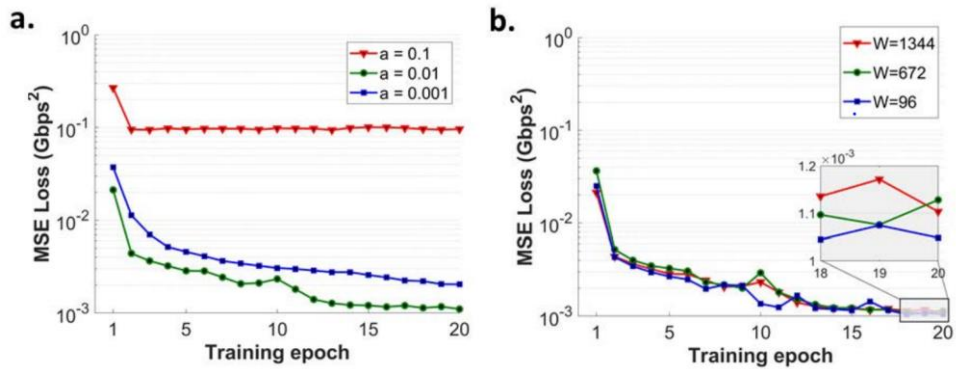


Figure 33: Mean squared error (MSE) loss curve as a function of the training epochs for different values of learning rate a (panel a) and lookback window length W (panel b).

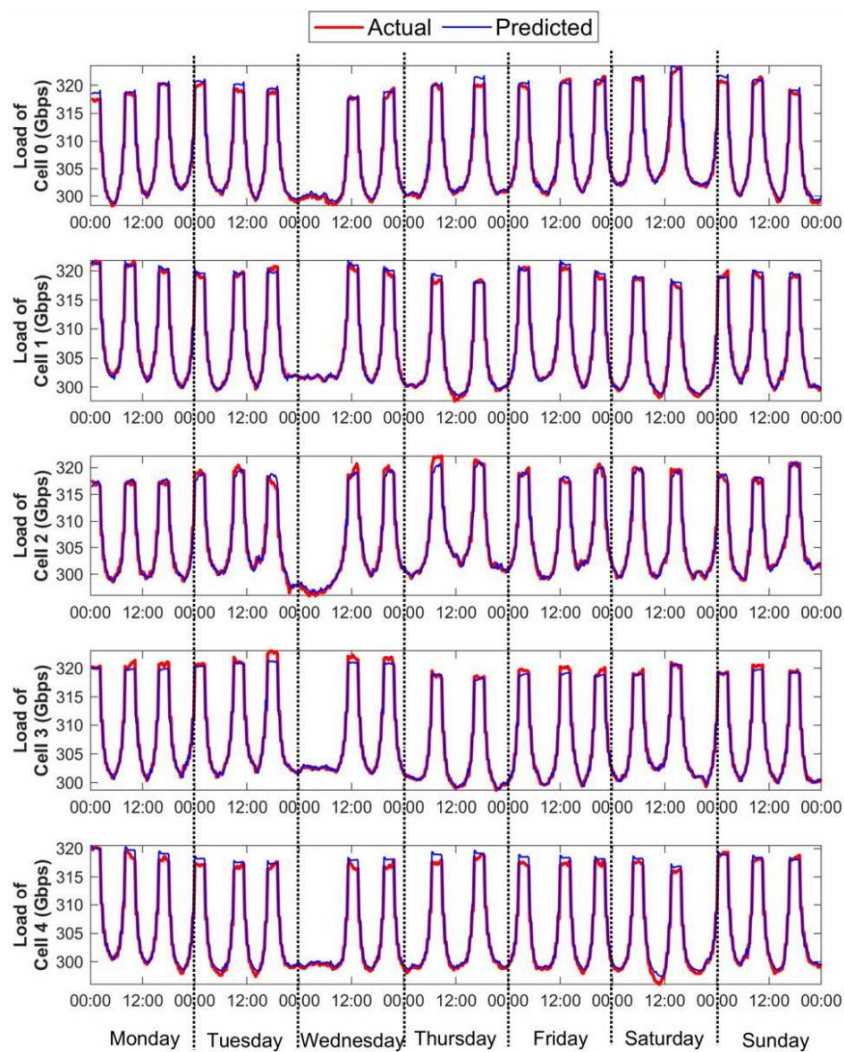


Figure 34: Actual and predicted load (Gbps) for each IoT area/cell-specific LSTM model. Validation data refers to 1-week total load values used for the testing set.

Load anomaly detector

The goal of the supervised anomaly detector in this case was to predict the correct class (anomaly or not), or equivalently, only the probability of the class ‘anomaly’ (the probability of class ‘no anomaly’ is complement). To this end, different classification models were compared for purposes of identifying the best detector. Figure 35 depicts the classification accuracy (%) amongst different models, with the Decision Tree being proved the optimal model for identifying anomalies (99.86% accuracy). We also observed the quasi-identical results using Random Forest, but we finally considered the Decision Tree as the best detector, due to its lower complexity (Random Forest is the Ensemble learning version of the Decision Tree). Note that, the same schemes were contrasted in each IoT area with similar results, whereas Figure 35 shows only the outcomes derived for IoT area No 1, without loss of generality.

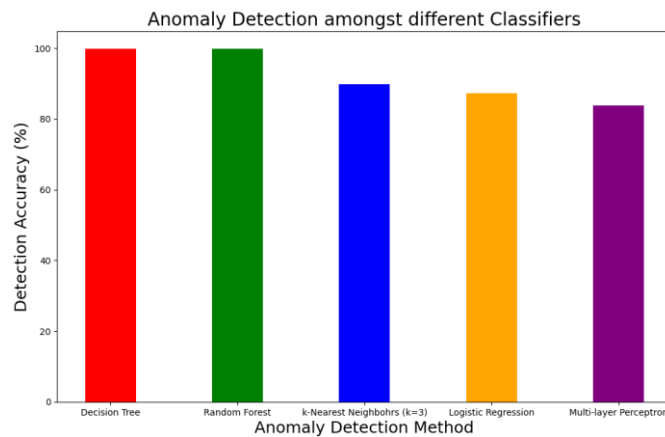


Figure 35: Percentage anomaly detection accuracy across different supervised classification models

Annex III: Intelligence Layer Coordination API Development Guide

A development guide has been circulated among T4.2 partners to integrate models in the API. This guide, also part of the Intelligence Layer Coordination API repository covered in 3.4, can be found in the project document repository: <https://newrepository.atosresearch.eu/index.php/f/1260823>.

The repository can be opened on invitation. To request access to this repository, please contact:

- **Carmen San Román:** Eviden, Networks & Edge Technologies – Research & Innovation
carmen.sanroman@eviden.com

Document name:	D4.1 Data management, Intelligence and Security Layers (IT-1)	Page:	66 of 66				
Reference:	D4.1	Dissemination:	PU	Version:	1.0	Status:	Final