Transform safety-critical Cyber Physical Systems into distributed solutions for end-users and partners

D26 (D2.4)

Reference architectures for distributed safety-critical distributed cyber-physical systems v2

This project has received funding from the ECSEL Joint Undertaking (JU) under grant agreement No 101007260. The JU receives support from the European Union's Horizon 2020 research and innovation programme and Netherlands, Finland, Germany, Poland, Austria, Spain, Belgium, Denmark, Norway.

Document Information

Project	TRANSACT
Grant Agreement No.	101007260
Work Package No.	WP2
Task No.	T2.1
Deliverable No.	D26
Deliverable No. in WP	D2.4
Deliverable Title	Reference architectures for distributed safety-critical distributed cyber-physical systems v2
Nature	Report
Dissemination Level	Public
Document Version	v1.0
Date	19/12/2023
Contact	Jordi Arjona
Organization	ITI
Phone	+34 963877069
E-Mail	jarjona@iti.es

Authors Table

Name	Company	E-Mail
F. Javier Fernández-Bravo Peñuela	ITI	fjfernandez@iti.es
Miguel García Gordillo	ITI	miguelgarcia@iti.es
Jordi Arjona Aroca	ITI	jarjona@iti.es
Roel Hermans	PMS	roel.hermans@philips.com
Robert Hofsink	PMS	robert.hofsink@philips.com
Koen de Laat	PMS	koen.de.laat@philips.com
Krzysztof Oborzynski	PMS	krzysztof.oborzynski@philips.com
Abel Gómez Llana	UOC	agomezlla@uoc.edu
Iván David Alfonso Díaz	UOC	ialfonsod@uoc.edu
Eike Moehlmann	DLR	eike.moehlmann@dlr.de
Sebastian Vander Maelen	DLR	sebastian.vandermaelen@dlr.de
Javier Ferrer	KUM	javier@kumori.systems
Marco Jahn	ECL	marco.jahn@eclipse-foundation.org
Martijn Hendriks	TUE	m.hendriks@tue.nl
Sara Pastor	NUN	sara.pastor@nunsys.com
Anders Holme	NVT	anders.holme@navtor.com
Ralph Weissnegger	CISC	r.weissnegger@cisc.at
Elisabeth Salomon	TUG	elisabeth.salomon@tugraz.at
William Lindskog	DNDE	w.lindskog@eu.denso.com
Juhani Latvakoski	VTT	juhani.latvakoski@vtt.fi
Mika Jaakonaho	FLEET	mika.jaakonaho@fleetonomy.ai
Marek S. Tatara	DAC	marek.tatara@dac.digita
Wolfram Ratzke	AVL	wolfram.ratzke@avl.com

Bjørn Åge Hjøllo	NVT	Bjorn.hjollo@navtor.com

Reviewers Table

Version	Date	Reviewer
v0.3	10/12/2023	Astrid Rakow (DLR)
v0.3	11/12/2023	Jeroen Voeten (TU/e)
v0.3	13/12/2023	Teun Hendriks (TNO)

Change History

Version	Date	Reason for Change	Affected pages
v0.1	02/03/2023	First version of the deliverable providing the document structure and examples for the component and scenarios to be described.	All
v0.2	01/11/2023	Internal review	All
v0.3	22/11/2023	Document ready for review	All
v1.0	19/12/2023	Consolidated document after final review	All

List of participants

No.	Participant organisation name	Short name	Country
1	Philips Medical Systems Nederland BV	PMS	NL
2	PS-Tech BV	PST	NL
3	Fleetonomy.ai Oy	FLEET	FI
4	Teknologian tutkimuskeskus VTT Oy	VTT	FI
5	Nodeon Finland Oy	NOD	FI
6	AVL Software and Functions GmbH	AVL	DE
7	Eclipse Foundation Europe GmbH	ECL	DE
8	Denso Automotive Deutschland GmbH	DNDE	DE
9	Fraunhofer-Gesellschaft - Fraunhofer Institute for Experimental Software Engineering	IESE	DE
10	DAC Spolka Akcyjna	DAC	PL
11	Technische Universitaet Graz	TUG	AT
12	CISC Semiconductor GmbH	CISC	AT
13	NAVTOR AS	NVT	NO
14	Instituto Tecnológico de Informática	ITI	ES
15	NUNSYS SL	NUN	ES
16	Kumori Systems	KUM	ES
17	Fundació per a la Universitat Oberta de Catalunya	UOC	ES
18	Deutsches Zentrum fur Luft- und Raumfahrt EV	DLR	DE

Table of Contents

List of Figures

List of Tables

Glossary

Term	Definition
Allocation (*aka* task allocation)	Task allocation refers to the decision of task placement and scheduling associated with the resource management.
Architecture	The fundamental organization of a system embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution.
Architecture framework	Conventions, principles and practice for the description of architectures established within a specific domain of application and/or community of stakeholders.
Component	One of the parts that make up a system.
Concept	An abstraction; a general idea inferred or derived from specific instances.
Cyber-Physical System	Digital system that semi-automatically interacts with its physical environment as integral part of its functionality. It integrates computation with physical processes, where system properties are determined by both cyber and physical parts.
Device	Physical entity embedded inside, or attached to, another physical entirety in its vicinity, with capabilities to convey digital information from or to that physical entity.
Digital twin	A digital twin is a digital model of an actual physical system, which has a “live” connection (digital thread) with the physical system, so that it represents its actual status, and is used to derive a higher-level representation of the system’s status and performance.
Edge Computing	Edge computing is a new architectural paradigm in which the resources of an edge server are placed at the edge of the Internet, in close proximity to cyber-physical systems, mobile devices, sensors and IoT endpoints.
Framework	A framework is an abstraction in which “engineering bricks” providing generic functionality can be selectively changed by system engineers, thus providing application-specific solutions. It provides a standard way to build and deploy CPS.
Mechanism	An established process by which something takes place or is brought about.
Method	A method consists of systematic steps for performing a task, in other words, it defines the “how” of each task.
Methodology	A collection of related formalisms, techniques, processes, methods, and tools. A methodology is essentially a “recipe” and can be thought of as the application of related processes, methods, and tools to a class of problems that all have something in common.
Middleware	Middleware is computer software that provides services to software applications beyond those available from the operating system. It can be described as “software

	glue". Middleware makes it easier for software developers to implement communication and input/output, so they can focus on the specific purpose of their application.
Mission-critical function	A mission-critical function is a function that is essential to the survival of a business or organization in an application. When a mission-critical function fails or get interrupted, business operations are significantly impacted.
Mixed-criticality application	An application containing computer hardware and software that can execute several functions of different criticality, such as safety-critical and non-safety-critical, or of different SIL. Different criticality functions are engineered to different levels of assurance, with high criticality functions being the costliest to design and verify.
Offloading (*aka* computation or task offloading)	Computation offloading is the transfer of resource intensive computational tasks to a separate processor, such as a hardware accelerator, or an external platform, such as a cluster, grid, or a cloud. Offloading computing to an external platform over a network can provide computing power and overcome hardware limitations of a device, such as limited computational power, storage and energy.
Orchestration	Type of composition where one particular element is used by the composition to oversee and direct the other elements. Note: The element that directs an orchestration is not part of the orchestration itself.
Platform	A collection of interoperable system "engineering bricks" that can be used to set up a system engineering environment in a company. A technology or engineering brick can be a software tool/product, a software component to build a software tool/product, a system engineering methodology, an interface, a standard, or means for establishing interoperability that is needed for the efficient development of safety-critical embedded systems.
Process	A process is a logical sequence of tasks performed to achieve a particular objective. A process defines "what" is to be done, without specifying "how" each task is performed.
Reference Architecture	A Reference Architecture (RA) is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements. It captures the essence of the architecture of a collection of systems. The main purpose of a Reference Architecture is to provide guidance for the development of architectures. One or more reference architectures may be derived from a common reference model, to address different purposes/usages to which the Reference Model may be targeted. (OASIS Standard, 2012)
Reference Model	A reference model is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or

	other concrete details. A reference model may be used as a basis for education and explaining standards to non-specialists.
Safety-critical CPS	A safety-critical CPS is a cyber-physical system where the failure or malfunction may result in one (or more) of the following outcomes: death or serious injury to people, loss or severe damage to equipment/property, environmental harm.
Safety-critical function	A function whose failure or malfunction may result in one (or more) of the following outcomes: death or serious injury to people, loss or severe damage to equipment/property, environmental harm.
Service	Services are the mechanism by which needs and capabilities are brought together.
Service migration	Dynamically moving (migrating) the service (or task) from one processing element to another at runtime.
Software component	An independent software unit that communicates with the surrounding system through explicitly specified interfaces.
Solution	A means of solving a problem or dealing with a difficult situation.
System	A combination of interacting elements organized to achieve one or more stated purposes.
System component	A system architectural element.
Technique	Technical and managerial procedure that aids in the evaluation and improvement of the [system] development process.
Tool	A tool is an instrument that, when applied to a particular method, can enhance the efficiency of the task; provided it is applied properly and by somebody with proper skills and training.

Table 1: Terms and Definitions

Term	Definition
AI	Artificial Intelligence
API	Application Programming Interface
AWS	Amazon Web Services
BDaaS	Big Data Analytics as a Service
CapEx	Capital Expenditures
CPS	Cyber-Physical System

CVEs	Common Vulnerabilities and Exposures
Dx.x	Deliverable x.x
DICOM	Digital Imaging and Communication in Medicine
E2E	End-to-end
HSM	Hardware Security Module
IaaS	Infrastructure as a Service
IoT	Internet of Things
IP	Intellectual Property
KPI	Key Performance Indicator
MAPE-K	Monitor, Analyze, Plan and Execute over a Knowledge base
MCR	Mode Change Request
ML	Machine Learning
mTLS	mutual Transport Layer Security
N/A	Not Applicable
OPEX	Operational Expenditures
OS	Operating System
OSS	Open-Source Software
PaaS	Platform as a Service
PKI	Public Key Infrastructure
QoS	Quality of Service
RA	Reference Architecture
RBAC	Role-Based Access Control
SaaS	Software as a Service
SCDCPS	Safety-Critical Distributed CPS
SIEM	Security Information and Event Management
SIL	Safety Integrity Level

SLI	Service Level Indicator
SLO	Service Level Objective
SotA	State of the Art
SSH	Secure Shell
TLS	Transport Layer Security
TPM	Trusted Platform Module
TSR	Technical System Requirement
UC	Use Case
UI	User Interface
VPN	Virtual Private Network
WP	Work Package
WWTP	Wastewater Treatment Plant
XaaS	*X* (Anything/Everything) as a Service

Table 2: Abbreviations and Definitions

1 Introduction

The overarching goal of TRANSACT is to develop a universally applicable distributed solution architecture concept, a framework and a transition methodology for the transformation of standalone safety-critical CPS into distributed safety-critical CPS solutions. The rationale behind this purpose is manifold, from the reduction of manufacturing costs of devices by simplifying them, extending their lifetime by complementing them with additional services deployed in the near edge, or even improving performance when no critical KPIs are endangered.

However, achieving these objectives is complicated, as there are multiple aspects to be considered, such as being able to ensure privacy, security, performance and safety, among others. Considering and guaranteeing these aspects is complex, as the execution of the service is not constrained to the device anymore, and other variables come into play, such as the reliability of the network, especially in those cases where the cloud is considered; the latency involved, expected to be low when the device relies on the edge tier, but potentially unbounded when the cloud tier is involved; the need to shield communications, wired or wireless; or extending the potential number of users that may have access to the data being generated.

Still, the benefits far outweigh these risks. First, it allows embracing data analytics and AI without requiring a device with extensive resources that would be underused most of the time. Edge and cloud tiers can provide computational support for services leveraging the data produced or captured by the device. Similarly, the abundance of resources in these tiers can be leveraged to outsource the execution of services and provide improved performance in multiple dimensions, e.g., reducing execution time, improving precision in the computations, providing richer results, etc. This outsourcing must, of course, ensure that a series of KPIs or SLOs are continuously met, strictly monitoring the service execution and retake the execution of the service in question if any indicator is in risk of not being met or safety may be compromised. This alternation in the execution (in-device/off-device) can be controlled through the management of different operational modes extending over the different tiers. The result is a distribution of the complexity, simplifying the device, reducing its costs, making it easier to update or upgrade and integrating it into an ecosystem rather than having it operating in isolation.

The goal of this document is to continue the work that was started in Deliverable D7 (D2.1), leveraging the knowledge acquired through the study of the state of the art to design a reference architecture for distributed safety-critical cyber-physical systems. To do so, this document revises and refines the reference architecture that was provided during the proposal phase, diving into its expected functionalities, components and potential application scenarios.

All in all, this deliverable aims to provide the guidelines on how organizations can migrate from monolithic approaches to distributed ones by adopting the TRANSACT reference architecture.

1.1 Purpose and structure of the deliverable document

The purpose of this document is to provide a clear, consistent and comprehensive reference architecture to help organizations migrate safety-critical CPS from monolithic approaches to distributed solutions, while ensuring safety, performance, security and privacy. This can be broken down into the following major tasks:

- Assimilate previous work from the requirements and architecture deliverables into a reference architecture framework.
- Describe the structure and properties of the reference architecture.
- Define the components of the reference architecture.
- Display the behavioural view of the reference architecture, revealing features covered by it and how the different components interact during execution. This is showcased upon some of the key

workflows and scenarios (performance management, change of operational mode, remote updates, access control...) that it can manage.
- Set the architectural basis for the horizontal demonstrator in WP5.
- Provide an architectural framework to get different use case demonstrators to get aligned.
- Help adopting the reference architecture and transitioning to it.

The document contents are arranged according to the next structure:

- Section 2 presents the reference architecture framework, set along a compute continuum of three tiers that encompass device, edge and cloud, including an overview of its services and expected functionalities.
- Section 3 complements the architectural design with insights from technical system requirements, concepts and solutions, establishing a link to the work results from WP2 and WP3.
- Section 4 explains crosscutting non-functional properties that complement the reference architecture and, as well, derive from the aforementioned work packages.
- Section 5 details a static view of the reference architecture in isolation: the components (services and functions) that act as its building blocks. A detailed description of the different components that have been observed is provided for each tier, specifying the differences in those components present in more than one.
- Section 6 provides a dynamic view of the reference architecture that displays how these components would potentially interact in different situations, undertaking detailed descriptions for up to six representative use cases where, for each one of them, different possible scenarios are contemplated, offering examples of different ways in which the TRANSACT various functions can interact. These cases span health monitoring, performance management, change of operational mode, remote updates, access control and end-to-end secure communications.
- Finally, Section 7 presents several recommendations for adopting the TRANSACT reference architecture to migrate from monolithic approaches to distributed ones. In addition, this last section provides an enumeration of open-source tools and services that can be used to implement the different functionalities presented throughout the document.

1.2 Relationship to other TRANSACT documents

This document relates to the following TRANSACT deliverables:

- D5 (D1.1) Use case descriptions, end user requirements, SotA and KPI's (TRANSACT D05, 2022)
  Expression of needs that, through the formulation of requirements, are captured in the definition of a reference architecture that will support them.
- D6 (D1.2) Technical requirements and TRANSACT transition methodology commonalities (TRANSACT D06, 2022)
  Definition of technical system requirements. Every one of them must be covered by, at least, one component in the reference architecture.
- D7 (D2.1) Reference architectures for SCDCPS v1 (TRANSACT D07, 2022)
  First version of the present document. It provides an initial design of TRANSACT's reference architecture. It also features an extensive study of the state of the art that enables the progress towards a more advanced phase into which that design is refined into a complete reference architecture.

- D8 (D3.1) Selection of concepts for end-to-end safety and performance for distributed CPS solutions (TRANSACT D08, 2022)
  Definition of concepts for safety and performance, extracted from requirements. They must be supported by the components in the reference architecture.
- D9 (D3.2) Selection of concepts for end-to-end security and privacy for distributed CPS solutions (TRANSACT D09, 2022)
  Definition of concepts for security and privacy, extracted from requirements. They must be supported by the components in the reference architecture.
- D15 (D1.4) Use case evaluation and impact assessment v1 (TRANSACT D15, 2023)
  Informative report about the industrial use cases. It supplies information about how TRANSACT's reference architecture can be applied to various workflows and scenarios.
- D16 (D3.3) Solutions for end-to-end safety and performance for distributed CPS v1 (TRANSACT D16, 2022)
  Definition of solutions for safety and performance, summarized from concepts that, in turn, are extracted from requirements. They must be supported by the components in the reference architecture.
- D17 (D3.4) Solutions for end-to-end security and privacy for distributed CPS v1 (TRANSACT D17, 2022)
  Definition of solutions for security and privacy, summarized from concepts that, in turn, are extracted from requirements. They must be supported by the components in the reference architecture.
- D18-D22 (D5.3-D5.7) TRANSACT UCx demonstrations and validation v1 (TRANSACT D18-D22, 2023)
  Demonstrators from the industrial use cases. Their application of the TRANSACT approach serve as an inspiration for the definition of the dynamic view presented in this document.
- D23 (D1.3) TRANSACT transition guide to facilitate safety-critical distributed CPS solutions v1 (TRANSACT D23, 2023)
  Transition guide for the transformation of a monolithic cyber-physical system into a distributed solution that retains its non-functional properties. The transition methodology presented helps into providing insights for the adoption of the reference architecture.
- D33 (D3.5) Solutions for end-to-end safety and performance for distributed CPS v2 (TRANSACT D33, 2024)
  Second version of the deliverable document about solutions for safety and performance. Currently under writing process.
- D34 (D3.6) Solutions for end-to-end security and privacy for distributed CPS v2 (TRANSACT D34, 2024)
  Second version of the deliverable document about solutions for security and privacy. Currently under writing process.

The results obtained by this deliverable document D2.4 will be used to create the horizontal demonstrator in WP5 and get also aligned with the different use case demonstrators. It uses as an input the requirements specified by WP1 and the concepts and solutions devised by WP3.

2 Reference architecture framework and concepts

This section describes the reference architecture framework and concepts. First, the overall structure is given and each of the three tiers is described individually, then the system functionality is described.

A reference architecture is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements (OASIS Standard, 2012). It captures the essence of the architecture of a collection of systems. The main purpose of a reference architecture is to provide guidance for the development of architectures (Muller, 2008). In this scope, the concept of the TRANSACT three-tier reference architecture, displayed in Figure 1, is based on a computing continuum that spans from the CPS device (first tier), through the edge (second tier), to the cloud (third tier). It brings together CPS end-devices at the edge of the network, with edge computing servers and cloud computing facilities, hosting multiple mixed-criticality applications.

A key element of the TRANSACT concept is that both these off-device tiers (edge and cloud), in addition to being multi-application, multi-device and possibly multi-tenant, support scalable and interoperable, as well as safe and secure, solutions. Applications can be flexibly deployed over this distributed architecture. Novel services and applications can share edge and cloud infrastructures, re-use domain services, and in general be faster developed, tested, and released independently from the safety-critical device itself. This newly distributed architecture concept will enable new business models by transforming system manufacturers into solution providers.

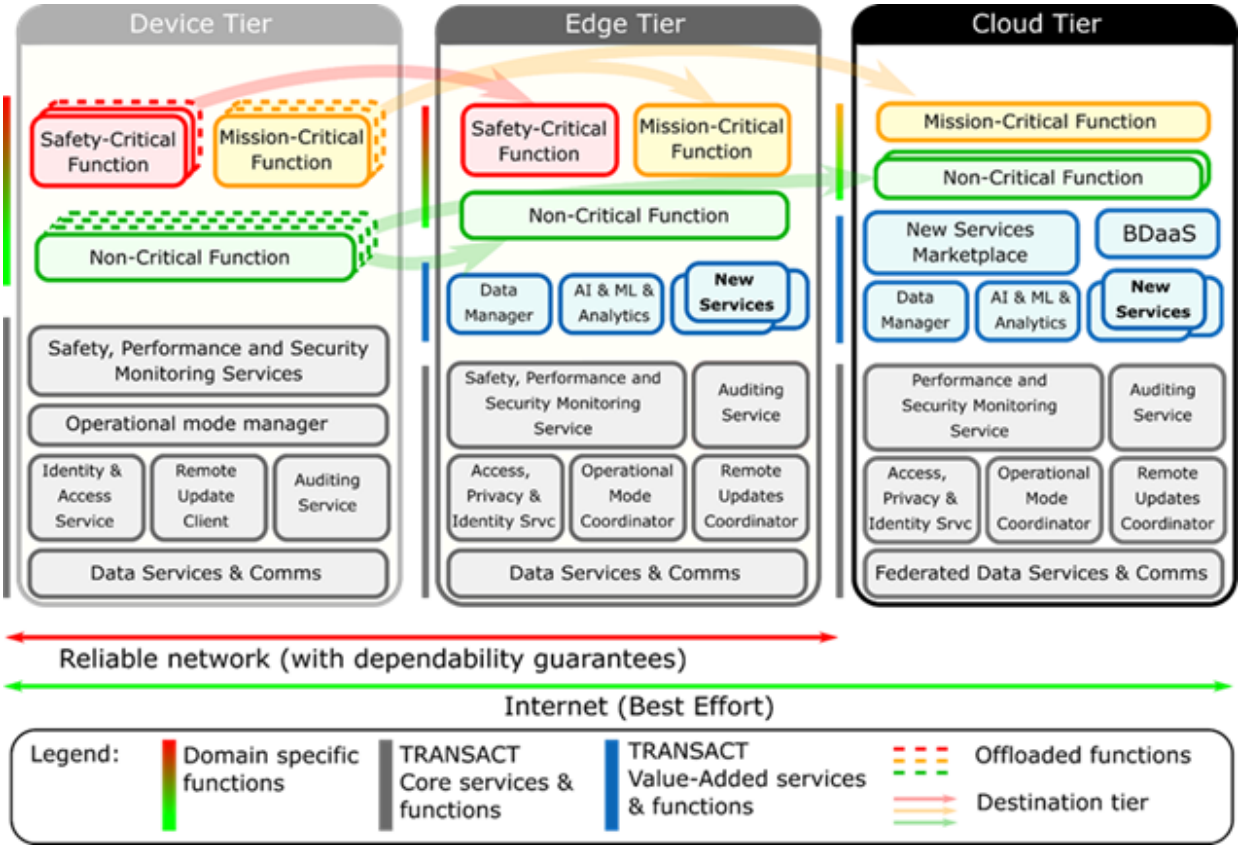

Figure 1: Components of the TRANSACT reference architecture long its three tiers.

2.1 Three-tier architecture

Along the three tiers that the TRANSACT architecture concept (cf. Figure 1) distinguishes, the CPS end devices are linked via the edge infrastructure to the cloud computing facilities, where the end device and the edge tier are connected via a reliable network, whereas the cloud tier has only access to the internet. TRANSACT defines the Core components (shown in grey) for all three layers that allow functions to be offloaded from the end device to off-device tiers.

The main system functionalities are distributed across the three tiers. Each tier provides a specific quality of service level especially with respect to performance aspects (such as response times and data transfer guarantees, ranging from best effort to reliable and time-deterministic data transfers) which are critical for the safety-critical and mission-critical functions. For example, safety-critical functionalities often have hard real-time constraints which lead to severe failures when missed, whereas the mission-critical functions may have soft real-time constraints which may degrade the system's quality of service when missed, but do not necessarily lead to failures. Next to safety, also security and updates are important aspects of the distributed system.

The TRANSACT reference architecture defines firstly the safety- and mission-critical functions, and non-critical functions. These domain specific functions or components, which are depicted in red, yellow and green depending on their criticality, may be offloaded from the device to other tiers. Furthermore, the reference architecture defines the Core TRANSACT components, available to every use case, which are depicted in grey. Finally, blue components refer to potential Value-Added functions that may be included depending on the use case.

The safety- and mission-critical functions are key in the safety-critical CPS. TRANSACT aims at improving the existing CPSs by, first, stripping the device of the functions that are not safety- or mission-critical and can be executed remotely; and second, by offloading certain safety-critical functions to the edge tier. The functions to be offloaded are identified at design time and deployed in the required tiers. As a result, the device would only keep the basic and safety-critical functions while offloading the remaining functions to the other tiers. Such an approach presents numerous advantages such as: improved reliability and performance of the device (as fewer services are running on the device), improved efficiency of the offloaded functions due to usage of better hardware at the edge or cloud, improved innovation speed of the distributed CPS solution as the new or upgraded functions can be deployed with greater ease at the edge and cloud.

2.2 TRANSACT services and functions

Therefore, this architecture introduces several Core services deployed across all the tiers to ensure safety, performance, security and privacy of the new solution. TRANSACT's Core services and functions are listed below, introducing their domains and related responsibilities. Static view (Section 5) and dynamic view (Section 6) in this document give an elaborate overview of such services and functions, including the description and roles of these Core services and relevant scenarios where they are essential, respectively.

As already mentioned, when considering offloading functions from the device, it is critical to ensure CPS solution's end-to-end performance, safety, security and privacy. Therefore, several dedicated Core services, which are briefly introduced next, are featured to realize that objective cooperatively. The *Safety, Performance and Security Monitoring Services* are responsible for **monitoring, detecting and preventing safety, security and performance failures**. In addition, they measure the Service Level Indicators (SLIs), such as latency, throughput, accuracy or uptime. The scenarios described in the *Health monitoring* and *Performance management* sections leverage these monitors to ensure the system is working as expected.

The Service Level Objectives (SLOs), measured by the SLIs, are used by the *Operational Mode Manager* (running on the device) and the O*perational Mode Coordinator* (running at the edge/cloud tier) to **decide at run-time the proper operational mode** to be executed. E.g., function offloading from the device is associated with operational thresholds or ranges described in a SLO. If this SLO is compromised or out of range, the operational mode coordinator will request the device to switch to an operational mode where the function is executed locally, to ensure the required performance. To better understand adapting the system behaviour, *Change of operational mode* scenarios implement these Core services in different domains.

Another concern that the TRANSACT reference architecture addresses is updating the distributed solution. To achieve **safe and predictable system updates**, the following Core services have been identified: the R*emote Update Client* (running on the device) and the *Remote Update Coordinator* (running at the edge/cloud). Those services cooperate across tiers to perform remote automatic updates of the different device services in a secure and safe way. The updates ensure uniform software versions on the tiers and keep the system services up to date with the latest functionality. In addition, the automatic updates allow rolling-out a new functionality or introducing new value-added services, also minimizing system downtime. Each update activity is coordinated with the operational mode coordinator services to keep the system in a safe state at any time. The dynamic behaviour of software updating is also explained through the scenarios of the *Remote updates* section in the dynamic view.

The **secure access** to the system functionality is managed by the *Access, Privacy and Identity Services*. These services are responsible for granting/denying access to the system resources based on the policies defining who (and which role) has access to which resources. Another Core service unit contributing to the system security is the *Auditing Service*. This service collects information about access and usage of the system to help detecting security policy violations when the system is accessed by unauthorized users or in unauthorized way. The security aspects are also addressed by the *(Federated) Data Service and Communications* helping in efficient and secured data handling, both in transit and at rest. *Access control* and *End-to-end secure communications* sections describe relevant scenarios where security aspects are explained in specific domains.

The TRANSACT reference architecture also defines *Value-Added services and functions* that **enhance the system capabilities**. Those functions can be introduced during the system design or after the system release (as part of the system updates). Within these services are the *Data Manager* services that encompass the storage, protection and access to data in the cloud and edge tiers. They ensure data is validated and fully accessible to services and applications when needed. Also, the *Big Data as a Service (BDaaS)* component included in the cloud tier enables the infrastructure to provide other services and applications with advanced analyses on large data sets.

Further Value-Added services are the *AI & ML & Analytics* services giving **insights into the collected data by extracting valuable information** that helps improving user activities. For example, in the healthcare domain, those services can enhance the specific algorithms assisting doctors in disease diagnostics and supporting them in making clinical decisions; in the automotive domain, those services can enhance the routes' predictability or leverage the risk analysis to assist in better automatic and manual driving decisions. In addition to the user task improvements, such services can help optimise an organization's operational performance and costs, for example, by better equipment utilization in the healthcare or transportation domains.

The new *Value-Added* services can also be made available via the *New Services Marketplace*. Such a marketplace service opens possibilities to **provide new solutions** (applications, services, algorithms, AI models, etc.), not only by the system builder but also by third-party vendors.

In the TRANSACT project, this universally applicable distributed solution architecture has been elaborated and augmented with a framework and transition methodology for transforming safety-critical CPS into distributed safety-critical CPS solutions. The architectural solution can host a diverse set of mixed-criticality applications, which can be distributed over multiple resources, e.g., CPS devices, high-end computing resources hosting multiple applications of mixed-criticalities, or in the cloud. It also supports a variety of interesting trade-offs by allowing the allocation of mixed-criticality functions over the computing continuum of device-edge-cloud.

3 Relation of the reference architecture to requirements and solutions

In order to have a consistent story, TRANSACT's functions and services of the reference architecture must be related to the technical system requirements defined in Deliverable D6 (D1.2). Since these were partially elucidated from the use case requirements defined in Deliverable D5 (D1.1)[1], these were not sufficient and complete for an overall reference architecture and, therefore, additional TSRs were defined. Every one of these requirements must fall into one or multiple components.

Furthermore, the Deliverables from Work Package 3 present concepts and solutions focused on topics related to safety and security. All components must also be pointed by at least one requirement and concept, validating that the architecture provides a complete coverage of such requirements, concepts and solutions, at the same time it does not include spurious elements that are not actually required nor used.

This section displays the relation between TRANSACT's Reference Architecture to technical system requirements and derived concepts and solutions, as it is described in the present document. More specifically, a mapping from those requirements and concepts to architectural elements is devised, highlighting their relation, providing a two-way traceability between them and ensuring the validity and completeness of the reference architecture.

3.1 Requirements and architecture mapping

Table 3 below displays a mapping between the technical system requirements and components in the reference architecture. Requirements are referenced according to their identifiers in D6 (D1.2), and split in five thematic groups, also including those requirements that were expressly included from an architectural perspective or added in response to safety and security additional considerations. Each cell reflects the number of requirements (per group) that are related to that component or fulfilled by it.

As overall observations, more than half of the TSRs concern Artificial Intelligence and Machine Learning, and therefore match the *AI & ML & Analytics* and *BDaaS* components. The *Safety, Performance and Security Monitoring Services*, *Access, Privacy & Identity Service* and *(Federated) Data Services & Communications* components are heavily used, noting the distributed nature of the architecture and its focus on safety and security. The *New Services Marketplace* is aimed to provide new domain-specific services, not defined in the initial version of specific solutions, which the use cases, the horizontal demonstrator and further applications can benefit from.

[1] Deliverables D5 (D1.1) and D6 (D1.2) are confidential documents. They are referenced here to provide evidence that the architectural components match a series of needs expressed by stakeholders, even if those needs and requirements cannot be displayed on this present document.

Architecture component	Number of (grouped) related requirements					*Total associated requirements*
	Security and privacy (TSR-1 to TSR-20)	Safety and performance (TSR-21 to TSR-60)	Additional safety and performance TSR (TSR-SP-1 to TSR-SP-4)	Additional TSR from architecture perspective (TSR-ARCH-1 to TSR-ARCH-3)	AI base ML pipeline (TSR-61 to TSR-141)	
Safety, Performance and Security Monitoring Services	11	11			7	**29**
Operational Mode Manager/Coordinator		2	1			**3**
Access, Privacy & Identity Service	20	1		1		**22**
Remote Updates Client/Coordinator		2	1			**3**
Auditing Service	3			1		**4**
(Federated) Data Services & Comms	19	13	1			**33**
Data Manager	7	10	1		11	**29**
AI & ML & Analytics		3			64	**67**
BDaaS					36	**36**
New Services		1	1		6	**8**
New Services Marketplace				1		**1**

Table 3: Mapping from technical system requirements to architectural components

3.2 Concepts and solutions architecture mapping

Complementing technical system requirements, D8 (D3.1) and D9 (D3.2) identify a series of relevant concepts for safety and performance, and security and privacy, respectively, associating them to the TSRs they are inferred from. From them, a series of solutions on those topics, applicable across different tiers of the reference architecture, are derived. These solutions are presented in D16 (D3.3) and D33 (D3.5) (safety and performance), and D17 (D3.4) and D34 (D3.6) (security and privacy), which point out how they fit with the reference architecture and link them to specific components in the three tiers. Table 4 synthetises and

displays this information. This reflects how the reference architecture support these concepts and solutions, briefly described below, which in turn can be traced back to the requirements.

- **Safety and performance** preserving solutions:
    - **Application solutions** target the *applications*. An application is the functionality that implements a particular solution/technique/method to help the end-user to perform a specific task. It can be composed of a monolithic service or a group of distributed services which are executed in different and distributed targets in the device-edge-cloud continuum.
    - **Cross-cutting solutions** are system-level methods and techniques for linking application and platform. They include concepts for designing an employing the application on the platform, as well as analysing and runtime monitoring and managing its behaviour. They can be divided in two groups, the first focusing on **performance** and the second comprising **safety, security, health and risk**.
    - **Platform solutions** target the *platform*. The platform is the environment in which the application is executed. It comprises the complete infrastructure in the device-edge-cloud continuum to execute the application, including hardware, network, hypervisors, operating system, containers, cloud computing services and runtime libraries.
    - This categorization allows system designers to distinguish the platform (the infrastructure of a system) from the applications that run on this platform or are enabled by this platform. Any solution that considers both the status of the application and the platform into account would then be categorized as a cross-cutting solution.
- **Security and privacy** preserving solutions:
    - **Secure communication-based solutions** target the end-to-end security and integrity of data transmissions of data between tiers, including wireless and public channels, as well as over-the-air updates.
    - **Identity access and services** target secure authentication management and role-based access control rules, in addition to the generation of customized view of security-related information.
    - **Safety, security, and performance monitoring services** target the security and privacy of elements such as remote attestations and trust-related solutions for various domains.
    - **Cloud security** targets cloud-based security and posture management tools, as well as cloud detection and response capabilities.
    - **Data communication security** targets the privacy of solutions whose application domains range from traffic monitoring in public spaces to medical/healthcare DICOM solutions.
    - **AI/ML-based solutions** preserve user privacy and detect anomalous activities in cloud environments by leveraging machine learning techniques and models.
    - **Risk analysis** targets solutions to minimize security and privacy risks of components and applications deployed in the edge or cloud. One solution example is the SIRENA tool, while a potential application is the medical & healthcare domain.

- **Hardware-based security** targets hardware devices able to act as a source of trust, such as HSM devices for the generation and storage of keys, and TPM chips to provide end-to-end security.
- **Security solutions for new services** target solutions that may enforce the security of third-party services and applications, such as security-by-contract frameworks.

Architecture components	Related solutions												
	Safety and performance				Security and privacy								
	Application solutions	Cross-cutting -performance solutions	Cross-cutting - safety, health, risk, security	Platform solutions	Secure communication-based solutions	Identity and access	Safety, security and performance	Cloud security	Data communication security	AI/ML-based solutions	Risk analysis	Hardware-based solutions	Security solutions for new services
Monitoring Services	•	•	•	•			•	•	•	•	•	•	
Operational Modes	•	•	•	•									
Access, Privacy & Identity Service						•		•	•			•	
Remote Updates Client/Coordinator										•			
Auditing Service									•	•			
(Federated) Data Services & Comms				•	•			•	•	•			
Data Manager	•												
AI & ML & Analytics	•									•			
BDaaS						•							
New Services & Marketplace	•												•
Non-Critical Function									•				
Mission-Critical Function									•				
Safety-Critical Function										•			

Table 4: Mapping from solutions to architectural components.

4 Crosscutting non-functional properties of the reference architecture

TRANSACT's functions and services of the reference architecture are based on the concepts and solutions described in WP3. This section provides insights into how to consider those concepts and solutions. Following the decomposition developed in its work package, they can be split into "safety and performance" and "security and privacy".

4.1 Insights for safety and performance

As has been described in D8 (D3.1), safe operation of safety-critical distributed CPSs that have been deployed on a distributed device-edge-cloud continuum requires both design-time and run-time techniques to model, analyse, and monitor performance of these systems as well as their safety (and security) properties. Moreover, there is a need for proactive resource management for the resources on the network, cloud, and edge platforms with the goal of guaranteeing certain timing properties (e.g., for real-time applications running on the cloud) or improving performance and resource efficiency (for real-time and non-real-time applications).

Crosscutting solutions for safety and performance have been identified in two categories:

1. Solutions for predictable performance.
2. Solutions for safety and health monitoring, risk analysis, and safety and security assurance.

4.1.1 Predictable performance

Performance aspects may be of different levels of importance or criticality depending on the system nature. Typical examples of performance aspects are end-to-end latency or response times, throughput, and scalability. For real-time systems, timing performance is an integral aspect of the functional behaviour of the system. For safety-critical systems, performance aspects may be integrally essential to meet safety requirements.

In distributed CPS solutions, designing a system for required or optimal performance is extremely challenging. Their distributed nature makes performance an emergent property of the interaction of many components that are physically distributed, often heterogeneous, and not necessarily subject to a single point of control. Some of the resources, such as cloud resources, are shared and managed in unpredictable ways and may not always provide the levels of reliability required by an application. The satisfaction of performance requirements and optimization of performance objectives can therefore not be dealt with at design time only, but also requires active runtime management.

To address predictable performance, three areas have been recognized:

1. Performance monitoring.
2. Performance modelling.
3. Performance management.

The goal of the first area is to monitor a system's performance on the device-edge-cloud continuum. For performance modelling partners have investigated solutions using AI-based, simulation-based, and formal-method techniques. With respect to the third area, partners introduce a scenario-based performance

management and reconfiguration technique that allows adjusting system resources to the applications and adapting their mode at runtime while preserving some properties about the system performance.

4.1.2 Safety and health monitoring, risk analysis, and safety and security assurance

Safety engineering (Lutz, 2000) has traditionally been concerned with assuring that engineered systems provide acceptable levels of safety. Traditional methods include failure mode and effects analysis (FMEA), fault tree analysis (FTA) and bow-tie analysis. (Ferdous, 2011)

When safety-critical or mission-critical functions are offloaded to the edge or cloud, then monitoring at runtime is necessary to ensure that the system is still operating within specifications (and is safe to use). Health/safety monitoring can encompass the monitoring of availability of resource instances on the device-edge-cloud continuum, as well as the liveness, responsiveness and integrity of applications. (Sanden, 2021)

To address these topics, solutions have been described for:

1. Risk management planning/monitoring.
2. Real-time machine-learning based solutions for detecting safety, security, and privacy anomalies,
3. Mapping and scheduling techniques across device, edge, and cloud.
4. Service continuity monitoring.

While the first solution is carried out as a risk assessment involving representatives of the use cases, the other three solutions relate to functionalities which should be considered in the TRANSACT architecture.

4.1.3 Affected architecture components

The actual implementation of safety and performance aspects in the architecture will be application specific. To this end, the safety concept of an application will typically employ the components related *to Safety, Performance and Security Monitoring Services* and *Operational Mode Coordinator* at the respective tiers (Figure 2), which are thoroughly described in Section 5. It is important for the application's development to not only create a distributed solution for the specified safety, performance and security requirements, but also to verify and validate that these requirements have been realized.

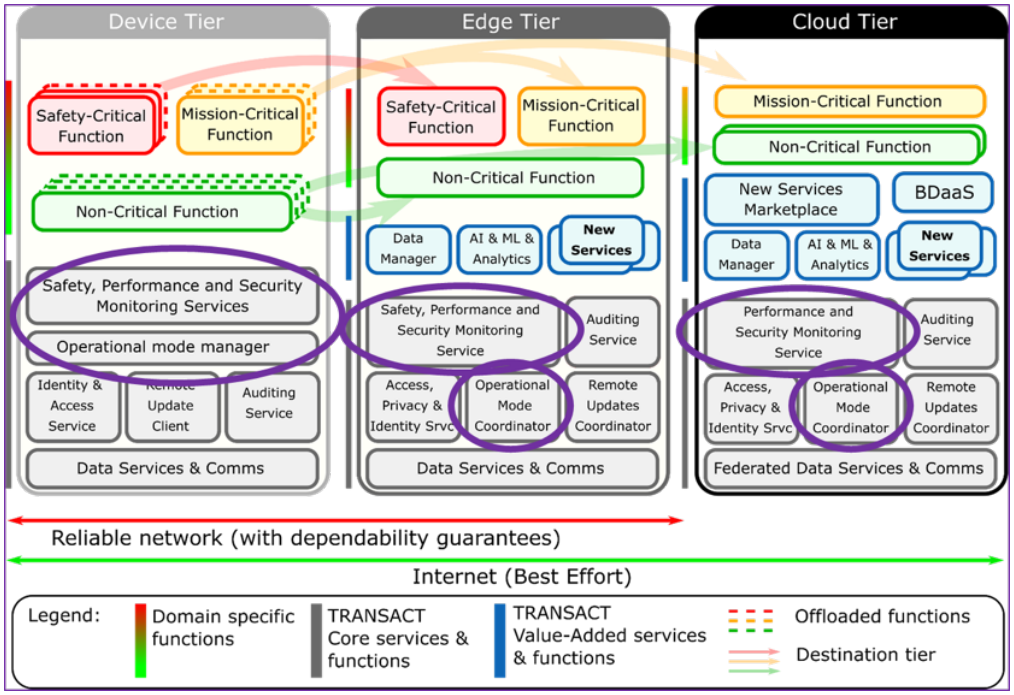

Figure 2: Allocation of safety and performance functionalities in the TRANSACT architecture

4.2 Insights for security and privacy

TRANSACT deliverable D9 (D3.2) has described security and privacy concepts for four categories:

1. Security & Privacy Concepts for Domain-Specific Functions.
2. Security & Privacy Concepts for TRANSACT Value-Added Services & Functions.
3. Security & Privacy Concepts for TRANSACT Core Services & Functions.
4. Application-Specific Security and Privacy Concepts.

Together they cover a wide variety of concepts which should be considered when developing cloud-based solutions. The mapping on the reference architecture depicted in Figure 3 shows which components should be considered for which category.

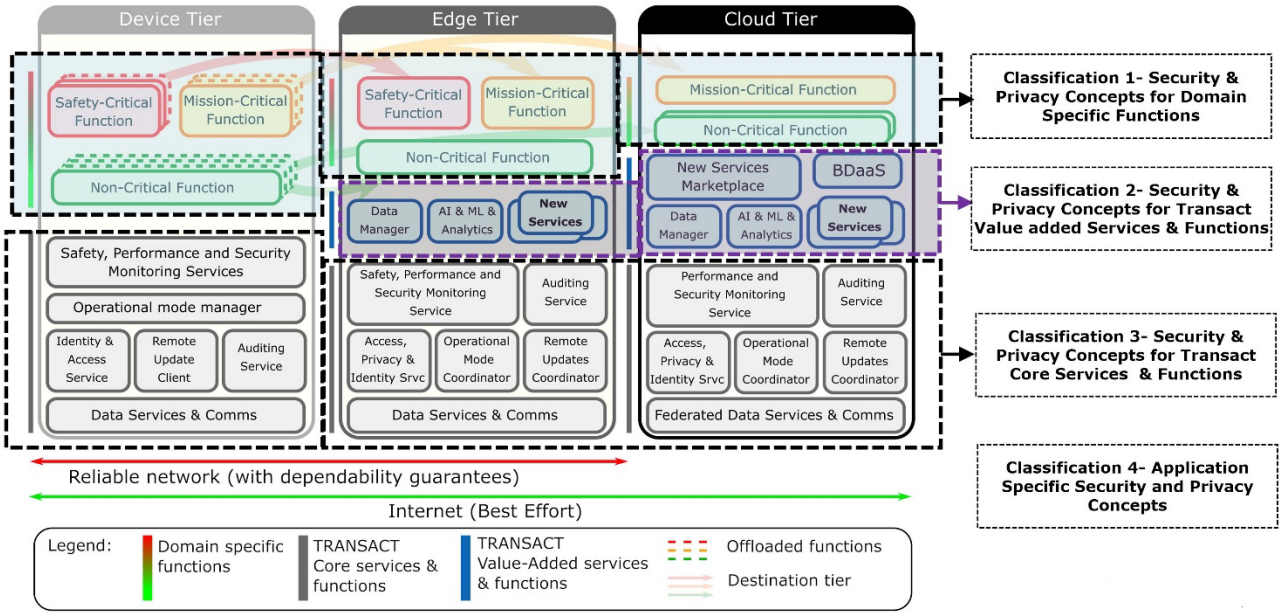

Figure 3: Allocation of security and privacy functionalities in the TRANSACT architecture

5 Static view of the reference architecture

This section outlines the static view of the TRANSACT reference architecture, which describes a high-level representation of the key components of the architecture. The purpose of this section is to provide a clear, high-level understanding of the structure of the architecture without including its temporal behaviour and to serve as a concept map for designing distributed systems based on TRANSACT. In doing so, the architecture developed in D7 (D2.1) is refined by detailing their component descriptions.

The reference architecture is divided into various components, each responsible for specific functions or features of the system. These components represent the abstract building blocks of the system that can be implemented using one or different modules with a common objective.

The static view provides the model of the three tiers from a non-dynamic point of view, including the roles and functionality of each component, their configurations and interfaces, the relationship with the other components, with the Technical System Requirements, and with the TRANSACT concepts and solutions.

A common approach to describe the different components composing an autonomic system is a MAPE-K loop (Monitor-Analyse-Plan-Execute over a Knowledge base) (Arcani, Riccobene, & Scandurra, 2015). The Knowledge (K) manages the data of the system that is observed by the Monitors (M). Analise (A) components perform data analysis to decide whether an adaptation is required, following the actions defined by the Plans (P) that are carried out by the Executors (E). The components detailed in the static view are described in terms of this model in order to classify them into the different roles of an autonomic system.

The components belonging to the three tiers of the reference architecture are described in the following sections, differentiating the functionality implemented at each tier.

5.1 Cloud tier

The cloud tier (Figure 4) hosts the domain-specific functions with higher computational requirements and the need for greater scalability. These functions are implemented on top of the TRANSACT cloud Core services and functions (Section 5.1.1) and the TRANSACT cloud Value-Added services and functions (Section 5.1.2).

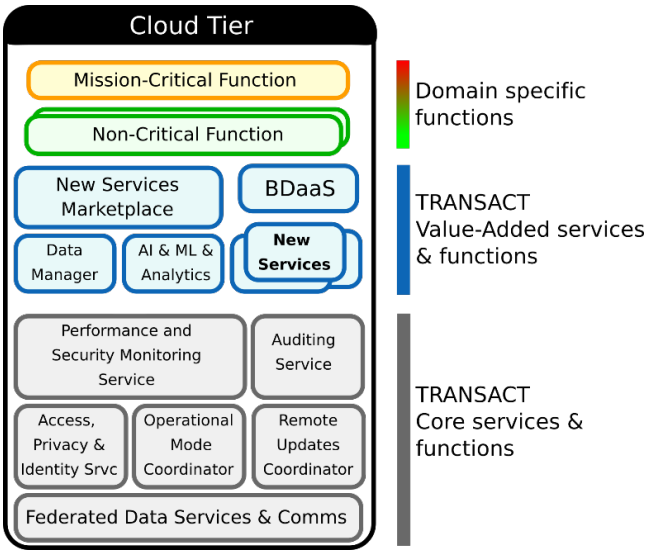

Figure 4: Static view of the cloud tier

5.1.1 Core services and functions

The cloud Core services support the mission-critical and non-critical functions at the cloud tier. These core components encapsulate the safety and performance, and security and privacy high-level functionalities of the system, which are responsible for providing the necessary support to monitor, coordinate and configure the rest of the system.

5.1.1.1 Performance and Security Monitoring services

Monitoring service in the cloud comprise the monitors in charge of collecting performance and security indicators.

The aim of the performance monitoring service is to monitor performance indicators from the various components and systems in the cloud. This might include monitor the cloud infrastructure to scale resources dynamically based on demand or monitor the allocation of virtual machines, containers or serverless functions to optimise resource usage (such as aspects as CPU, memory, and disk usage). Performance monitoring service is also in charge of collecting information about communication performance, such as bandwidth usage, packet loss and jitter or communication latency.

Apart from monitoring performance indicators the service is also responsible, depending on configuration, to respond to out-of-bound indicators. These responses could be in the form of alarms (e.g., email, sms, ...) or actual mitigations to these conditions. These mitigations can for example include automatic scaling (in or out) of components or request more (or less) cloud compute resources.

The aim of the security monitoring service is to detect and prevent security violations. The preventive part will continuously monitor all components and their compliance with the security policies that are defined. Optionally, it can also scan continuously for known Common Vulnerabilities and Exposures[2] (CVEs). Both parts are preventive steps to avoid and/or mitigate security issues.

Next to the preventive measures, the security monitoring service can also be equipped with a Security Information and Event Management (SIEM) functionality. This functionality provides real-time analysis of security alerts generated by the components and applications. In essence, it detects anomalies in the audit logs.

Furthermore, the services receive performance and security measurements of the edge and device tiers.

The overview and additional information are detailed in Table 5.

Role/functionality		• MAE in the MAPE cycle (extend functionality) • Monitoring performance and security indicators of different components • Mitigations to out-of-bound indicators
Configuration	Mandatory	• If KPIs are monitored though pull operations, the endpoints for the KPIs have to be configured

[2] https://cve.mitre.org/

	Optional	• Criteria/ranges to specify/describe the monitoring of the KPI (e.g., SLOs, thresholds)
Input interfaces	Mandatory	• Performance KPIs of components (system/service KPIs). It may be required to push or pull the KPIs, depending on the scenario.
	Optional	• Security policies
Output interfaces	Mandatory	• The set of KPIs themselves
	Optional	• The analysis related • Predictions on KPIs values, e.g., regarding the crossing of a threshold • Outputs/alarms/events
Endpoints		• Alerting APIs • Scaling APIs
Covered requirements		• [TSR-30] Architecture should be self-organising, that implies self-monitoring capabilities. • [TSR-38] Constantly monitor available resources to detect software failures and aging in the architecture • [TSR-46] Cloud computing components in the architecture will enhance the performance by processing IoT data in the cloud • [TSR-55] The components in the architecture should be able to monitor events, resources and errors. • [TSR-56] The architecture should support the monitoring of the data transmission rate of the devices. • [TSR-58] The TRANSACT system architecture should support error detection. • [TSR-102 to 109] Monitoring and modelling systems
Related concepts & solutions		• [D3.1] Application service level agreements • [D3.1] [D3.3] [D3.5] Concepts and solutions for AI monitoring • [D3.1] [D3.3] [D3.5] Concepts and solutions for predictable performance • [D3.1] [D3.3] [D3.5] Concepts and solutions for health and safety monitoring • [D3.2] Runtime verification
Related components		• Auditing Service • Operational Mode Manager/Coordinator

Table 5: Performance and monitoring services in the cloud

5.1.1.2 Auditing service

An auditing process consists of examining and evaluating processes in terms of their level of computerisation and data processing, also of verifying controls in the processing of information and verifying the installation of security systems.

The aim of this process is:

- To determine if the controls implemented are enough and efficient.
- Identify the root of an existing problem or areas of improvement.
- Determine preventive and corrective actions so the system remains trustworthy and safe.

When a system is audited, the technical components of the system are analysed for vulnerabilities or threats that could have a negative impact on the system. Besides, an audit process is useful to identify the traceability of the processes, to evaluate the technical aspects and to identify potential threats and possible failures regarding safety and security.

With the aim of significantly enhance auditing processes, the TRANSACT reference architecture includes Auditing services, which provide automation, efficiency, and improved accuracy in the monitoring, analysis, and reporting of audit data. These services are responsible for tracking user access and actions within the cloud tier, logging data access and modifying sensitive data or any event related to a security incident. Additionally, Auditing services should provide compliance reports that meet industry standards and regulations.

Regarding deploying these services into the cloud tier, this solution offers some advantages in front of deploying at edge or device: it is more accessible and more flexible, and using the cloud avoids expending funds on hardware, software and infrastructure.

It is necessary to highlight the importance of checking, not only the computerised equipment but also the controls applied and their use.

When using a cloud platform, it is important to consider that when a service is outsourced, the “owner” lost control over safety and security. That means it could be some risks which in on premise environment would appear. So, the safeguards and measures applied must be thinking and oriented to compensate this lack of control.

Some safeguards that are needed to be implemented are those aimed to protect against network failures; using a sandbox so the “real” and “testing” data are in a distinguished environment. Also, cloud suppliers must adopt measures to make sure that data centres are in a protected environment (e.g., in a fireproof room, with a regular temperature, access control, etc.).

Besides, since the cloud provider can process personal data, it must be considered as a processor in compliment of personal data regulation.

Lastly, another regulation that must be taken into consideration, is the Directive NIS2[3], since it establishes cybersecurity safeguards. These regulations and particularities must have considered in a process of auditing.

In summary, auditing services should simplify the work of auditors with process automation and support for better decision-making and risk management.

The overview and additional information are detailed in Table 6.

[3] https://digital-strategy.ec.europa.eu/en/policies/nis2-directive

Role/functionality		• Check safeguards. • Identify areas for improvement. • Review compliance with various applicable laws. • User and access control tracking. • Data access logging. • Track anomalies in resource consumption. • Generate compliance reports to meet industry standards and regulations • Provide alerts for security incidents
Configuration	Mandatory	• The policies and proceedings implemented must comply with the applicable regulation. • These policies and proceedings need to be correctly implemented in the structure of the company.
	Optional	• It is advisable to involve all the areas in the process of auditing.
Input interfaces	Mandatory	• Coordination between departments and auditor. • Information on the current status. • Set a level of auditing.
	Optional	• N/A
Output interfaces	Mandatory	• Reporting. • Analytic results. • Advisory output.
	Optional	• N/A
Endpoints		• Interchange proceedings and policies.
Covered requirements		• [TSR-13] Protection against authentication and authorization attacks • [TSR-18] Mitigation plan in case of system is compromised and how to participate in the investigation and prosecution. • [TSR-19] Data is not shared to unauthorised third parties.
Related concepts & solutions		• [D3.1] Requirements and concepts for end-to-end safety and performance assurance • [D3.2] [D3.4] [D3.6] Security and privacy concepts for secure remote driving operation

	• [D3.2] Security and privacy requirements and patterns for the healthcare DICOM data and applications
Related components	• Performance and Security Monitoring Services. • Operational Mode Coordinator. • Access, Privacy and Identity Services.

Table 6: Auditing service in the cloud

5.1.1.3 Access, Privacy and Identity services

In the section below, emphasis is put on the Access & Identity part of the service. The privacy preserving parts are documented in Section 5.2.1.3, where aspects of the edge tier are discussed. The rationale is that the decision whether privacy preserving functions are deployed at the cloud or at the edge typically depends on legislation and the processing power. Since more strict legislation often restricts Privacy preserving measures to be executed on the Edge, but the capabilities are largely the same, the Privacy services is documented in Section 5.2.1.3.

The Access and Identity (A&I) services aim to authorize access to resources and authenticate users. Authorization protects the access of resources owned by an entity by only giving access to those resources to entities that are entitled to access those resources (e.g. by consent of the owner of the resources). Authentication verifies a users' claimed identity. Both contribute to safeguard privacy, but only form a small part in privacy protection. Specific privacy aspects are detailed out in Section 5.2.1.3.

Typical aspect for the cloud part of these services is their usage in microservices architectures, allowing for scalability and configurability. It allows all components (services/APIs, devices (sensors, apps), etc), to obtain secrets that hold authentication/authorization information from a single extensible authorization server, removing the needs to link different user identities, exchange different means of defining access permissions (scope, role) as long as the components are internet connected. These secrets are implemented in different ways, supporting various scenarios: credentials, tokens, certificates, API keys, etc.

Often these secrets are created, managed and provisioned by a (central) authorization server, but integration with third-party authorization server (delegation, impersonation) is often required to connect components from different systems together. When migrating from a local CPS to a distributed solution, often integration with other identification and authorization protocols is required, like SAML, Active Directory etc is required.

The flow for Identification and Authorization is typically triggered by devices, like sensors triggering the flow, or users triggering the flow. Communication to and from the edge is optional, where the edge can serve as a proxy, perform routing, filtering etc. Access & Identity management in case there is no connection to cloud is available is part of the edge/device description.

The overview and additional information are detailed in Table 7.

Role/functionality		• Establish identity of a user (E)
		• Authorize the access of an entity (user/client/device) to a protected resource (E)
		• Configuration & Management of users/groups, client/device, scopes/ roles/permissions, secrets etc. For application/service developers & for end-user admins. (P)
		• Monitor and audit access request (for exploit prevention, intrusion detection). (M,A)
		• Creation, (secure) storage, management, and provision of secrets (tokens) that are to be provisioned to edge / devices (to enable local authentication & authorization). (P)
		• PKI infrastructure (to sign keys/tokens) (P)
		• Optional:
		  ○ Self-Service Portal for configuration
		  ○ Federation
		  ○ Single Sign On

	Mandatory	• Runtime: All configuration to support role/functionality/token/key creation should be at runtime. Supported by configuration API/ self-service portal. • At configuration time: Specification of access rules describing role permissions on system resources and services.
	Optional	• Configuration of use of already integrated external Identity Providers.

Input interfaces	Mandatory	• Requests input according to relevant authentication protocol (e.g. OAuth, SAML)
		  ○ Authorize: Token Request
		  ○ Evaluate: Token Introspection, User Info
		  ○ Invalidate: Token Revoke, Session Termination
		  ○ Revalidate: Token Refresh, Session Refresh.
		• Configuration API

	Optional	• UI/WebSite for self-service configuration
Output interfaces	Mandatory	• Authorization codes, access token • Access/denied to an entity, resource, or service
	Optional	• N/A

Endpoints	• API with authorization and token endpoints (e.g. according to Ouath2/OpenID) • HealthCheck (alarms), • Invalid Login Check (alarm)
Covered requirements	• [TSR-1] This component provides access control exchanged over security (encrypted channel), including measures that safeguard the token exchange (e.g. auth code exchanged for token via back channel) • [TSR-10] I&A should support against (malware) requested tokens e.g. with challenge-response or request-response association • [TSR-11] I&A components should be developed such that (zero-day) vulnerabilities are mitigated (asap), and should deploy defences against CSRF/SSRF (e.g. state in Oauth2.0) • [TSR-13] Use OAuth >1.0 • [TRS-17] Security by (logical) separation in orgs/apps/users etc. • [TSR-18 to 20] Component should (indirectly) contribute.
Related concepts & solutions	• [D3.4] [D3.6] Authentication Management • [D3.4] [D3.6] Role Based Access Control • [D3.4] [D3.6] Security, privacy and trust related solutions for remote driving operation
Related components	• Auditing Service • Cloud/edge tier: TRANSACT Value-Added services & functions. These services and functions may require the access, privacy and identity component to verify the access permissions of a user.

Table 7: Access, privacy and identity services in the cloud

5.1.1.4 Operational Mode Coordinator

The Operational Mode Coordinator in the cloud tier aims to manage the transitions between operational modes at system level. This responsibility is not only to control mode changes, but also to synchronise Operational Model Coordinators in the edge tier (see Section 5.2.1.4), to maintain a consistent execution. An operational mode defines which behaviours and configurations the system can currently have. Modes are defined for all phases of any execution. There are separate modes for normal operation and there are failure modes, where e.g. mitigation mechanisms are triggered. A mode transition corresponds to a change in the system behaviour and is triggered by events or alarms provided by other services.

Services in charge of collecting and analysing data in the compute continuum should generate the events (or alarms) to activate the transitions between modes. E.g., Service Level Indicators (SLI), in comparison to Service Level Objectives (SLO), could be used to generate an alarm if the SLO is out of range or could be compromised. In this case, the monitoring service in charge of measuring that SLI should produce the alarm to be used in the Operational Mode Coordinator.

Regarding the synchronisation method with Coordinators in the edge tier, Operational Mode Coordinator in the cloud tier must be responsible for propagating requests to perform the mode change, ensuring application consistency. The coordinators in the edge handle, in turn, the requests to Operational Mode Managers in the devices. This is further developed in their corresponding sections (see Section 5.2.1.4 and Section 5.3.1.4).

The overview and additional information are detailed in Table 8.

Role/functionality		• Manage the transitions between operational modes at system level. • Synchronize different coordinators at the edge tier. • Execution phase (E) in the MAPE cycle. • Optional Planning phase (P) on the MAPE cycle.
Configuration	Mandatory	• At configuration time: ○ Operational modes at system level. ○ Allowed mode transitions. ○ Rules relating events/alerts and transitions.
	Optional	• N/A
Input interfaces	Mandatory	• Events/alerts that enable mode changes
	Optional	• Status of the Operational Mode Coordinators in the edge tier.
Output interfaces	Mandatory	• Mode Change Requests to edge tier
	Optional	• N/A
Endpoints		• Mode Change Request • Status • Other communication may take place through publish/subscribe mechanisms
Covered requirements		• [TSR-60] Multimode approach to support online error recovery and repair • [TSR-SP-2] Allowing multimode to support system reconfiguration and fall-back solutions
Related concepts & solutions		• [D2.1] Operational mode management • [D3.1] Concepts for operational modes and change transitions • [D3.3] [D3.5] Solutions for operational modes and change management: Mode change coordination.

Related components	• Cloud tier:
	  ○ Performance and Security Monitoring Service: Provider of events and alarms to enable mode changes.
	• Edge tier:
	  ○ Operational Mode Coordinator: Propagation of operational modes to edge and devices.

Table 8: Operational mode coordinator in the cloud

5.1.1.5 Remote Updates Coordinator

In the following, we discuss a setting where the Remote Update Coordinator is in charge of ensuring that all clients run the software version as specified at the cloud. An end device owner can request an update from the cloud and has to approve to updates suggested by the cloud.

Concrete update mechanisms range from (a) full device control, i.e. the device requests the available versions from the cloud, selects the updates, approves the updates, and finally installs the updates to (b) full remote control, the cloud knows the state of the device, selects appropriate updates, approves the updates in the cloud and forces the installation on the devices without any intervention. In this section, we focus on the setting where the Remote Update Coordinator is in charge of ensuring that all clients run the software version as specified at the cloud.

For the use cases considered in TRANSACT, the Remote Update Coordinator aims to ensure all its clients (devices and edge servers) run the correct software version at all times. It keeps a representation of each client including what software versions it should be running, what we call its desired state. The Remote Update Coordinator running in the cloud tier oversees the overall management of the updates and is considered the main source of truth. A platform operator can decide a software update by interacting with the cloud Remote Update Coordinator to set a new version in the desired state of the device or edge server. The cloud and edge Remote Update Coordinators must always keep their data in sync, so a device receives the same information no matter who it can contact, the edge service or the cloud service. Moreover, the Remote Update Coordinator, both in the cloud and edge tiers, may also be responsible for storing and serving the software versions themselves. The Remote Update Client running on the device will periodically check what is its desired state by making a request to a Remote Update Coordinator. If the desired state it receives differs from its current state, it must perform the software updates that will make it reach the desired state.

The Remote Update Coordinator component ensures that devices have access to the latest software versions approved by Operators. It serves as the source of truth and provides a flexible update mechanism, playing a vital role in maintaining the system's integrity. It allows platform Operators to plan, schedule and launch updates, then monitor the update process and perform rollbacks if something goes wrong.

The overview and additional information are detailed in Table 9.

Role/functionality	• Controls the version of the software running in the clients (devices and edge servers) • Synchronize different coordinators at the edge tier

		• Execution phase (E) in the MAPE cycle. • Optional Planning phase (P) on the MAPE cycle
Configuration	Mandatory	• Software Release channels and versions for each software element • Setup synchronization with Edge tier Remote Update Coordinators
	Optional	• Maintenance windows for updates • Rollout and rollback schedules and strategies • Select working mode (push/polling)
Input interfaces	Mandatory	• Software versions, channels, schedules and strategies • Current state of the clients
	Optional	• Remote Update Coordinators in edge tier
Output interfaces	Mandatory	• Send desired state to clients via push or polling mechanism • Reporting
	Optional	• Synchronize state with coordinators in the edge tier • Serve software updates (artifact repository)
Endpoints		• REST API for managing software versions, channels, schedules and strategies • REST API for managing Edge tier Remote Update Coordinators • REST API for clients to retrieve their desired state • REST API for downloading software updates (artifacts) • REST API for reporting
Covered requirements		• [TSR-37] Offer support for software updates • [TSR-SP-4] Independent releasing of modular software parts
Related concepts & solutions		• [D1.2] Technical requirements and TRANSACT transition methodology commonalities • [D2.1] Software updates in safety-critical systems • [D3.1] Concepts for safe and secure modular updates • [D4.2] Strategies for continuous updating and independent releasing

Related components	• Edge tier:
	  ○ Remote Update Coordinator: keep all edge servers in sync
	• Device tier:
	  ○ Remote Update Client: propagation of new desired software state to devices

Table 9: Remote updates coordinator in the cloud

5.1.1.6 Federated Data services and Communications

The Federated Data services and Communications component in the cloud tier plays a crucial role in enabling seamless and efficient data management, communication and collaboration within the device-edge-cloud continuum. This component aims to collect and publish data to be shared between various edge nodes and cloud services, ensuring that data from different sources is transformed into a common format using standardised protocols, facilitating interoperability and consistency.

The communication service must offer mechanisms to manage the quality of communication services, optimising factors such as latency, bandwidth and reliability based on application requirements. Additionally, it must implement mechanisms to detect and recover from communication failures, ensuring that the unavailability of a particular cloud service or edge node does not result in a complete interruption of communication, minimising downtime and ensuring the resilience of the system.

Usually, due to the highest computational power (as compared to edge and device tiers, excluding dedicated accelerators), the most power-demanding computing happens in the cloud tier, therefore data services encompass also various processing algorithms. Some of them could (and preferably should) be delegated to edge tier, yet the computational capabilities of such nodes should be considered.

The overview and additional information are detailed in Table 10.

Role/functionality		• Manage data exchange and communications between edge nodes and cloud services. • Establish and maintain connections between cloud services and edge nodes. • Manage the QoS in communications. • Failure detection and recovery in communications. • This component is related to execution phase (E) in MAPE cycle.

Configuration	Mandatory	• At configuration time:
		  ○ Specification of communication protocols and security settings.
		  ○ Specification of QoS requirements for communications.
		  ○ Configuration of fault tolerance mechanisms, such as redundancy and failover settings.

	Optional	• At configuration time: ○ Identification of allowed edge nodes and cloud services.
Input interfaces	Mandatory	• Configuration interfaces. • Data ingestion interfaces (data streams, messages, …) • Performances monitoring metrics to track the health and efficiency of the communication system.
	Optional	• N/A
Output interfaces	Mandatory	• Data output interfaces, that are responsible of transmitting processed data to edge nodes.
	Optional	• Notification and alert interfaces to alert on critical events
Endpoints		• Configuration endpoint • Data ingestion endpoint
Covered requirements		• [TSR-1] Protection of data involved in transfer or transmissions. • [TSR-14 to 16] Protection against threads. • [TSR-17] Ensure security requirements of confidentiality, authenticity and integrity. • [TSR-18 to 20] Ensure data privacy in cloud. • [TSR-26 to 27] Ensure cloud high availability. • [TSR-28] Data encryption on the continuum. • [TSR-54] Resource availability and lower response latency. • [TSR-SP-1] Ensure data integrity across continuum.
Related concepts & solutions		• [D3.2] Concepts to centralized machine learning with decentralized data. • [D3.2] Security and privacy requirements and patterns for the healthcare DICOM data and applications. • [D3.2] Safety and privacy of off-the-shelf components. • [D3.4] [D3.6] Secure communication-based solutions. • [D3.4] [D3.6] Data communication security solutions.

Related components	• Cloud tier:
	  ○ Performance and Security Monitoring services: Provide metrics to ensure QoS in communications.
	  ○ Access, Privacy and Identity Services: Ensure communication permissions.
	• Edge tier:
	  ○ Data Services and Communications: Data exchange with edge nodes.

Table 10: Federated data services and communications in the cloud

5.1.2 Value-Added services and functions

The cloud Value-Added services enhance the system capabilities of the mission-critical and non-critical functions in the cloud. These value-added components encapsulate the high-level services in charge of providing the device-edge-cloud continuum with specific functionalities, such as data analysis, anomaly prediction or decision-making, allowing the system to be improved.

5.1.2.1 Big Data as a Service (BDaaS)

The Big Data as a Service (BDaaS) component is deployed on the cloud tier as the means for facilitating the execution of data analytics processes (and related tasks). Candidate services and applications that make use of this component are those whose computing power requirements, due to the massive volume of the data sets they must process and analyse, are so demanding that they are not suitable for their allocation in the local premises of the edge tier.

The BDaaS automates the deployment and management of the technologies utilised to the execution of tasks devoted to processing a massive volume of data, such as those that fall under the scope of Big Data Analytics, also providing a simple way for setting their configuration. Since many of these technologies are distributed, their deployment isn't as easy as launching a centralized process in a single machine. It usually requires deploying many processes across a cluster of interconnected machines, whose roles and network connections also need to be properly configured. Moreover, the horizontal scalability factor (the number of machines currently deployed for carrying out the task) also may dynamically vary depending on the system's needs and the availability of resources (which is theoretically nearly unlimited on the cloud tier), which is known as elasticity.

The automated deployment of technologies performed the BDaaS also manages and abstracts its users from the underlying complexity. The technologies being deployed are managed as services, exposing concrete endpoints and easing their integration with other services and their integration with the running applications that take advantage of data analytics techniques. Usually, the service demanding the execution of tasks in the BDaaS component will originate in the AI & ML & Analytics component, that will delegate the execution of jobs and algorithms on the technologies provisioned by the BDaaS, supplying them from the data collected from the Data Manager.

The overview and additional information are detailed in Table 11.

Role/functionality		• Automate the deployment and configuration of technologies enabling the execution of data analytics and related tasks. • Monitor and scale the deployment along the execution of jobs, according to their current workload. • Analysis phase (A) in the MAPE cycle. • Optional Planning phase (P) in the MAPE cycle.

Configuration	Mandatory	• At configuration time:
		  ○ Technology to be deployed
		  ○ Deployment settings
		• At runtime:
		  ○ Data set to be processed/analysed

	Optional	• At runtime: ○ Job execution parameters
Input interfaces	Mandatory	• Request of the execution of data analytics tasks by other services, usually the AI & ML & Analytics component
	Optional	• N/A
Output interfaces	Mandatory	• Data output, stored on cloud storage or returned to the calling process
	Optional	• N/A
Endpoints		• Technology deployment request • Execution of a job, requested on an endpoint allocated for a technology having been deployed.
Covered requirements		• [TSR-SP-3] Elastic horizontal scalability of provisioned computational resources, attending to current workload • [TSR-61] AI framework • [TSR-64] Machine Learning pipeline • [TSR-65] Automated Machine Learning pipeline • [TSR-78] Split datasets for training predictive models • [TSR-80] Algorithms for model training • [TSR-82] Parallelization within model training • [TSR-90] Selection of the model scoring the highest performance • [TSR-100] Evaluation of the model on a subset of the input data

	• [TSR-110] Capabilities of the components in the Machine Learning pipeline • [TSR-111] Supervised learning algorithms • [TSR-112] Unsupervised learning algorithms • [TSR-113] Semi-supervised learning algorithms • [TSR-114] Reinforcement learning algorithms • [TSR-116] Support to classification analysis • [TSR-117] Allocation of the component on the cloud tier
Related concepts & solutions	• [D2.1] Big Data as a Service (BDaaS) • [D4.1] Integration of AI services into distributed SC CPS solutions

Related components	• Cloud tier:
	  ○ AI & ML & Analytics: Requests the execution of tasks on deployed technologies.
	  ○ Data Manager: Supplies the data that will feed the jobs, tasks and algorithms to execute.
	• Edge tier:
	  ○ AI & ML & Analytics: Requests the execution of data analytics tasks, offloading it to the cloud tier, where extensive computing power is available.

Table 11: Bid Data as a service in the cloud

5.1.2.2 Data Manager

The Data Manager component manages the storage, protection and access to data assets in the cloud tier. It ensures that data is validated and fully accessible to services and applications when needed.

This component receives data from the edge and device tiers, and conveniently stores them for being provided to other components that may use them, such as the AI & ML & Analytics and the BDaaS components, or high-level applications. Core components in the cloud tier can also act as data sources, such as the Auditing and Monitoring services. The transmission of data allocated by the *Data Manager* to other components (especially when they are placed on different tiers) will take place in the context of the Federated Data Services & Communications component.

It is important to remark, especially when data can be used to train AI models or to perform analyses whose results can lead to take business decisions, that data quality is a dimension that must be considered. Low quality data can result in poor AI models and/or misleading analysis conclusions which, at the same time, lead to poor business decisions. The data manager component is the best suited component in the TRANSACT architecture to host data quality functionalities. These can inspect or evaluate different dimensions of the data, from a business or statistical perspective providing insights of whether data is valid as an input for further training or analysis, or whether corrective actions must be undertaken in the ingestion or collection

process to improve it. Examples of the potential indicators could be completeness, precision, timeliness, statistic distributions or histograms or, from a business-aware perspective, defining specific KPIs that can trigger an alarm when data quality deteriorates.

Therefore, the purpose of this component is to safely store and make available data from other components across the three tiers, enabling and easing their exploitation by various Value-Added services and functions, as well as monitoring its quality.

The overview and additional information are detailed in Table 12.

Role/functionality		• Manage the storage, protection and access to data in the cloud tier. • Ensure that data is validated and fully accessible to services and applications that require them. • Monitor data quality. • Knowledge Base phase (K) in the MAPE-K self-adaptive cycle.

Configuration	Mandatory	• At configuration time:
		  ○ Storage systems
		  ○ Storage resources (volumes)
		  ○ Data ingestion mechanisms
		• At runtime:
		  ○ Data sources and sinks

	Optional	• At configuration time:
		  ○ Relevant data quality KPIs to evaluate
		• At runtime:
		  ○ Access control policies

Input interfaces	Mandatory	• Data sources: components and services providing data
	Optional	• N/A
Output interfaces	Mandatory	• Data sinks: interfaces through which components, services and applications can request data from the Data Manager
	Optional	• Data quality results
Endpoints		• Submission of new data • Request of stored data
Covered requirements		• [TSR-1] Cloud Data Security • [TSR-15] Data Poisoning • [TSR-18] Data handling and security

	• [TSR-19] Data protection • [TSR-19] Cloud requirements • [TSR-28] Back-up and recovery • [TSR-29] Data availability • [TSR-43] Cost-efficient cloud storage • [TSR-44] Storage for datasets intended for AI training algorithms • [TSR-67] ML pipeline: Data storage • [TSR-68, 70, 71] ML pipeline: Data collection and ingestion • [TSR-SP-1] Data integrity
Related concepts & solutions	• [D3.3] [D3.5] Data integrity • [D3.4] [D3.6] Secure cloud-based infrastructure • [D4.1] Cloud as an enabler for AI tasks requiring greater storage capabilities than the other tiers can provide • [D4.1] Combination of data sources from different services

Related components	• Cloud tier:
	  ○ AI & ML & Analytics: Feeds from data in the Data Manager to process and analyse it.
	  ○ Big Data as a Service (BDaaS): Feeds from data in the Data Manager, potentially being in turn instrumentalized by the AI & ML & Analytics component.
	  ○ Federated Data Services & Communications: Data from and to other components and services is transmitted via this Core function, especially when the component providing or requesting data is on a different tier.
	  ○ Performance and Security Monitoring Service: Data from monitoring services can be stored in the Data Manager for further analysis and decision-making support.
	  ○ Auditing Service: Audit data can be stored in the Data Manager for further analysis.
	  ○ Access, Privacy & Identity Service: Access records and related data can be stored in the Data Manager for being checked later.
	• Edge tier:
	  ○ Data Manager: Data from the homonymous component in the edge tier can be sent to the cloud, where larger storage resources are available. Data from the cloud can also be sent

	back to the edge in case of being required by processes making use of the Data Manager in the edge tier. ○ AI & ML & Analytics: Definite data from the cloud can be sent to the edge's Data Manager for being locally processed and analysed. ○ Data Services & Communications: Data from services in the edge can be sent to the cloud for proper storage. • Device tier: ○ Data Services & Communications: Data from devices can be sent to the cloud tier for proper storage.

Table 12: Data manager in the cloud

5.1.2.3 AI and ML and Analytics

The AI, ML and Analytics components are responsible for handling operations related to extracting valuable insight from provided data and produce new knowledge through various forms of analysis. These components can be located in the edge and/or cloud tier of the system and will provide advanced capabilities such as natural language processing, image recognition, predictive analytics, and anomaly detection. The components are classified as either traditional Analytics or AI/ML based.

Traditional analytics typically involves the use of statistical techniques to analyse data and identify patterns or trends. This approach relies on human analysts to define the parameters of the analysis, interpret the results, and make decisions based on those results. Traditional analytics is often used to identify historical trends, understand customer behaviour, or monitor business performance.

On the other hand, AI-based analytics uses machine learning algorithms to automatically identify patterns and make predictions based on large datasets. These algorithms are trained on historical data and can identify complex relationships and trends that might be difficult for humans to detect. AI-based analytics is often used to make predictions about future outcomes, optimize business processes, or automate decision-making.

At the heart of an AI/ML component lies a complex algorithm that has been trained on a large dataset using advanced machine learning techniques. This training process typically involves feeding the algorithm with vast amounts of data and teaching it how to recognize patterns and make predictions based on the input data. The resulting model is then used to perform specific tasks, such as classifying images, translating languages, or predicting future outcomes based on historical data.

The overview and additional information are detailed in Table 13.

Role/functionality	• Process incoming data and update learning model (ML) • Host a manually created analytics model • Analyse new data by applying learning or analytics model • Orchestrate clients and data exchange • Manage data usage and access

		• Store and visualize results
		• Related Phases von MAPE cycle
		  ○ Monitor: e.g.: Data drift, model results
		  ○ Analyse: e.g., anomalies
		  ○ Plan: e.g., retraining
		  ○ Execute: e.g., inference, model update

Configuration	Mandatory	• Input data sources • Analytics models • Connection to clients • Data storage
	Optional	• Remote update of models running on edge
Input interfaces	Mandatory	• Learning data streams or packages • Operational data stream
	Optional	• Log files • Heart beat • Trained model or weights
Output interfaces	Mandatory	• Analytics results • Advisory output
	Optional	• Trigger for retraining • Parameters for client operation • Newly trained models or weights
Endpoints		• Edge: Message Broker, API (gRPC, REST) • Cloud Frontend for user interface and visualization
Covered requirements		• [TSR-15] Data Poisoning • [TSR-18] Data handling and security • [TSR-19] Data Protection • [TSR-20] Organizational policies and trust • [TSR-23] Cloud requirements • [TSR-24] Legal Compliance • [TSR-31] Timely data analysis

	• [TSR-32] Timely data analysis • [TSR-33] Time constraints • [TSR-37] Updates • [TSR-39] Anomaly detection • [TSR-40] Message Broker • [TSR-43] Storage • [TSR-44] Storage • [TSR-45] Scalability • [TSR-57, TSR-131] Commercial intelligence cloud services • [TSR-SP-1] Data Integrity • [TSR-ARCH-1] Protection of system resources • [TSR-ARCH-3] New Models • [TSR-61] AI frameworks • [TSR-63] Cascading framework for artificial intelligence • [TSR-64 to 110] Machine Learning Pipeline • [TSR-111 to 117, 121, 122] Support different algorithms
Related concepts & solutions	• [D3.1] Application service level agreements for AI-model based distributed CPS solutions • [D3.1] Concepts for ensuring data integrity (across the device-edge-cloud continuum) • [D3.2] Anonymization: Prevent Personal Data leak • [D3.2] Centralized Machine Learning with Decentralized Data • [D3.2] User and entity behavioural analytics (UEBA) concept for cloud security • [D3.2] Cloud detection & response (Cloud DR) orchestration for multiple clouds • [D3.3] [D3.5] Solutions for ensuring data integrity • [D3.4] [D3.6] Privacy preservation by leveraging the concept of federated learning • [D3.4] [D3.6] User behavioural massed ML models for anomalous activities in cloud environments • [D4.1] Combining on-device intelligence or autonomy with edge or cloud-based AI services to ensure safe handling of safety-critical situations

	• [D4.1] Defining the Level of safety-criticality to which AI services can be used, and the type of safeguards or degraded modes of operation that are needed • [D4.1] Defining what novel services can be created by the combination of data from various local systems that can exploit global insights beyond the local CPS only viewpoint • [D4.1] Updating AI services incorporating new or updated functionality (gradually) available to ensure early field feedback yet maintain safe operation • [D4.2] Strategies for retraining and releasing AI models • [D4.2] AI monitoring to detriment if/when to update

Related components	• Cloud tier:
	  ○ Mission Critical Function: If results are relevant for Function.
	  ○ Non-Critical Function: If results are relevant for Function.
	  ○ New Services & Marketplace: Algorithms can be provided by third party.
	  ○ BDaaS: Analytics can be based on Big Data
	  ○ Data Manager: Data handling must be managed.
	  ○ New Services: Analytics enables New Services
	  ○ Performance and Security Monitoring Service: Supervision of results quality
	  ○ Auditing: Ensures traceability of parameter and model updates, and any other manual changes.
	  ○ Access, Privacy & Identity Services: Crucial as sensitive data may be processed.
	  ○ Remote Updates Coordinator: Updating models and weights
	  ○ Federated Data Services & Comms: Enables third-party integration and communication with edge clients.
	• Edge tier:
	  ○ AI & ML & Analytics: Hosts client, algorithm or manages data exchange

Table 13: AI & ML & Analytics in the cloud

5.1.2.4 New services & Marketplace

The New services & Marketplace components enable the integration of new added-value services. These services, provided by third-party vendors to TRANSACT, aim to be used by applications u other components of the architecture. The services provided by this component are available via TRANSACT's own Marketplace service. This marketplace enables offering and integrating new solutions (applications, services, algorithms, AI models, etc.) not only by the systems builders themselves but also by third-party vendors.

These instances of the components are deployed on the cloud tier. Although the "New services" component is conceptually identical to its edge tier counterpart, the technologies it offers here match services that, due to their nature, purpose and computing infrastructure requirements, make sense to be deployed on the cloud tier of TRANSACT's architecture.

The overview and additional information are detailed in Table 14.

Role/functionality		• Enable the acquisition and integration of new technologies (possibly developed by third-party vendors) as services on the cloud tier, for their consumption by TRANSACT's applications, services and components. • N/A phase on the MAPE cycle
Configuration	Mandatory	• Listing of the technologies available • Executable/package of the technology to be acquired • Mechanism to integrate new third-party technologies via the marketplace service
	Optional	• Category and description of the technology to be acquired • Specification of the interfaces in the technology to be acquired
Input interfaces	Mandatory	• Request point for technologies available via the marketplace. • Interface for submitting new third-party as services to the marketplace.
	Optional	• N/A
Output interfaces	Mandatory	• Supply of packaged technologies, served by request.
	Optional	• N/A
Endpoints		• Endpoints exposed by the technology to be acquired, in case it operates as a server that attends and processes requests by clients.
Covered requirements		• [TSR-ARCH-3] Inclusion of new services • [TSR-88] Inclusion of a library providing multiple evaluators • [TSR-107] Availability of multiple monitoring tools • [TSR-115] Support to multiple machine learning algorithms

	• [TSR-131] Commercial machine learning frameworks
Related concepts & solutions	• [D1.1] Use cases specification and end user requirements • [D4.1] Integration of AI services • [D4.3] Fast development of innovation using distributed solutions • [D4.4] New business models enabled by edge and cloud services
Related components	• Edge tier: ○ New Services: Integration of new third-party technologies as services in TRANSACT.

Table 14: New services & Marketplace in the cloud

5.2 Edge tier

The edge tier (Figure 5) hosts the domain-specific functions that must be deployed on-premises due to low latency and high availability requirements, as well as the need for greater security and data privacy. These functions are implemented on top of the TRANSACT edge Core services and functions (Section 5.2.1) and the TRANSACT edge Value-Added services and functions (Section 5.2.2).

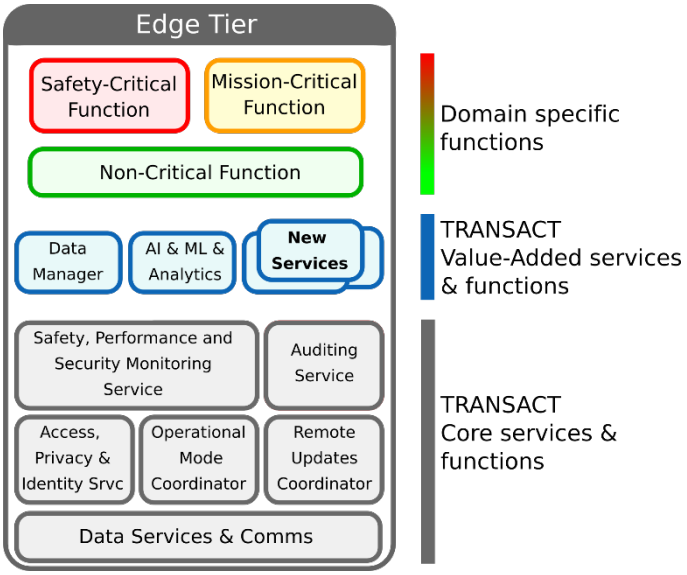

Figure 5: Static view of the edge tier

5.2.1 Core services and functions

The edge Core services support the safety-critical, mission-critical and non-critical functions at the edge tier. These core components encapsulate the safety and performance, and security and privacy edge-level functionalities of the system, which are responsible for providing the necessary support to monitor, coordinate and configure the devices and synchronize with cloud services.

5.2.1.1 Safety, performance and security monitoring services

Monitoring services at the edge comprise the monitors in charge of collecting safety, performance and security indicators.

The aim of the performance monitoring service is to monitor performance indicators from the various components and systems on the edge tier. This might include latency, throughput but also resource aspects. such as CPU, memory, and disk usage.

The aim of the security monitoring service is to detect and prevent security violations. The preventive part will continuously monitor all components and their compliance with the security policies that are defined. This service is responsible for measuring the indicators that facilitate detecting potential vulnerabilities or attacks and monitoring network traffic with devices to identify anomalies or malicious activity.

Safety monitoring service primarily deals with collecting data to facilitate the identification and prevention of potential hazards, risks and failures within a system. While these monitors are distinct, they often overlap in practice with performance and security monitors. For example, a significant performance degradation might impact the safety of a system, especially in critical applications where response time is crucial. Also, security breaches can introduce safety risks, in case of a manipulation of the system to cause harm.

In the use cases, a setting has been considered where these services will forward the events to the cloud tier for further analysis. Depending on the criticality of application and the sensitivity of the performance indicators, a local reaction to the measurements and a minimization of data transmission might be required.

The edge tier will also forward the events received from the monitoring services of the device tiers.

The overview and additional information are detailed in Table 15.

Role/functionality		• M in the MAPE cycle • Monitoring safety, performance and security indicators of different components
Configuration	Mandatory	• If KPIs are monitored though pull operations, the endpoints for the KPIs have to be configured
	Optional	• Criteria/ranges to specify/describe the monitoring of the KPI (e.g., SLOs, thresholds)
Input interfaces	Mandatory	• Performance KPIs of components (system/service KPIs). It may be required to push or pull the KPIs, depending on the scenario.
	Optional	• Security policies
Output interfaces	Mandatory	• The set of KPIs themselves
	Optional	• Predictions on KPIs values, e.g., regarding the crossing of a threshold • Outputs/alarms/events
Endpoints		• Receiving device-tier monitoring events
Covered requirements		• [TSR-2 to 12] The architecture should be protected against most attacks on edge computing infrastructures

	• [TSR-30] Architecture should be self-organising, that implies self-monitoring capabilities. • [TSR-32 to 33] Ensure architecture in near-real-time or real-time performance • [TSR-38] Constantly monitor available resources to detect software failures and aging in the architecture • [TSR-46] Cloud computing components in the architecture will enhance the performance by processing IoT data in the cloud • [TSR-55] The components in the architecture should be able to monitor events, resources and errors. • [TSR-56] The architecture should support the monitoring of the data transmission rate of the devices. • [TSR-58] The TRANSACT system architecture should support error detection. • [TSR-102 to 109] Monitoring and modelling systems
Related concepts & solutions	• [D3.1] Application service level agreements • [D3.1] [D3.3] [D3.5] Concepts and solutions for AI monitoring • [D3.1] [D3.3] [D3.5] Concepts and solutions for predictable performance • [D3.1] [D3.3] [D3.5] Concepts and solutions for health and safety monitoring • [D3.2] Runtime verification
Related components	• Auditing Services • Operational Mode Manager

Table 15: Safety, performance and security monitoring services at the edge

5.2.1.2 Auditing service

Auditing consists of examining and evaluating processes in terms of their level of computerisation and data processing, also of verifying controls in the processing of information and verifying the installation of security systems, as defined in auditing services at the cloud.

When a system is audited, the technical components of the system are analysed for vulnerabilities or threats that could have a negative impact on the system.

Auditing services can significantly enhance auditing processes by providing automation, efficiency, and improved accuracy in the monitor, analysis, and reporting of audit data. These services at edge can be responsible for registering the status of devices, such as detected failures, firmware updates and configuration changes; tracking suspicious network activity or any event related to a security incident.

It is necessary to highlight the importance of checking, not only the computerised equipment but also the controls applied and their use.

Edge technology may offer some advantages such the proximity (that helps to have better control over it), it works isolated from the rest of the network, reduce on latency and immediacy. All these characteristics conclude on an improvement in cybersecurity. Nonetheless, it is not out of harm's way. Whilst cloud computer shows great flexibility, being able to access to cloud, wherever and whenever, not all gadgets may be compatible with the Edge. Also, the implementation and maintenance of the Edge may be more laborious than a device architecture or Cloud.

When operating on edge, it is essential to take into consideration its location. Usually, the edge may not be accessible physically, so a remote secure connection is required. Also, the protocols for connection need to be safe. Furthermore, the edge needs to be protected physically, so it is not accessible. Therefore, it is necessary to invest resources in the maintenance of the edge.

The overview and additional information are detailed in Table 16.

Role/functionality		• Check safeguards. • Identify areas for improvement. • Revise the compliment of various applicable laws. • Registering status of devices (failures, updates, configuration changes) • Track anomalies in resource consumption. • Provide alerts for security incidents
Configuration	Mandatory	• The policies and proceedings implemented must comply with the applicable regulation. • These policies and proceedings need to be correctly implemented in the structure of the company.
	Optional	• It is advisable to involve all the areas in the process of auditing.
Input interfaces	Mandatory	• Coordination between departments and auditor. • Information on the current status. • Set a level of auditing.
	Optional	• N/A
Output interfaces	Mandatory	• Reporting. • Analytic results. • Advisory output.
	Optional	• N/A
Endpoints		• Interchange proceedings and policies.

Covered requirements	• [TSR-13] Protection against authentication and authorization attacks • [TSR-18] Mitigation plan in case of system is compromised and how to participate in the investigation and prosecution. • [TSR-19] Data is not shared to unauthorised third parties.
Related concepts & solutions	• [D3.1] Requirements and concepts for end-to-end safety and performance assurance • [D3.2] [D3.4] [D3.6] Security and privacy concepts for secure remote driving operation • [D3.2] Security and privacy requirements and patterns for the healthcare DICOM data and applications
Related components	• Safety, Performance and Security Monitoring Services. • Operational Mode Coordinator. • Access, Privacy and Identity Services.

Table 16: Auditing service at the edge

5.2.1.3 Access, privacy and identity services

In the edge tier, we focus on the privacy related services. The identity and access services are already covered by cloud tier description.

5.2.1.3.1 Identification and access service

The identification and access (I&A) functionality enables the interaction with cloud-based I&A services and enables the local/offline I&A management with devices. Hence, the edge can have a dual role, acting as client in one case (see device section), and as authorization server (see cloud section) in the other.

The main difference is that, being on the edge, the specifics of the edge-device relation can trigger additional capabilities (e.g. other communication protocols, proximity relation etc). Below we focus on those differences.

The overview and additional information are detailed in Table 17.

Role/functionality	• Support KPI infrastructure for key signing • Onboarding of devices by token exchange (including safeguards, e.g. ensuring physical presence of the device) • Act as authorization service to support the local/offline storage

Table 17: Identification and access service at the edge

5.2.1.3.2 Privacy service

The privacy functionality enables de-identification of Personal Identifiable Information. Deployment can vary between device/edge/cloud, depending on the legislation by which private information is (or is not) allowed to be transferred across legal boundaries. For example, legislation that requires medical information to be pseudonymized once it leaves the (legal) premise of the hospital.

The overview and additional information are detailed in Table 18.

Role/functionality		**Privacy functionality.**
		• De-identification of data in various formats
		  ○ Anonymization
		  ○ Pseudonymization
		• Support of de-identification profiles where standards apply (e.g. medical domain – DICOM/FHIR)
		  ○ Clear, remove, change (hash, date-shift, ...)
		• Re-identification to support dataflow back into legal boundary that is allowed to contain private information (data owner) – for pseudonymized data only.
		• Support of filtering non-textual data (e.g. images, video, ...) means of selectable profile (e.g. complete image blurring, object detection blur, …)
		• Encryption / Decryption key-management
		• Support customized processing / processing actions
		• Pipes & job-management
		• Batch processing (e.g. discrete datasets) and stream processing (e.g. continuous signals)
		• MAPE: Plan (Setup of pipeline/jobs/configuration) & Execute (processing of the data)

Configuration	Mandatory	• Source/sink, profile, pipe, batch/stream
	Optional	• Custom processor
Input interfaces	Mandatory	• Input data • Profile configuration
	Optional	• N/A
Output interfaces	Mandatory	• (Result) Batch Data Access, (Result) Data Stream
	Optional	• Job status

Endpoints	• API – REST / CLI, WebPage (config)
Covered requirements	• [TSR-9] (partially)
Related concepts & solutions	• [D3.2] Anonymization, • [D3.4] [D3.6] Data Communication Security (partially)
Related components	• Data Manager

Table 18: Privacy service at the edge

5.2.1.4 Operational mode coordinator

The Operational Mode Coordinator in the edge tier aims to manage the transitions between operational modes at the edge level. The responsibilities of this component comprise both, propagating mode change requests coming from the Coordinator in the cloud (see Section 5.1.1.4) and synchronising Operational Mode Managers in the device tier (see Section 5.3.1.4). Each operational mode defines a behaviour or configuration that the system has to be in each phase of the execution, including both, normal operation and failure mitigation mechanisms. A mode transition corresponds to a change in the system behaviour, responding to events or alarms provided by other services.

The Operational Mode Coordinator in the edge works closely together with the Operational Mode Coordinator in the cloud. Both coordinators in the cloud and the edge tier must be synchronized, propagating the global operational mode and ensuring it is reached in all of them when a change is requested. In addition, events (or alarms) generated by other services, such as monitoring services, have to be considered to perform isolated transitions at the edge level, usually with the aim of mitigating a localized failure.

Regarding the synchronization method with devices, the Operational Mode Coordinator in the edge tier must be responsible for propagating requests to perform the mode change in the devices connected to it. It is expected that devices could be legacy and not be designed to fit into this architecture. For this reason, the Coordinator in the edge must provide the adaptation layer to synchronize each of the connected devices, ensuring application consistency.

The overview and additional information are detailed in Table 19.

Role/functionality		• Manage the transitions between operational modes at the edge level. • Synchronize Operational Mode Managers in the device tier • Execution phase (E) in the MAPE cycle. • Optional Planning phase (P) on the MAPE cycle.
Configuration	Mandatory	• At configuration time: ○ Local operational modes at edge level if these are different from the modes at system level
	Optional	• At configuration time:

		○ Operational modes at system level provided by Coordinator in the cloud. ○ Allowed mode transitions. ○ Additional rules relating events/alerts and local transitions (for isolated fault mitigation purposes at edge level). ○ Device mode adaptation in case of legacy devices.
Input interfaces	Mandatory	• Mode Change Requests coming from Coordinator in the cloud tier. • Events/alerts that enable local mode changes
	Optional	• Status of the devices regarding the current operational mode.
Output interfaces	Mandatory	• Mode Change Requests to device tier
	Optional	• Current status regarding the current operational mode to be sent to Coordinator in the cloud
Endpoints		• Mode Change Request • Status • Other communication may take place through publish/subscribe mechanisms
Covered requirements		• [TSR-60] Multimode approach to support online error recovery and repair • [TSR-SP-2] Allowing multimode to support system reconfiguration and fall-back solutions
Related concepts and solutions		• [D2.1] Operational mode management • [D3.1] Concepts for operational modes and change transitions • [D3.3] [D3.5] Solutions for operational modes and change management: Mode change coordination.
Related components		• Cloud tier: ○ Operational Mode Coordinator: Mode change requests • Edge tier: ○ Safety, Performance and Security Monitoring Service: Provider of events and alarms to enable local mode changes. • Device tier: ○ Operational Mode Manager: Propagation of operational modes to devices.

Table 19: Operational mode coordinator at the edge

5.2.1.5 Remote updates coordinator

The Remote Update Coordinator aims to ensure all devices run the correct software version at all times. It keeps a representation of each device including what software versions it should be running, what we call its desired state. The Remote Update Coordinator running in the edge tier oversees managing the updates of the devices that are connected to it. Its source of truth is the Remote Update Coordinator running in the cloud tier, so it must permanently keep in sync with it. When a platform operator decides for a software update by interacting with the cloud Remote Update Coordinator to set a new version as the desired state of the device, the change is automatically propagated to all the Remote Update Coordinators running in the edge tiers. This way, a device will receive the same information no matter what Remote Update Coordinator it can contact, the edge service or the cloud service. Moreover, the Remote Update Coordinator, both in the cloud and edge tiers, may also be responsible for storing and serving the software versions themselves. The Remote Update Client running on the device will periodically check what is its desired state by making a request to a Remote Update Coordinator. If the desired state it receives differs from its current state, it must perform the software updates that will make it reach the desired state.

The Remote Update Coordinator component ensures that devices have access to the latest software versions approved by Operators. It serves as the source of truth and provides a flexible update mechanism, playing a vital role in maintaining the system's integrity. It allows platform Operators to plan, schedule and launch updates, then monitor the update process and perform rollbacks if something goes wrong.

The overview and additional information are detailed in Table 20.

Role/functionality		• Controls the version of the software running in the devices • Synchronize with the coordinators in the cloud tier • Execution phase (E) in the MAPE cycle • Optional Planning phase (P) on the MAPE cycle
Configuration	Mandatory	• Software Release channels and versions for each software element • Setup synchronization with cloud tier Remote Update Coordinator
	Optional	• Maintenance windows for updates • Rollout and rollback schedules and strategies • Select working mode (push / pull)
Input interfaces	Mandatory	• Software versions, channels, schedules and strategies • Current state of the devices • Remote Update Coordinators in cloud tier
	Optional	• N/A
Output interfaces	Mandatory	• Send desired state to devices via push or pull mechanism • Reporting
	Optional	• Synchronize state with coordinators in the edge tier

	• Serving of software updates (artifact repository)
Endpoints	• REST API for managing software versions, channels, schedules and strategies • REST API for managing cloud tier Remote Update Coordinators • REST API for devices to retrieve their desired state • REST API for downloading software updates (artifacts) • REST API for reporting
Covered requirements	• [TSR-37]: Offer support for software updates • [TSR-SP-4]: Independent releasing of modular software parts
Related concepts & solutions	• [D1.2] Technical requirements and TRANSACT transition methodology commonalities • [D2.1] Software updates in safety-critical systems • [D3.1] Concepts for safe and secure modular updates • [D4.2] Strategies for continuous updating and independent releasing

Related components	• Cloud tier:
	  ○ Remote Update Coordinator: keeps its own software database in sync with cloud
	• Device tier:
	  ○ Remote Update Client: propagation of new desired software state to devices

Table 20: Remote updates coordinator at the edge

5.2.1.6 Data services and communications

The Data services and Communications component at the edge tier is responsible for managing data exchange and communications between the different devices connected to the edge and with the services provided by the cloud. One of the main objectives is to facilitate devices to leverage the services provided by cloud and edge tiers, encapsulating the communication processes. The other one is to facilitate cloud and edge running algorithms to collect and aggregate data provided by the different devices. Ensuring data transformation from different sources to a common and compact format using standardised protocols is critical due to the heterogeneity of the devices, with the aim of facilitating interoperability and consistency between the different services and resources.

In the same way as in the cloud tier, the communication service at the edge must offer mechanisms to manage the quality of communication services, optimising factors such as latency, bandwidth and reliability based on application requirements. Additionally, it must implement mechanisms to detect and recover from communication failures, ensuring that the unavailability of a particular connected device does not result in a complete interruption of communication, minimising downtime and ensuring the resilience of the system.

The overview and additional information are detailed in Table 21.

Role/functionality		• Manage data exchange and communications between devices and edge and cloud services. • Manage connection with different devices. • Establish and maintain connections with cloud tier. • Manage the QoS in communications. • Failure detection and recovery in communications. • This component is related to execution phase (E) in MAPE cycle.

Configuration	Mandatory	• At configuration time:
		  ○ Specification of communication protocols and security settings.
		  ○ Specification of QoS requirements for communications.
		  ○ Configuration of fault tolerance mechanisms, such as redundancy and failover settings.

	Optional	• At configuration time: ○ Identification of allowed devices and edge services.
Input interfaces	Mandatory	• Configuration interfaces. • Data ingestion interfaces (data streams, messages, …) • Performances monitoring metrics to track the health and efficiency of the communication system.
	Optional	• N/A
Output interfaces	Mandatory	• Data output interfaces, that are responsible of transmitting processed data to edge nodes.
	Optional	• Notification and alert interfaces to alert on critical events
Endpoints		• Configuration endpoint • Data ingestion endpoint
Covered requirements		• [TSR-2 to 12] Protection against attacks on edge computing. • [TSR-14 to 16] Protection against threads. • [TSR-17] Ensure security requirements of confidentiality, authenticity and integrity. • [TSR-28] Data encryption on the continuum. • [TSR-31] Usage of adequate storage mechanism and effective data-transmission protocols in the CPS network.

	• [TSR-40] Usage of message brokers with large number of devices. • [TSR-42] Availability of bandwidth to deal with IoT app requirements. • [TSR-45] Optimization of bandwidth according to IoT demands. • [TSR-54] Resource availability and lower response latency. • [TSR-SP-1] Ensure data integrity across continuum.
Related concepts & solutions	• [D3.2] Concepts to centralized machine learning with decentralized data. • [D3.2] Security and privacy requirements and patterns for the healthcare DICOM data and applications. • [D3.4] [D3.6] Secure communication-based solutions. • [D3.4] [D3.6] Data communication security solutions.

Related components	• Cloud tier:
	  ○ Federated Data Services and Communications: Data exchange with cloud services.
	• Edge tier:
	  ○ Performance and Security Monitoring Services: Provide metrics to ensure QoS in communications.
	  ○ Access, Privacy and Identity Services: Ensure communication permissions.
	• Device tier:
	  ○ Data Services and Communications: Data exchange with devices.

Table 21: Data services and communications at the edge

5.2.2 Value-Added services and functions

The edge Value-Added services enhance the system capabilities of the safety-critical, mission-critical and non-critical functions at the edge. These value-added components encapsulate the high-level services in charge of providing specific on-premises functionality with low latency and high availability requirements, as well as greater security and data privacy than services provided by the cloud tier.

5.2.2.1 Data manager

The Data Manager component manages the storage, protection and access to data assets in the edge tier. It ensures that data is validated and fully accessible to services and applications when needed.

This component receives data from the device tier, which get conveniently stored for being provided to other services and applications that may use them, such as the AI & ML & Analytics component. Data can be also sent to the cloud, where larger storage resources are available, as well as being supplied back from the cloud tier to be stored in the edge in case low-latency access to those data is required. The transmission of data

allocated by the Data Manager to other components (especially when they are placed on different tiers) will take place in the context of the Data Services & Communications component. The same remarks observed in the cloud tier regarding data quality apply in this component. However, if the data manager is present in both edge and cloud tiers, the data quality analysis could be performed in only one of them.

Therefore, the purpose of this component is to safely store and make available data from end-devices and other components on the edge tier, enabling and easing their exploitation by various Value-Added services and functions, as well as (potentially) monitoring its quality.

The overview and additional information are detailed in Table 22.

Role/functionality		• Manage the storage, protection and access to data in the edge tier. • Ensure that data is validated and fully accessible to services and applications that require them. • Monitor data quality. • Knowledge Base phase (K) in the MAPE-K self-adaptive cycle.

Configuration	Mandatory	• At configuration time:
		  ○ Storage systems
		  ○ Storage resources (volumes)
		  ○ Data ingestion mechanisms
		• At runtime:
		  ○ Data sources and sinks

	Optional	• At configuration time:
		  ○ Relevant data quality KPIs to evaluate
		• At runtime:
		  ○ Access control policies

Input interfaces	Mandatory	• Data sources: components and services providing data
	Optional	• N/A
Output interfaces	Mandatory	• Data sinks: interfaces through which components, services and applications can request data from the Data Manager
	Optional	• Data quality results
Endpoints		• Submission of new data • Request of stored data
Covered requirements		• [TSR-1] Cloud Data Security • [TSR-9] Edge Computing Side Channel Attacks

	• [TSR-15] Data Poisoning • [TSR-19] Data Protection • [TSR-28] Back-up and recovery • [TSR-29] Data availability • [TSR-43] Cost-efficient storage • [TSR-67] ML pipeline: Data storage • [TSR-68, 70, 71] ML pipeline: Data collection and ingestion • [TSR-SP-1] Data integrity
Related concepts & solutions	• [D3.3] [D3.5] Data integrity • [D3.4] [D3.6] Secure cloud-based infrastructure (edge is actually a resource-constrained cloud on local premises) • [D4.1] Cloud (edge) as an enabler for AI tasks requiring greater storage capabilities than CPS can provide • [D4.1] Combination of data sources from different services

Related components	• Cloud tier:
	  ○ Data Manager: Data can be sent to the homonymous component in the cloud tier, where larger storage resources are available. Data from the cloud can also sent back to the Data Manager in the edge in case of being required by processes in this tier.
	  ○ AI & ML & Analytics: Having previously transferred data from the edge's Data Manager to the matching cloud component, this service feeds from those data to process and analyse them.
	  ○ Big Data as a Service (BDaaS): Having previously transferred data from the edge's Data Manager to the matching cloud component, this service feeds from those data, potentially being in turn instrumentalized by the AI & ML & Analytics component on that tier.
	  ○ Federated Data Services & Communications: Data from devices can be sent to the cloud tier for proper storage. They can also be sent back to the edge in case low-latency data access is required.
	• Edge tier:
	  ○ AI & ML & Analytics: Feeds from data in the Data Manager to process and analyse it.
	  ○ Data Services & Communications: Data from and to other components and services is transmitted via this Core function,

	especially when the component providing or requesting data is on a different tier. ○ Safety, Performance and Security Monitoring Service: Data from monitoring services can be stored in the Data Manager for further analysis and decision-making support. ○ Auditing Service: Audit data can be stored in the Data Manager for further analysis. ○ Access, Privacy & Identity Service: Access records and related data can be stored in the Data Manager for being checked later. • Device tier: ○ Data Services & Communications: Data from devices are sent to the edge tier for proper storage (possibly being aggregated beforehand). ○ Safety, Performance and Security Monitoring Services: Data from monitoring services in devices can be stored in the edge's Data Manager for further analysis and providing hints in areas such as anomalies detection and predictive maintenance.

Table 22: Data manager at the edge

5.2.2.2 AI and ML and analytics

The AI, ML and Analytics components are responsible for handling operations related to extracting valuable insight from provided data and produce new knowledge through various forms of analysis. These components can be located at the edge and/or cloud tier of the system and will provide advanced capabilities such as natural language processing, image recognition, predictive analytics, and anomaly detection. The components are classified as either traditional Analytics or AI/ML based.

Traditional analytics typically involves the use of statistical techniques to analyse data and identify patterns or trends. This approach relies on human analysts to define the parameters of the analysis, interpret the results, and make decisions based on those results. Traditional analytics is often used to identify historical trends, understand customer behaviour, or monitor business performance.

On the other hand, AI-based analytics uses machine learning algorithms to automatically identify patterns and make predictions based on large datasets. These algorithms are trained on historical data and can identify complex relationships and trends that might be difficult for humans to detect. AI-based analytics is often used to make predictions about future outcomes, optimize business processes, or automate decision-making.

At the heart of an AI/ML component lies a complex algorithm that has been trained on a large dataset using advanced machine learning techniques. This training process typically involves feeding the algorithm with vast amounts of data and teaching it how to recognize patterns and make predictions based on the input data. The resulting model is then used to perform specific tasks, such as classifying images, translating languages, or predicting future outcomes based on historical data.

Depending on the requirements and constraints, the AI/ML components can be located solely on the edge tier. This may be the only solution if time criticality (latency) and communication dependability are hard

requirements; at the cost of available computational power and hardware leading to so-called CapEx. On the other hand, extending the system by integrating the cloud tier enables to leverage scalable on-demand computational power. Further, cross-communication between different clients can be realized more easily. And, supervision by personnel (human-in-the-loop) is easier to integrate.

The overview and additional information are detailed in Table 23.

Role/functionality		• Process incoming data and update learning model (ML)
		  ○ Host a manually created analytics model
		  ○ Analyse new data by applying learning or analytics model
		  ○ Manage data usage and access
		  ○ Store results
		• MAPE:
		  ○ Execute: Inference

Configuration	Mandatory	• Analytics models • Connection to Cloud
	Optional	• Data storage • Connection to Update Server
Input interfaces	Mandatory	• Learning data streams • Operational data stream
	Optional	• Cloud Connection • Trigger for retraining • Parameters for operation • Newly trained models or weights
Output interfaces	Mandatory	• Inference results
	Optional	• Log files • Heart beat • Trained model or weights • Pre-Processed data stream
Endpoints		• Cloud: Message Broker, API (gRPC, REST) • Data Source connection (e.g., CAN bus, sensor)
Covered requirements		• [TSR-15] Data Poisoning • [TSR-18] Data handling and security

	• [TSR-19] Data Protection • [TSR-31] Timely data analysis • [TSR-32] Timely data analysis • [TSR-33] Time constraints • [TSR-37] Updates • [TSR-39] Anomaly detection • [TSR-40] Message Broker • [TSR-44] Storage • [TSR-SP-1] Data Integrity • [TSR-ARCH-1] Protection of system resources • [TSR-ARCH-3] New Models • [TSR-61] AI frameworks • [TSR-62] Embedded vision • [TSR-63] Cascading framework for artificial intelligence • [TSR-64 to 110] Machine Learning Pipeline • [TSR-111 to 117, 121, 122] Support different algorithms • [TSR-118 to 120, 124] Data reduction
Related concepts & solutions	• [D3.1] Application service level agreements for AI-model based distributed CPS solutions • [D3.1] Concepts for ensuring data integrity (across the device-edge-cloud continuum) • [D3.2] Anonymization: Prevent Personal Data leak • [D3.2] Centralized Machine Learning with Decentralized Data • [D3.2] User and entity behavioural analytics (UEBA) concept for cloud security • [D3.2] Cloud detection & response (Cloud DR) orchestration for multiple clouds • [D3.3] [D3.5] Solutions for ensuring data integrity • [D3.4] [D3.6] Privacy preservation by leveraging the concept of federated learning • [D3.4] [D3.6] User behavioural massed ML models for anomalous activities in cloud environments

	• [D4.1] Combining on-device intelligence or autonomy with edge or cloud-based AI services to ensure safe handling of safety-critical situations • [D4.1] Defining the Level of safety-criticality to which AI services can be used, and the type of safeguards or degraded modes of operation that are needed • [D4.1] Defining what novel services can be created by the combination of data from various local systems that can exploit global insights beyond the local CPS only viewpoint • [D4.1] Updating AI services incorporating new or updated functionality (gradually) available to ensure early field feedback yet maintain safe operation • [D4.2] Strategies for retraining and releasing AI models • [D4.2] AI monitoring to determine if/when to update

Related components	• Edge tier:
	  ○ Safety-Critical Function: If results are relevant for Function.
	  ○ Mission Critical Function: If results are relevant for Function.
	  ○ Non-Critical Function: If results are relevant for Function.
	  ○ Data Manager: Data handling must be managed.
	  ○ New Services: Analytics enables New Services
	  ○ Performance and Security Monitoring Service: Supervision of results quality
	  ○ Auditing: Ensures traceability of parameter and model updates, and any other manual changes.
	  ○ Access, Privacy & Identity Services: Crucial as sensitive data may be processed.
	  ○ Remote Updates Coordinator: Updating models and weights
	  ○ Data Services & Comms: Enables third-party integration and communication with edge clients.
	• Cloud tier:
	  ○ AI & ML & Analytics: Manages client connection and data exchange.

Table 23: AI & ML & analytics at the edge

5.2.2.3 New services

The New services component enables the integration of new added-value services to TRANSACT, to be used by applications and other architectural components. This instance of the component is deployed on the edge tier. Although the "New services" component is conceptually identical to its cloud counterpart, the technologies it offers here match services that, due to their nature, purpose and computing power and infrastructure requirements, make sense to be deployed on the edge tier of TRANSACT's architecture. More specifically, they usually will be services whose latency requirements are not critical and whose resource requirements fit those of the edge tier premises, not needing to be offloaded to the cloud tier.

The overview and additional information are detailed in Table 24.

Role/functionality		• Enable the acquisition of new technologies as services on the edge tier, for their integration in TRANSACT's applications, services and components. • N/A phase on the MAPE cycle
Configuration	Mandatory	• Listing of the technologies available
	Optional	• Category and description of the technology to be acquired • Specification of the interfaces in the technology to be acquired
Input interfaces	Mandatory	• N/A
	Optional	• N/A
Output interfaces	Mandatory	• Attend requests by serving and supplying packaged technologies.
	Optional	• N/A
Endpoints		• Endpoints exposed by the technology to be acquired, in case it operates as a server that attends and processes requests by clients.
Covered requirements		• [TSR-88] Inclusion of a library providing multiple evaluators • [TSR-107] Availability of multiple monitoring tools • [TSR-115] Support to multiple machine learning algorithms • [TSR-131] Frameworks for commercial machine learning
Related concepts & solutions		• [D4.1] Integration of AI services • [D4.3] Fast development of innovation using distributed solutions • [D4.4] New business models enabled by edge and cloud services

Related components	• Cloud tier: ○ New Services: Integration of new technologies as services ○ New Services Marketplace: Supplying of new technologies, developed by TRANSACT partners or by third-party teams.

Table 24: New services at the edge

5.3 Device tier

The device tier (Figure 6) implements the domain-specific functions that represent the embedded devices (sensors, actuators, displays…) and other specific devices. They are usually deployed in very limited platforms in terms of computing capacity and consumption. They are also defined by specific functionality and, typically, real-time requirements. These functions are implemented on top of the TRANSACT device core services and functions (Section 5.3.1).

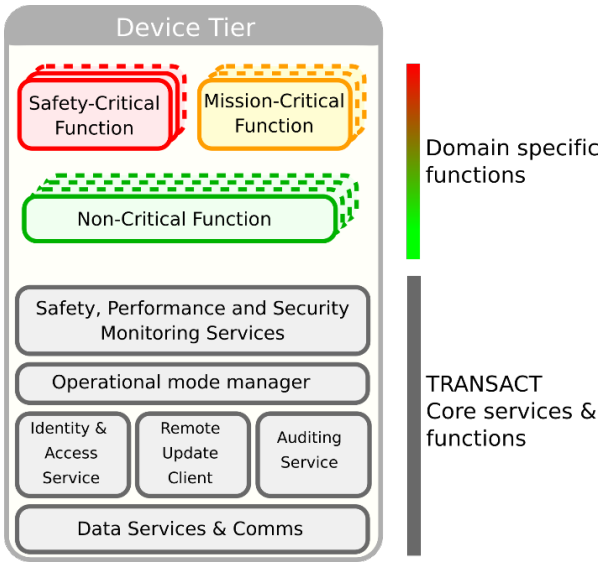

Figure 6: Static view of the device tier

5.3.1 Core services and functions

The device Core services support the safety-critical, mission-critical and non-critical functions at the device tier. These core components encapsulate the safety and performance, and security and privacy low-level functionalities of the system, which are responsible for providing the necessary support to monitor, coordinate and configure the devices.

5.3.1.1 Safety, performance and security monitoring services

Monitoring services in the device tier comprise the monitors in charge of collecting safety, performance and security indicators.

The aim of the performance monitoring service is to monitor performance indicators from the various components and systems on the device tier. This service might include response time of the local process ensuring performance requirements; resource utilisation aspects, such as CPU, memory, and disk usage; or power consumption to help optimise energy efficiency.

The aim of the security monitoring service is to detect and prevent security violations. The preventive part will continuously monitor all components and their compliance with the security policies that are defined. This service focuses on detecting security threads at device level, such as malware, unauthorized access or physical tampering This functionality heavily depends on the capabilities of the device tier in terms of computing resources available for these tasks. If the device isn't capable of doing these types of tasks, they can be offloaded to the edge tier.

Safety monitoring service primarily deals with collecting data to facilitate the identification and prevention of potential hazards, risks and failures within a system. This service is especially relevant in safety-critical devices, where avoiding or mitigating hazardous failures must be ensured.

Monitoring services will forward the events to the edge or cloud tier for further analysis.

The overview and additional information are detailed in Table 25.

Role/functionality		• M in the MAPE cycle • Monitoring safety, performance and security indicators of different components
Configuration	Mandatory	• If KPIs are monitored though pull operations, the endpoints for the KPIs have to be configured
	Optional	• Criteria/ranges to specify/describe the monitoring of the KPI (e.g., SLOs, thresholds)
Input interfaces	Mandatory	• Performance KPIs of components (system/service KPIs). It may be required to push or pull the KPIs, depending on the scenario.
	Optional	• Security policies
Output interfaces	Mandatory	• The set of KPIs themselves
	Optional	• Predictions on KPIs values, e.g., regarding the crossing of a threshold • Outputs/alarms/events
Endpoints		• Receiving device-tier monitoring events
Covered requirements		• [TSR-30] Architecture should be self-organising, that implies self-monitoring capabilities. • [TSR-32 to 33] Ensure architecture in near-real-time or real-time performance

	• [TSR-38] Constantly monitor available resources to detect software failures and aging in the architecture • [TSR-46] Cloud computing components in the architecture will enhance the performance by processing IoT data in the cloud • [TSR-55] The components in the architecture should be able to monitor events, resources and errors. • [TSR-56] The architecture should support the monitoring of the data transmission rate of the devices. • [TSR-58] The TRANSACT system architecture should support error detection.
Related concepts & solutions	• [D3.1] Application service level agreements • [D3.1] [D3.3] [D3.5] Concepts and solutions for AI monitoring • [D3.1] [D3.3] [D3.5] Concepts and solutions for predictable performance • [D3.1] [D3.3] [D3.5] Concepts and solutions for health and safety monitoring • [D3.2] Runtime verification
Related components	• Auditing Services • Operational Mode Manager

Table 25: Safety, performance and security monitoring services in the device

5.3.1.2 Auditing service

Auditing consists of examining and evaluating processes in terms of their level of computerisation and data processing, also of verifying controls in the processing of information and verify the installation of security systems, as defined in auditing services at the cloud.

When a system is audited, the technical components of the system are analysed for vulnerabilities or threats that could have a negative impact on the system.

It is necessary to highlight the importance of checking, not only the computerised equipment but also the controls applied and their use.

Some of the specific risks that need to be considered in an auditing process of the device tier are those relating to updating processes or the lack of encryption. It is evident too, how important is to implement an access control to avoid non-authorised access.

Auditing services can significantly enhance auditing processes by providing automation, efficiency, and improved accuracy in the collection, analysis, and reporting of audit-related data. These services can be responsible for tracking device configuration and updating; logging user activities such as unauthorised accesses and suspicious actions or any event related to a security incident.

It is often challenging to develop and maintain dedicated auditing services tailored for each end-devices, due to resource limitations and since no off-the shelf solutions are available. Safety and security expertise are required, and, moreover, any changes in regulations must be timely matched.

The overview and additional information are detailed in Table 26.

Role/functionality		• Check safeguards. • Identify areas for improvement. • Revise the compliment of various applicable laws. • Log device configuration and updating • Track anomalies in resource consumption. • Provide alerts for security incidents.
Configuration	Mandatory	• The policies and proceedings implemented must comply with the applicable regulation. • These policies and proceedings need to be correctly implemented in the structure of the company.
	Optional	• It is advisable to involve all the areas in the process of auditing.
Input interfaces	Mandatory	• Coordination between departments and auditor. • Information on the current status. • Set a level of auditing.
	Optional	• N/A
Output interfaces	Mandatory	• Reporting. • Analytic results. • Advisory output.
	Optional	• N/A
Endpoints		• Interchange proceedings and policies.
Covered requirements		• [TSR-13] Protection against authentication and authorization attacks • [TSR-18] Mitigation plan in case of system is compromised and how to participate in the investigation and prosecution. • [TSR-19] Data is not shared to unauthorised third parties.
Related concepts & solutions		• [D3.1] Requirements and concepts for end-to-end safety and performance assurance

	• [D3.2] [D3.4] [D3.6] Security and privacy concepts for secure remote driving operation • [D3.2] Security and privacy requirements and patterns for the healthcare DICOM data and applications
Related components	• Safety, Performance and Monitoring Services. • Operational Mode Manager. • Access, Privacy and Identity Services.

Table 26: Auditing service in the device

5.3.1.3 Identity and access service

There is a wide range of devices, both simple and complex. Some can be, for example, independent sensors with communication capabilities, whereas others can be applications running on a client system that can have some user interaction. In all cases, one of their main capabilities is getting access to a resource. For devices, the main capability is to get access to a resource. The device should support the authentication and authorization flows with or without user interaction, depending on the device. The applications can include the use of an agent (browser). The flows interact with an identity and access service that can be on the cloud (online) or edge (offline/local).

The overview and additional information are detailed in Table 27.

Role/functionality		• Enable (end-users) to perform token authentication and authorization flows
		  ○ Online (mandatory)
		  ○ Offline (optional)
		• Secure secrets storage (hardware or software)
		• Optional
		  ○ Support exchange via different communication protocols (BLE/NFC/TCP-IP/HTTP(s))

Configuration	Mandatory	• Configuration time: Device / app onboarding & token storage
	Optional	• N/A
Input interfaces	Mandatory	• API to trigger authentication/authorization flows on device ○ Challenge (in Challenge Response) • API for configuration
	Optional	• N/A

Output interfaces	Mandatory	• API to trigger authentication/authorization flow toward the authorization server (premise or cloud) ○ Response (in Challenge Response)
	Optional	• N/A
Endpoints		• N/A
Covered requirements		• [TSR-10] I&S should support against (malware) requested token via challenge-response or request-response association • [TSR-13] Use OAuth >1.0 • [TSR-17] Secure Key storage • [TSR-18 to 20] Component should (indirectly) contribute.
Related concepts & solutions		• [D3.4] [D3.6] Authentication Management • [D3.4] [D3.6] Security, privacy and trust related solutions for remote driving operation
Related components		• Auditing Service

Table 27: Identity and access service in the device

5.3.1.4 Operational Mode Manager

The Operational Mode Manager aims to manage the transitions between operational modes at the device level. In addition, this component is responsible both for enabling tasks and configurations to be executed based on the selected operational mode and for selecting the correct protocol to ensure a safe transition between modes. Each operational mode defines a behaviour or configuration that the system has to be in each phase of the execution, including both normal operation and failure mitigation mechanisms. A mode transition corresponds to a change in the system behaviour, responding to events or alarms provided by other services.

Operational Mode Coordinator in the edge tier must define the operational mode to be selected at the device level, and the Operational Mode Manager must follow their requests. In addition, events (or alarms) internally generated by the device application have to be considered to perform isolated transitions at the device level, usually with the aim of mitigating a localized failure.

The overview and additional information are detailed in Table 28.

Role/functionality	• Manage the transitions between operational modes at device level • Enable tasks and configurations to be executed based on the selected operational mode • Select the correct protocol to ensure a safe transition between modes

		• E (Execute) on the MAPE cycle (extend functionality) • Optionally P on the MAPE cycle

Configuration	Mandatory	• At configuration time:
		  ○ Operational modes at device level.
		  ○ Allowed mode transitions.
		  ○ Mode change protocols.
		  ○ Additional rules relating events/alerts and local transitions (for isolated fault mitigation purposes at edge level).

	Optional	• N/A
Input interfaces	Mandatory	• From the Execution point of view, requests for mode changes • Be aware of mode changes (feedback) or failure in applying them
	Optional	• Partial or component wise mode changes • From the Planning point of view, you need the inputs, events that allow the component to propose a change
Output interfaces	Mandatory	• Execution of the proposed mode change • Acknowledgement/failure of the proposed mode change to other components
	Optional	• Execution/acknowledgement/failure of partial mode changes
Endpoints		• Mode Change Request • Status • Other communication may take place through publish/subscribe mechanisms
Covered requirements		• [TSR-60] Multimode approach to support online error recovery and repair • [TSR-SP-2] Allowing multimode to support system reconfiguration and fall-back solutions
Related concepts and solutions		• [D2.1] Operational mode management • [D3.1] Concepts for operational modes and change transitions • [D3.3] [D3.5] Solutions for operational modes and change management: Mode change management on the device

Related components	• Edge tier:
	  ○ Operational Mode Coordinator: Mode change requests
	• Device tier:
	  ○ Safety, Performance and Security Monitoring Service: Provider of events and alarms to enable local mode changes.

Table 28: Operational mode manager in the device

5.3.1.5 Remote updates client

The Remote Update Client allows a device to communicate with a Remote Update Coordinator, in the edge tier as the first option and in the cloud tier as the fall-back option, to check what its desired state is. Once it receives its desired state, it compares it to its current state, to determine if any software running on the device must be updated. If so, the device Remote Update Client will fetch the new software installer from the Remote Update Coordinator or any other source and perform the software update. Once all the necessary updates are completed, the current state of the device should match its desired state.

The overview and additional information are detailed in Table 29.

Role/functionality		• Manage the software updates of software running in the devices • Retrieves a desired state from a Remote Update Coordinator • Downloads and applies the necessary updates to reach the desired state • Reports the device software state • Execution phase (E) in the MAPE cycle • Optional Planning phase (P) on the MAPE cycle
Configuration	Mandatory	• Software list • Is user confirmation required before applying updates? • Hot update required? (maintain service during the process)
	Optional	• List of Remote Update Coordinators
Input interfaces	Mandatory	• Desired state (software versions) • Software updates download
	Optional	• N/A
Output interfaces	Mandatory	• Current state (software versions) • Update process status
	Optional	• N/A

Endpoints	• REST API for pushing desired software state • REST API for reporting (current state, update progress)
Covered requirements	• [TSR-37] Offer support for software updates • [TSR-SP-4] Independent releasing of modular software parts
Related concepts & solutions	• [D1.2] Technical requirements and TRANSACT transition methodology commonalities • [D2.1] Software updates in safety-critical systems • [D3.1] Concepts for safe and secure modular updates • [D4.2] Strategies for continuous updating and independent releasing

Related components	• Cloud tier:
	  ○ Remote Update Coordinator: checks for new desired state and reports status
	• Edge tier:
	  ○ Remote Update Coordinator: checks for new desired state and reports status

Table 29: Remote updates client in the device

5.3.1.6 Data services and communications

The Data services and Communications component at the device tier is responsible for managing data exchange and communications with the edge node to which the device is connected. The objectives are to facilitate devices to leverage the services provided by cloud and edge tiers and to provide edge and cloud services with the collected data.

The communication service at the device must implement mechanisms to detect and recover from communication failures, ensuring that an error in the connection with the edge does not result in a complete system failure that compromises its availability, ensuring the resilience of the system.

Here, stacks of various communication protocols are contained, and the aspects related to requesting credentials, authorization information or identity from the Identity and Access Service. It should be emphasized that for ensuring end-to-end security, data encryption should take place. In some cases, the device tier would have limited computational resources, therefore the methods employed at this tier should be matched to the capabilities of the devices, ensuring seamless data flow and adequate security at the same moment.

The overview and additional information are detailed in Table 30.

Role/functionality	• Manage data exchange and communications with edge nodes. • Failure detection and recovery in communications.

		• This component is related to execution phase (E) in MAPE cycle.

Configuration	Mandatory	• At configuration time:
		  ○ Specification of communication protocols and security settings.
		  ○ Configuration of fault tolerance mechanisms, such as redundancy and failover settings.

	Optional	• At runtime: ○ Change of connection parameters.
Input interfaces	Mandatory	• Configuration interfaces. • Data ingestion interfaces (data streams, messages, …)
	Optional	• N/A
Output interfaces	Mandatory	• Data output interfaces, that are responsible of transmitting processed data to the edge.
	Optional	• N/A
Endpoints		• Configuration endpoint • Data ingestion endpoint
Covered requirements		• [TSR-17] Ensure security requirements of confidentiality, authenticity and integrity. • [TSR-28] Data encryption on the continuum. • [TSR-31] Usage of adequate storage mechanism and effective data-transmission protocols in the CPS network. • [TSR-40] Usage of message brokers with large number of devices. • [TSR-42] Availability of bandwidth to deal with IoT app requirements. • [TSR-45] Optimization of bandwidth according to IoT demands. • [TSR-SP-1] Ensure data integrity across continuum.
Related concepts & solutions		• [D3.2] Concepts to centralized machine learning with decentralized data. • [D3.2] Security and privacy requirements and patterns for the healthcare DICOM data and applications. • [D3.4] [D3.6] Secure communication-based solutions. • [D3.4] [D3.6] Data communication security solutions.

Related components	• Edge tier:
	  ○ Data Services and Communications: Data exchange with edge node.
	• Device tier:
	  • Access, Privacy and Identity Services: Provides identity of the device and manages access control for communication purposes and data provisioning. Also provides encryption of the data and cryptographic engine for communication purposes.
	  • Auditing Service: Could be coupled with d*ata services and comms* to monitor activities, aiming to detect suspicious actions or potential attacks.

Table 30: Data services and communications in the device

6 Dynamic view of the reference architecture

The dynamic view of the TRANSACT reference architecture describes features covered by the architecture and provides insights into how the different components interact during execution. This view represents the behaviour of the components in the device-edge-cloud continuum to ensure that the system operates as expected. This view explains the role of the components regarding a specific functionality and the relationship between them.

Six functional capabilities have been selected to facilitate the understanding of the TRANSACT reference architecture. These capabilities describe the system features from the perspective of the end-user and their needs. Each of them is characterised by its workflow as the different stages during the solution process, defining the relationship between the components of the reference architecture.

The workflow of each functional capability describes a solution to the user's needs and defines the methods and steps to achieve it. This description does not refer to any specific case, but it should be possible to implement it in different circumstances.

A selection of relevant scenarios has been included for each capability. Each scenario illustrates a concrete usage of a workflow. A scenario describes a concrete story in a particular context within the framework of the problem. The behaviour of the system is detailed using the components of the reference architecture to make it easier to understand how to implement the workflow in this given context.

Table enumerates the components of the static view that have been included in the selected functional capabilities of the dynamic view. This relationship does not exclude the components from being used together with other components not detailed in this table.

		Core services & functions						Value-Added services & functions			
		Safety, Performance and Security Monitoring	Operational modes	Access, Privacy & Identity	Remote Updates	Auditing Service	Data Services & Comms	Data Manager	AI & ML & Analytics	BDaaS	New Services & Marketplace
Functional capabilities	Health monitoring	•	•						•	•	
	Performance management	•	•					•	•		
	Change of operational mode	•	•				•	•			
	Remote updates				•						•
	Access control			•		•	•				
	End-to-end secure communication			•	•	•	•				

Table 31: Relation between dynamic and static view

6.1 Health monitoring

Health monitoring in a distributed system within the device-edge-cloud continuum involves using software components to collect, analyse, and manage health-related data across different system tiers. This practice is essential for ensuring the efficient operation of systems and achieving various technical and business goals, such as early detection of issues, preventive maintenance or resource planning.

At the device level, sensors and software agents collect diverse health-related data, from maintenance parameters to performance metrics. This raw data can be initially and partially preprocessed on the device, aiming to reduce bandwidth communications and leverage the results to make internal decisions. More advanced processing occurs at the edge. Runtime analytics and anomaly detection algorithms operate there, providing quick insights into health status. This edge tier acts as an intermediary, performing initial data analysis, filtering, transformation and aggregation before transmitting relevant information to the centralized cloud. In the cloud tier, the availability of extensive computational capabilities is leveraged for in-depth analysis, in the context of Big Data Analytics techniques. Advanced machine learning models enable predictive analytics and decision-making, fostering an accurate, efficient and sophisticated approach for handling large volumes of health-related data.

In essence, health monitoring functionalities must be consistent with continuous monitoring and analytics services, aligned with BDaaS cloud capabilities, that provide automatization and scalability in data processing according to the necessary workload.

6.1.1 Workflow

Implementing a health monitoring solution in the device-edge-cloud continuum involves using a collection of components of the TRANSACT reference architecture, such as monitoring services, *AI & ML & Analytics* services and *Big Data as a Service (BDaaS)*. Several steps are interconnected to perform this functionality:

1. **Continuous Monitoring**: Deploy comprehensive monitoring services on each tier (device, edge, cloud) to continuously assess the health of components and detect anomalies. They are usually located in the device tier as sensors but, in practice, can be deployed in the whole system.
2. **Device Local Processing**: Implement basic preprocessing on the device to filter, transform, batch and aggregate data before transmission.
3. **Edge Data Aggregation and Preprocessing**: Edge nodes aggregate data from multiple devices and leverage edge computing for more advanced preprocessing, including AI analytics, such as early anomaly detection.
4. **Cloud data Ingestion**: Cloud services receive and ingest the transmitted data, leveraging BDaaS principles for scalability and storage in the efficient handling of large datasets.
5. **Big Data Processing**: Utilize cloud BDaaS tools and frameworks, such as Apache Spark, Flink, or Hadoop, for in-depth analysis, predictive modelling and trend identification. Usually, these processes are either batch-oriented or stream-oriented, depending on how they ingest and process available data.
6. **Decision-making**: Implement machine learning models for advanced health analytics and decision-making.
7. **Scalability and Resource Management**: Utilize containerization (e.g., Docker) for packaging applications and their dependencies, enhancing scalability and portability. Also, implement

orchestration platforms and tools (e.g., Kumori) to manage the deployment and scaling of containers efficiently.

Following this workflow, a robust health monitoring solution can be established in the device-edge-cloud continuum, leveraging monitoring services and BDaaS principles for efficiency, scalability, and comprehensive data analysis.

6.1.2 Scenario 1: Predictive maintenance strategy for wastewater treatment plants

Industrial monitoring systems are employed to proactively manage and optimize preventive maintenance in industrial setups. The approach is focused on using monitoring technologies and advanced analysis to assess the health and performance of critical machinery and equipment. Continuous monitoring captures data on relevant factors, while predictive analytics models analyse historical data to identify patterns indicative of future potential equipment failures, allowing for preventive measures. These model-based algorithms evaluate machinery's health and functioning parameters and triggers maintenance actions when deviations occur.

Health monitoring processes perfectly integrate with existing maintenance workflows, ensuring analysis results translate into both manual and automated maintenance tasks. The benefits include minimized downtime, extended equipment lifespan, cost savings, improved operational efficiency, data-driven decision-making, and enhanced safety in industrial environments.

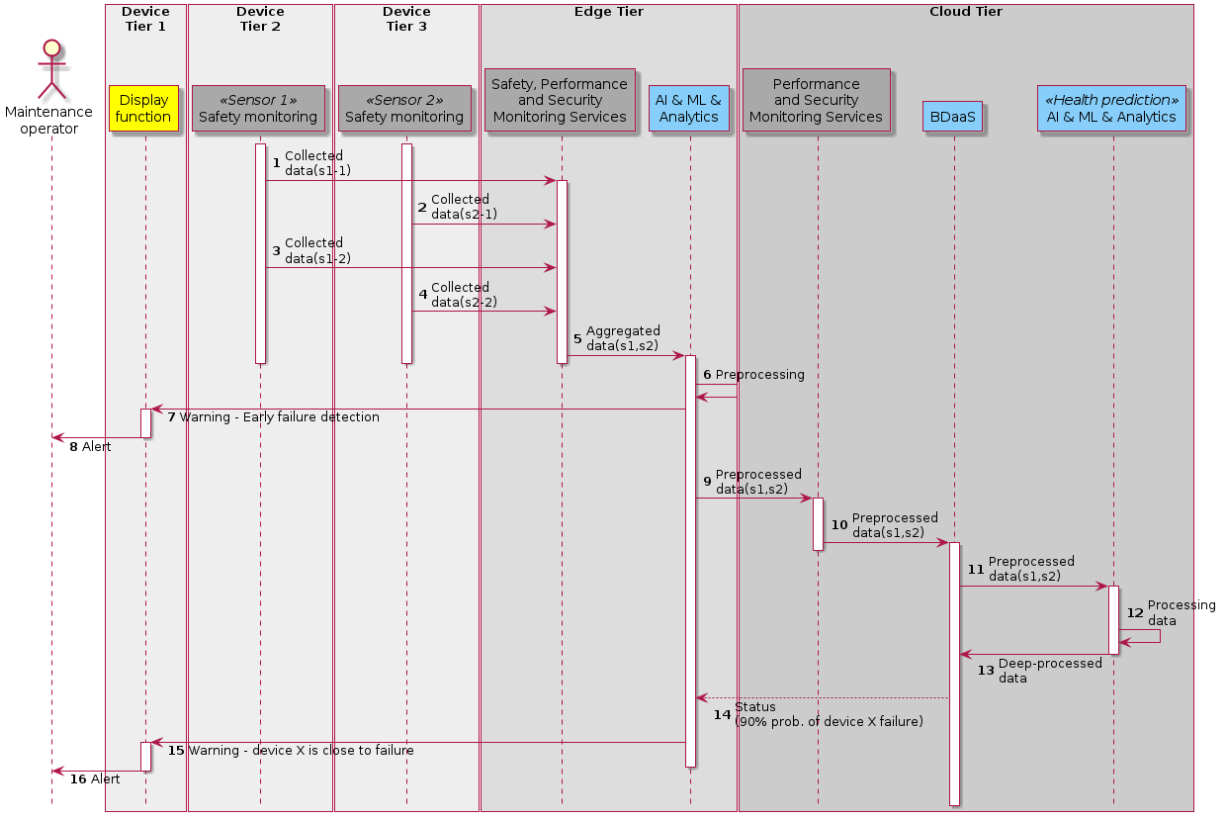

Figure 7: Predictive maintenance strategy

The scenario displayed here is based on a network of wastewater treatment plants (Figure 7), whose machinery is continuously monitored to ensure the proper functioning of their critical infrastructure and equipment, such as pumps, motors or sensors. Temperature or vibration values are aggregated from different strategic points distributed around the plant. An initial edge preprocessing can help detect possible failures early, warning the maintenance operator before they occur. A deep analysis could be performed in the cloud with complex AI analytics, in the context of current operational conditions, that allows predicting the future behaviour of the infrastructure. This aims to diminish downtime periods and decrease OPEX costs, since maintenance windows can be calculated and planned according to predictions, replacing worn away components before their malfunctioning affects the overall running premises.

This scenario shows the behaviour of a CPS composed of an edge processor connected to various sensor devices and display intended to alert the maintenance operator acting as the end-user. The edge can offload heavy data processing tasks to the cloud tier. All devices are managed from the edge tier, where the relevant information collected is preprocessed and aggregated by using the *AI & ML & Analytics* services. If this early preprocessing detects a warning, an alarm is displayed. Next, the resulting data is delivered to the cloud, where it is deeply processed by means of the *BDaaS* component. The probability of a device failing is sent back to the edge, which, in case of exceeding a 90% threshold, instructs the device tier to raise a visual alarm.

6.1.3 Scenario 2: IoT battery health monitoring

Battery health monitoring is deployed to enhance the longevity and performance of electronic devices reliant on rechargeable batteries. Its primary goal is the proactive assessment and management of battery health, addressing issues related to degradation, capacity loss and overall efficiency.

The system involves continuous runtime monitoring of relevant battery parameters, allowing for the detection of variations and potential anomalies. This data is used to analyse capacity trends, optimize charging processes, and implement predictive maintenance strategies. A user interface provides information about battery health, notifying users of alerts and critical events. The focus is on extending the device's lifespan, achieving cost and energy savings and promoting environmental sustainability through responsible consumption and waste reduction.

This scenario shows the behaviour of a CPS composed of an edge processor and several devices connected to it, including the battery, an engine and a user interface (Figure 8). The edge will be responsible of managing all devices, collecting the relevant information and processing it using *AI & Analytics* services. The system status is provided to the user on runtime and, in case of detecting an issue the *Operational Mode Coordinator* at the edge has the ability to execute the appropriate mitigation action, as well as alert the user. On the other hand, the edge itself is connected to *BDaaS* services in the cloud to send the relevant system telemetry, usually preprocessed and compressed in batches to save bandwidth. This information, collected from all systems connected to the cloud, is used by engineers to evaluate the performance of the design, identify potential action points and introduce improvements in future releases.

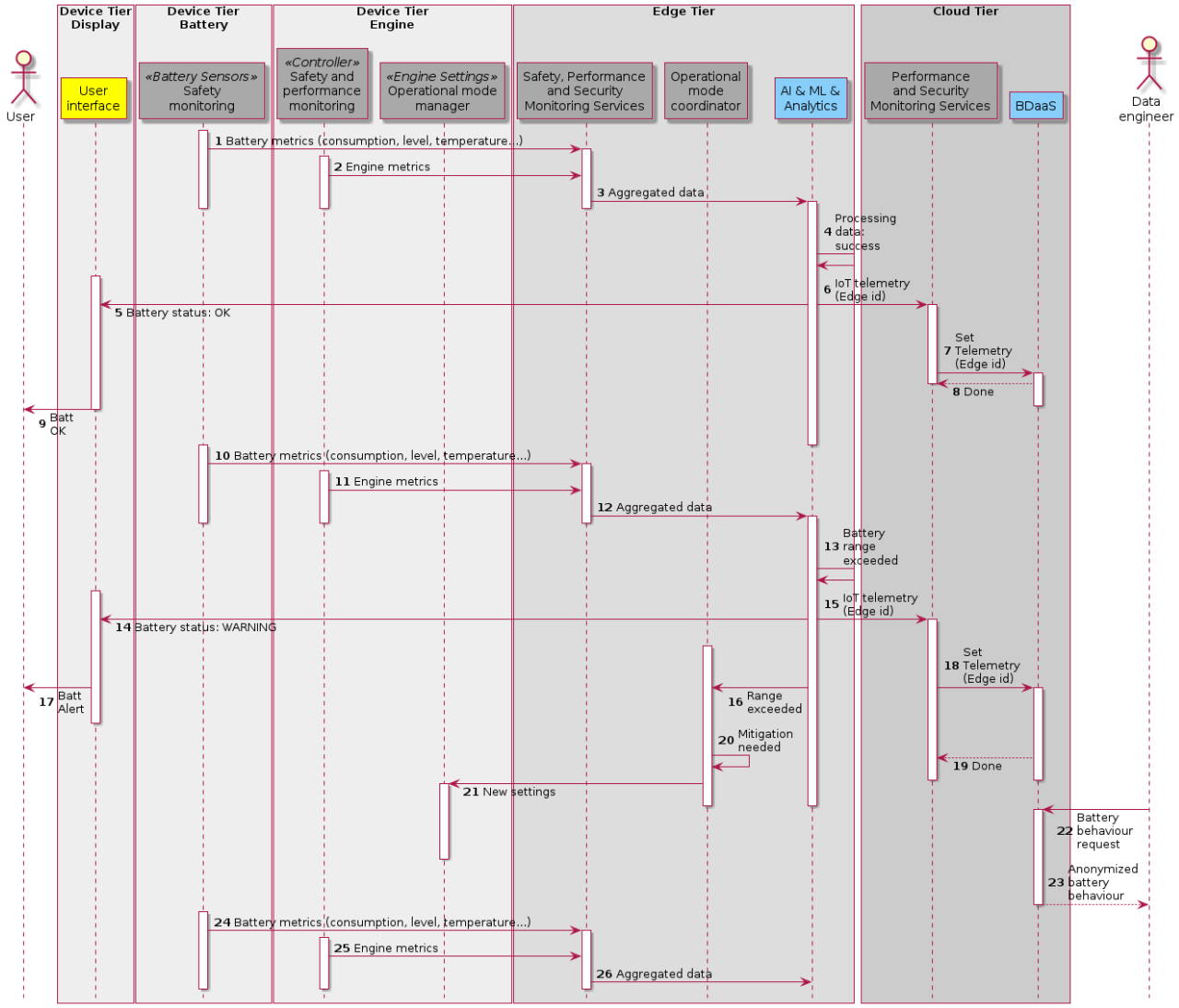

Figure 8: IoT battery health monitoring

6.2 Performance management

Timing performance often plays an important role in cloud-based cyber-physical systems. In TRANSACT UC4, for example, the 3D reconstruction of a sequence of medical X-ray images taken in a hospital by a scanner, is mission-critical functionality. The end-to-end latency for 3D image reconstruction requests that are issued from hospitals to a compute cloud should be bounded with a certain probability to meet the system requirements. Multiple scanners in a single hospital, however, share the bandwidth to the cloud. If multiple scans happen at the same time, then the bandwidth can become a bottleneck that may significantly influence the end-to-end latency. The system thus must adapt to changing resource availabilities and workloads to ensure that the timing requirement are met in a dynamic environment. In this section we illustrate how the TRANSACT reference architecture and its solutions can be used to address this kind of performance-management challenges.

6.2.1 Workflow

Performance management can be realized by following the MAPE-K paradigm. Design-time activities are the foundation on which the run-time performance management is built:

A. Determine the performance metrics to be managed.

B. Develop a predictive model for the metrics, based on several system parameters.

C. Design a monitoring strategy for the system parameters that are input for the predictive model.

D. Design the normal/happy flow, and design mitigation strategies for situations in which the (predicted) performance of the happy flow does not meet the requirements.

The monitoring strategy, predictive models and mitigation strategies can then be used at runtime to manage the system performance:

1. (Monitor) Continuously monitor the system parameters needed for the predictive model.
2. (Analyse) When functionality is requested, make a performance prediction of the happy flow.
3. (Plan) Decide whether to use the happy flow or apply a mitigation strategy.
4. (Execute) Fulfil the request.

The *Performance Monitoring Service* is responsible for the Monitoring and Analysis. The *Operational Mode Coordinator* can be used for the Planning and serve as a gateway for the Execution of the request. The distribution of these responsibilities over the edge or cloud instances of these services is highly dependent on the specific use case.

6.2.2 Scenario 1: Cloud-based 3D reconstruction

The first scenario is inspired by TRANSACT UC4. It considers mission-critical image processing functionality (3D reconstruction) for a medical scanner. The image processing has been deployed in the cloud, and there is a low-quality backup option on the edge. The key performance metric (A) is the probability that the end-to-end latency of a 3D reconstruction requested by a scanner in the hospital is below some threshold (denoted by $(e2e < x)$.

It is anticipated that the bandwidth of the hospital to the cloud can be a bottleneck, e.g., when multiple scanners in the same hospital are scanning at the same time. A predictive model (B) needs to be developed that predicts $\mathbf{P}(e2e < x)$ based on the selected X-ray protocol, the available bandwidth, and the activity of the other scanners in the hospital.

The main system parameters that need to be monitored (C) thus are the bandwidth and the activity of the scanners.

The happy flow is the normal 3D reconstruction by the cloud. If this is not fast enough, then the fall back to the 3D reconstruction on the edge can be used, which has lower quality but a guaranteed end-to-end latency. The mitigation strategy (D) thus is a simple switch to fall back on the edge if $\mathbf{P}(e2e < x)$ is smaller than the required probability.

Figure 9 shows a sequence chart that follows the happy flow. The *Performance Monitoring Service* on the edge continuously monitors the bandwidth (1). When the scanner interface is used to select an X-ray protocol, then this triggers the analysis phase (2): a prediction is made for $\mathbf{P}(e2e < x)$. The outcome of the prediction is used to plan (3) the reconstruction in the cloud, which consequently is executed (4).

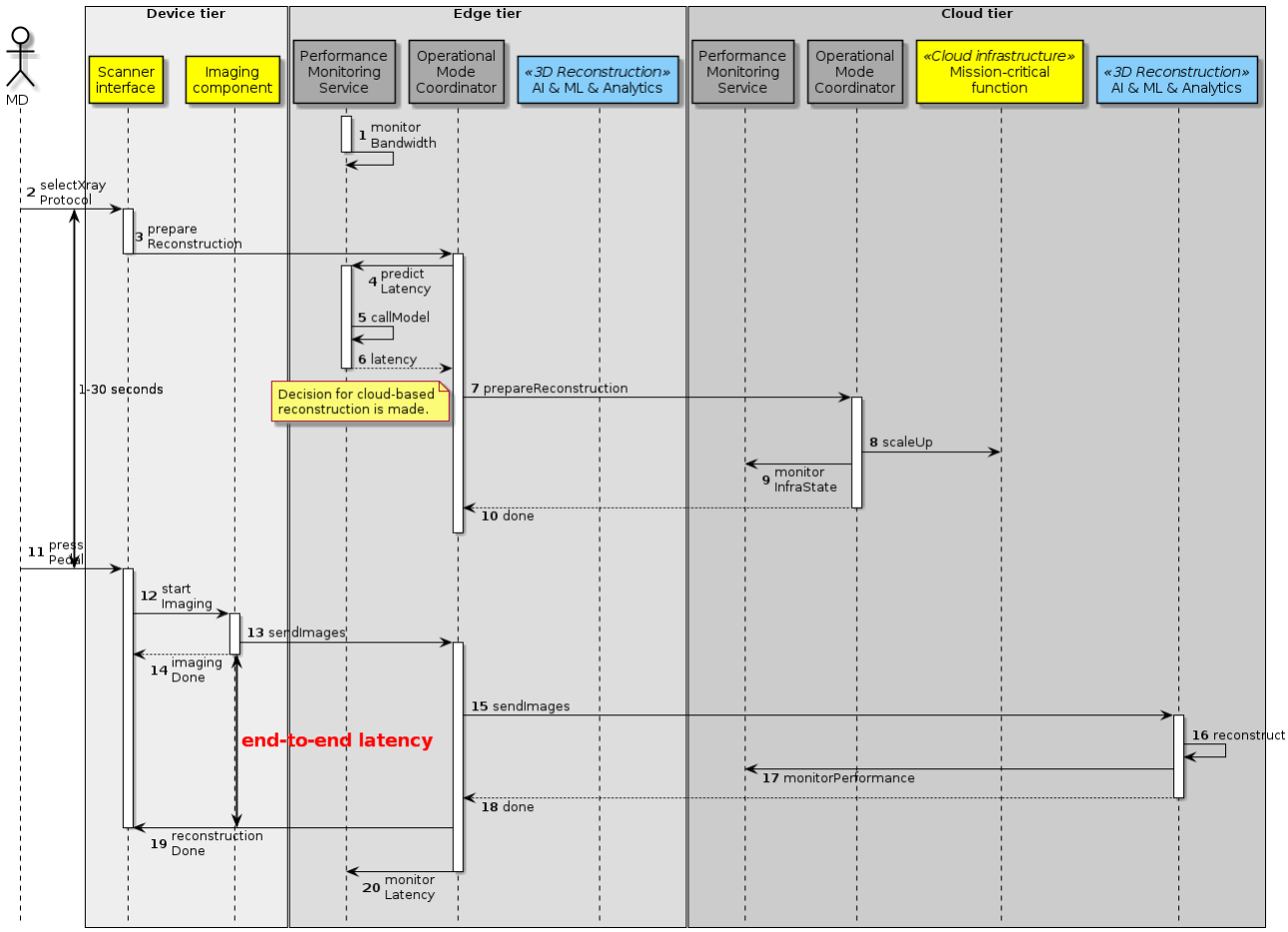

Figure 9: Cloud-based 3D reconstruction

6.2.3 Scenario 2: Edge-based 3D reconstruction

The second scenario is similar to the first scenario. Instead of the happy flow, now the outcome of the prediction results in a plan to use the fall back on the edge.

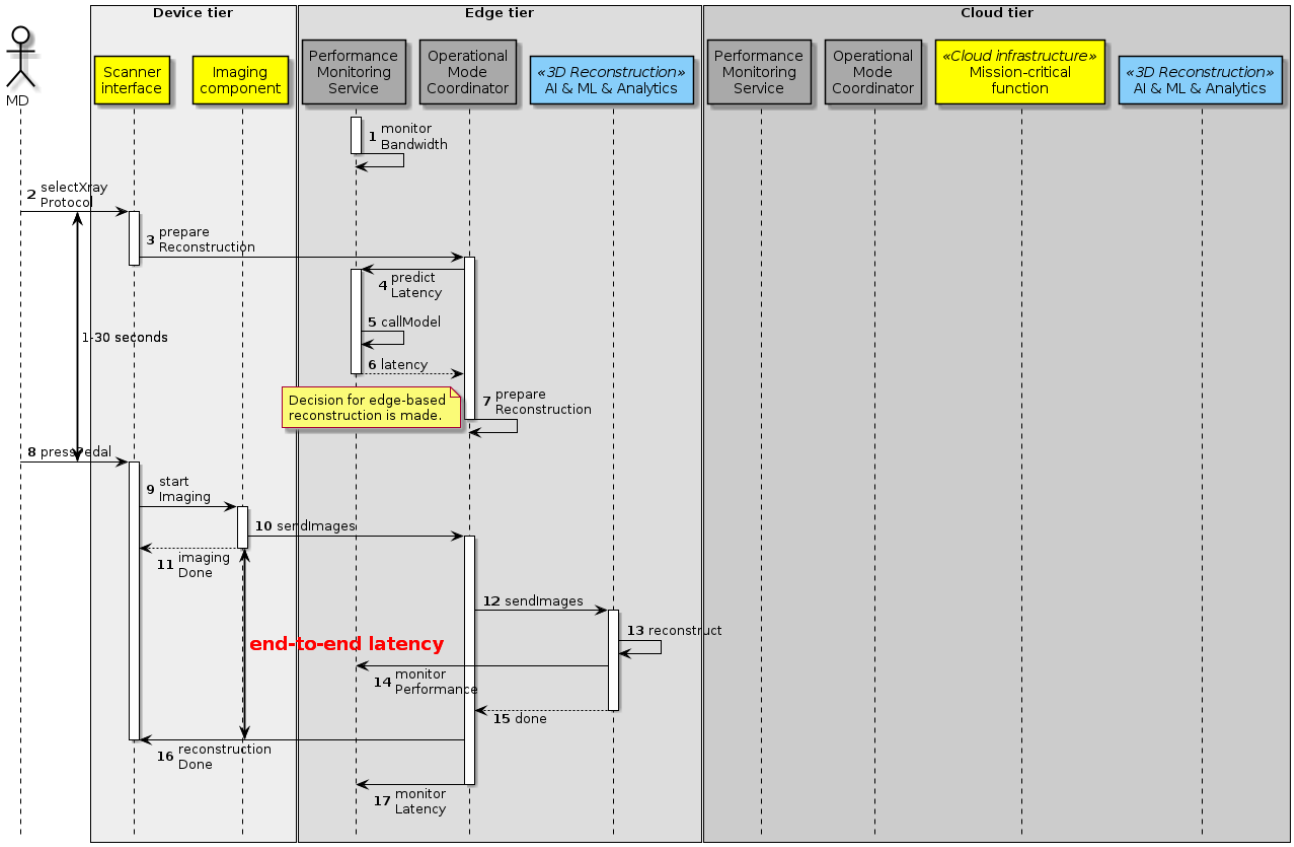

Figure 10: Edge-based 3D reconstruction

6.2.4 Scenario 3: Edge-based maritime advisory service

This scenario stems from the TRANSACT UC2 (maritime advisory service framework). This scenario addresses the safety-critical functionality of creating and updating routes onboard a vessel. Here, the route procession is deployed in the cloud as-well-as on the edge. The cloud-based version in general has higher performance and produces more optimal routes by enabling to consider extra information such as weather and ocean conditions, tidal streams, the routes of other vessels, and sessional or temporal preferable or undesirable regions (like those with brash or fast ice, maritime sensitive areas, piracy warning, etc). However, the edge-based version might not have updated global dynamic information, especially the frequently updated, global weather forecast, while the routes of the nearby vessels might be anticipated or locally communicated. The edge-based version serves as a safe backup which can generate routes which are indeed safe but might not be optimal and, hence, might lead to increased fuel-consumption, detours, extended sea time, etc. However, the backup is needed as the cloud-based service might not be available at all time. Especially in worse weather conditions (which are to be avoided by the service), it might not be possible to contact the cloud-based service in order to request an updated route.

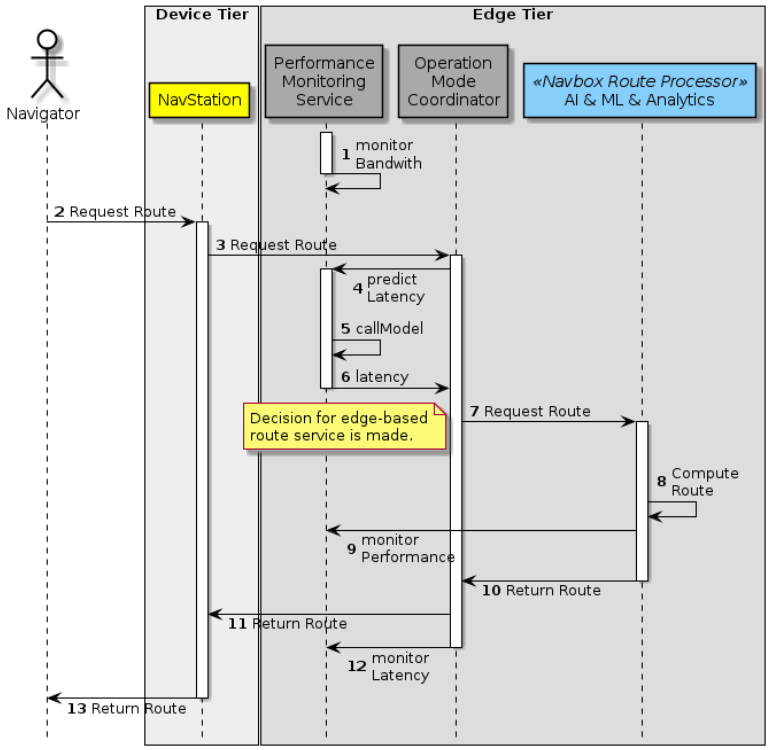

Figure 11: Edge-based maritime Advisory Service

In this scenario, the navigator requests the generation of a new route or an update of an existing route via the NavStation onboard the vessel. This request is than forwarded to the Advisory Request/Response Manager (*Performance Monitoring Service* & *Operational Mode Coordinator*). The manager decides to make use of the edge-based Navbox Route Processor (*AI & ML & Analytics*) as the cloud-based service is not available, e.g. due to a previous time-out or the cloud-connection being down. As a result, the response (the route) is generated by the Navbox Route Processor (*AI & ML & Analytics*) and is returned to the Advisory Request/Response Manager (*Performance Monitoring Service* & *Operational Mode Coordinator*) which, in turn, returns it to the NavStation, where the Navigator can inspect and accept the new route. The following figures show two sequence charts. Figure 11 shows the normal case where the edge-based service is available, and Figure 12 shows undesired case where additionally the edge service is unavailable. In the latter case, the Navigator is informed in order to take appropriate action such as manual route calculation or executing a full stop.

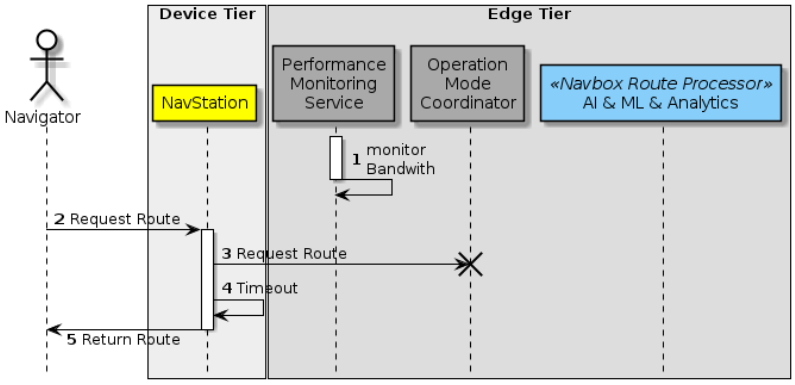

Figure 12: Edge-based maritime Advisory Service – Edge not available

6.2.5 Scenario 4: Cloud-based maritime advisory service

This scenario highlights the cloud-based maritime advisory service. It is similar to Scenario 3, however, the cloud-based advisory service is available and as such the Advisory Request/Response Manager (*Performance Monitoring Service & Operational Mode Coordinator* in edge tier) is able to contact the cloud-based service with a sufficient performance and latency via an API to the Operation Mode Manager (cloud tier) in order to save the request on the ADV Storage (*Data Manager*) and inform the ADV EventGrid about the new request.

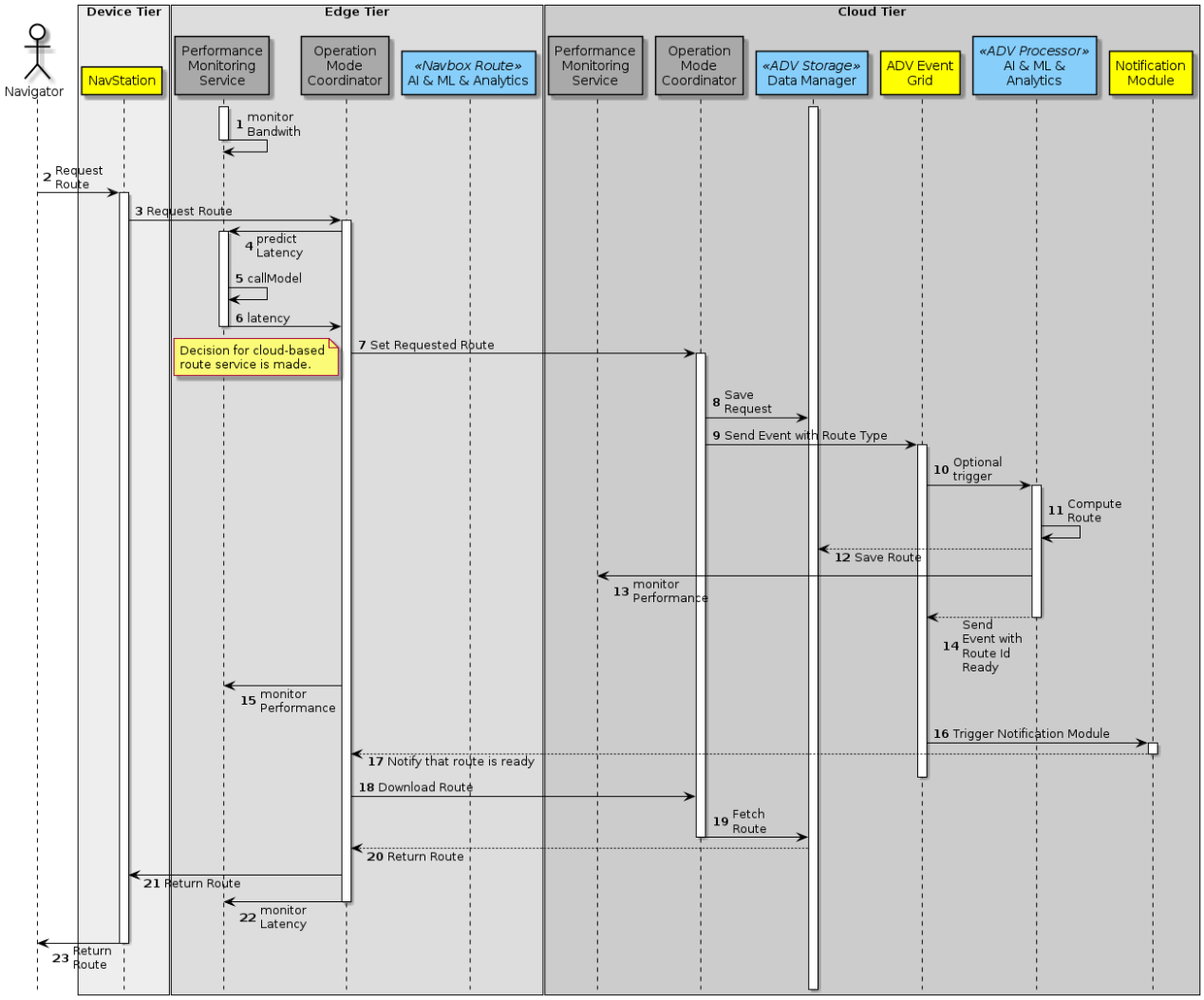

Figure 13: Cloud-based maritime Advisory Service

The EventGrid will activate the ADV Processor (*AI & ML & Analytics*) which calculates a new route, saves the resulting route to the ADV Storage and informs the ADV EventGrid about the calculated route. The ADV EventGrid triggers the Notification Module to inform the Advisory Request/Response Service *(Performance Monitoring Service* & *Operation Mode Coordinator* about the availability of the calculated route which, in turn, downloads the route via the Internal API from the ADV Storage (*Data Manager*) and makes the route available to the Navigator via the NavStation. Just like in Scenario 3 the navigator's task is to inspect and accept the calculated route. This activity is visualized in Figure 13.

6.3 Change of operational mode

Distributed cyber-physical systems based on the TRANSACT device-edge-cloud continuum combine physical components and computing components for their operation. It is well known that the high variability of the environment can lead to unpredictable failures. To adopt a CPS to unexpected or newly evolved environmental needs, one way is to adapt its configuration. This configuration must remain consistent across the device-edge-cloud continuum, therefore the transition between the configurations must be performed in a safe and coordinated manner.

It follows from the needs of the TRANSACT use-cases and their requirements that the behaviour of a safety-critical application should be adapted at runtime. This capability is intended to avoid or mitigate system failures, or just to ensure minimum levels of QoS, such as maintaining service continuity when application responsiveness must be guaranteed.

This section presents a solution to adapt the functionality of a system based on multi-mode applications by using the TRANSACT reference architecture, as a method to adapt the behaviour of the application in the device-edge-cloud continuum.

6.3.1 Workflow

Multimode applications exhibit specific behaviour for each operational mode that cover the different needs during execution. Through these multimode applications and the management of their operational modes, it is not only possible to manage the system malfunctions, but also to define the behaviour that the system must have in its normal execution phases.

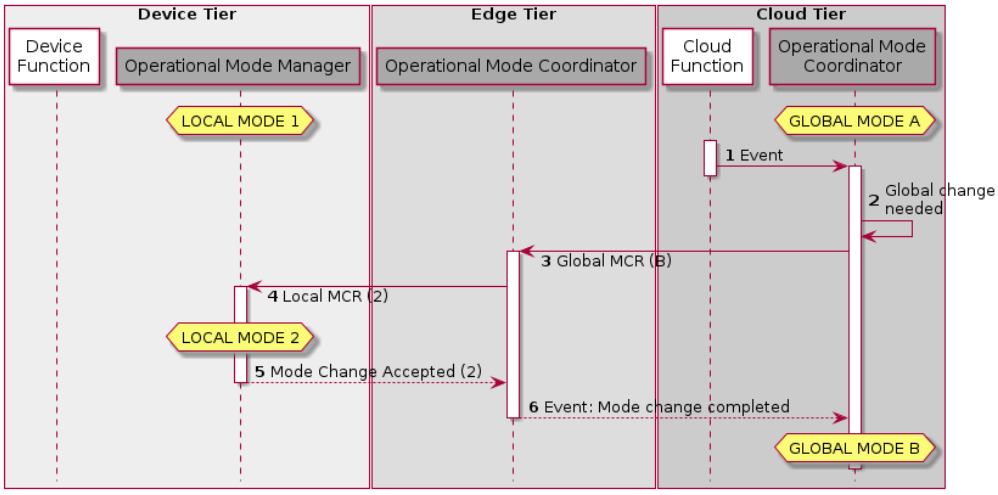

Figure 14: Change of operational mode workflow. Event captured from cloud

In the case of distributed systems, the difficulty of implementing this type of application increases. This is due to the necessary synchronization between the different nodes during the transition from one operational mode to the next. It is at this point that a coordination mechanism becomes essential.

As shown in Figure 14, an event captured in the cloud (1) can define the need for a behavioural change at the global level (2). For this change to become effective, the Mode Change Request (MCR) must be propagated from the cloud to the edge (3) and to the different devices (4). A device can define different operational modes to those defined at the system level, so it is necessary to differentiate between global

modes and local modes. The coordinator at the edge is in charge of translating which local mode in the device should be activated in each global mode of the system. Once all the devices have been configured with the required local mode (5), the global mode can be considered to have been reached (6) and the behaviour of the system has been updated.

On the other hand, as depicted in Figure 15, the need for an operational mode change can also be detected locally on a device (7, 8), that means the device must notify to the edge and the cloud of the mode change (9, 10), in order to take the necessary actions at the system level (11, 12), as it may induce a global mode change.

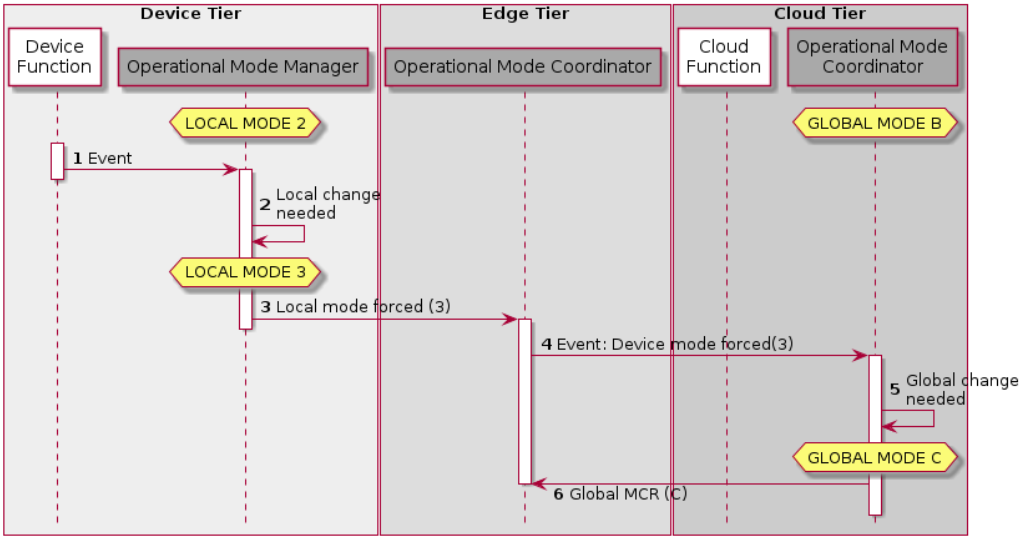

Figure 15: Change of operational mode workflow. Event captured from device

From the point of view of the architecture components involved in this workflow, all of them with the role of Analyse (A) in the MAPE-K cycle can generate events or alarms to notify a change in the environment that must be managed or mitigated. Such as *Monitoring Services* (Sections 0, 5.2.1.1and 5.3.1.1), *Data Manager* (Sections 0 and 5.2.2.1) and *AI & ML & Analytics* (Sections 5.1.2.3 and 5.2.2.2). These events or alarms are based on rules that use metrics generated across the entire device-edge-cloud continuum, including Core services, Value-Added services and domain specific functions.

Operational Mode Coordinator in the cloud tier (Section 5.1.1.4) and in the edge tier (Section 5.2.1.4) are responsible for generating and propagating a Mode Change Request (MCR) in the appropriate moment, i.e., when an event or an alarm must be processed depending on the defined plan. In addition, *Operational Mode Manager* at the device (Section 0) receives MCRs and is in charge of managing the transitions between operational modes when the mode change is performed.

Regarding the related solutions defined in WP3, this workflow fits in the following concepts defined in D3.1:

- *Concepts for operational modes and change transitions*
- *Concepts for predictable performance*
- *Concepts for platform service level specification*
- *Concepts for safety-critical platforms*

In addition, this workflow is related with the following solutions defined in D3.3:

- *Solutions for operational modes and change management*, including *Mode change management on the device* and *Mode change coordination*.
- *Solutions for predictable performance*, including *Performance modelling and prediction* and *Performance management*.
- *Solutions for safety and health monitoring, risk analysis, and safety and security assurance*, focusing on *Mapping and scheduling techniques across device, edge, and cloud*.

Since a mode change means change of the behaviour, ensuring that the modes are well aligned across the different tiers can be a safety-critical. The mode specification, i.e. "what behaviour does the system show in which mode", is a prerequisite for a mode management. To coordinate the modes across all tiers, the mode specifications of the different systems/components hence must be made available to the mode manager.

Considering the potential risks in implementing this workflow, one of the most important aspects is system heterogeneity. An operational mode in the cloud may not correspond to the same operational mode at the edge or device, or different legacy devices may have different modes of operation. A possible solution could be to define adaptation layers between each device and the edge, or between the cloud and the edges. This adapter would be in charge of translating a transition in the upper layer to the necessary actions or transitions in the lower layer.

6.3.2 Scenario 1: Change of operational mode to enable cloud supervision

Distributed applications must maintain consistency across the device-edge-cloud continuum, ensuring that each part behaves as expected by the rest of the system. Therefore, global guidelines and plans must define the local behaviour of each of the devices.

In this scenario, *Operational Mode Coordinator* in the cloud defines the global plan as a list of rules configured to respond to events and alarms generated by the different services. The *Operational Mode Coordinator* at the edge receives the mode change request and coordinates devices connected to it.

A transition between two operational modes is depicted in Figure 16. The *Monitoring Service* collects the tracepoints (1, 2, 4, 5) from the application (*Non-Critical Function*) and generates the metrics (3, 6) with the parametrised behaviour. These metrics are analysed by the *Analytics Service*, which causes an alarm (7) to notify a change in behaviour. The *Operational Mode Coordinator* in the cloud tier receives an alarm that implies that an update of the global operation mode (8) of the system is necessary. This Coordinator initiates a process to request edge for the expected mode change (9) and waits for it to be completed (16), to ensure that the entire system is synchronised and consistent.

The *Operational Mode Coordinator* at the edge is in charge of configuring the devices according to its plan. Because a device may define different operational modes than those specified at the system level, this Coordinator must define which local mode for each device should be requested to comply with the desired global mode. In Figure 16, *Devices 1* and *Device 2* should be configured in *Local Mode 2* (10, 11) and *Device 3* needs to change from *Local Mode 5* to *Local Mode 4* (12). Once all the devices are in the expected mode (13, 14, 15), The Coordinator at the edge can notify to the Coordinator in the cloud to complete the mode change operation (16).

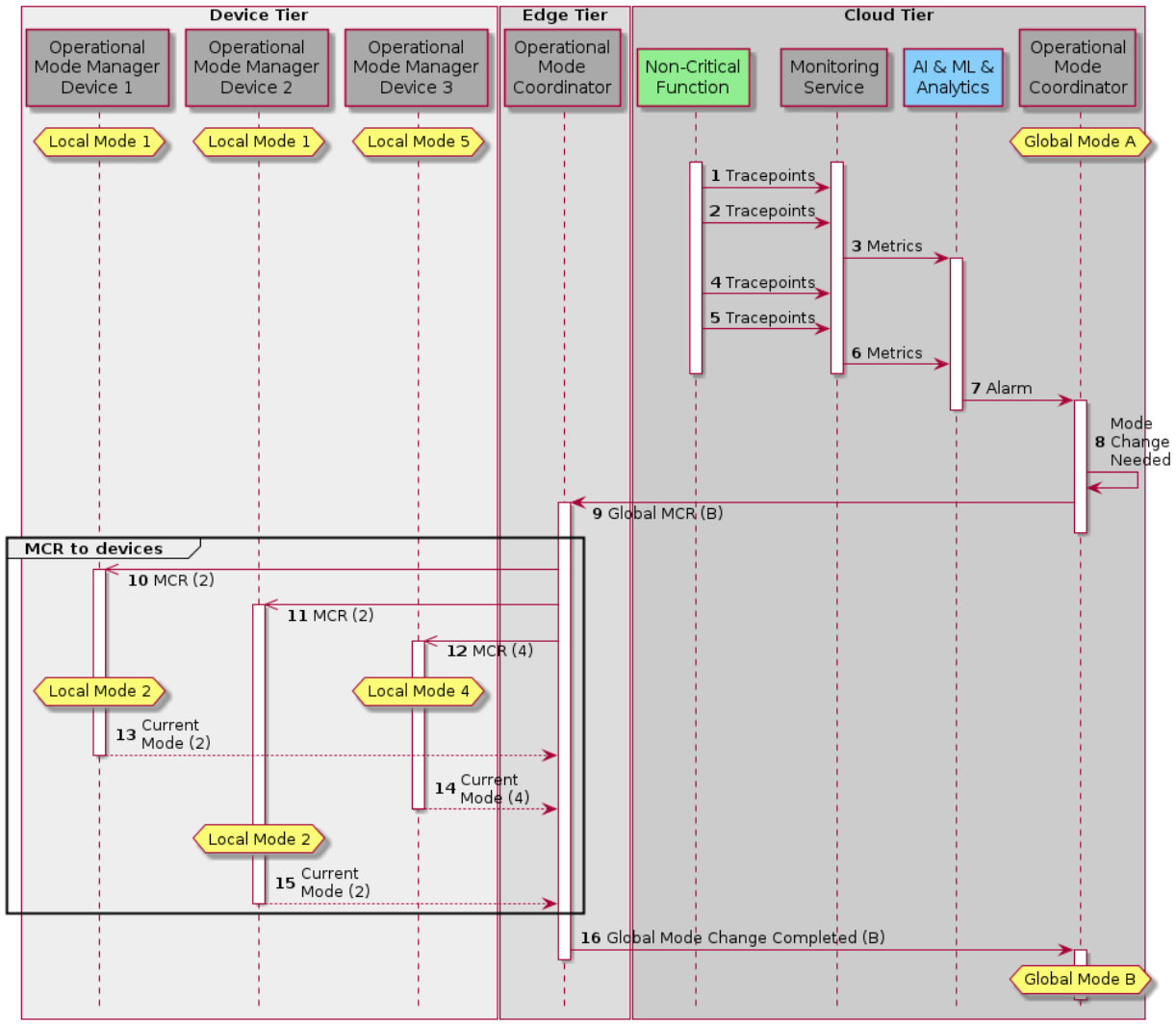

Figure 16: Change of operational mode to enable cloud supervision

6.3.3 Scenario 2: Change of operational mode when losing connection to the cloud

In many cases, distributed applications on the device-edge-cloud continuum are designed to provide cloud services to enhance capabilities at device-edge level. These value-added services in the cloud allow the execution of powerful algorithms that would be not possible otherwise, as in the case described in UC4, where cloud-based solutions are used to enhance image guide therapy procedures.

Safety-critical and mission-critical functions in device-edge tiers usually require full availability, but cloud services do not ensure this capability (Figure 17 – Normal operation). In case of a lack of connectivity between the edge and cloud or a service in the cloud refuses a request due to overload, the availability must be guaranteed in the device and edge tiers (Figure 17 – Connection lost).

The proposed solution involves modifying the behaviour of the system, allowing the functionality that cannot

be executed in the cloud to be run on the device or the edge. This fallback mechanism will maintain the availability of the service but with a reduced quality due to the lack of computing power (Figure 17 – System adaptation).

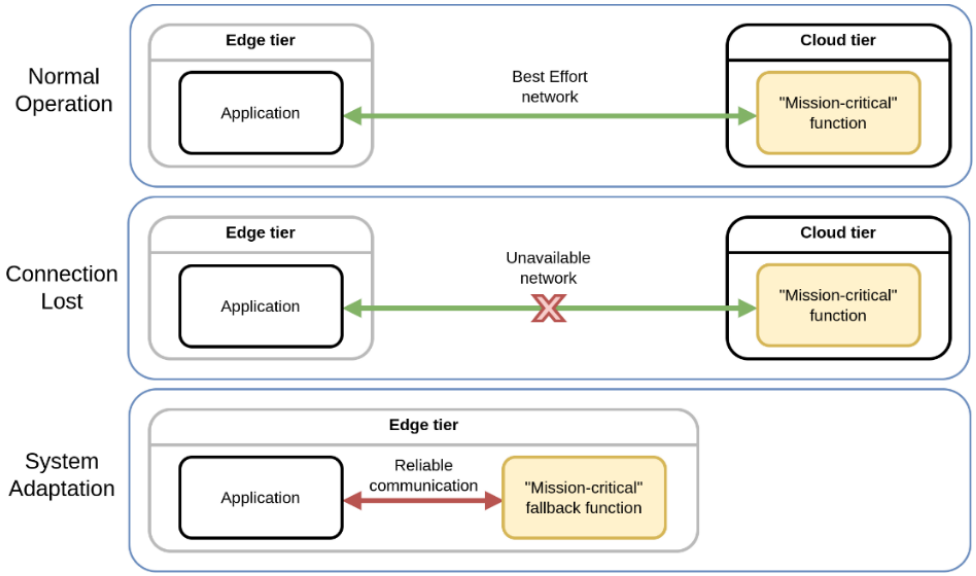

Figure 17: Change of operational mode when losing connection to the cloud

In the specific case of this scenario, the imaging component device captures images of the patient. These images are processed to generate a 3D image reconstruction and are displayed to the user. In the *Normal Operation* mode (Figure 18), *Operational Mode Coordinator* at the edge enables the cloud processing (1) to send the images to the cloud. Initially, raw images are sent from the Scanner Interface to the edge Data Manager (2 and 3), that in this case is responsible for safely storing and making available images from end-devices. Image Manager recovers these images to send them to the 3D reconstruction cloud service, which returns the 3D image to the Image Manager and are stored in *Data Manager* (4 to 13). Finally, this 3D image is shown in the Scanner Interface (14). The computing power of the cloud allows the image processing service to run powerful algorithms that otherwise could not be executed at the edge itself.

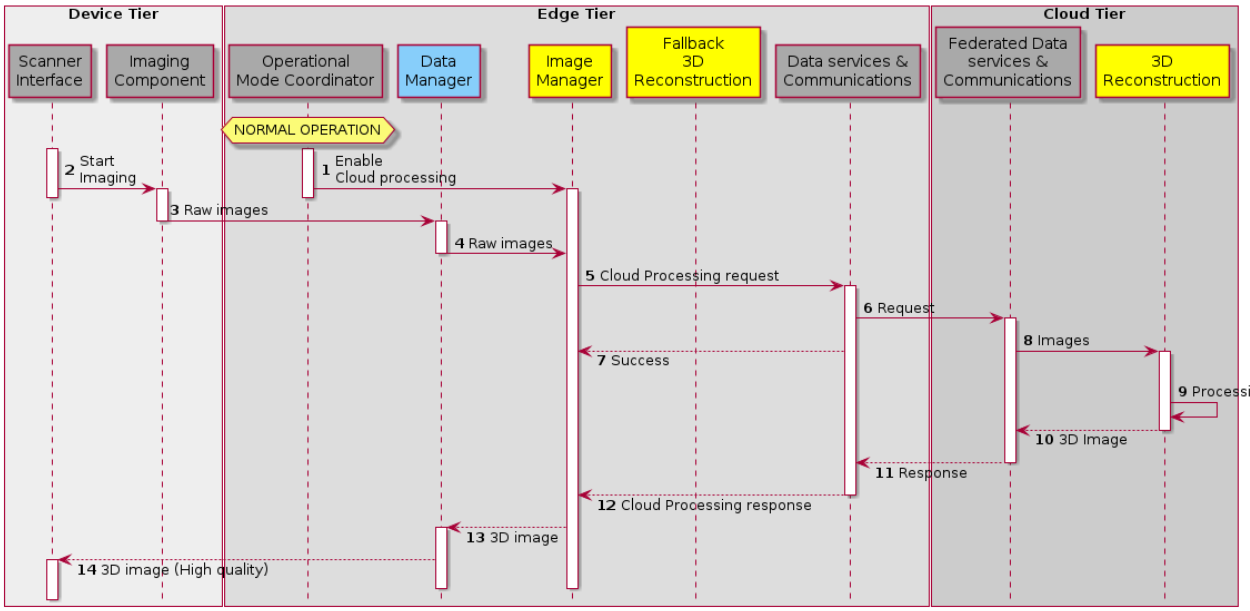

Figure 18: Change of operational mode when losing connection to the cloud - Normal operation

If connectivity with the cloud is interrupted (1 to 9), the availability of the 3D images must be ensured (Figure 19). As a result, the Edge *Operational Mode Coordinator* activates the *Isolated Operation* mode (11), updating the application behaviour by activating two new tasks at the edge, one for local processing of the images (12) and another to try to reconnect the device with the cloud (14). Once the connection has been re-established (21), the *Normal Operation* mode is enabled again (22-24).

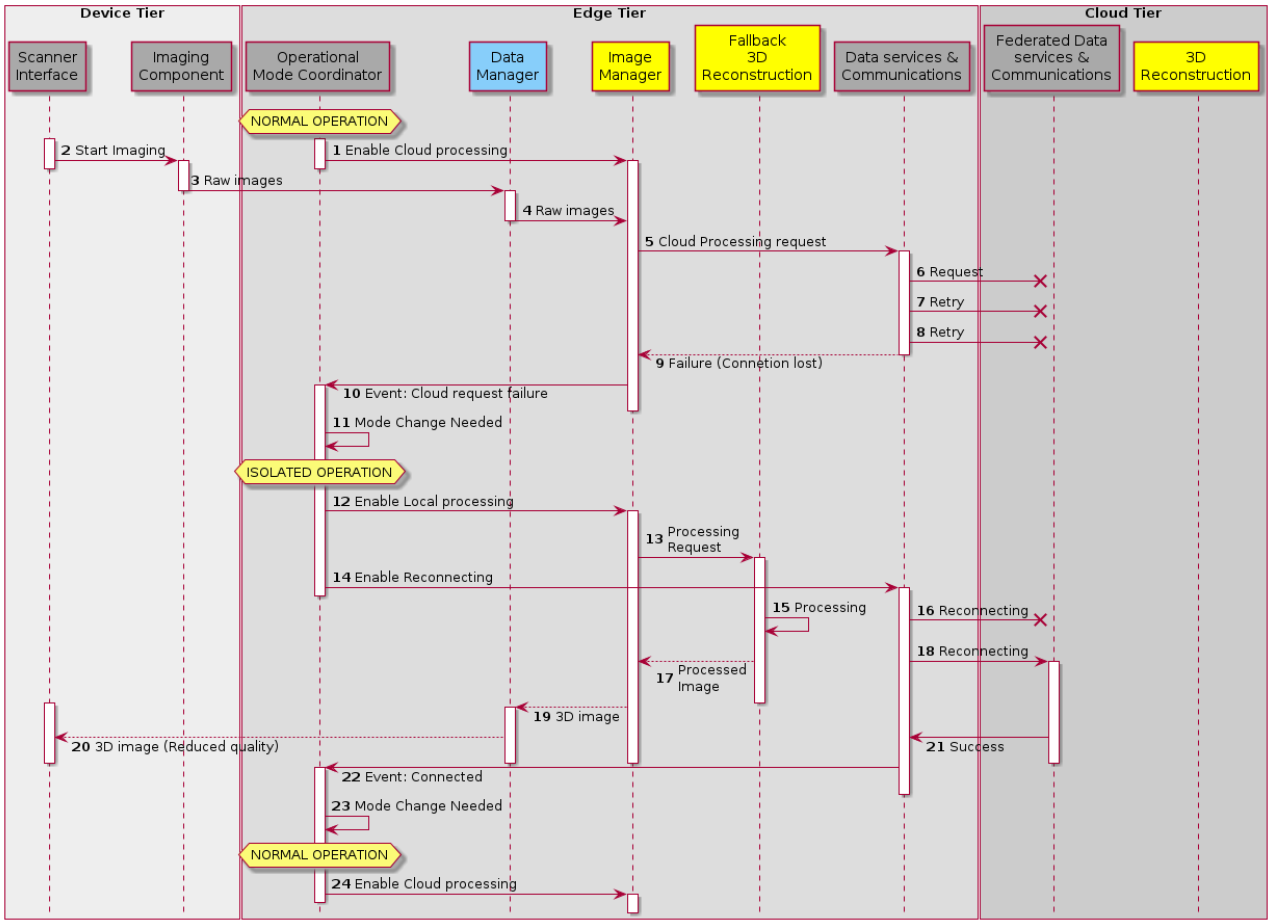

Figure 19: Change of operational mode when losing connection to the cloud - Isolated operation

6.3.4 Scenario 3: Change of operational mode when device detects a failure

The behaviour of the distributed system should include not only the option to coordinate mode changes from edge and cloud tier but also be able to request changes from a device, such as in the case of fault detection.

Figure 20 depicts a scenario when a behavioural change is forced from one of the devices. A failure is detected locally in a device (2), and for criticality reasons, the device must react without waiting for the rest of the system, causing a change of local mode.

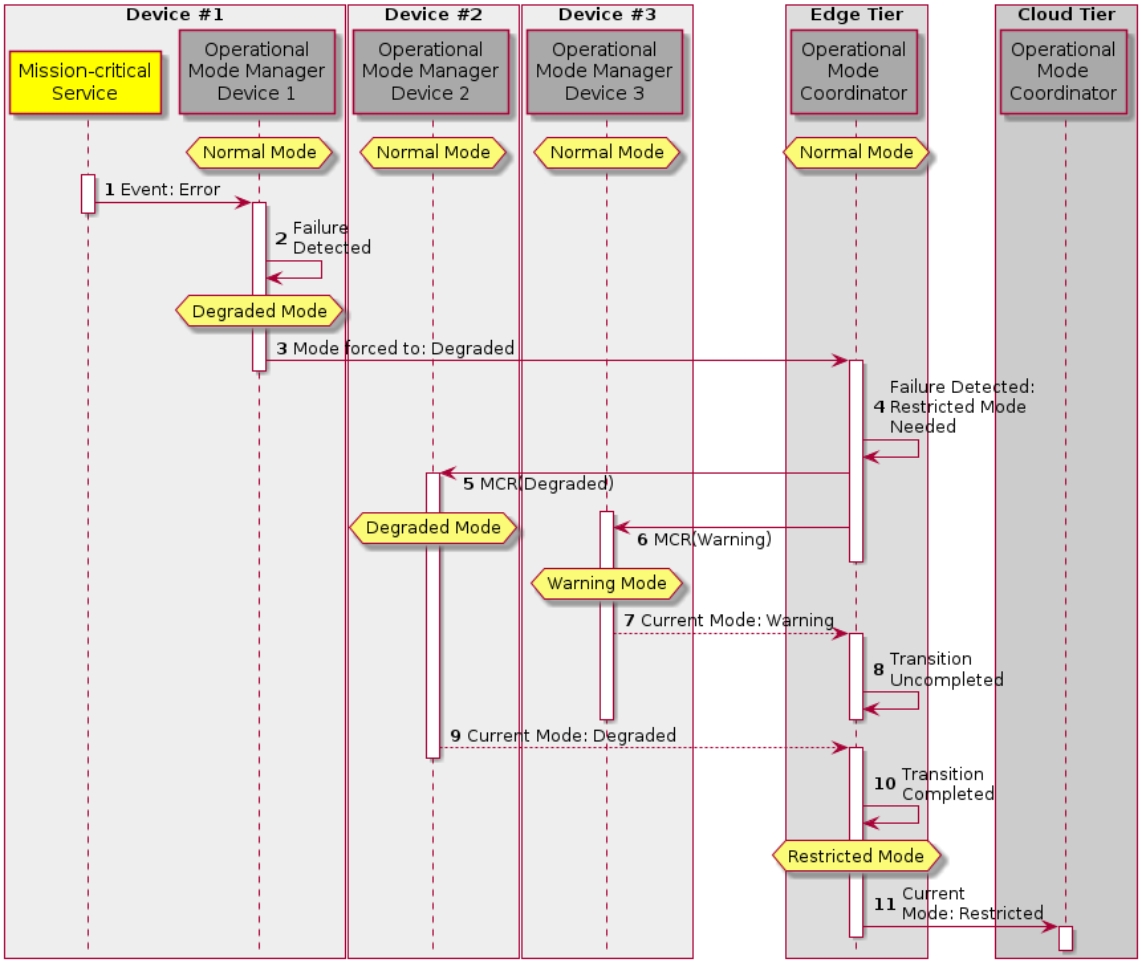

Figure 20: Change of operational mode when device detects a failure

The device notifies the edge (3) propagating the current operational mode, to ensure consistency in the execution of the distributed system. The Edge *Operational Mode Coordinator* must take the appropriate action to mitigate the detected failure. In this case, the coordinator detects the need to change the global behaviour of the system (4).

The operation mode of the devices must be updated (5,6) to be consistent with the global mode needed (4). When all the nodes of the system are in the requested mode (10) it will be considered that the global mode has been reached. In this case, to maintain consistency, the Cloud *Operational Mode Coordinator* should be informed of the new mode reached (11).

6.4 Remote updates

In distributed cyber-physical systems (CPS) based on the TRASACT device-edge-cloud continuum functionality is split into a set of components, allowing for each component to be developed independently, often by different teams, and have its own life-cycle and release schedule. In such scenario, updates of the different components will inevitably be necessary and must be precisely planned and coordinated, to maintain inter-component compatibility at all times. Having a single coordination point for all software updates enables

platform operators to plan and execute updates of multiple components in an orderly manner. Also, devices are usually located in many different locations or may not be easily accessible, so an operator applying updated on-site is not a cost-effective option or, in some cases, not an option at all. The capability to perform updates remotely via a network connection becomes paramount.

This section describes how a software update is planned by an operator and applied to a set of devices using the Remote Update components of the TRANSACT reference architecture.

We consider an Operator to be an entity, a human or an automated process, in charge of registering new software versions and deciding what version is used in each device.

6.4.1 Workflow

The *Remote Update Coordinator* plays a critical role within the TRANSACT reference architecture. Its primary purpose is to act as the central repository of software version information for all the different devices involved. By acting as the source of truth, the Coordinator ensures that all devices have access to the latest software versions approved by platform Operators.

To feed software version data into the Coordinators, there are two main methods. First, information can be automatically fed into the Coordinator by configuring connections with CI/CD pipelines or software trains. This allows for seamless integration and ensures that the Coordinator is always automatically aware of the latest software versions. Second, operators can also manually upload or configure new versions in the Coordinator. This flexibility allows for greater control and fine-grain customization when necessary. New services can also be published via the Coordinator, so it becomes available for devices to install and use them.

When a device needs to determine the appropriate software versions to run, it will reach out to a Coordinator located either on the edge or cloud tier. The Coordinator acts as a point of contact for devices, providing them with the necessary software versions information. If a newer version is available for any of its software elements, the device will retrieve the update package and proceed with the installation.

To deal with the diverse capabilities of different devices, the Coordinator should support various modes of interaction. Devices can make individual requests for each software they are running, managing each software in a more granular way or. alternatively, they can make a single request for the entire list of software they are interested in, receiving a full list, a desired state, that details the expected version for each software element. This flexibility allows devices to choose the most suitable approach based on their specific requirements or capabilities.

In three tier architecture scenarios, *Remote Update Coordinators* can be deployed in both the cloud and the edge tier. In such cases, the Coordinator in the cloud tier serves as the ultimate source of truth. The edge tier Coordinators act as delegates, mirroring all the software version data configured in the cloud Coordinator. This mirroring or synchronization process ensures that the edge Coordinators stay in sync with the latest software versions. The synchronization can be achieved autonomously, with the edge Coordinators periodically polling the cloud Coordinator for updates. Alternatively, an operator can manually trigger the synchronization by sending a "synchronization request" to the edge Coordinator.

Devices can be either active or passive in terms of how they will receive updates. Active devices will take a proactive approach by periodically polling a *Remote Update Coordinator* for information about the software versions they should be running. If new versions are available, they will automatically initiate the download and installation process. On the other hand, passive devices rely on platform Operator to send them Update Request messages. These messages specify the updates that need to be applied, and the passive devices will simply be waiting for these instructions and, once they receive them, start the necessary update processes.

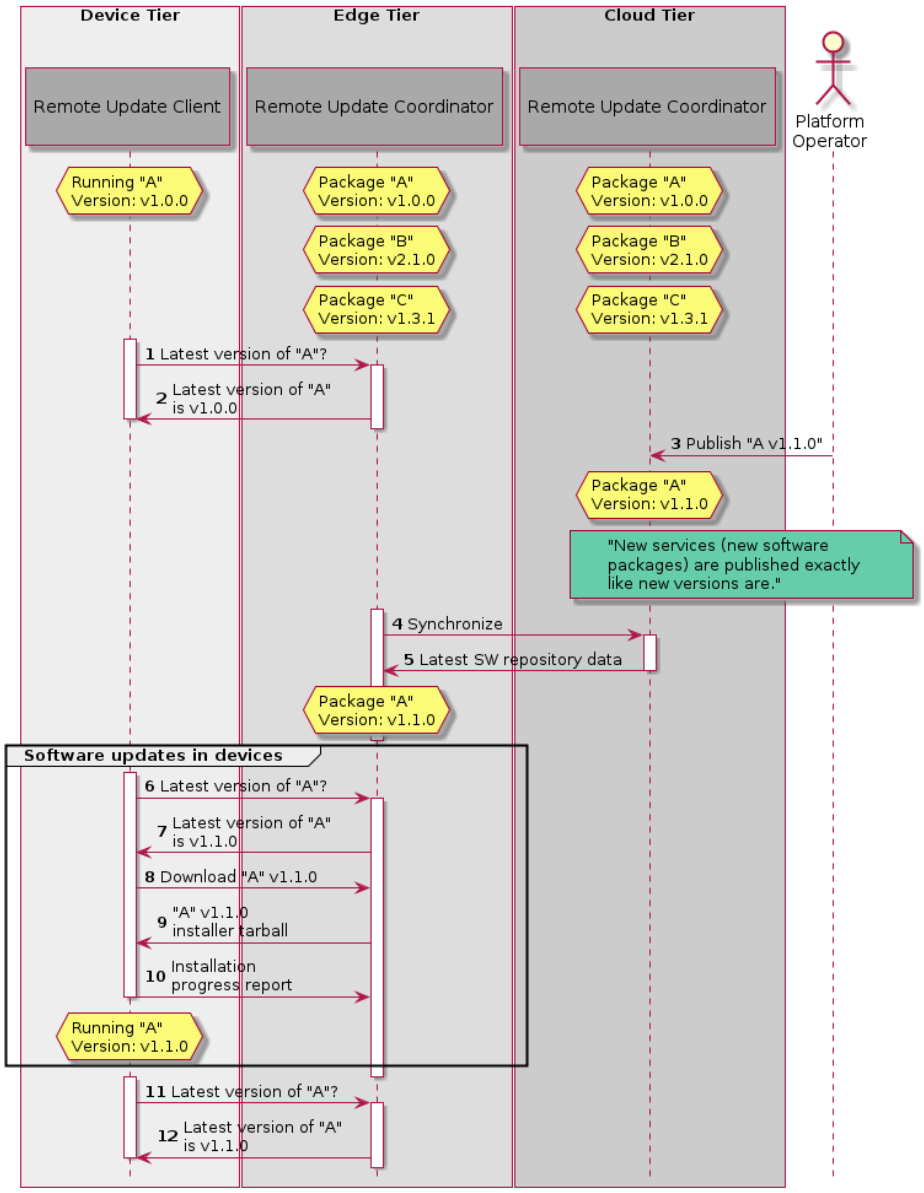

Figure 21: Remote Update System workflow in automatic mode (polling)

When Devices apply software updates, they must report some kind of update progress to the Coordinators to keep them informed about the status of the update process. Knowing this status allows Coordinators to have real-time visibility on the progress of each update, enabling them to take appropriate actions should any issues arise. Reporting the software update progress to the Coordinators, also enables more precise monitoring of the process. The Coordinators have accurate information and are able to track the overall progress, identify any possible issues, and take appropriate actions to correct them. The reporting mechanism should be reliable so no important information is lost and based on the communications capabilities of both Device and Coordinator.

Devices should periodically send status updates to the Coordinator, including progress of the update as percentage of completion, current stage of the update process (downloading, installing, verifying, applying, self-checking), output messages of the process or errors encountered. If any errors or issues occur during the

update process, Devices should immediately report them to the Coordinator. This will allow the Coordinator or an Operator to take appropriate actions, for example initiate a rollback to a previous version.

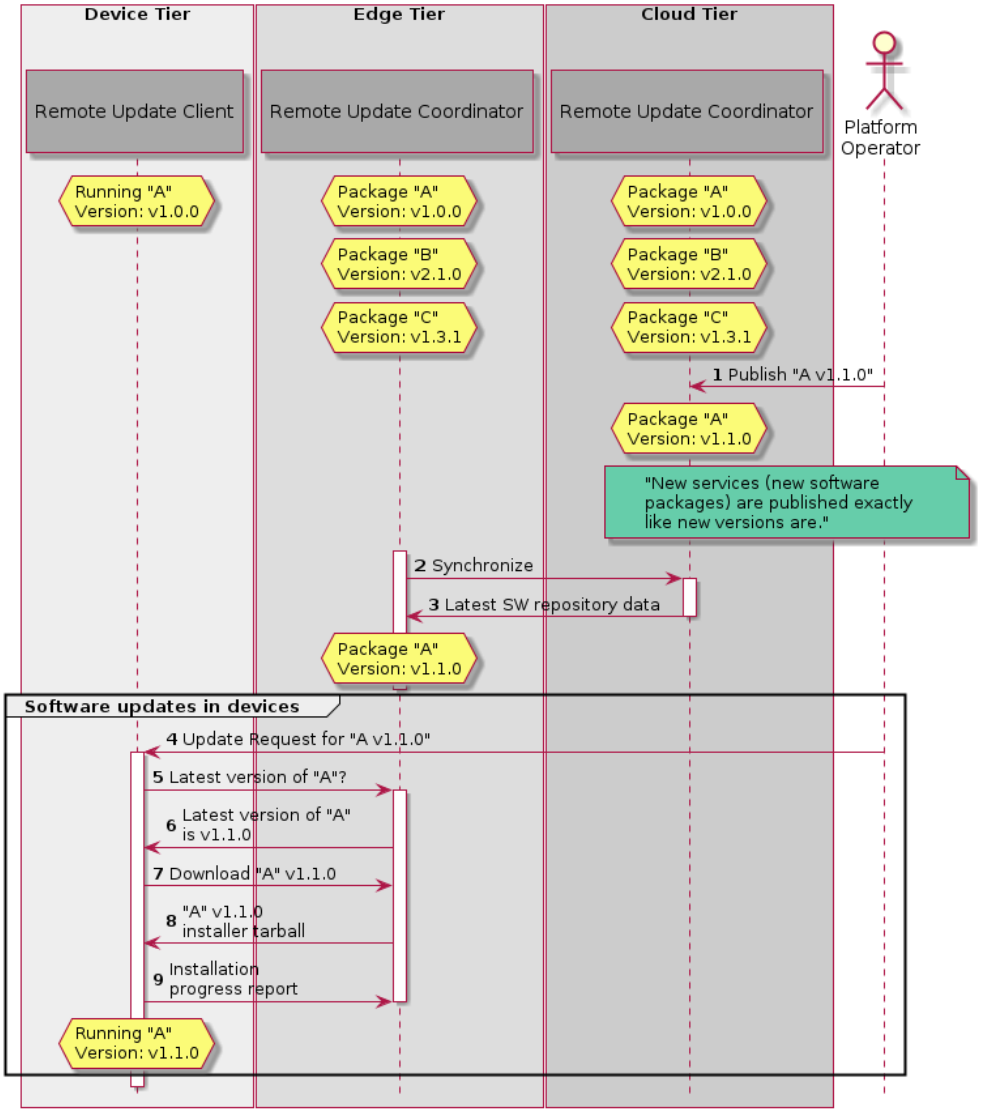

Figure 22: Remote Update System workflow in manual mode

6.4.2 Scenario 1: Remote software update of vessel side devices

In UC2 the device-cloud-edge continuum consists of the Remote Update Client- NavStation navigator's planning station (device), the *Remote Update Coordinator - NavBox* (edge) and the NavCloud (cloud). TRANSACT remote update solutions leveraging the full continuum is employed for this task.

The NavStation software updating process includes the following steps, depicted in Figure 23 and Figure 24:

1. New NavStation software version is added to the NavBox cloud side back-end using a cloud side provisioning tool. Currently all NavStation software versions metadata is stored in the ERP Database, the binaries itself are stored in a Blob Storage.

2. Then one can generate NavStation Update Request from the provisioning tools (for a single NavBox or for a group of NavBoxes).
3. The provisioning tool through the NavBox back-end sends a command to *Remote Update Coordinator - NavBox* to initiate updating process (in case of group updating, a separate command is generated for every *Remote Update Coordinator - NavBox* in the group). The *Remote Update Coordinator- NavBox* checks the version of the software available on disk and sends a *Remote Update Client- NavStation* Update Request to the server, specifying current NavStation version on the NavBox and the requested NavStation version.
4. NavBox back-end receives the NavStation Update Request and produces a response message containing the software update. Then it sends it to the *Remote Update Coordinator- NavBox*.
5. The *Remote Update Coordinator- NavBox* receives the message with the software update and verifies it. When the software update verifies successfully, earlier version of the NavStation uptate is removed from the NavBox.
6. NavStation Software is available through the NavBox Public REST API, which allows to enumerate available software versions (currently, no more than one NavStation version is available at any time), get list of content of the installation, fetch the software installation binaries.
7. Installed NavStation on a client computer checks periodically available version on *the Remote Update Coordinator- NavBox*, and if a newer version is available, it prompts the user to install it.

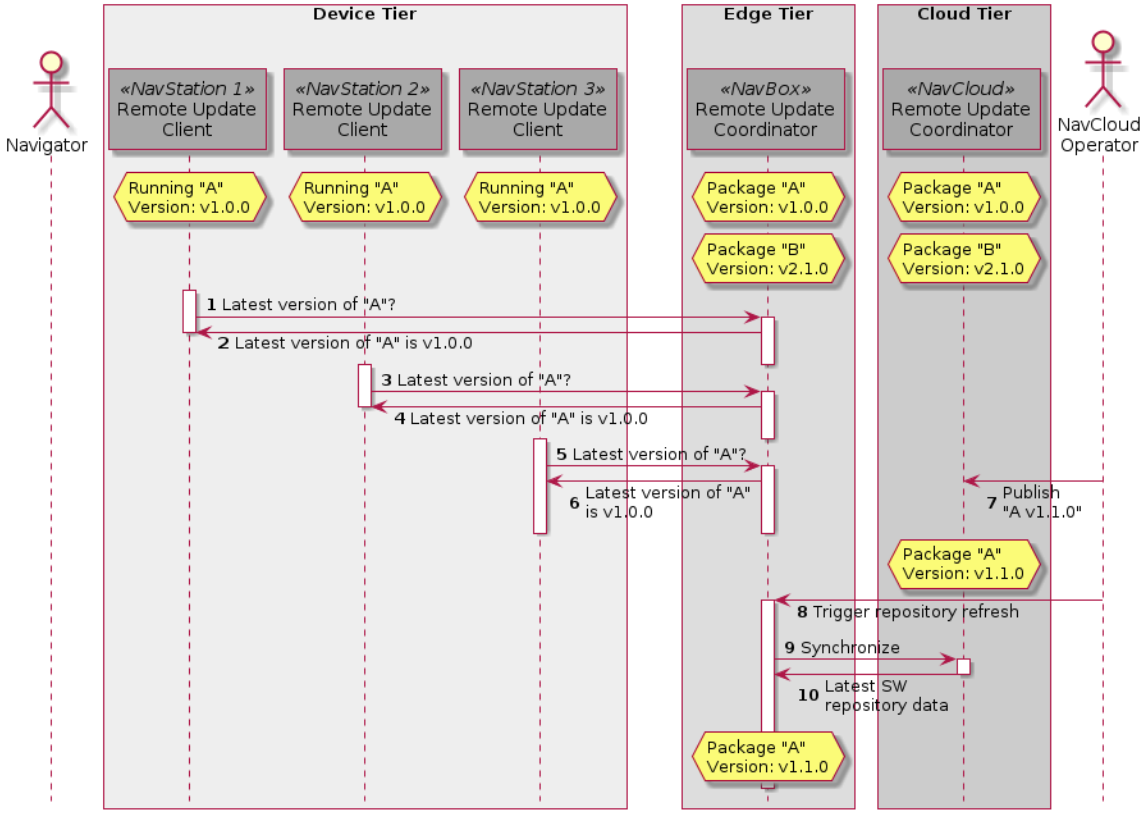

Figure 23: NavStation – New version of Package "A" released

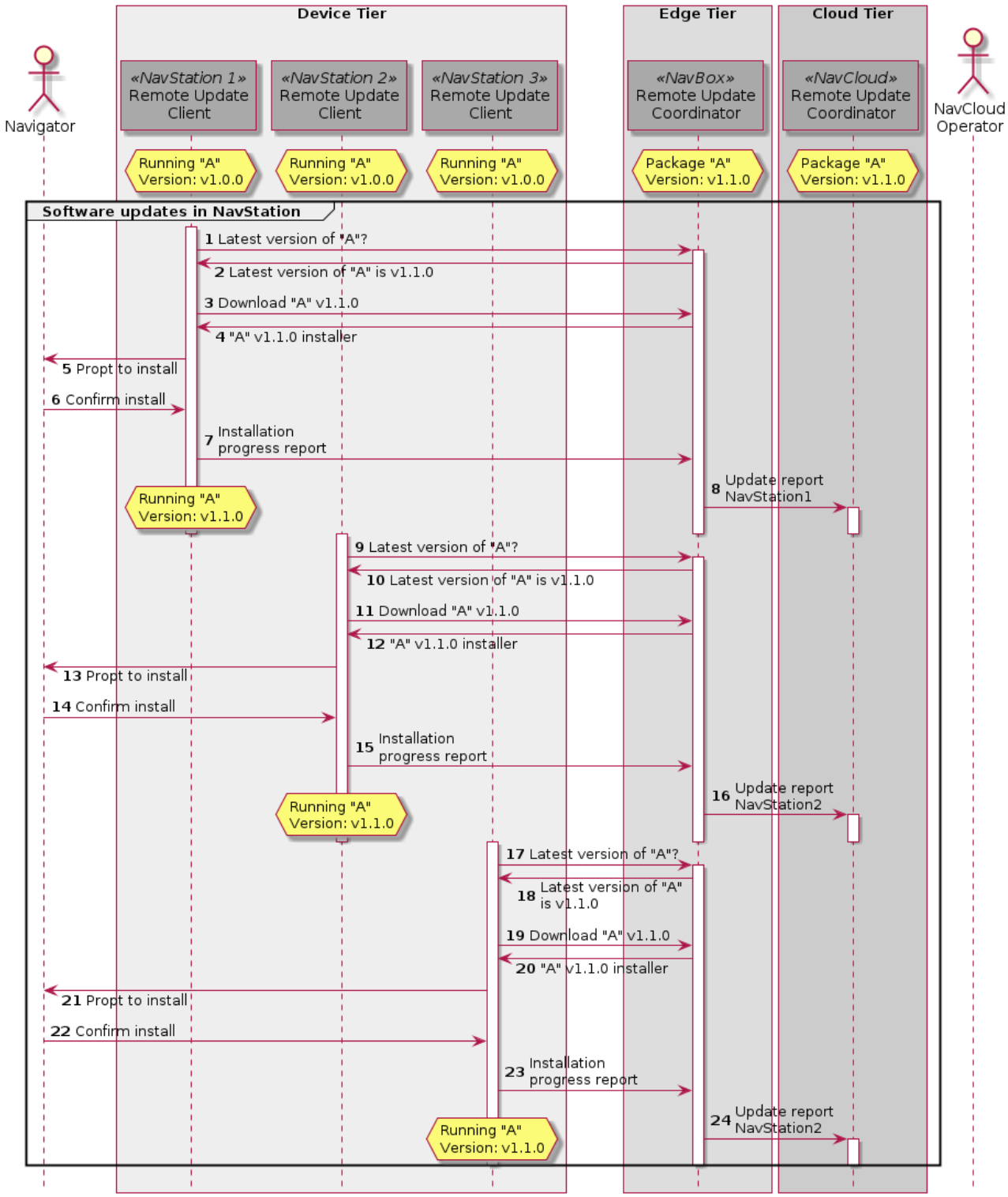

Figure 24: NavStation - Software update workflow

6.4.3 Scenario 2: Secure proximity update of wireless IoT sensors

Secure onboarding of wireless sensor devices via wireless charging (NFC) (Figure 25). This tool aims at being a data provider for other systems in a local cloud. It also exposes its public key over the NFC interface for onboarding the following features:

- NFC-based IoT sensors with base station
- Advanced security with secure element (i.e. NXP SE050)
- Additional feature: NFC-based wireless charging
- NFC Interoperability evaluations

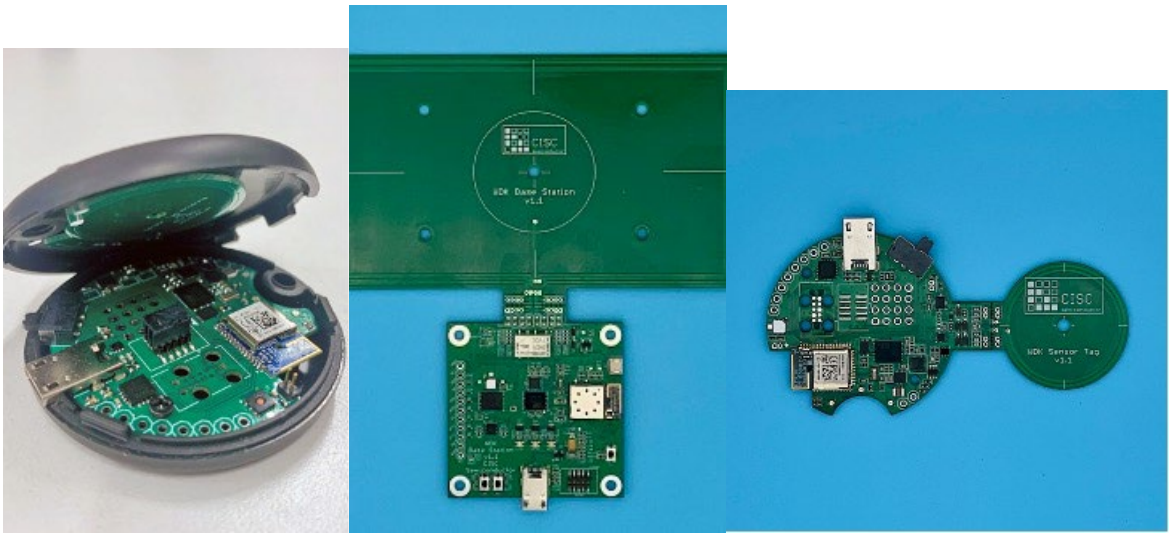

Figure 25: Wireless IoT sensors

The remote update feature is currently integrated on the Field Gateway (edge tier) hardware for interfacing the wireless sensors (device tier). This will enable discovery, authentication and registration of new sensor nodes, key exchange, and their configuration (Figure 26).

The onboarding process consists of the IoT node (to be attached to the existing infrastructure), on-boarding application, management of the cloud and an edge gateway. The device exposes its credential (e.g., public key) over NFC interface to the onboarding application. The onboarding application has a GUI for configuration of access rights of the node to particular clouds (one or more). The management infrastructure is a tool enabling remote supervisory control over edge gateways (being implementation and the central point of local clouds) and indirectly – over the edge nodes. The management backend sends the information whether a node should be updated and authorized to access the local cloud, and what are its credentials. After the node is admitted communicating with the cloud, it is powered on and in proximity of the field gateway, it enters auto discovery mode and established the connection with the edge gateway, registering in the cloud. This concludes the onboarding process, and shows the functionality of the onboarding toolchain.

The main concept contributing to the reduction of engineering costs is the proposed onboarding procedure that automatizes the configuration, authentication and authorization of new IoT devices. The onboarding process is designed to be simplified enough that the visit of a qualified technician on site is not necessary. Furthermore, connection to the management infrastructure and dynamic reconfiguration of the device tier

from the cloud tier allows for remote diagnostics and adaptation of the device to new requirements (again – without the need of qualified technician's visit).

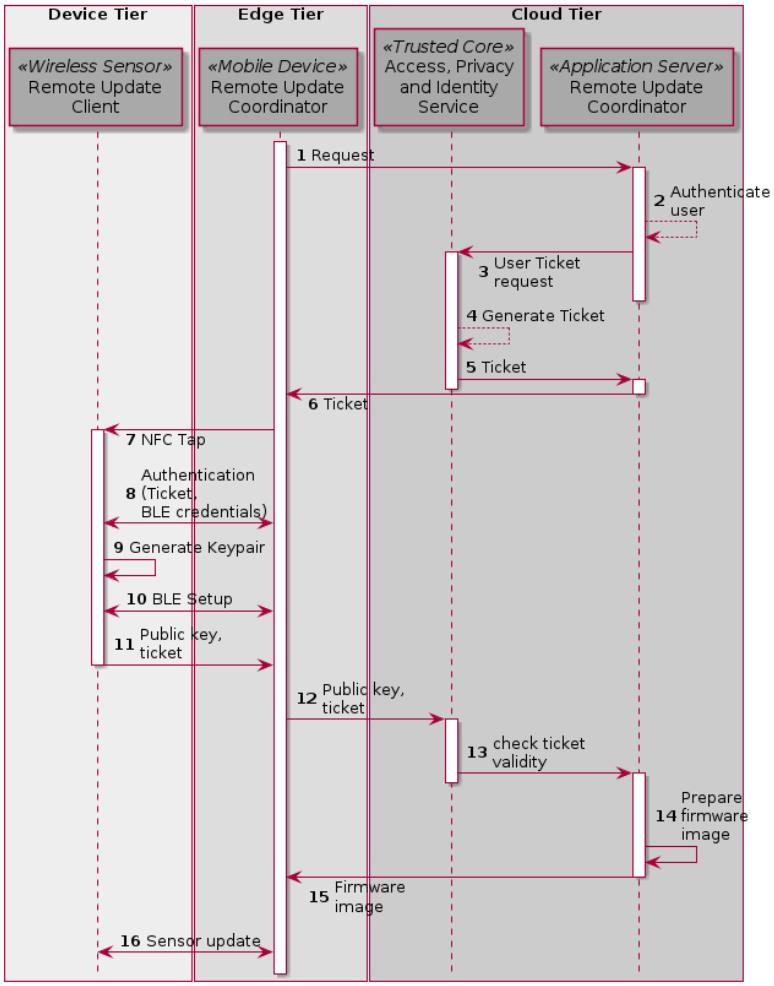

Figure 26: Secure proximity update

6.5 Access control

The TRANSACT reference architecture involves the design and deployment of different components, tasks, services, and functions that can manage sensitive information or control critical system processes. Safeguarding the integrity, confidentiality, and availability of sensitive information, as well as protecting the components and services deployed across the three layers (device, edge, and cloud), is one of the primary needs for implementing distributed safety-critical CPS solutions. By restricting access, unauthorized individuals are prevented from tampering, damaging, or stealing resources, thereby reducing the risk of security breaches and potential adverse consequences. Additionally, by implementing robust access control, specific policies and privileges can be established for each user or entity, ensuring proper resource management, and preventing abuse or malpractice.

This section introduces the access control approach within the TRANSACT reference architecture, along with illustrative scenarios inspired by the project's use cases.

6.5.1 Workflow

Due to the TRANSACT reference architecture's three-tier deployment of services and resources, it becomes imperative to incorporate a component in each tier that governs resource access. Fulfilling this role, the *Access, Privacy & Identity* component takes charge of managing access and operations on system resources. The sequential process involved when a user or device requests access to a system resource is illustrated in Figure 27. This control flow could occur in scenarios where a user retrieves information from a database or a sensor device transmitting data to a messaging broker.

As depicted in Figure 27, the requester (user or device) must possess a token (1) that enables verification of their privileges and permissions when interacting with a protected resource. This token can be acquired beforehand through an authentication process. Subsequently (2), the token is forwarded to the *Access, Privacy & Identity* component for validation and verification of the requester's privileges (3). If the requester possesses the necessary permissions, the operation on the resource is executed (4, 5, and 6). Conversely, if the requester lacks the required permissions, the request is denied (7 and 8).

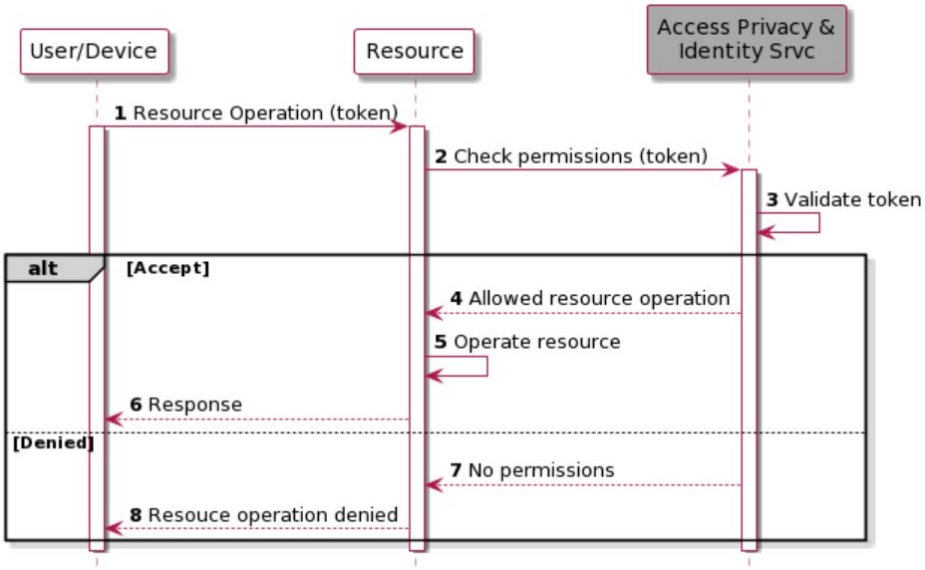

Figure 27. Access control workflow

6.5.2 Scenario 1: Role-Based Access Control (RBAC) for access control to sensor data

The first access control scenario is inspired by the TRANSACT UC5. Wastewater Treatment Plants (WWTPs) contain sensors to monitor variables to control the different physicochemical treatments (unit operations) and detect anomalous events. These sensors collect data continuously and are deployed in the unit operations of the plant. For example, the Font de la Pedra WWTP in eastern Spain (whose operation is composed of 8-unit operations in the water line and three in the sludge line) monitors variables such as pH, conductivity, total suspended solids (TSS), chemical oxygen demand (COD), Temperature, and Total Organic Carbon (TOC), in some of the unit operations. However, access to this data in real time (via the user interface) is restricted according to the role of the user or worker. For example, the chief plant operator should be able

to consult the data collected in all unit operations and manipulate the control systems (i.e., have permissions on most resources), but a sludge line technician should have access to limited resources (e.g., only permissions to consult certain variables).

The management of access to different system resources for this scenario is based on an RBAC approach managed mainly by the *Access, Privacy & Identity Service* component of the reference architecture. Figure 28 shows the sequence diagram for this scenario: to access the dashboard and consult the information collected by the sensors in real time, the *User* (e.g., a plant operator) performs the authentication process (1) to the *Web system*. The *Access Privacy & Identity* component then checks the user's credentials (2). Based on the RBAC rules previously configured by the system manager, the *Access Privacy Identity* component returns the access token (3) with the information of the resources authorized for the user according to his role. Then, the *Web System* performs the necessary queries to the database (4 and 5) to obtain the sensor data, build the visualizations, and display them to the user (6). Only the sensor data that the user has permission to view will be displayed. On the other hand, if the user/role does not have access to the system, the request is denied (7 and 8).

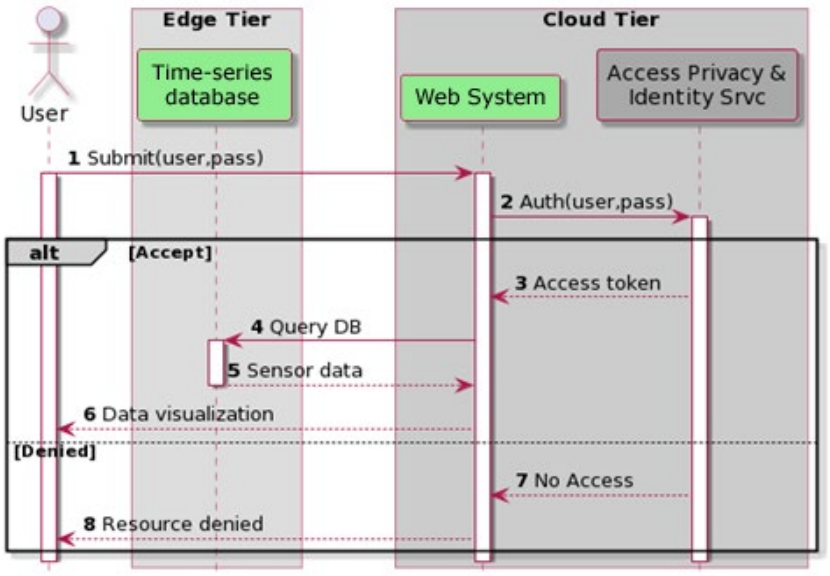

Figure 28: RBAC for access control to sensor data – scenario 1

6.5.3 Scenario 2: Token based access for battery management

This access control scenario is inspired by the TRANSACT UC3 (Cloud-featured battery management systems).

Today, the data produced by electrical vehicles, and especially by the battery management system, is stored on an offline data logger in the car – if at all. In irregular intervals these local and isolated data silos are read out via a wired connection. In terms of security/safety, the current standard is the use of a VPN connection established using VPN certificates stored on an embedded multimedia card. For each read-out of the logger and software update, the car must be handed out to a specialist. During this process, the car is not available for the owner.

UC3 aims to break up these silos. The data produced by a battery management system and additional vehicle information are forwarded to a central gateway within the vehicle via a CAN bus infrastructure. The gateway pre-processes the data and transmits it via a secure and wireless connection to a message broker. The broker in turn forwards the data – which can originate from different vehicles – to a central cloud. The data are further processed and stored in an optimized database. In the backend, the data are aggregated and made available for several purposes.

To secure the data transfer between vehicle and cloud measures apply that are illustrated below using the example of a telemetry data transfer from the vehicle to the cloud.

6.5.3.1 Onboarding and Registration of a Vehicle with the Cloud system

In UC3, access and rights control take place using digital certificates. For this purpose, all participants in the ecosystem must have a credible digital certificate assigned. For the gateway of a vehicle, this means:

1. Creation of an X.509 certificate[4] (per vehicle gateway).
2. Creation of a policy which determines access rights of the specific vehicle gateway (e.g., connect, pub, sub).
3. Creation of an AWS IoT Thing in the Cloud.
4. Assignment of a policy to an IoT Thing.
5. Assignment of the X.509 certificate to an IoT Thing.
6. Assignment of the vehicle's X.509 certificate to the corresponding policy.
7. Copying the gateway certificate, the CA certificate, and the private key to the vehicle gateway (e.g. secure memory).

The process above can also be partially automated to enable quick onboarding of new devices.

6.5.3.2 Access control with the transmission of telemetry data of a vehicle to the cloud

After the mutual authentication of a vehicle (edge) with the cloud, an encrypted transfer of data from the edge to the cloud takes place.

[4] https://www.itu.int/rec/T-REC-X.509

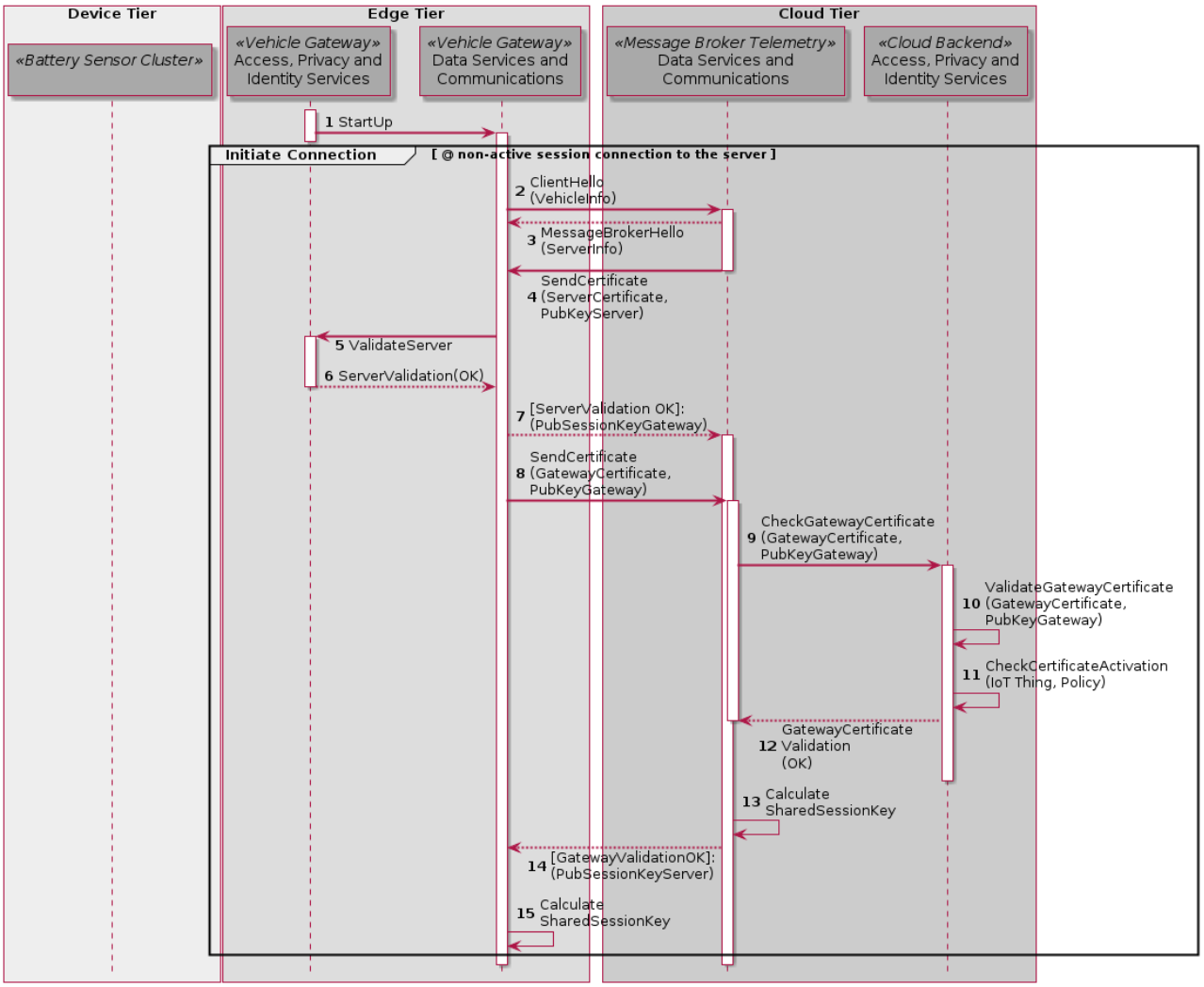

Figure 29: Access Control for Telemetry Data – Connection establishment (UC3)

The following entities are involved in this process:

Device Tier:

- «Battery Sensor Cluster», acting as *Safety, Performance, and Security Monitoring service* entity.

Edge Tier:

- «Battery Management System», acting as *Safety, Performance, and Security Monitoring service* entity.
- «CAN Bus», acting as *Data Services and Communications* entity.
- «Vehicle Gateway», acting as *Access, Privacy, and Identity Services* entity.
- «Vehicle Gateway», acting as *Data Services and Communications* entity.

Cloud Tier:

- «Message Broker Telemetry», acting as *Data Services and Communications* entity.
- «Cloud Backend», acting as *Access, Privacy, and Identity Services* entity.

The process itself can be depicted as follows. Before starting a data transmission, a connection between edge tier and cloud tier must be established (Figure 29). Therefore, the «Vehicle Gateway» in the edge and the cloud's «Message Broker Telemetry» in collaboration with the «Cloud Backend» authenticate each other via mutual Transport Layer Security (mTLS).
The transmission of the telemetry data is initiated by the vehicle since the cloud does not know when the vehicle can establish a connection to the server in the cloud.

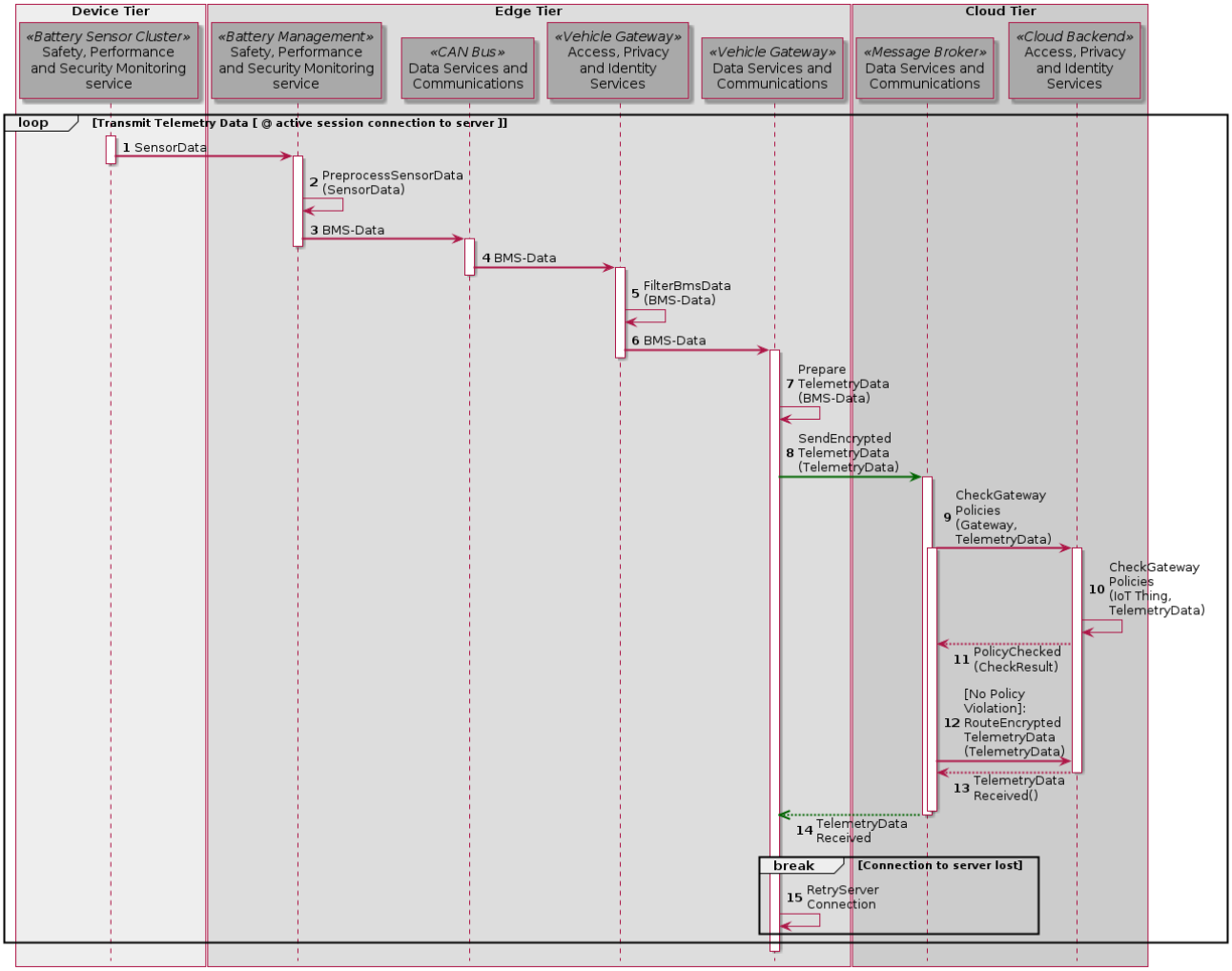

Figure 30: Access Control for Telemetry Data – Transmission of data (UC3)

The «Battery Management System» preprocesses data from the «Battery Sensor Cluster» and sends the results via the «CAN Bus» to the «Vehicle Gateway» (Figure 30). If the mutual authentication with the cloud was successful, the «Vehicle Gateway» starts transmitting the vehicle's telemetry data to the cloud's «Message Broker Telemetry» according to agreed rules/policies.

The «Cloud Backend» checks the compliance with the policies on the part of the «Vehicle Gateway». In the event of a rule violation or conspicuous behavior of the «Vehicle Gateway» regarding data transmission, the «Cloud Backend» initiates appropriate measures.

Policies can be changed at any time. Based on the currently active policy, the «Cloud Backend» checks every single action of the «Vehicle Gateway» on whether it has the privileges to perform the respective action or not. In this way, changes become active immediately.

6.5.4 Scenario 3: Access control and privacy by encryption in medical image processing

The sections below provide the dynamic view of the *Access, Privacy, and Identity Services* in a medical domain. The dynamic views have been generalized to some extent, incorporating the most important functionality as stated in section 5.1.1.3, applicable to medical domain:

- A workflow for configuring access to resources, covering functions in cloud-tier (section 5.1.1.3) and device tier (section 5.3.1.3)
- A workflow to obtain access to these resources, covering functions in cloud-tier (section 5.1.1.3) and device tier (section 5.3.1.3)
- A workflow for privacy protection of these resources by means of de-identification / re-identification, covering functions in edge tier (section 5.2.1.3)

6.5.4.1 Access Control Configuration

Before the access control can be executed, various onboarding and configuration steps are needed. Typically, a portal can be used to administrate such configuration. It is assumed the organization is onboarded to make use of the services provided (at minimum of the Access Control Service) and has an admin user with appropriate admin permissions. Figure 31 shows the workflow. The admin logs in to the Access Control Service by the same Auth0[5] flow, as mentioned in section 6.5.4.2. This implies some initial configuration (for the admin) already should be in place as part of onboarding. The login results in a token, which is passed to API calls and used by Authorization Service to introspect the permissions & org of the admin user.

The permissions are specified by the various Resource Services. The configuration is created at run-time, i.e. the Authorization Server can immediately use the configuration without a restart. Various other aspects can be configured as well, like impersonation / federation (IODC) or onboarding of other identities like services/devices. As in Figure 32, logins and API calls are audited.

[5] https://auth0.com/

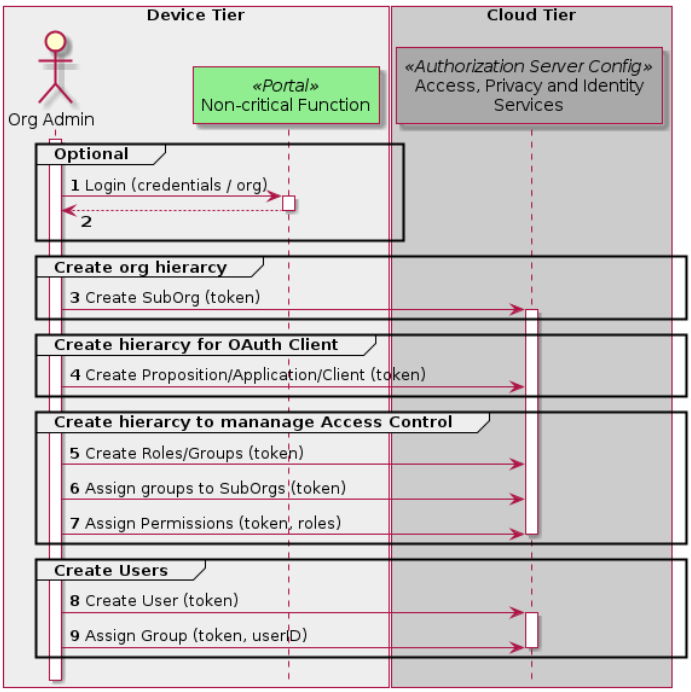

Figure 31: Access Control Configuration.

6.5.4.2 Access Control

Access to protected resources is granted via an Auth0 based access control.

The user needs to identify him/herself when starting the application. The application is represented by an OAuth client, called Client below. In Figure 32, the authentication for the users' identity is needed by the application to audit user access, and authorization is needed to verify if the user is allowed to access the applications APIs (resources) as well as other resources exposed by APIs, for example the resource server.

For public clients (e.g. browser applications), the challenge-response needs to be enhanced by mechanics like PKCE to ensure the authorization service can trust the authorization code and subsequent access token request. Often the Authorization and Authentication Server are implemented by same service, but Authentication can be delegated to a third-party Identity Service Provider (ISP).

Figure 32 depicts the typical workflow. Note, that in this example, with a browser application, the application (resources) is served via a cloud hosted web-server, but *execution* of logic runs in the browser in the device tier. It can be augmented by PKCE in steps 3 and 7. Security can be enhanced by restricting the requested scope for the token to the minimal necessary. The refresh token should not be kept by the public client; a silent re-login flow, or token refresh by a supporting backend for frontend pattern should be considered. To access the resources in step 14/15, the resource server can perform an organization (hierarchy) based introspect to assess if the org to which the user belongs has access to the resource.

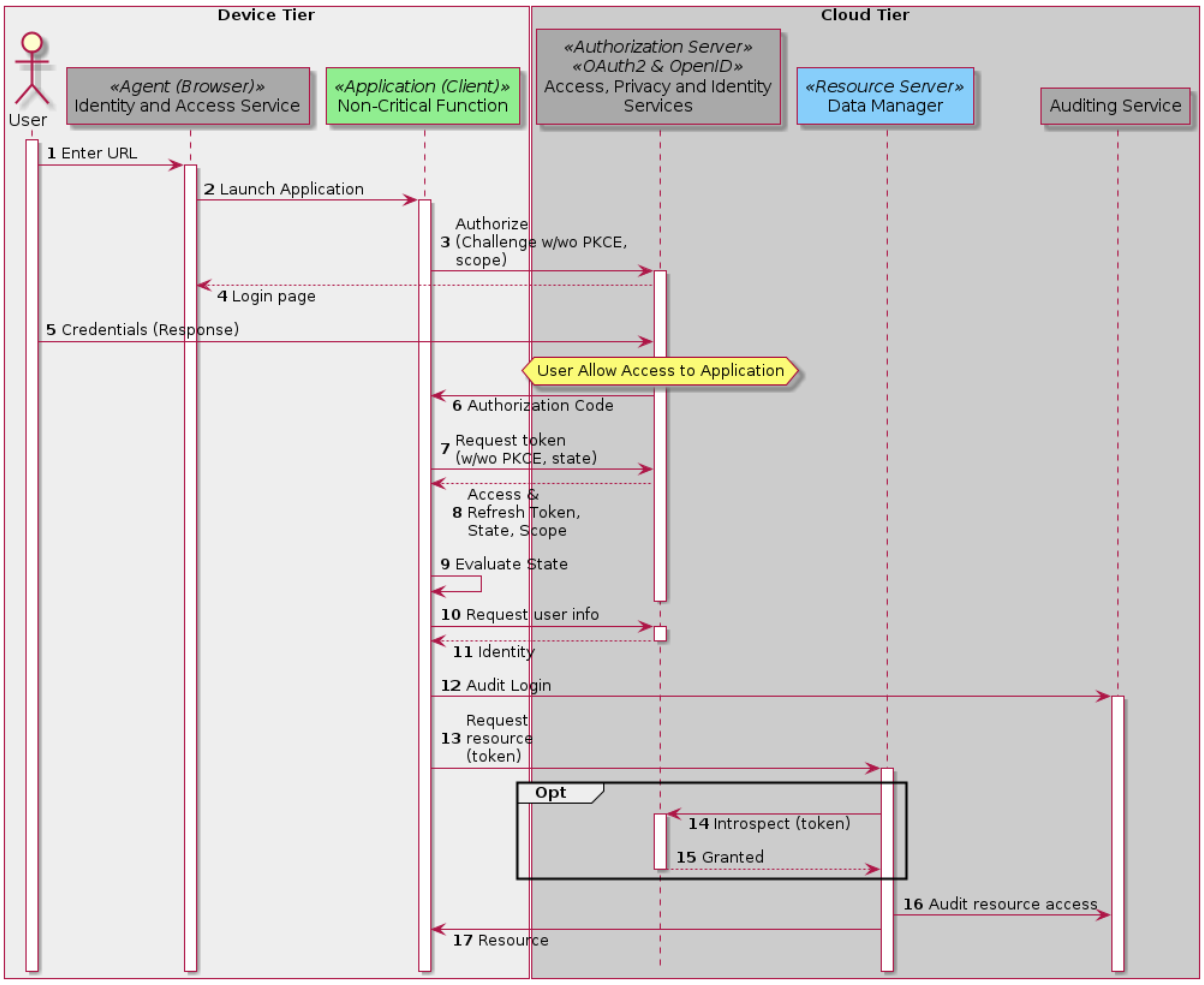

Figure 32: Authentication and Authorization flow using OAuth2.0 using Auth Code Grant type.

6.5.4.3 De-identification

If access control is not a sufficient means to protect the resources, then de-identification can be applied, as specified in section 5.2.1.3. This scenario assumes that the (clinical) user is allowed to see full personal identifiable information (PII), but the third-party algorithm that is executed on the data, should not have access to PII. To enhance security, the keys should be managed by a PKI service and rotated frequently. The manner of de-identification is defined by the profile. In this example, the de-identification is defined by the de-identification service itself and allows customization per the profiles, but no further creation of a processing pipeline with customer pluggable processing steps. Processing goes via batches, not via a continuous stream. Figure 33 depicts the workflow for De-identification and Re-identification by the Privacy Service. This flow is generic, but in medical imaging domain standards like DICOM are often used.

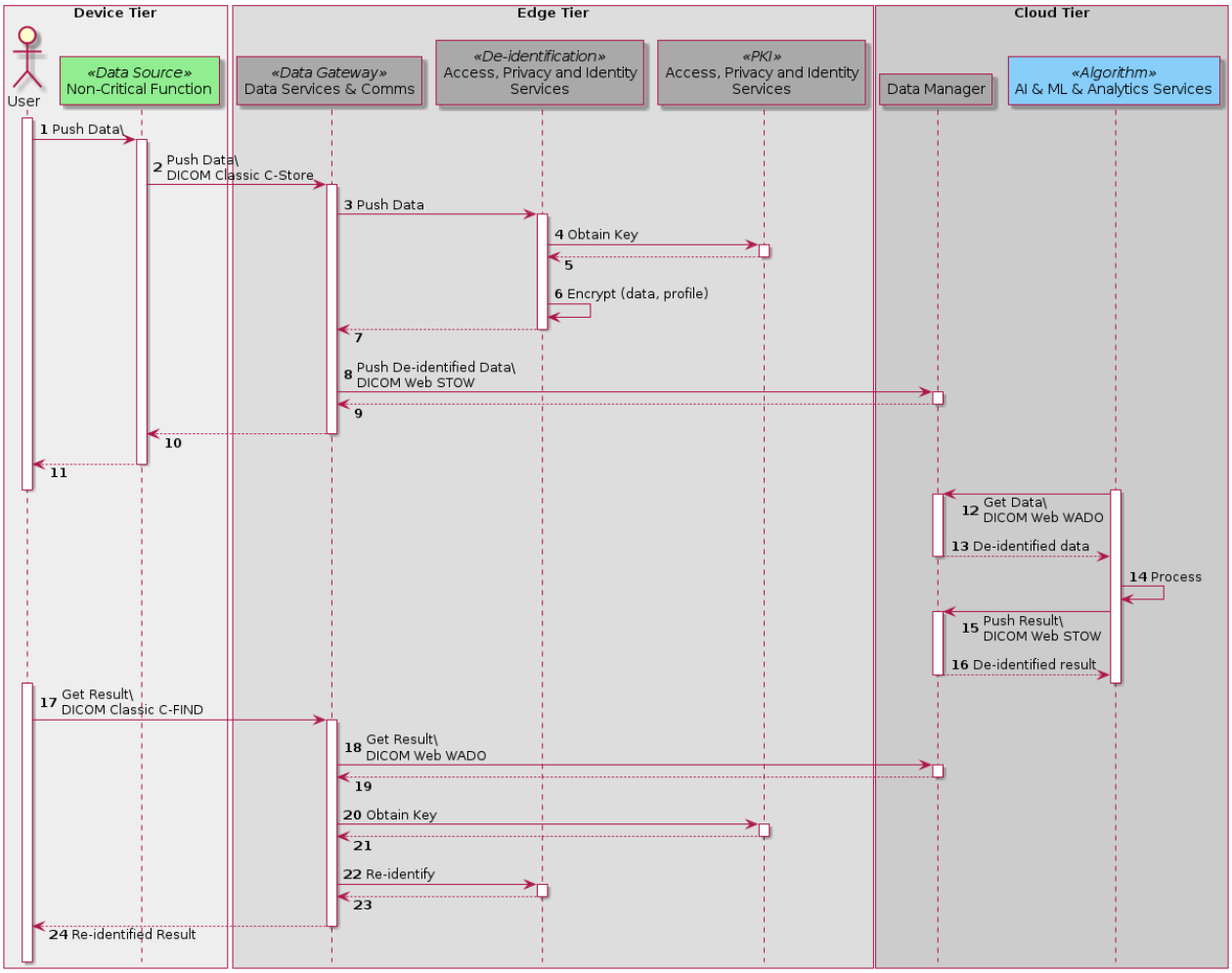

Figure 33: De-identification and Re-identification by the Privacy Service.

6.6 End-to-end secure communication

The general workflow of ensuring the end-to-end (E2E) security requires some steps to be taken in order to ensure that both ends of the communication are secure. Some of them should be done during the device provisioning phase, where either software-based or hardware-based measures can be employed. The flows of device provisioning and ensuring E2E security are described below, specifying which parts are related to *Identity and Access Service*, *Data Services and Comms* and *Auditing Service*. Note that the flow can be done either between Device and Edge tiers, or Edge and Cloud tiers.

6.6.1 Workflow

Workflow of Device Provisioning

- [Identity and Access Service] Create a unique device identity or certificate for secure communication.
- [Identity and Access Service] Generate a Certificate Signing Request (CSR) that includes the device certificate as well as other important information.

- [Identity and Access Service] Submit the CSR for verification and signing to a Certificate Authority (CA).
- [Identity and Access Service] After successful verification, the CA issues the device with a signed certificate.
- [Identity and Access Service] Securely store the device certificate within the device's Trusted Platform Module (TPM).
- [Identity and Access Service, Data Services and Comms] Set up the device with the appropriate network settings, security policies, and access control lists.
- [Identity and Access Service, Data Services and Comms] For secure data transmission, generate encryption key pairs (public and private keys).
- [Data Services and Comms] Create secure channels between the device and remote servers or endpoints for key exchange.
- [Identity and Access Service] Encrypt data before transmission using the device's public key.
- [Data Services and Comms] Use the device's unique certificate to authenticate with remote servers.
- [Data Services and Comms] Connect the device to the desired network, which could be a local network or cloud infrastructure.
- [Auditing Service] Constantly monitor the device's behaviour and performance for any irregularities or security breaches.
- [Auditing Service] Keep a record of provisioning activities for auditing and compliance.
- [Identity and Access Service, Data Services and Comms] Manage updates, renew certificates, and decommission devices securely throughout the device's lifecycle.
- [Auditing Service] To track provisioning activities and security incidents, keep detailed logs and generate reports.
- [Identity and Access Service, Data Services and Comms] Create procedures for safely retiring or repurposing devices when they reach the end of their operational life.

Implementing the TPM module into the overall architecture allows for introducing the End-to-End (E2E) communication. Alternatively, a software-based crypto module can be employed instead, yet hardware-based solution ensures a higher degree of security against the potential attacks. One method of completing this type of secure data transfer may be achieved using the following steps. This workflow describes on a low level the creation of a secure session key.

End to End communication

- [Identity and Access Service] Both sides of communication medium choose random large prime number and generator number.
- [Identity and Access Service] Both sides generate a temporary secret key.
- [Identity and Access Service, Data Services and Comms] Using the temporary secret key, a public shared key associated with each side of is created.

- [Identity and Access Service, Data Services and Comms] Both sides share their public key with each other via chosen communication medium.
- [Data Services and Comms] Each device creates a new private session key based on its own temporary private key and a public key from the other device.
- [Identity and Access Service] The final session key, used for further communication, is stored inside the secure storage of the TPM module.
- [Identity and Access Service, Data Services and Comms] This key can now be used to perform E2E secure data transfer via chosen communication medium.

6.6.2 Scenario 1: Battery monitoring secure communication

The ability to remotely monitor and manage car batteries has become indispensable as electric vehicles gain popularity and connected car technologies advance. End users expect real-time information about their battery's health, charge status, and performance, which is critical for optimizing vehicle range, lifespan, and safety. Furthermore, remote management is useful for proactively identifying and addressing problems, preventing breakdowns, and ensuring a smooth driving experience.

Continuous remote monitoring, on the other hand, raises serious security concerns. Robust security measures are required to protect sensitive battery data from unauthorized access, tampering, or cyber threats. Certificates and Trusted Platform Modules (TPMs) come into play here, and are controlled through Identity and Access Service on the Device, Edge and Cloud Tiers. The unique endorsement key stored within the TPM is a key component of this security framework. The TPM's unique endorsement key serves as a unique identifier for the vehicle or battery management system (in this case, on the Edge tier). It is critical in establishing trust between the edge layer and remote servers (cloud), enabling secure communication. This key is used to encrypt all data and meta-data stored on device so unique certificate stored inside non-volatile memory of TPM can be used to encrypt/decrypt all data transmitted from the device to the cloud, making it extremely difficult for unauthorized entities to intercept or manipulate the information. This shows the close interconnection between Identity and Access Service and Data Services and Comms components.

Certificates, in conjunction with the unique endorsement key stored in the TPM, provide a secure framework for encrypting and decrypting data, identifying the device, and establishing a secure channel for continuous monitoring of the battery. By addressing these security concerns, we can give end users the peace of mind they need, knowing that their car battery's data is safe and that their electric vehicles are safe and efficient.

E2E encryption mechanisms can be implemented in battery management systems. Continuous telemetry of the battery involves the real-time monitoring of its various parameters. As an example, these parameters may include voltage levels, temperature, charging and discharging rates, health, detected failures etc. This descriptive information can be collected to make decisions based on the battery performance, introduce optimization strategies or predict potential issues with it. All the data related to the battery needs to be transferred in a proper encrypted manner. First, the measurements should be passed from the device to edge tier, which involves the involvement of Identity and Access Service and Data Services and Comms components on device and edge tiers to ensure the secure communication between the device and edge. Depending on the type of connection, the measures to ensure E2E communication on this level may be integrated in the communication protocol specifications, but may be also backed by the hardware layer. The backbone of the secure telemetry lies in the implementation of E2E encryption. This type of secure data transfer ensures that the data is securely transmitted to the cloud/server and remains safe throughout this route. E2E encryption prevents unauthorized access of the data at any point during the transfer process. The raw data can be read only by each side of the transmission. E2E encryption not only makes the data

confidential but also authenticates the sender. It allows for identifying the specific device from which the telemetric data came from. Each device with its unique set of encryption keys/certificates makes it possible to track back the source of the information. The authentication process adds an extra layer of security. It makes it more challenging to tamper the communication by injecting malicious data. The encrypted data remains intact during the transmission, which guarantees its integrity. Any alteration of the data during the communication can be detected, and Auditing Service, employed on the edge and cloud tiers can be used to monitor and detect misbehaviour. In general, the use of TPM in a form of E2E encryption allows for secure transfer of any device-related information. The general flow of the communication and data exchange in the Battery Management System is visualized in Figure 34, Figure 35 and Figure 36.

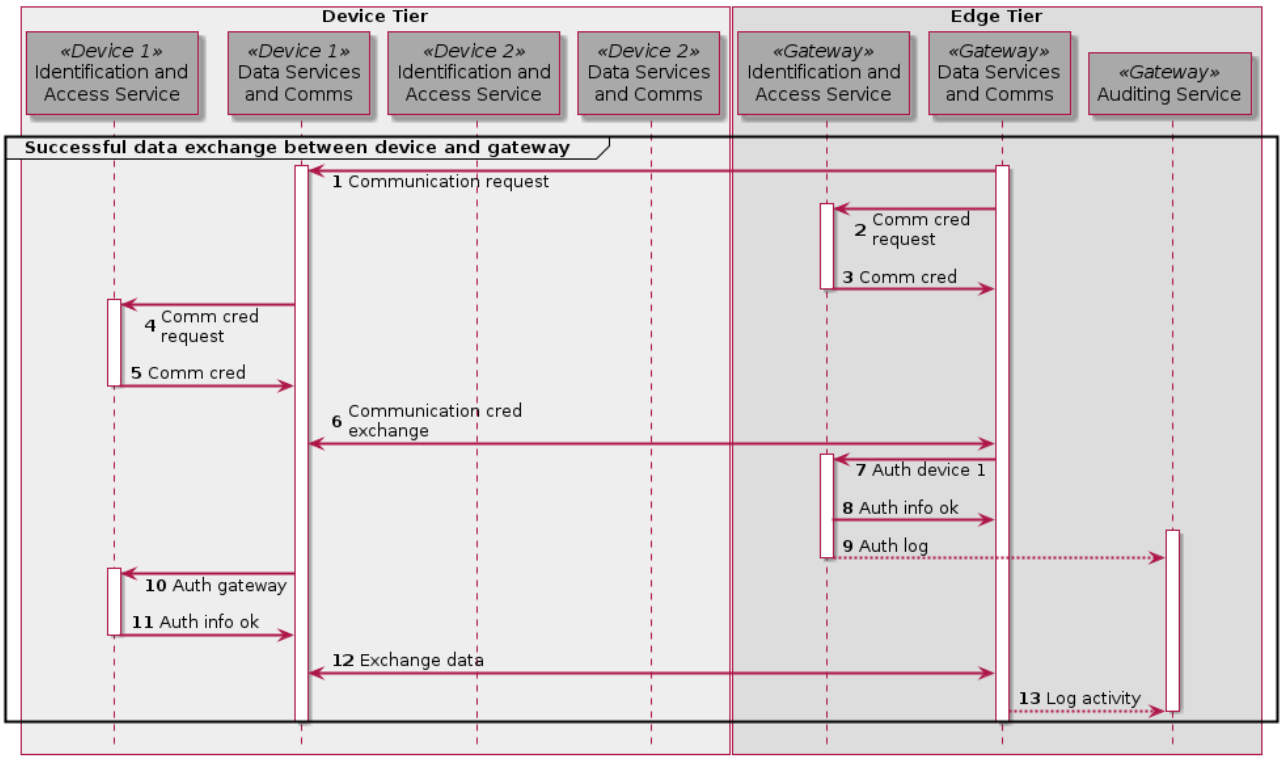

Figure 34: Battery Management System: Data exchange between device and gateway.

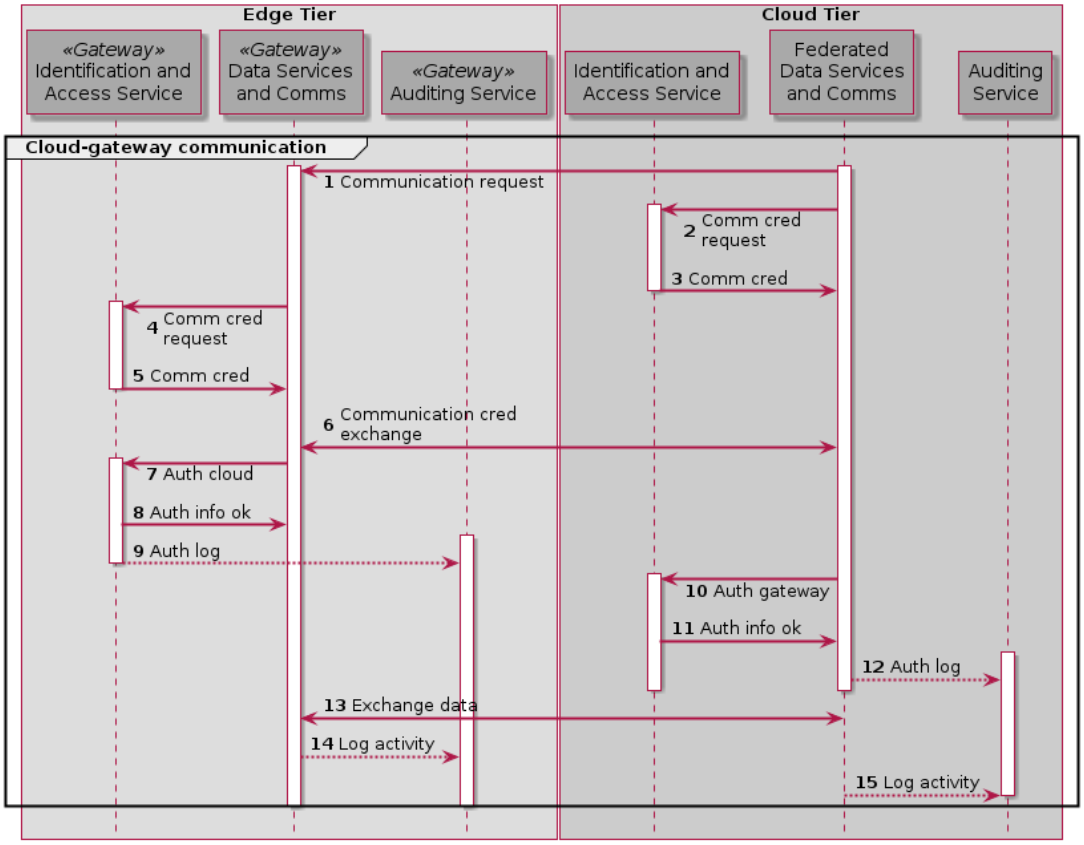

Figure 35: Battery Management System: Cloud-gateway communication

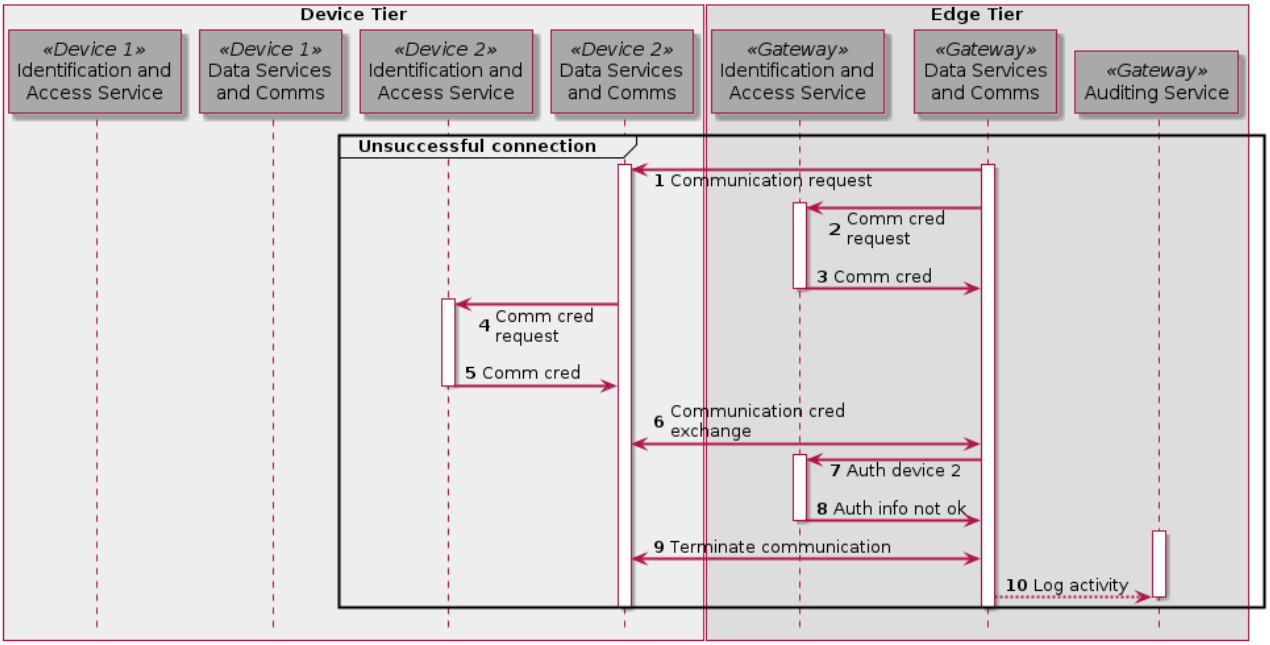

Figure 36: Battery Management System: Unsuccessful connection

6.6.3 Scenario 2: Secure commissioning and update

This scenario is inspired by TRANSACT UC3. Managing a large number of IoT devices in the field often entails connectivity and stability issues caused by mobile LTE connections. However, to ensure integrity of collected data as well as regularly maintain devices it is essential to achieve a secure and stable layer of communication between services in edge and cloud tiers and the device. Temporary loss of connection at any point in time as well as high spikes in delays must be dealt with at both sides of this layer.

Commissioning

Data managed and communicated by IoT devices is often sensitive and needs to be protected from malicious third parties. When compromised large fleets of devices can pose a significant threat to businesses and even our society. Therefore, ensuring security is a key topic in IoT. This has to be addressed already during design of the commissioning process of IoT devices. It still happens today that devices are commissioned generically with static default passwords for whole fleets leaving them very vulnerable to attacks.

Ensuring that devices can be identified and authenticated individually is a key requirement for a secure IoT infrastructure. This poses a challenge to the provisioning and commissioning process since application software cannot simply be cloned from device to the next. Our approach to this process includes a fully automated just-in-time commissioning of devices using the Amazon Webservices (AWS) IoT toolkit; In the design phase.

Direct Maintenance Access

Although large IoT fleets are mostly managed in groups containing multiple devices it is often necessary to access a single device in order to implement a device specific configuration or solve a distinct failure. A common way to achieve this is to activate a secure shell (SSH) daemon on the device making it accessible from the outside. This approach enables encrypted and secure communication provided the respective services are updated regularly. Configuration of such an access point is critical. Many of the first-generation devices have default passwords that are shared by all devices. Many contain design flaws that allow attackers to take full control of the devices. There have been several attacks where the number of IoT devices taken under control has been in the millions.

It should be clear that leaving an SSH daemon active permanently on an IoT device poses a security risk: An attacker can try to gain access at any time using new vulnerabilities that might appear. Generally, for the vast majority of the time direct access to specific devices is not necessary. It is only required under special circumstances. Therefore, enabling the SSH daemon only when actually needed would greatly improve IoT security.

MQTT Shelltunnel

Our approach to this problem involves completely refraining from making use of SSH and utilizing another channel to gain direct access to a device: We use a MQTT message broker to facilitate application specific communication. While authentication is achieved via X.509 certificates, traffic is sent securely over Transport Layer Security (TLS). To prevent man-in-the-middle attacks a central authority certificate is checked upon connection to the MQTT broker. AWS with its IoT Core is our vendor of choice for the broker implementation. It provides sate of the art security as well as a practical interface to manage fleets of IoT devices.

Instead of listening for SSH connections our implementation is subscribed to a device specific MQTT topic. When receiving a message on this channel it is interpreted as a standard shell command and forwarded to the operating system. Any shell output is saved and published to another device specific topic making it accessible for the maintainer that issued the command.

Appropriate configuration of access policies makes sure that only authorized users are able to publish and subscribe to these topics. Even if a device is compromised, i.e. an attacker gains access to the MQTT broker, it is not possible to send commands to other devices, since permissions for publishing and subscribing are managed centrally and device specific.

Another problem this approach solves is when devices are located behind a firewall rendering their SSH daemon hidden from the outside. Since the shelltunnel takes a client role no port forwarding is required to gain access to it.

SSH Tunnel

Sometimes it is more convenient to have an actual SSH connection; When a maintenance session becomes more involved it can be easier to have access to an actual terminal. AWS IoT offers a service called “Secure Tunneling” that enables temporary SSH connections to devices even if they are hidden behind a firewall. In order to use this service, we implemented an agent that listens to a specific MQTT topic on the IoT device. Once a tunnel is opened via the AWS web interface this agent receives a security token that is then used to establish the SSH connection. A secure tunnel’s lifetime is limited to 12 hours maximum after which it is closed automatically. Furthermore, security tokens are newly generated for every session and are required on both sides of the connection; This greatly increases the security of this approach. Currently, the pricing for this service is rather high at 5 to 6 USD per tunnel opened depending on the region. This is why using this as a default access method when maintaining multiple devices is economically unfeasible.

Mechanism for updating device groups

When the number of managed devices grows larger it is no longer feasible to regularly maintain them individually. Therefore, a mechanism to apply software or configuration updates to groups of devices is crucial. Keeping an overview of device states and success or failure of an update are some of the problems that need to be dealt with here. We utilized some tools provided by AWS IoT to solve this challenge: The “Jobs” interface can be used to define a set of operations that are sent to a specifiable group of devices. It also provides functionality to express and display device state, i.e. a job can fail, succeed or time out for a specific device. Communication between devices and the AWS IoT core is realized via MQTT. However, AWS does not provide any implementation that can execute operations on a device. There are SDKs that abstract access to the MQTT broker, nevertheless, to make full use of the jobs interface we implemented an application called “update agent” that reacts appropriately to the incoming messages.

The update agent was designed to be able to update a specific application running on the same device as well as execute arbitrary shell commands and store their outputs to the cloud. To achieve this, it is subscribed to a device specific topic waiting for a new job to be deployed. A job contains a custom job document which we defined to consists of one or more commands. The update agent currently supports four commands:

- UPDATE – Upon receiving this command the update agent first downloads a file specified as a parameter from a dedicated cloud store (AWS S3). This file contains the updated target application. Second, it sends a shutdown signal via a pipeline to the target application. Then it waits for shutdown approved signal from target. If the approved signal is not received within a specifiable timeout the target is shut down forcibly using a predefined operating system operation. After shutting down the target application is replaced with the downloaded version and the target is restarted. Subsequently the update agent waits for a startup approved signal again via a pipeline. If the expected signal is received within a timeframe of 5 seconds the returned job result is SUCCEEDED otherwise it is FAILED.
- UPDATE_SELF – This command is used to update the update agent itself. Again, a file containing the new version is downloaded from the cloud. Then, a script located on the device is executed. This script replaces the update agent with the downloaded version and restarts it. The job returns FAILED if either the download fails, or an exception is thrown when accessing the system runtime to execute the script otherwise it returns SUCCEEDED.
- DOWNLOAD – This command simply downloads the file specified in a parameter. The job result FAILED is returned if an error occurred during download. Otherwise SUCCEEDED is returned.
- EXECUTE – This command can be used to execute an arbitrary shell script on the device. The script is downloaded from the location specified in a parameter and passed to the operating system for execution. All outputs are saved to a device specific cloud store. We use the AWS IoT shadow to that end. The job result FAILED is returned when an error occurs during download or during access of the systems runtime, otherwise SUCCEEDED is returned.

These commands provide enough flexibility to conduct almost any maintenance task at scale. Furthermore, AWS IoT handles the case when devices are offline during deployment by redelivering jobs as soon as a device registers itself at the MQTT broker.

The update agent also keeps some device specific information such as version numbers of installed modules stored in a device specific cloud store, i.e. its shadow. This helps assessing device state even when a device goes offline. The shadow functionality is provided by AWS and also allows querying to find specific groups of devices in your IoT fleet, e.g. ones that run a certain version of a module.

IoT Platform Management

Nowadays, IoT devices often need to be more than mere stateless sensors. As soon as a device needs to interact smartly with other devices and a backend server managing an IoT fleet becomes more challenging; Device monitoring and proper handling of state when faced with unstable internet connectivity becomes crucial when IoT devices are part of business-critical infrastructure.

Device State Synchronization

To tackle the issues described above, we make use of the “Shadow” service provided by AWS IoT. This is a device specific cloud store that can be used manage state. The advantage of this service is that it provides functionality that helps synchronizing state with the cloud when a device loses and regains internet connectivity. Additionally, it is accessible even when a device is offline which makes backend implementation easier.

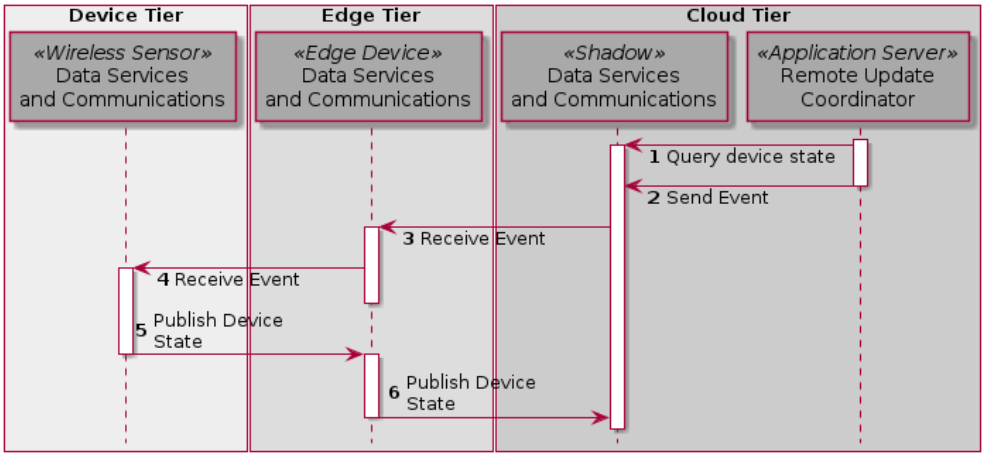

Figure 37: Device state synchronization

Figure 37 shows how the application server communicates with the devices it manages; to retrieve device state the shadow is queried. Since this is a web service it should be rare that it is not accessible, hence, this can be treated as an exceptional situation rather than a common one. Similarly, when the server needs to send information it simply stores an event in the shadow independent of the online status of the device. In case of the device being offline this event will be delivered once it reconnects making the state synchronization complete. The device itself publishes its state to the shadow which completes the cycle of communication.

AWS IoT provides a possibility to query for the online status of a device, however, this functionality comes with restrictions that make it unavailable for our application design; To determine if a device is online or not it must be ensured that only a single application running on it is connected to AWS IoT. Since we use the shelltunnel, update agent as well as a business application which all need a separate connection, we cannot meet this requirement. To solve this issue each device regularly publishes a message to a dedicated MQTT topic. The application server is subscribed to this topic and can in this way determine which devices are online.

Monitoring and Logging

When IoT devices are part of critical infrastructure it is mandatory to monitor operating system health as well as being able to access system logs in case of failure. For monitoring we make use of an open-source application called Netdata. It allows detailed monitoring of the operating system. When a monitored resource, e.g. number of file handles or CPU/RAM utilization, reaches a defined threshold Netdata raises an alarm. When that happens different means of notification are possible such as AWS SNS or email. However, since we also wanted a persistent log of all events for failure tracking, we decided to redirect them to a file and upload them to AWS Cloudwatch periodically. Cloudwatch offers searchable logs and also allows for definition of custom alarms based on them. All our IoT applications also use Cloudwatch for application specific logging. This enables us to quickly assess any failure and trace back the state of a devices operating system as well as its applications even when the device went out of service.

Mobile to IoT device communication

When mobile devices and IoT devices are part of an application communication management can become a difficult task. For large numbers of devices, the sum of possible lines of communication becomes large leading

to a heavy load on an application server if it is responsible for orchestration of communication. The MQTT protocol is a valid solution for this problem, relieving the backend of any direct communication management tasks.

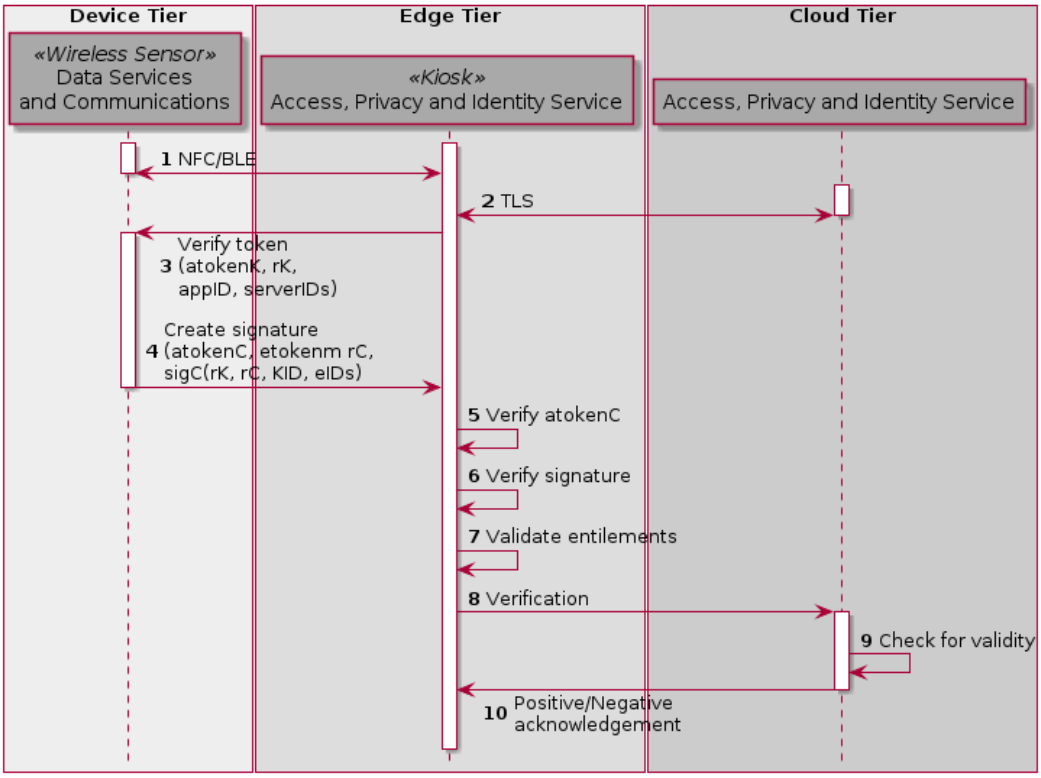

Figure 38: Mobile to IoT communication

Figure 38 shows our implementation to realizing this approach. We support Bluetooth Low Energy (BLE) as a means for proximity communication. This channel is used to establish a connection between a specific mobile user and IoT device. In that way, a shared and unique MQTT topic is established that can later be used to enable direct communication. The backend application server can be excluded completely in this process. When physical proximity to an IoT device is part of a use case, e.g. charging of an electric car, this is an effective approach. A user can subscribe to the charging station when he initiates the charging session and then receive data specific to that session directly to his mobile device even when the BLE connection has been terminated.

7 Adopting TRANSACT's reference architecture

This section provides a series of recommendations to facilitate the adoption of the TRANSACT reference architecture. Firstly, relevant aspects are presented to consider from the point of view of both architecture and infrastructure. Additionally, various concepts are described to adopt the architecture with an open-source approach, listing existing projects that can be used to implement the components defined in the TRANSACT reference architecture.

7.1 Relevant recommendations

The section below covers relevant recommendations for adopting the TRANSACT reference architecture. The recommendations are scoped to those areas indicated in green in Figure 39. These are the areas required to transform the safety critical CPS product in terms of its architecture and infrastructure.

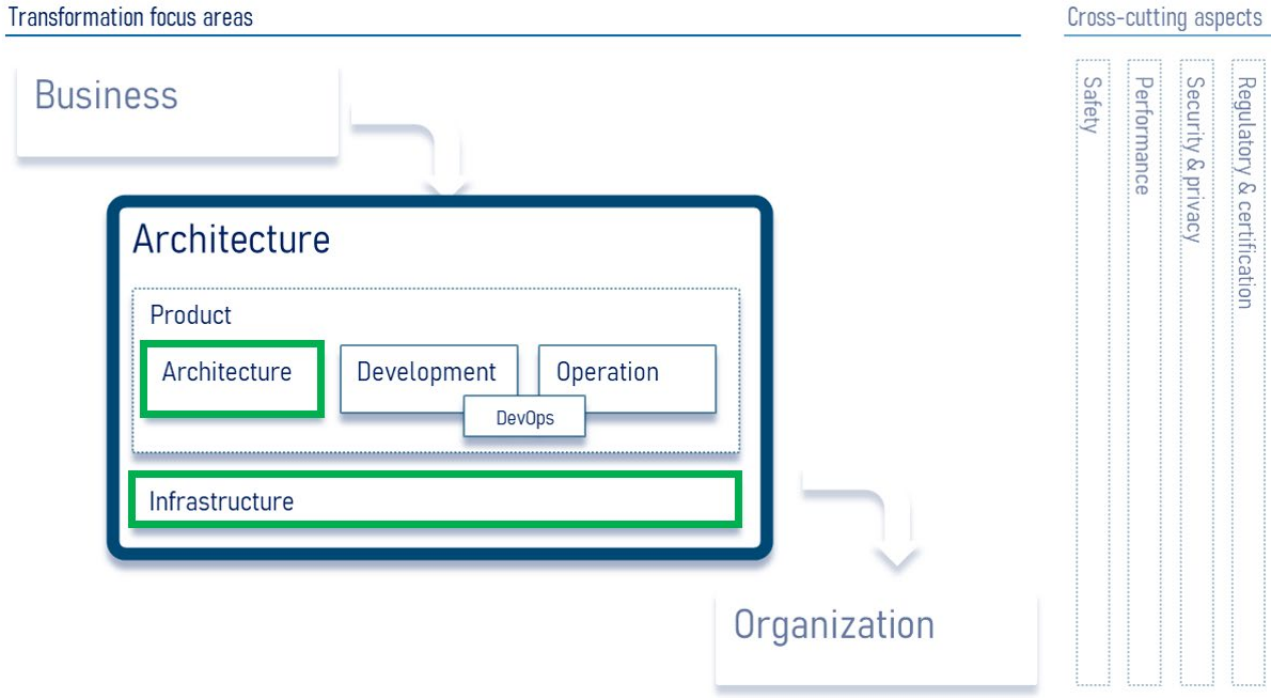

Figure 39: Scope of relevant recommendation

7.1.1 Architecture

Driven by the use cases, in Task 2.1 the following aspects have been identified as main challenges to tackle when transforming on-device CPS to a distributed CPS:

- Architectures for distributes safety-critical solutions
- Migrating safety-, mission- and non-critical functions to new architecture
- System security
- System updates
- Observability

An overview of the Core services and Value-Added services that contribute to such aspects is presented in Table 32. Recommendations in these areas and clarification of why a service contributes to certain aspect(s) are documented in the table.

		Core services & functions						Value-Added services & functions			
		Safety, Performance and Security Monitoring	Operational modes	Access, Privacy & Identity	Remote Updates	Auditing Service	Data Services & Comms	Data Manager	AI & ML & Analytics	BDaaS	New Services & Marketplace
Architecture transformation aspect	Architectures for distributes safety-critical solutions	U	U	U	U	U	U	U	U	U	U
	Migrating safety-, mission- and non-critical functions	P	P	U	U	U	U	U	U	U	U
	System Security	P/ U	U	P / U	P / U	P / U	P / U	U	U	U	U
	System Updates	P / U	P /U	U	P	U	U	U	U	U	U
	Observability	P / U	P / U	P	P / U	P	P	P	P	P	P

Table 32: Mapping of Architecture Transformation Aspect to Core- and Value-Added Services. P: contributes to the architecture aspect, U: uses the architecture aspect (for its own functionality).

Considerations and recommendations for architectures for distributed safety-critical solutions:

- The microservices architecture pattern is often recommended to leverage the scaling capabilities of the cloud, but should be considered after evaluating other strategies, like those in "*6 Rs*" (Orban, 2016))[6], where different parts of the solution can be migrated in different ways.
- As can be seen from previous sections, higher level services often depend on lower-level services and services that support the cross—cutting concerns. Therefore, when migrating to (cloud) microservices it is recommended to only apply this microservices pattern
    - once Core-services like security, updates, and observability are in place;
    - to migrate those parts that really need the scalability and can be deployed on cloud from a safety & security perspective;
    - By starting at the bottom of the dependencies, e.g. ensure data management is well-architected before migrating the higher-level services.

[6] The AWS migration strategies are extending the Gartner's "5 Rs" approach.

Considerations and recommendations for migrating safety-, mission- and non-critical functions:

- The more critical a function is, the clearer it needs to be what makes this function critical and how to guarantee its safety and performance in the distributed architecture. Again, as in the overall architecture, the different strategies from should be evaluated. Typically, the more critical the functionality, the more conservative the approach should be, where *retaining* is most conservative, up to *refactor* being least conservative.

- If migrating to cloud, then a redundancy service should be considered. This is made possible by the *Safety and Performance Monitoring* and *Operation Mode Coordination* services. When there is sufficient statistical evidence from the field that a fallback is not required anymore, the removal/reduction of the fallback can be considered. This can be made possible by measuring the KPIs in the field, often relying on Observability, leveraging output of the Safety and Performance Monitoring.

Considerations and recommendations for system security:

- System security must be designed into the distributed solution. Where component running on-device could often be secured by device level measures, after migration of (parts of the) system to edge and cloud, the security context (e.g. logged in user) propagates from the device (where user typically logs in) down into the services. Care should be taken that a security issue on the device cannot penetrate into the edge/cloud, potentially affecting multiple systems instead of only the local device. In addition, special attention should be targeted to prevent internal attackers from getting access to critical CPS resources. The *Identity and Access Services*, *Auditing Service, and Data Services and Communication* and *Remote Update* have a crucial role.

Considerations and recommendations for system updates:

- For on-device CPSs, the update can be seen as an atomic action, during which period the device is in a special mode. The orchestration of updates for a distributed system becomes more complex. It is recommended to:
    - first support updates of only the device via cloud-distributed software, before migrating functionality to the edge/cloud
    - create interfaces to ensure components that are intended to be migrated to cloud/edge have proper (forward/backward) compatibility;
    - apply strict versioning semantics to indicate breaking changes and avoid misconfiguration;
    - ensure there are mechanisms to guarantee the integrity of the distributed update, for example via code signing with digital signatures or a checksum

  This allows the Remote Update Coordinator to grow from a single on-device update coordinator to a coordinator to make a truly distributed update possible. For example, an update where both the on-device component is updated and the device is configured to use latest API of an updated cloud service used by that component.

Observability

- Just like security, observability should be designed into the solution. Like security, observability for an on-device CSP was restricted to collecting data on that single device. In a distributed solution it becomes a distributed problem, where a runtime relation between all independent services needs to be made. It is key to correlate all data, like logs (what is happening on system), traces (how does

information flow across tiers/services) and metrics (what is the state of the computing infrastructure the services are running on).

- Observability is a shared responsibility which all components (on-device, edge, tier) need to implement. Once in place it is useable as input for monitoring all Core and Value-Added services, up to complete product/solution. and potentially trigger Operation Modes and Remote Updates.
- It is recommended to follow open standards like (OpenTelemetry, 2023) and build in observability in lower-level services first.

7.1.2 Infrastructure

The infrastructure spans across all three tiers: device, edge, and cloud. The infrastructure needs to be catered such that it supports the following phases: product development (including testing) and product use (including deployment). The aspects in Table play a crucial role in defining and managing this infrastructure to support the products/solutions, consisting of core- and non-core services (e.g. value adding). The column 'Most Important Supporting Core-Services' mentions the TRANSACT core services that play an important role for supporting the mentioned aspects. There are also additional aspects, not yet covered by the core-services that play an important role. A few examples are mentioned in Table ; here 'Most Important Supporting Core-Services' is left empty.

To support the infrastructure, third-party infrastructure suppliers are often used. These providers typically offer infrastructure on various levels in the *X*aaS model: IaaS, PaaS, SaaS (Infrastructure, Platform, and Software as a Service, respectively). This is offered on cloud tier, and more frequently on edge tier as well. Choosing a level XaaS and Tier level depends on many factors. The "*6 Rs*" (Orban, 2016))[7] provides some guidance, for example a lift-and-shift often user lower XaaS level infrastructure.

Irrespective of the XaaS level, there is almost always a shared responsibility between cloud provider and client (user). For example, to ensure the overall security and compliance of the services and infrastructure used in the product/solution. Typically, the lowest level, i.e. the hardware / OSs in the IaaS level, are kept secure by the cloud providers, but usage of higher levels (e.g. data storage in PaaS) puts a fair share of responsibility on the client as well (e.g. security of data at rest by encryption and access controls). Also, the monitoring of the product/solution, becomes a shared responsibility: the cloud provider provides relevant observability capabilities (logs, traces, metrics) and adheres to standards in that area (e.g. OpenTelemetry) to enable the observability. The clients need to configure the thresholds, alarms and actions based on those metrics, and ensure client-specific components also adhere to same standards to realize full end-to-end observability.

This shared responsibility thinking needs to be taken into account when migrating functionality to edge/cloud. It doesn't only have consequences for the components/services hosted in the cloud, but also for the device design, for example:

- a device should not become an access point to compromise the security of the edge/cloud-hosted services (potentially affecting many more devices than the device itself);
- the development environment should be guarded and limited to those cloud-services that have appropriate certification.

[7] The AWS migration strategies are extending the Gartner's "5 Rs" approach.

Aspect	Most important Supporting Core-Services	Relevant Tiers [device, edge, cloud]	Most Relevant Phase [development, use]	Rationale
Health Status	Safety and Performance Monitoring	All	Use	To control both hardware (device, edge) and software (device, edge, cloud), the health should be monitored. Observability should be designed for *across* tiers to enable development / operations to investigate performance issues.
Updates	Remote updates,	All	Use	Cloud offers the opportunity to leverage as extension of compute power to postpone costly device/edge hardware device updates. Updates of cloud infrastructure itself is often not under control of the product owner; migrating cloud-services to newly introduced cloud-infra (e.g. newly supported hardware) needs to be actively pursued. Use Core Service to manage SW updates on device or edge.
Redundancy	Safety and Performance Monitoring	All	All	Ensure redundancy for mission critical / safety critical cloud hosted services on edge/device until there is sufficient trust that probability of failure for on-cloud service is justified (by safety risk management).
Shared Security Responsibility	Access, Privacy & Identity Services,	Cloud	All	Control deployments to ensure it's secure by default, e.g. by limiting developers access to cloud services that are compliant with relevant legislation, using blueprints that have controls in place.

Aspect	Most important Supporting Core-Services	Relevant Tiers [device, edge, cloud]	Most Relevant Phase [development, use]	Rationale
Technology management	-	All	All	Where possible reuse software and deployment technologies to harmonize development and deployment. Balance to give development teams freedom in technology choice vs freedom to easily rotate developers between teams.
Cost management	-	Cloud	All	For development it's important to manage the cost of used cloud-services (e.g. for testing). For production it is crucial to ensure business profit outweighs cost. Especially managing cost of hardware (IaaS), serviced (SaaS) that are not owned by business is new wrt a device-only CPS.
Testability	-	All	Development	Design a balanced test-architecture where testing majority of functionality can happen quickly to get short feedback cycles. For example, allow dependencies of devices on cloud-services (e.g. database) to be stubbed by a more performant (on-device / in memory variant). Setup the test infrastructure accordingly, to test in a staged way, with locally run test giving feedback, where full product-like deployment give confidence in actual product use.

Table 33: Infrastructure recommendations

7.2 Adopting the reference architecture in open source

7.2.1 An experimentation environment for open source adoption

A reference architecture is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realizes a predetermined set of requirements. As such, it can be realized in different kinds of systems, implementing specific use cases. Such systems can be composed of proprietary software and/or open-source software.

Open source enables open innovation and open collaboration and can be a means to foster adoption of technologies. Open, industry-grade software platforms allow organisations to collaborate on core technologies and compete on value-added products and services building on open source. The definition of open source implies that source code is distributed under a license in which the copyright holders grant others the power to run, access, modify and re-distribute the software to anyone and for any purpose (Open Source Initiative, 2023), thus enabling development under an open, collaborative model. Over the last decades, the open-source software (OSS) development model has gained more and more popularity around the globe. Nowadays, open-source components are the core building blocks of application software in most innovative domains, providing developers with an ever-growing selection of off-the-shelf possibilities that they can use for assembling their products faster and more efficiently (MEND - Microsoft, 2021).

Whether consuming or contributing to open source, one needs to have a sound knowledge of licensing and IP (Intellectual Property) management. When assembling third-party open-source software into a new product, it is important to understand their license conditions and potential compatibility conflicts. Further, to foster adoption, it is important to implement a set of best practices and governance guidelines, such as providing necessary meta information in repositories, having contributor agreements in place, etc.

To account for these challenges, we provide such an environment that allows adopting the TRANSACT architecture under open source best practices. This environment is provided as GitLab organisation, called Eclipse Research Labs[8] (see Figure 40).

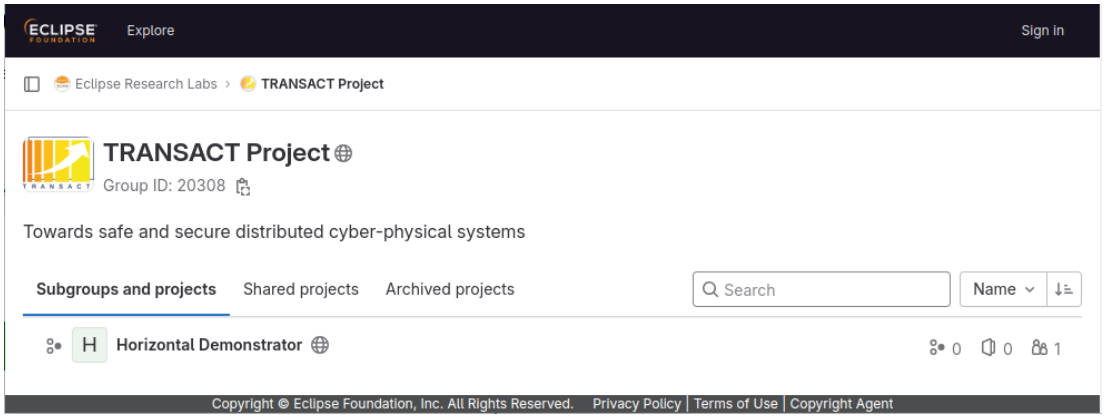

Figure 40: Eclipse Research Labs - TRANSACT Project

It allows to implement open source best practices and increase the chances of delivering high-quality, professional open source. In concrete terms, this means enabling:

[8] https://gitlab.eclipse.org/eclipse-research-labs/transact-project

- **Open Collaboration:** An open infrastructure (e.g., GitLab/GitHub) for doing open source successfully in research projects. It also enables good practices of (open-source) software development such as integrating often and soon, automating the process with continuous testing and integration scripts to identify bugs or broken APIs (Application Programming Interface) as early as possible.
- **Community Governance:** Contributor Agreements allow to identify and list the contributors to the open-source project(s) from beginning. Signing the Eclipse Contributor Agreement (ECA) is a prerequisite for pushing code in one of the Research Labs repositories.
- **Open Source License Compliance:** IP reviews help to analyse shared code as early and often as possible to identify potential intellectual property issues, such as missing copyright headers in source files, metadata files in repositories or third-party dependencies with incompatible licenses. Tool support for third-party IP checks will be provided.

TRANSACT Use Case implementations consist of both, proprietary and open-source components. Hence, the horizontal demonstrator will provide a basic implementation of the TRANSACT reference architecture based mainly on open source. Being publicly available in Eclipse Research Labs, it will act as a place for experimentation, guidance and adoption of the reference architecture. Eclipse Research Labs implementation will ensure proper compliance of open source best practices as described above.

Details of the horizontal demonstrator and accompanying open-source environment will be laid out in D31 (D5.2) TRANSACT basic horizontal demonstrator and D32 (D6.5) Open source project proposal.

7.2.2 Mapping the Eclipse IoT and Edge ecosystem

The Eclipse IoT and Edge Working Groups are collaborations of industry and academic partners who are building a set of open-source technology for the Edge and IoT ecosystem. The focus of these collaborations is on building 1) open-source implementations of IoT and Edge standards and protocols, 2) open-source frameworks and services that will be used by IoT and Edge solutions, and 3) tools for IoT and Edge developers.

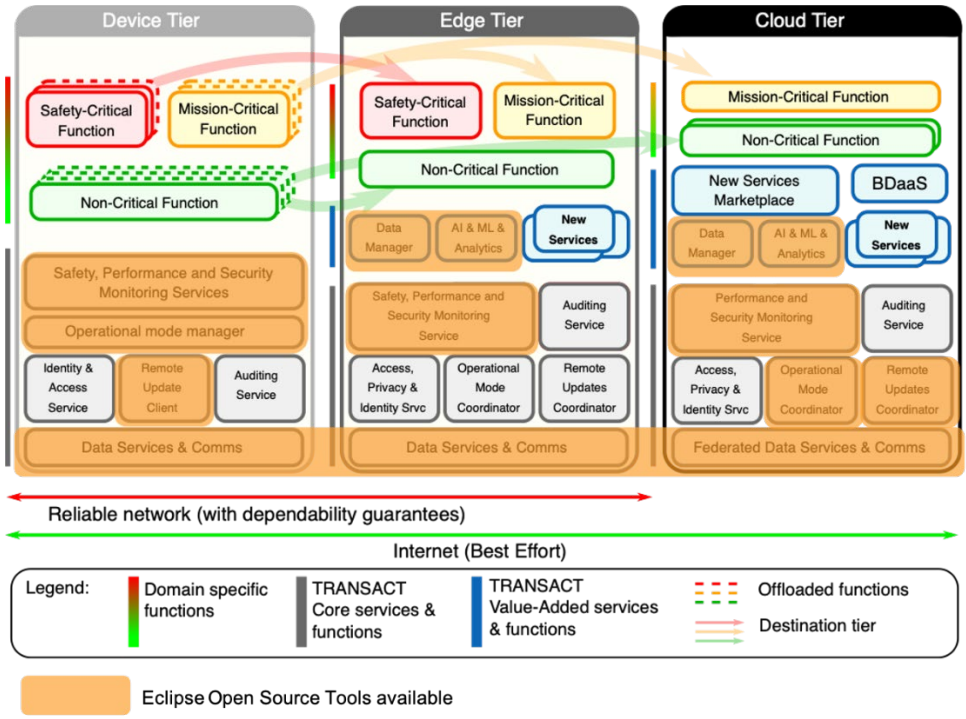

Figure 41: Mapping Eclipse Open Source Tools to the TRANSACT Reference Architecture

Currently, the Eclipse IoT and Edge Working Groups comprise around 50-member companies who collaborate on more than 40 open source projects that help building professional applications for the Edge-Cloud Continuum.

In the following we provide an overview of the analysis of Eclipse Open Source Tools that can address the requirements of certain components of the TRANSACT reference architecture (Figure 41 highlights these components). Each of these tools is available under a business-friendly open source license such as Apache-2.0 or EPL-2.0.

7.2.2.1 Data Service & Comms

The Eclipse IoT Working Group provides open-source implementations of various **protocol specifications**:

Name	Description	Link
Californium	Eclipse Californium is a powerful **CoAP** framework targeting back-end services communicating with smaller Internet of Things devices. Stronger Internet of Things devices may use Californium as well. It provides a convenient API for RESTful Web services that support all of CoAP's features.	https://eclipse.dev/californium/
CycloneDDS	Eclipse Cyclone DDS™ is an implementation of the **OMG Data Distribution Service (DDS)** specification (see http://www.omg.org/spec/DDS/) and the related specifications for interoperability (see http://www.omg.org/spec/DDSI-RTPS/)	https://cyclonedds.io/
Leshan	Eclipse Leshan™ is an OMA Lightweight M2M (**LWM2M**) implementation in Java.	https://eclipse.dev/leshan/
Milo	Eclipse Milo is an open source implementation of **OPC UA** (currently targeting 1.03). It includes a high-performance stack (channels, serialization, data structures, security) as well as client and server SDKs built on top of the stack.	http://www.eclipse.org/milo
Mosquitto	Eclipse Mosquitto provides a lightweight server implementation of the **MQTT** protocol that is suitable for all situations from full power machines to embedded and low power machines.	https://mosquitto.org/
Paho	The Eclipse Paho project provides open source, mainly client side, implementations of **MQTT** and **MQTT-SN** in a variety of programming languages.	https://eclipse.dev/paho/
Tahu	Eclipse Tahu™ is a set of references implementations for the **Sparkplug** specification. It addresses the existence of legacy SCADA/DCS/ICS protocols and infrastructures and provides a much-needed definition	https://eclipse.org/tahu

	of how best to apply MQTT into these existing industrial operational environments.	
Wakaama	OMA **Lightweight M2M** C implementation designed to be portable on POSIX compliant systems.	https://eclipse.dev/wakaama/
Zenoh	Eclipse Zenoh is a **pub/sub/query** protocol unifying data in motion, data at rest and computations. It elegantly blends traditional pub/sub with geo distributed storage, queries and computations, while retaining a level of time and space efficiency that is well beyond any of the mainstream stacks.	https://zenoh.io/

On top of protocol implementations, the Eclipse ecosystems provides open-source tools for **connectivity, interoperability, and device management & integrations:**

Name	Description	Link
Ditto	Eclipse Ditto™ is a framework for providing the "Digital Twin" pattern for IoT applications in order to interact with IoT devices. That means that Ditto mirrors physical devices as digital representations in the cloud.	https://eclipse.dev/ditto/
Hono	Eclipse Hono provides uniform (remote) service interfaces for connecting large numbers of IoT devices to a (cloud) back end. It specifically supports scalable and secure data ingestion (*telemetry* data) as well as *command & control* type message exchange patterns and provides interfaces for provisioning & managing device identity and access control rules	https://eclipse.dev/hono/
ThingWeb	The Eclipse Thingweb node-wot is a framework for implementing **Web of Things** servers and clients in Node.js	https://www.thingweb.io/

7.2.2.2 Safety, Performance, and Security Monitoring Services and Operational Modes

Name	Description	Link
POOSL	Eclipse POOSL offers a general-purpose method and tool for describing and simulating complex systems' architecture for the early evaluation of their key structural and behavioural aspects, requirements and performance.	https://www.poosl.org/
Trace4CPS	Eclipse TRACE4CPS™ (TRACE) maintains and evolves a visualization and analysis tool for the performance	https://eclipse.dev/trace4cps/

	engineering of cyber-physical systems. It has a strong focus on industrial applicability.	

7.2.2.3 Remote Updates

Name	Description	Link
Hawkbit	Eclipse hawkBit™ is a domain independent back-end framework for rolling out software updates to constrained edge devices as well as more powerful controllers and gateways connected to IP based networking infrastructure.	https://eclipse.dev/hawkbit/
Hara	Eclipse Hara™ provides a reference agent software implementation featuring the Eclipse hawkBit device API. Such reference implementations are initially driven by operating systems and application frameworks that today constitute the main platforms for the majority of IoT and embedded devices. These devices include but are not limited to: Open Embedded, Android, QT, etc.	https://github.com/eclipse/hara-ddiclient

7.2.2.4 Data Manager

Name	Description	Link
Kapua	Eclipse Kapua™ is a modular integration platform for IoT devices and smart sensors that aims at bridging Operation Technology with Information Technology. Eclipse Kapua can archive the telemetry data sent by the devices into a persistent storage for application retrieval. A reference message payload is defined which allows for a timestamp, a geo position, strongly typed message headers and an opaque message body.	https://eclipse.org/kapua/
Streamsheets	With Eclipse Streamsheets™ the everyday technical or business end user can create stream processing applications just by using their existing spreadsheet knowledge (e.g. from Microsoft Excel or Google Sheets). Eclipse Streamsheets give non-programmers the opportunity to work with event streams with a power and flexibility that would otherwise only be available to an experienced software programmer.	https://eclipse.org/streamsheets/

7.2.2.5 Machine Learning

Name	Description	Link
Deeplearning4j	The goal of Eclipse Deeplearning4j is to provide a core set of components for building applications that incorporate AI. AI products within an enterprise often have a wider scope than just machine learning. The overall goal of a distribution is to provide smart defaults for building deep learning applications.	https://deeplearning4j.konduit.ai/

7.2.2.6 Setting up a device to cloud stack

The Eclipse IoT Working Group provides **Eclipse IoT Packages™**[9], an effort to create easy to deploy Eclipse IoT based, end-to-end scenarios, on top of Kubernetes and Helm. The Eclipse IoT Packages™ project provides a home for use case focused IoT deployments, based on Eclipse IoT technology. It takes the building blocks that the different projects provide, and adds the necessary glue code, needed to create more complete setups. Packages not only provide deployment scripts, but also descriptions of what the benefit of the integrated package is, along with some initial, tutorial like steps, to play with the setup, once it is deployed. Because integrating different software components is already hard enough, the IoT packages project chose Kubernetes as its cloud side deployment platform. Currently two IoT packages are available:

Name	Description	Link
Cloud2Edge	Connecting and managing sensor style devices. Connecting sensors to the cloud and processing data with a digital twin platform. Cloud2Edge includes **Eclipse Hono** and **Eclipse Ditto**.	https://www.eclipse.org/packages/
Telemetry end-to-end	A package showing telemetry data acquisition end-to-end: Microcontroller firmware to cloud side data processing, using Drogue IoT and Apache Kafka in the process. It includes **Eclipse Ditto**, **Eclipse Kura**, **Eclipse Mosquitto**, and **Eclipse Streamsheets.**	https://www.eclipse.org/packages/

[9] https://www.eclipse.org/packages/

8 Conclusions

The present document has provided a detailed description of all the relevant aspects related to the reference architecture proposed by TRANSACT. It has not only undertaken the purpose and mission of the architecture at a high level but established the connection between the different components in its static view and the requirements that were defined by stakeholders in the context of Work Package 1. Moreover, one of the more relevant aspects was establishing the relevance of the architecture in four fields that are key for the project: performance, safety, security and privacy, which has been settled providing insights on each one of these dimensions and certain particular areas within.

The most important work, however, has been the fine-grained definition of the different components in the static view of the reference architecture, as well as for the dynamic view. In the first case, a thorough work has been performed providing information related to the role of each component, especially regarding the MAPE cycle, its relevant aspects regarding the configuration and input/output interfaces, separating the mandatory aspects from those that can be considered as optional or extensions. The connection to the requirements has been made as well at component level, also describing endpoints or the related components or concepts, paving the ground for the posterior dynamic view section. In the latter, six different functional capabilities were described to provide examples of how the reference architecture can be applied for different purposes. For each one, a general workflow was provided and described, and then different scenarios -more than 15-, more concrete and that can be seen as particular cases of the workflow, were provided. Altogether, they cover a wide range of cases and situations where the reference architecture provides the frame on how to act with the components designated and proves its flexibility and adaptability.

Finally, the document concludes by presenting a series of recommendations from the architectural and infrastructure point of view, as well as on how to adopt the reference architecture or examples of open-source existing technologies that can be adopted to implement the different reference architecture components. Here, in fact, a special reference is made to the Eclipse IoT and Edge ecosystem, indicating in what components Eclipse can provide specific solutions in order to facilitate the adoption.

The document has proven ambitious and extensive, especially in the level of detail achieved in the static and dynamic views. Understanding the potential capabilities of the components and ways of interacting among them will help potential users to adopt the proposed TRANSACT approach.

9 References

Arcani, P., Riccobene, E., & Scandurra, P. (2015). Modeling and analyzing MAPE-K feedback loops for self-adaptation. *2015 IEEE/ACM 10th International Symposium on Software Engineering for Adaptive and Self-Managing Systems* (pp. 13-23). IEEE.

Ferdous, R. K. (2011, Aug). Analyzing systm safety and risks under uncertainty using a bow-tie diagram: An innovative approach. *Process Safety and Environmental Protection, 1-2*(91), 1-18.

Lutz, R. R. (2000). Software Engineering for Safety: A Roadmap. *Proceedings of the Conference on The Future of Software Engineering* (pp. 213-226). New York, NY, USA: Association for Computing Machinery.

MEND - Microsoft. (2021). *The Complete Guide to Open Source Security.* Retrieved from https://www.mend.io/resources/white-papers/the-complete-guide-on-open-source-security/

Muller, G. (2008). *A reference architecture primer. White paper.* Eindhoven: Eindhoven University of Technology.

OASIS Standard. (2012). *OASIS Reference Architecture Foundation for Service Oriented Architecture v1.0.*

Open Source Initiative. (2023). *The Open Source Definition (Annotated). History of the OSI*. Retrieved from https://opensource.org/osd-annotated

OpenTelemetry. (2023, 05 03). *OpenTelemetry*. Retrieved from OpenTelemetry: https://opentelemetry.io/

Orban, S. (2016, 11 1). *6 Strategies for Migrating Applications to the Cloud*. Retrieved from 6 Strategies for Migrating Applications to the Cloud: https://aws.amazon.com/blogs/enterprise-strategy/6-strategies-for-migrating-applications-to-the-cloud/

Sanden, v. d. (2021). Model-driven system-performance engineering for cyber-physical systems. *International Conference on Embedded Software (EMSOFT)*, (pp. 11-22).

TRANSACT D05. (2022). *D5 (D1.1) Use case descriptions, end user requirements, SotA and KPI's. GUT. [Confidencial].* TRANSACT EU project.

TRANSACT D06. (2022). *D6 (D1.2) Technical requirements and TRANSACT transition methodology commonalities. DLR. [Confidencial].* TRANSACT EU Project.

TRANSACT D07. (2022). *D7 (D2.1) Reference architectures for SCDCPS v1. ITI. [Online] Available: https://transact-ecsel.eu/resources.* TRANSACT EU Project.

TRANSACT D08. (2022). *D8 (D3.1) Selection of concepts for end-to-end safety and performance for distributed CPS solutions. TNO. [Online] Available: https://transact-ecsel.eu/resources.* TRANSACT EU Project.

TRANSACT D09. (2022). *D9 (D3.2) Selection of concepts for end-to-end security and privacy for distributed CPS solutions. DTU. [Online] Available: https://transact-ecsel.eu/resources.* TRANSACT EU Project.

TRANSACT D15. (2023). *D15 (D1.4) Use case evaluation and impact assessment v1. GUT. [Confidencial].* TRANSACT EU project.

TRANSACT D16. (2022). *D16 (D3.3) Solutions for end-to-end safety and performance for distributed CPS v1. TUE. [Online] Available: https://transact-ecsel.eu/resources.* TRANSACT EU project.

TRANSACT D17. (2022). *D17 (D3.4) Solutions for end-to-end security and privacy for distributed CPS v1. FSC. [Online] Available: https://transact-ecsel.eu/resources.* TRANSACT EU project.

TRANSACT D18-D22. (2023). *D18-D22 (D5.3-D5.7) TRANSACT UCx demonstrations and validation v1. FLEET, NVT, AVL, PMS, DAM. [Confidencial].* TRANSACT EU project.

TRANSACT D23. (2023). *D23 (D1.3) TRANSACT transition guide to facilitate safety-critical distributed CPS solutions v1. PMS. [Online] Available: https://transact-ecsel.eu/resources.* TRANSACT EU Project.

TRANSACT D33. (2024). *D33 (D3.5) Solutions for end-to-end safety and performance for distributed CPS v2. TUE. [Draft].* TRANSACT EU Project.

TRANSACT D34. (2024). *D34 (D3.6) Solutions for end-to-end security and privacy for distributed CPS v2. DTU. [Draft].* TRANSACT EU Project.