



D2.2

Enhancing IoT Underlying Technology

WORKPACKAGE WP2

PROGRAMME IDENTIFIER H2020-ICT-2020-1

DOCUMENT D2.2

GRANT AGREEMENT ID 957246

REVISION V1.0

START DATE OF THE PROJECT 01/10/2020

DELIVERY DATE 31/05/2022

DURATION 3 YEARS

© Copyright of the IoT-NGIN Consortium

This project has received funding from the European Union's Horizon 2020 research and innovation programme under Grant Agreement No 957246



DISCLAIMER

This document does not represent the opinion of the European Commission, and the European Commission is not responsible for any use that might be made of its content.

This document may contain material, which is the copyright of certain IoT-NGIN consortium parties, and may not be reproduced or copied without permission. All IoT-NGIN consortium parties have agreed to full publication of this document. The commercial use of any information contained in this document may require a license from the proprietor of that information.

Neither the IoT-NGIN consortium as a whole, nor a certain party of the IoT-NGIN consortium warrant that the information contained in this document is capable of use, nor that use of the information is free from risk, and does not accept any liability for loss or damage suffered using this information.

ACKNOWLEDGEMENT

This document is a deliverable of IoT-NGIN project. This project has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement N° 957246.

The opinions expressed in this document reflect only the author's view and in no way reflect the European Commission's opinions. The European Commission is not responsible for any use that may be made of the information it contains.

D2.2 – Enhancing IoT Underlying Technology

PROJECT ACRONYM	IoT-NGIN
PROJECT TITLE	Next Generation IoT as part of Next Generation Internet
CALL ID	H2020-ICT-2020-1
CALL NAME	Information and Communication Technologies
TOPIC	ICT-56-2020 - Next Generation Internet of Things
TYPE OF ACTION	Research and Innovation Action
COORDINATOR	Capgemini Technology Services (CAP)
PRINCIPAL CONTRACTORS	Atos Spain S.A. (ATOS), ERICSSON GmbH (EDD), ABB Oy (ABB), INTRASOFT International S.A. (INTRA), Engineering-Ingegneria Informatica SPA (ENG), Bosch Sistemas de Frenado S.L.U. (BOSCH), ASM Terni SpA (ASM), Forum Virium Helsinki (FVH), Optimum Technologies Piroforikis S.A. (OPT), eBOS Technologies Ltd (EBOS), Privanova SAS (PRI), Synelxis Solutions S.A. (SYN), CUMUCORE Oy (CMC), Emotion s.r.l. (EMOT), AALTO-Korkeakoulusaatio (AALTO), i2CAT Foundation (I2CAT), Rheinisch-Westfälische Technische Hochschule Aachen (RWTH), Sorbonne Université (SU)
WORKPACKAGE	WP2
DELIVERABLE TYPE	REPORT
DISSEMINATION LEVEL	PUBLIC
DELIVERABLE STATE	FINAL
CONTRACTUAL DATE OF DELIVERY	31/05/2022
ACTUAL DATE OF DELIVERY	31/05/2022
DOCUMENT TITLE	Enhancing IoT Underlying Technology
AUTHOR(S)	Fiona Williams (EDD), Robert Farac (EDD), Tuana Ensezgin (EDD), Felix Maier (EDD), Alexandra Bach (EDD), Serge Fdida (SU), Tomás Lagos Jenschke (SU), Jose Costa-Requena (CMC), Jonathan Klimt (RWTH), Manuel Pitz (RWTH), Stefan Lankes (RWTH), Marios Sophocleous (EBOS), Reza Mosahebfard (I2CAT)
REVIEWER(S)	Manuel Pitz (RWTH)
ABSTRACT	In the IoT-NGIN project, we have defined and are developing enhancements to 5G technologies which will better enable 5G networks to meet the growing communications requirements of IoT and vertical sector use cases as they move towards commercial maturity and large-scale deployment. The deliverable describes the progress we have made to date in defining, specifying and implementing our 5G enhancements. This deliverable builds on the results of D2.1, submitted in February, 2022.
HISTORY	SEE DOCUMENT HISTORY BELOW
KEYWORDS	5G, M2M, MCM, API, IoT, TSN, 5GLAN, D2D, network slicing, time sensitive communications, 5G device management, 5G coverage extension, 5G resource management adapter, 3GPP, standardisation.

Document History

Version	Date	Contributor(s)	Description
V0.1	16/12/2021	EDD	First draft
V0.2	16/01/2022	CMC	CMC deliverable contribution plan
V0.3	19/01/2022	EDD	ToC proposal from EDD
V0.4	23/01/2022	EDD	5G API and requirements exploitation
V0.5	26/01/2022	EDD	First D2.2 meeting
V0.6	17/02/2022	EDD	WP2 meeting
V0.7	23/02/2022	EDD	5G related project activities
V0.8	24/02/2022	EDD	Final document structure
V0.9	25/02/2022	EDD	Draft text from EDD
V0.10	03/03/2022	EDD	Draft text from EDD on Chapter 5 and Annex
V0.11	04.03.2022	EDD	Chapters 5 and 8 restructured
V0.12	07.03.2022	SU, eBOS	First text draft from SU and eBOS (Ch2)
V0.13	08.03.2022	CMC	First text draft from CMC (Ch3)
V0.14	08.03.2022	EDD	Chapter 5 (5G API) restructured
V0.15	08.03.2022	EDD	Annex restructured
V0.16	23.03.2022	ALL	D2.2 progress meeting
V0.17	24.03.2022	i2CAT	I2CAT and EDD meeting
V0.18	26.03.2022	CMC	CMC-EDD meeting
V0.19	31.03.2022	RWTH	5G resource management API architecture description
V0.20	01.04.2022	CMC	5GLAN/TSN implementation and test description
V0.21	05.04.2022	EDD	Updates in Chapter 1, 2, 5, 8 and Annex
V0.22	06.04.2022	EDD	Updates in all chapters and Annex
V0.23	06.04.2022	EDD, RWTH	Document ready for the D2.2 progress meeting
V0.24	15.04.2022	i2CAT	Update from i2CAT

V0.25	25.04.2022	SU	Contribution to Chapter 1 and 3
V0.26	02.05.2022	RWTH	Contribution to Sub-chapter 5.3.3
V0.27	02.05.2022	RWTH	Contribution to Chapter 6
V0.28	02.05.2022	CMC	Contribution to Chapter 4 and 5
V0.29	02.05.2022	i2CAT	Contribution to Chapter 5
V0.30	05.05.2022	RWTH, i2CAT	Contributions to Chapter 5 and Annex 1
V0.31	06.05.2022	EDD	Adding the headers and footers missing from template
V0.32	09.05.2022	CMC	Contribution to Chapter 4
V0.33	09.05.2022	RWTH	Contribution to Chapter 5, 6 and Annex
V0.34	09.05.2022	EDD	Document ready for internal review
V1.0	31.05.2022	EDD	All comments addressed. Ready for submission.

Table of Contents

Document History	4
Table of Contents	6
List of Figures	9
List of Tables	11
List of Terms, Acronyms and Abbreviations	12
Executive Summary	15
1 Introduction.....	17
1.1 Intended Audience for this deliverable	17
1.2 Relationship to other IoT-NGIN activities.....	17
1.3 Structure of this deliverable	18
2 Building the synergies between 5G and IoT-NGIN application use cases	19
2.1 IoT and 5G technologies are merging.....	19
2.2 The method of work on the 5G enhancements and relationships to other work packages	21
2.3 Conclusions.....	23
3 Enhancing 5G functionality to improve 5G coverage.....	24
3.1 Introduction	24
3.2 D2D communications in the IoT-NGIN context	24
3.3 Characterization of connection latency	28
3.3.1 Measurement methodology	29
3.3.2 Experimental setup.....	29
3.3.3 Results.....	30
3.4 Merging elements: 5G core and D2D extension	32
3.5 Next steps in this work	34
3.6 Conclusions.....	34
4 Enabling 5G to support protocols for deterministic communications	36
4.1 Introduction	36
4.2 TSN communications in the IoT-NGIN context	36
4.3 IoT MCM communication optimization	38
4.4 TSN based IoT communication.....	40
4.5 Results.....	42
4.6 Next steps in this work	43
4.7 Conclusions.....	43

5	Enhancing 5G ease of use through improved 5G APIs.....	44
5.1	Introduction	44
5.2	IoT-NGIN 5G resource management API capabilities and architecture.....	45
5.3	The 5G capabilities exposure APIs	46
5.3.1	Features of the 5G capabilities exposure APIs.....	46
5.3.2	OpenAPI specifications of the 5G capabilities exposure APIs	48
5.3.3	Sequence Diagrams of the 5G capabilities exposure APIs	50
5.3.4	Implementation descriptions of the 5G capabilities exposure APIs.....	55
5.4	IoT-NGIN 5G resource management API	58
5.4.1	IoT-NGIN 5G resource management API features	58
5.4.2	OpenAPI Specifications of the IoT-NGIN 5G resource management API	59
5.4.3	Sequence Diagrams of the IoT-NGIN 5G resource management API	60
5.4.4	Implementation descriptions of the IoT-NGIN 5G resource management API..	63
5.4.5	Living Lab API feedback.....	64
5.5	Next steps.....	65
5.6	Conclusions.....	65
6	Enhancing 5G and IoT applications security with a secure edge cloud micro-services execution framework	67
6.1	Approaches to edge cloud infrastructure	67
6.2	Unikernel and microVMs	68
6.3	The IoT-NGIN Secure Edge Cloud Framework	69
6.4	Next steps of this work.....	72
6.5	Conclusions.....	72
7	Relationship of the 5G enhancements to markets and LL use cases	73
7.1	The target markets for IoT-NGIN 5G enhancements	73
7.2	The relationship of the 5G enhancements to the LL use cases	74
7.3	Conclusions.....	75
8	Conclusions.....	76
9	References.....	77
Annex 1	Features of the 5G capabilities exposure APIs	78
Annex 1.1	Features of the network slice management API.....	78
Annex 1.2	Features of the 5G device management API.....	78
Annex 2	OpenAPI Specifications of the 5G capabilities exposure APIs.....	81
Annex 2.1	OpenAPI specifications for the network slice management API	81
Annex 2.2	OpenAPI specifications for the 5G device management	81

Annex 3	Sequence diagrams of the 5G device management API.....	93
Annex 4	Implementation description of the 5G device management API	99
Annex 5	Sequence diagrams for the IoT-NGIN 5G resource management API.....	101
Annex 5.1	Start and stop service	101
Annex 5.2	Resource allocation.....	102
Annex 6	Security features of 5G today	104
Annex 6.1	Introduction to 5G security in IoT-NGIN.....	104
Annex 6.2	Introduction to 5G security features.....	104

List of Figures

Figure 1-1 The topics of interactions between WP2 and other WPs in IoT-NGIN	18
Figure 3-1 Concept of 5G coverage extension through D2D using the TSN Bridge.	25
Figure 3-2 Concept of 5G coverage extension through D2D using smartphones as relays ..	26
Figure 3-3 Architecture on experimental setup for tests on throughput	27
Figure 3-4 When A and B get sufficiently close to each other, it takes some time for the discovery mechanisms of the two devices to detect the presence of the respective neighbour. The consequence is the waste of communication resources during the discovery period.	28
Figure 3-5 D2D communication connectivity flow diagram.....	29
Figure 3-6 Bluetooth Classic latency using the Samsung S20 as a discovery device.	31
Figure 3-7 BLE discovery latency using the Samsung S20 as a discovery device.....	31
Figure 3-8 Scenario on the use of 5GLAN starting from position 1 (up) moving towards position 2 (down)	33
Figure 3-9 Architecture of experimental setups for future tests.	34
Figure 4-1 ABB TSN network topology	37
Figure 4-2 5GLAN architecture and network functions	40
Figure 4-3 3GPP TSN architecture	42
Figure 4-4 Lab measurements TSN synchronization	43
Figure 5-1 IoT-NGIN 5G resource management API	46
Figure 5-2 Sequence diagram for 5GLAN	51
Figure 5-3 Workflow example of the service instantiation using i2CAT's tool, SOE, over edge domain	52
Figure 5-4 Workflow example of device management using the 5G device management API	54
Figure 5-5 5GLAN and network configuration through the CumuCore Network Controller (CNC) API.....	56
Figure 5-6 Swagger IoT-NGIN 5G resource management API definition	60
Figure 5-7 Service migration Connectivity management.....	62
Figure 5-8 Connectivity management	63
Figure 5-9 Example code generation utilizing the OpenAPI tools	64
Figure 6-1 Comparing container technology-based OS-level virtualization and virtual machines.....	68
Figure 6-2 Container runtime based on a microVM	69
Figure 6-3 Comparing container runtimes based on the OCI runtime specification	70
Figure 6-4 Startup time for Hermit and Linux containers under increasing system load	71

Figure 6-5 Enhancing 5G and IoT applications security with a secure edge cloud micro-services execution framework..... 72

Figure 9-1 Start and stop sequence 102

Figure 9-2 Resource allocation sequence..... 103

List of Tables

Table 2-1 Issues in the cellular wireless communications sector	20
Table 2-2 A brief summary of the WP2 relationship to the other IoT-NGIN WPs.....	22
Table 3-1 Location of the devices under study.....	30
Table 5-1 Description of the 5G device management API features.....	47
Table 5-2 List of features.....	58
Table 5-3 List of 5G IoT-NGIN Resource Management API operations	59
Table 7-1 Target 5G markets addressed by IoT-NGIN 5G-Enhancements.....	73
Table 7-2 Living Lab field trial site use of Public 5G or Private 5G with project 5G enhancements	74
Table 9-1 The description of various 5G device management API features relevant for IoT-NGIN applications	78

List of Terms, Acronyms and Abbreviations

Terms

5G enhancements	We use the term 5G enhancements throughout this document to refer to our collective work on improving the support 5G provides to IoT.
Swagger	Swagger is a tool commonly used to prepare OpenAPI specifications.
OpenAPI	OpenAPI specification describes the specification of machine-readable interface files for describing, producing, consuming, and visualizing RESTful web services, overseen by the OpenAPI Initiative, an open-source collaboration project of the Linux Foundation
An open API	An open API (often referred to as a public API) is a publicly available application programming interface that provides developers with programmatic access to a proprietary software application or web service.
IoT-NGIN 5G resource management API	A generic 5G resource management API is developed in this project. This API simplifies 5G interactions for developers and users.

Abbreviations

3GPP	3rd Generation Partnership Project	API	Application Programming Interface
5GCN	5G Core Network	BLE	Bluetooth Low Energy
5GLAN	5G Local Area Network	BSI	German Federal Office for Information Security
5GS	5G System	CAG	Closed Access Group
ACIA	Alliance for Connected Industries and Automation	CNC	Centralized Network Configuration
AF	Application Function	CNFD	Cloud-Native Function Descriptor
AMF	Access and Mobility Management Function	CPU	Central Processing Unit
		D2D	Device to Device
		DB	Data Base

D2.2 – Enhancing IoT Underlying Technology

DHCP	Dynamic Host Configuration Protocol	LLDP	Link Layer Discovery Protocol
DNN	Data Network Name	LTE	Long Term Evolution
DNS	Domain Name System	M2M	Machine-to-machine
DS-TT	Device-side TSN Translator	MAC	Media Access Control
ETSI	European Telecommunications Standards Institute	MANO	Management and Orchestration Function
GMF	Group Management Function	MCC	Mobile Country Code
GMLC	Gateway Mobile Location Center	ML	Machine Learning
GPSI	General Public Subscription Identifier	MNC	Mobile Network Code
gPTP	Generalized Precision Time Protocol	MSISDN	Mobile Station International Subscriber Directory Number
GPU	Graphics Processing Unit	MTO	Multi-Tier Orchestrations
GSM	Global System for Mobile	NAI	Network Access Identifier
GUI	Graphical User Interface	NAT	Network Address Translator
HTTP	Hypertext Transfer Protocol	NEF	Network Exposure Function
IEEE	Institute of Electrical and Electronics Engineers	NFVO	Network Function Virtualization Orchestration
IMS	IP Multimedia Subsystem	NMS	Network Management System
IoT	Internet of Things	NPN	Non-Public Network
IP	Internet Protocol	NSA	National Security Agency
ITU	International Telecommunication Union	NSD	Network Service Descriptor
LL	Living Lab	NW-TT	Network-side TSN Translator
		O&M	Operation and Management

D2.2 – Enhancing IoT Underlying Technology

OAM	Operation and Maintenance	SOE	Slice and Orchestration Engine
OCI	Open Container Initiative	SUCI	Subscription Concealed Identifier
OPC-UA	Open Platform Communication Unified Architecture	SUPI	Subscriber Permanent Identity
OSM	Open Source MANO	TRL	Technology Readiness Level
P2P	Peer-to-peer	TSN	Time Sensitive Networking
PCF	Policy Control Function	UC	Use Case
PCI	Payment Card Industry	UDR	User Data Repository
PDU	Protocol Data Unit	UE	User Equipment
PLMN	Public Land Mobile Network	UPF	User Plane Function
PSFP	Per-Stream Filtering and Policing	UPnP	Universal Plug and Play
PVN	Private Virtual Network	URLLC	Ultra-Reliable Low-Latency Communication
QoS	Quality of Service	V2V	Vehicle-to-vehicle
RAM	Random-Access Memory	VIM	Virtual Infrastructure Manager
RAN	Radio Access Network	VLAN	Virtual Local Area Network
REST	Representational State Transfer	VM	Virtual Machines
RPC	Remote Procedure Calls	VN	Virtual Network
SEAL	Service Enabler Architecture Layer	WLAN	Wireless Local Area Network
SIM	Subscriber Identity Module	WP	Work Package
SM	Slice Manager		
SME	Small and medium enterprises		
SMF	Session Management Function		

Executive Summary

As the use of IoT technologies in use cases of application sectors (such as smart cities, smart agriculture, smart energy and smart industry) develops momentum, their need for communications networking support is rapidly increasing. At the same time, LTE and 5G technologies offer the support which many IoT use cases need today in their implementations. In the IoT-NGIN project, we have defined and are developing a challenging set of enhancements to 5G technologies. The technical ambition level of each enhancement and the complexity of organising exploitation in the context of global standardisation are high. These 5G enhancements will better enable 5G networks to meet the growing communications requirements of IoT in vertical sector use cases as they move towards commercial maturity and large-scale deployment.

Good development of relationships between partners in the project and the definition of the IoT-NGIN architecture have enabled us to define optimal plans for implementation, testing and deployment of the 5G enhancements we are developing in relation to the needs of the Living Lab use cases.

One aspect of our work on enhancements is focussed on improving coverage by optimising the use of relay connections. To do this, relay selection strategies which consider the characteristics of the network at the time of communication are needed. To this end, techniques and methods must be developed to measure the quality of the links in real situations, especially that of the device-to-device (D2D) links. We have already implemented a tool which enables us to characterize the connection latency of D2D connections. We have used this new tool to determine latencies of connections using Bluetooth Classic and Bluetooth Low Energy (BLE) connections as relay connections for D2D connectivity. These tests show that Bluetooth Classic meets the requirements of this project better than BLE. In addition, we have defined a physical architecture necessary for the D2D communication to work as an extension of a 5G core while ensuring that the architecture supports the solutions required by the IoT-NGIN use cases. We are setting up a lab test-bed and installing a new 5G core to enable testing of further link configurations in the coming months.

To enable and simplify future industry 4.0 process and application control (which could be based on edge cloud) connections to devices, such as frequency converters and motors (Variable speed drives), over private 5G networks are needed. Combining control functions, data collection & storage, analytics and flexibility to change configurations has the potential to improve and enable new features and benefits for industrial control. The deployment of such enhanced 5G network functionality requires supporting both the synchronisation of devices using 5G and the connection of the devices with 5GLAN. Such functionality is under development in the project and preliminary tests of the performance of the initial implementations have been performed. However, we have identified limitations in our initial implementations. These limitations will be addressed through optimising the implementation of network functions. The expected result of this work is that it will improve the level of accuracy of synchronisation between clocks in fixed network and user equipment. Our target level of precision is not realizable in our current implementations, which are based on current 5G standards.

Modern 5G-enabled services, in particular those deployed as private 5G networks, will play an important role in future industry 4.0 deployments. To simplify the use of the 5G services and functionality, new interfaces between IoT and 5G infrastructures are required. These

Interfaces are defined as APIs (Application Programme Interfaces) which are exposed to the IoT applications. In IoT-NGIN, we are defining and implementing a simple to use IoT-NGIN 5G resource management API. This API provides a more generic interface than those available today, hiding the complexity of 5G from the user. We have selected appropriate resource management functionality and described the API specifications and sequence diagrams of this functionality. Additionally, we are specifying and developing sequence diagrams for example of the IoT-NGIN 5G resource management API. The IoT-NGIN 5G resource management API provides a generalised version of a range of currently available standardised APIs and proposes a set of new APIs.

To improve the security of Edge Clouds and reduce the overheads associated with virtualisation of micro-services in such architectures, IoT-NGIN is researching a flexible edge-cloud framework. We have investigated and developed an approach based on the use of microVM's with a unikernel approach and made it the central component of the implementation of our IoT-NGIN flexible and secure edge-cloud framework. The integration of our approach with common and existing orchestration tools, such as Kubernetes, facilitates the uptake of our results. In IoT-NGIN, we created our own container spawner called runh based on Kubernetes functionality. Runh is able to spawn Hermit containers and also containers based on a microVM and the unikernel RustyHermit. A prototype implementation of our functionality is already available in GitHub.

The 5G enhancements being developed in the project target two communications network market segments: wide area public networks and local area networks, supporting the range of needs of the IoT-NGIN use cases.

The key takeaways from our work so far are that:

- Standard LTE and 5G networks offer good support for the current requirements of many of the Living Lab small scale implementations of the use IoT-NGIN cases studied,
- The specific 5G enhancements being developed in IoT-NGIN have the potential to make a strong contribution to enabling the large-scale commercial deployment of the IoT-NGIN use cases. We extend the capabilities of existing standard 5G networks with features enabling optimised performance of the network and we increase its ease of use for industrial sector users of 5G, and that
- Additionally, we have learned that our IoT-NGIN 5G enhancements can enable new use cases in industrial sectors going beyond those defined for the Living Labs of IoT-NGIN.

In summary, we have made excellent progress towards achieving our objective of improving the support which 5G can provide to the large-scale use of IoT use cases addressing smart agriculture, smart industry, smart cities and smart energy. Extensive results of our architecture investigations have been documented and are being implemented, prototypes have been developed, tested and are being improved, specifications, sequence diagrams and implementation descriptions of the IoT-NGIN 5G resource management API have been proposed and are being further developed and the standardisation of new techniques and functionality is being investigated and promoted in 3GPP.

1 Introduction

This deliverable is the second deliverable of the IoT-NGIN project describing project work on enhancing IoT underlying technology. Our approach in WP2 to enhancing the IoT underlying technology is to develop a set of enhancements to 5G network and edge cloud capabilities. We use the term 5G enhancements throughout this document to refer to our collective work on improving the support 5G provides to IoT.

During the 6-month period between the publication of our first description of the 5G enhancements in D2.1 in November, 2021 and the publication of this deliverable in May, 2022, our work has progressed well. Apart from the progress on each of the 5G enhancements themselves, very good progress was made on undertaking and planning the testing of the enhancements and on building the relationship to the Living Lab field trials of the IoT-NGIN project.

This chapter provides the context information to enable the reader to understand the structure of this deliverable and how the individual chapters relate to each other. This chapter relates the contents of this deliverable to other existing project deliverables and to the future deliverables of the IoT-NGIN project.

1.1 Intended Audience for this deliverable

This deliverable will be useful to a wide audience of readers. Within the IoT-NGIN project, it provides all partners with an in-depth description of the progress on 5G enhancements. It provides project reviewers and the European Commission with a description of our achievements.

Additionally, readers in the IoT, mobile communications, smart agriculture, smart industry, smart cities and smart energy sectors will benefit from the descriptions of 5G enhancements as they provide insight into potential new products, services and use cases which 5G could support in these sectors in the coming years.

1.2 Relationship to other IoT-NGIN activities

This deliverable reports on the work of all activities ongoing on WP2. It builds on input from the use cases defined in WP1 and feeds into the application development, integration and testing activities of WP3 to WP6, and the living lab planning in WP7. Results are regularly provided to WP8 for communication, standardisation and exploitation activities of the project.

As shown in Figure 1-1 below, WP1 provides guidelines related to the uses-cases and their potential requirements to WP2. Similarly, WP2 contributes to the overall discussion on the architecture. WP3 defines requirements that can be considered in WP2. WP4 is addressing specific applications and is expecting efficient IoT device discovery. WP5 is concerned with enhancing IoT cybersecurity and data privacy. WP2 will consider specific security requirements and how they are addressed by 5G technology.

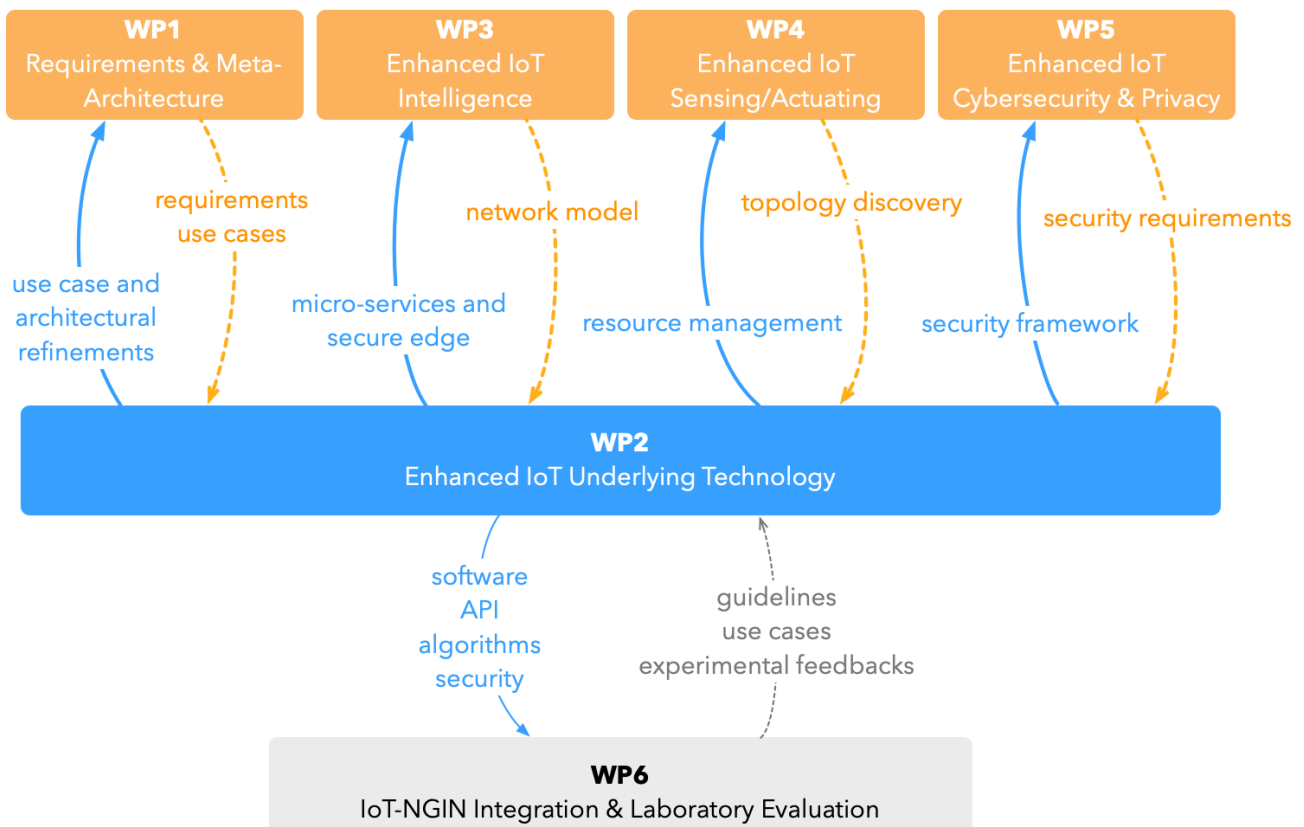


Figure 1-1 The topics of interactions between WP2 and other WPs in IoT-NGIN

1.3 Structure of this deliverable

In this deliverable, we describe the progress made in each of our enhancement topics and an overview of our plans to deploy and test each enhancement. We provide:

- An introduction to the challenges of addressing the merger of IoT and 5G technologies and the methodology of our work on 5G enhancements in IoT-NGIN,
- An updated description of our work to provide tools and algorithms which have the goal of enabling improved 5G coverage through the optimisation of existing device to device communications features,
- An updated description of our work on protocols which enable 5G deterministic communications,
- An initial description of the implementation of the IoT-NGIN 5G resource management API including communication sequence diagrams and OpenAPI specifications,
- A description of an initial version of a secure edge cloud framework for IoT micro-services,
- An overview of the target markets addressed by our 5G enhancements is provided, as is an overview of our plans for testing our 5G enhancements in laboratory and Living Lab field trials, and
- In the annexes, we provide additional 5G-API specifications, sequence diagrams and an overview of standard 5G security features supporting the security of our IoT-NGIN 5G enhancements.

2 Building the synergies between 5G and IoT-NGIN application use cases

This chapter provides an overview of the challenges driving change in the IoT and 5G sectors, 5G enhancements and the relationships we have developed to the other work packages in the project.

2.1 IoT and 5G technologies are merging

Both the IoT and 5G sectors are rapidly changing and both industrial sectors are increasingly interested to explore new business opportunities building on the synergies between the sectors. In this chapter, we provide information on the general trends and situation in the IoT and 5G sectors related to the challenges of integrating their technologies.

The **key driver** for change in the **IoT sector** is the growing need to support the digitalisation and communications requirements of many new applications. Digitalisation, supported by mobile communications, is needed to provide the data sets which can improve operational efficiency in the use of resources, and support for decision making, in many IoT use cases from crop spraying to manufacturing and power grid management. Climate change and pollution in cities are driving the need to decarbonise transport and to make multi-modal transport a reality, leading to IoT application requirements for reliable wireless communications support.

The architectures of IoT applications enabling digitalisation often benefit from the integration of mobile communications features. Such architectures for IoT use cases can benefit from cellular wireless communication system functionality and characteristics such as edge infrastructure, low latency, security, reliability, high availability and the guaranteed QoS which 5G network slicing could provide.

The 5G sector is characterized by long timescales related to standardisation cycles in the cellular wireless communications sector. There are two **key drivers** for change in the **cellular wireless communications sector** as the use cases for 5G expand in scope.

1. The need to support the growing communications requirements of industrial sectors is generating a new market for 5G networks

The rapid uptake of 5G technology in public networks in Europe and throughout the world, combined with the high reliability and low latency characteristics of 5G networks, has created the basis for the development of a market for 5G services for industrial sectors. Already in 2021, the population coverage of 5G networks had doubled compared to 2020 and 5G networks are now available to 62% of the European population. Almost all of the European population living in cities has coverage by 5G networks and coverage of motorways is also very good. Now that 5G coverage in urban areas is no longer an issue, the focus of the mobile sector is increasingly on addressing the needs of IoT use cases. The 5G public operators are interested to learn about experiences of using 5G to support industrial use cases through Living Labs and field trials.

2. The rapid growth in the market for private mobile networks

The rapid uptake of private campus-area LTE and 5G networks in the past few years has generated a rapidly growing market in which manufacturers offer standardised and proprietary versions of 5G networks tailored to the needs of specific application sectors such as the smart factory and energy sectors.

Key characteristics of the cellular wireless communications sector

The global standardisation of mobile communications enhancements often requires significant effort and long timescales as described in Table 2-1 below.

Table 2-1 Issues in the cellular wireless communications sector

Sector features	Cellular wireless communications standardisation
Time to market for new features and systems	<p>In private local mobile networks, new features or customisations can often be offered quickly by system developers if no standardisation of the features is planned or required</p> <p>For public mobile networks, the introduction of new network features, depending on the standardisation cycles for the mobile generation, can sometimes require timescales of several years for implementation</p>
Complexity of the development work needed to introduce network enhancements	<p>Cellular mobile communication systems, such as LTE and 5G, are very complex. Backward compatibility with previous mobile generations as well as standardisation, legal and regulatory requirements often make the development of network level enhancements complex and time consuming. New standardised generations of cellular systems have been introduced globally approximately every 10 years since the launch of the GSM system in 1990.</p>
Interoperability requirements fulfilment	<p>Following the global standardisation of new cellular systems and features, interoperability testing of such systems by manufacturers needs to be planned and undertaken.</p>
Test infrastructure requirements	<p>Globally standardised mobile networks are large and complex systems. Setting up 5G test infrastructures for broad scope enhancements is expensive, because a large range of system nodes may have to be purchased, configured and tested before the testing of enhancements can be started. Additionally, a team of international experts on the relevant system nodes may be required to run tests and to interpret the results of the tests.</p>

IoT and Cloud are now increasingly supported in mobile communications infrastructures. Mobile communications networks are based on the principles of IT infrastructures. For example, mobile communications network use containers, HTTP protocol and increasingly makes use of Open-Source code. Rather than basing the communications of IoT applications purely on Ethernet and WLAN networks, IoT providers today often include support for communications base on LTE and 5G in their systems.

2.2 The method of work on the 5G enhancements and relationships to other work packages

In this sub-chapter, we describe how the project work packages are combining their efforts to ensure that synergies between work items in the project are maximised as part of our preparation for the future commercial exploitation of the project results. Our enhancements address a set of issues that are hot topics in the 5G sector and which relate to the needs of the IoT-NGIN Living Labs and the IoT sector in general.

Our enhancements address different parts of the 5G network from device context to core network enhancements and edge enhancements. Due to the complexity and large scale of 5G systems integrating new enhancements into 5G products and services requires resources far beyond the available resources in the IoT-NGIN project. Therefore, we are developing the enhancements individually as proof-of-concept prototypes. Investigations of the possibilities to validate them in IoT-NGIN Living Labs and IoT-NGIN laboratory settings are on-going.

Our enhancements address a range of timescales to maturity in the market. The IoT-NGIN network slicing management, TSN functionality and the 5G device management API are close to market approaches which can be commercially offered within a few years. The full scope of the IoT-NGIN 5G API for resource management and the novel approach to virtualizing edge clouds require promotion and discussion in the 5G sector and a longer timescale to achieving market adoption of results into 5G and 6G standards and products of 5 to 10 years or even longer.

Our enhancements address different maturity levels concerning potential deployments in Living Lab field trials. The timescales and efforts required to bring 5G network enhancements to the maturity level required to deploy them in public networks are beyond the resources of the project. This restricts our ability to deploy 5G enhancements in the Living Lab field trials which require wide area cellular coverage. We are investigating deployments of private cellular networks for those use cases requiring campus area cellular coverage where our project budgets can support such a deployment.

Our enhancements address different approaches to valorisation and exploitation, ranging from the publication of results as open-source software to perspectives for the development of products implementing globally adapted 3GPP standards, based on the standardisation efforts of partners.

As a working method, we have organised numerous focussed virtual meetings between partners on key issues crossing work package boundaries. These efforts have resulted in cross-work package integration of themes and tasks as well as the integration of technologies and the development of a joint understanding by partners of the use cases of the IoT-NGIN Living Labs.

In Table 2-2 below, we summarise the relationship between the WP2 efforts on 5G enhancements and the complementary work of the other WPs in IoT-NGIN.

Table 2-2 A brief summary of the WP2 relationship to the other IoT-NGIN WPs

Work Packages	Description of the relationship to WP2
Requirements and architecture (WP1)	5G enhancements have been integrated into the architecture and use case requirements work of WP1. We are studying the context of the Living Lab use cases to identify how 5G enhancements can enable new use cases, going beyond those planned in the project work, thus extending the markets which the exploitation of project results can address.
Development (WPs 3,4,5)	WP2 has an ongoing dialogue with application developers in WPs 3,4 and 5 to develop synergies and mutual understanding of 5G enhancements. The integration of the Secure Edge Framework with the IoT-NGIN machine learning functionality has been investigated and is planned.
Laboratory trials (WP6)	Several sets of tests have already been run by partners developing 5G enhancements and further tests are ongoing and planned in the second reporting period. We are using 5G laboratory infrastructure in France, Germany and Spain. Laboratory proof-of-concept implementations of selected use cases are being investigated.
LL trials (WP7)	<p>The deployment of relevant 5G enhancements targeting private networks at Living Lab trial sites are being investigated. The availability of public standard 5G and the possibility to provide private 5G networks at Living Lab sites continue to be investigated.</p> <p>We have related the enhancements developed in WP2 to the roles they can play in the specific use cases of the Living Labs. In particular, some of our enhancements targeting private 5G campus area networks match requirements of use cases needing campus area network coverage. Other enhancements target wide area 5G networks matching use cases requiring wide area coverage. As our resources are insufficient to consider supporting the deployment of 5G enhancements in all use cases, we focus our efforts on deploying specific enhancements of one or two use cases sites. Some of our enhancements are not mature enough to deploy in the field or have restrictions because of ongoing standardisation of the enhancement.</p> <p>As part of initial partner exploitation planning for the 5G enhancements, WP2 prepared a structured questionnaire in late 2021. It was distributed to living lab partners in early 2022 to elucidate their views on the role which the IoT-NGIN 5G enhancements could play in enabling and accelerating the uptake of commercial large-scale deployments of the Living Lab applications in the coming years.</p>
Dissemination and exploitation (WP8)	WP2 partners have been active to contribute to the project dissemination activities and exploitation planning. They have already made contributions to several associations and are

Project Management (WP9)	<p>investigating and preparing standards contributions to the mobile communication global standards body, 3GPP. Contributions to Open-Source repositories are planned by WP2. The results of WP2 are being promoted in relevant public events, such as the EuCNC & 6G Summit to be held in Grenoble on 7-10 June 2022, and in publications.</p> <p>Input from WP2 to all project management tasks in WP9 and, as relevant, to the open calls of WP9, has been prepared.</p>
--------------------------	---

2.3 Conclusions

During the project work so far, partners have developed a mutual understanding of the different ways of working and timescales to bring new functionality to market in the IoT and 5G domains.

We have integrated this mutual understanding into:

- The IoT-NGIN architecture,
- The planning for the development of the IoT-NGIN platform,
- The use of 5G in Living Lab field trials and laboratory trials, and into
- Project discussions on the dissemination and exploitation contexts relevant to project results.

3 Enhancing 5G functionality to improve 5G coverage

3.1 Introduction

Almost every city in the world is deploying 5G public networks as 5G cell phones become available in the global market. In the IoT-NGIN project, we are investigating scopes and methodologies in device-to-device (D2D) communications to extend network coverage and thus maximize the support that available networks can offer to applications and their users.

We propose a simple but effective methodology for coverage extension by establishing a D2D communication between the node outside the cell coverage area and a relay inside the coverage area. An exchange of metrics is established between the participants to select the most suitable relay for the target performance. This selection process essentially involves three steps:

1. Characterization of the links between potential relays and destination nodes, an
2. Assessment of the dynamics of the network, and an
3. Algorithm to select the most appropriate relay, if any.

The implementation of the process is available in the AtomD tool. We have developed functionality as part of the AtomD application that allows us to repeat and automate the neighbour discovery procedure and log the discovery and connectivity request processes. AtomD was configured to allow devices to establish point-to-point P2P links. In addition, we configured the application to perform a set of experiments. The results of these experiments with Bluetooth Classic and BLE are described in this chapter.

The AtomD tool is intended to be provided as open-source software, creating accessibility of our results to a global community of developers. Our AtomD tool addresses the need to extend the coverage of wide area 5G networks. It could also be useful in privately owned campus 5G networks for applications without stringent latency and reliability requirements. The use of this functionality in the IoT-NGIN use cases is being investigated.

3.2 D2D communications in the IoT-NGIN context

One of the common challenges on the implementation of 5G in wide area networks is the short network coverage of approximately 500 m depending on the frequencies used and the attenuation of signals in a given location. In applications such as smart cities (UC1 of IoT-NGIN), due to the short coverage range, a high density of base stations is required for full coverage. IoT-NGIN proposes to use Device-to-Device (D2D) communications to enhance coverage in areas not well served by public networks. D2D technique is a communication method that provides the connection between two wireless devices without passing through an existing network infrastructure. Therefore, when devices communicate via D2D, data transmission can be shared between them, thus mitigating traffic on the overall core network at lower bandwidth since only one device is connected to the network. IoT-NGIN is also investigating its implementation and implications in the different Use Cases that the project intends to develop. D2D applications within wireless networks can contribute to other solutions within the telecommunications industry. In fact, if we focus on smart cities, D2D can

D2.2 – Enhancing IoT Underlying Technology

be exploited as a means of traffic distribution or data dissemination for parking prediction (UC1). One of the approaches investigated in smart cities is the distribution of different static beacons throughout the city interconnected to a base station, thus delivering data as a service. For example, the user (the driver) located near a beacon can be provided with traffic distribution information. This can be achieved using D2D communication as the transmission medium between the beacon and the user. Currently, there are no 3GPP standards on D2D communication however releases 16 and 17 started drafting the standards through the Vehicle-to-Vehicle (V2V) concept, which is very similar. Unfortunately, there aren't any quantitative standards yet although such standards are expected to be established with release 18.

However, the use of this approach give rise to challenges. The first challenge is the capabilities of such connections such as throughput, bandwidth, latency, jitter etc. The second challenge is the discovery and authentication of available relays/beacons. It is expected that this method will be used for non-time-critical applications such as sensor readings hence the required bandwidth/throughput and latency is not of high importance. D2D can provide coverage when devices are outside the network coverage of base stations. However, several issues arise in doing that, such as the IP generation for the out-of-coverage device. In fact, if another smartphone is used as the relay node, the base station and the operator will not be aware of the presence of the out-of-range device. Although this might not seem like an issue, if the relay node has a limited amount of data available from the operator, the data download on behalf of the out-of-range device will be drawn from the relay's account. However, the use of a TSN Bridge will allow the acknowledgement of the out-of-range device by the base station, and an IP will be given to that device from the operator allowing the correct allocation of data roaming. In IoT-NGIN we focus on developing a TSN bridge, which can be used as a relay node that will solve this IP issue, as shown in Figure 3-1.

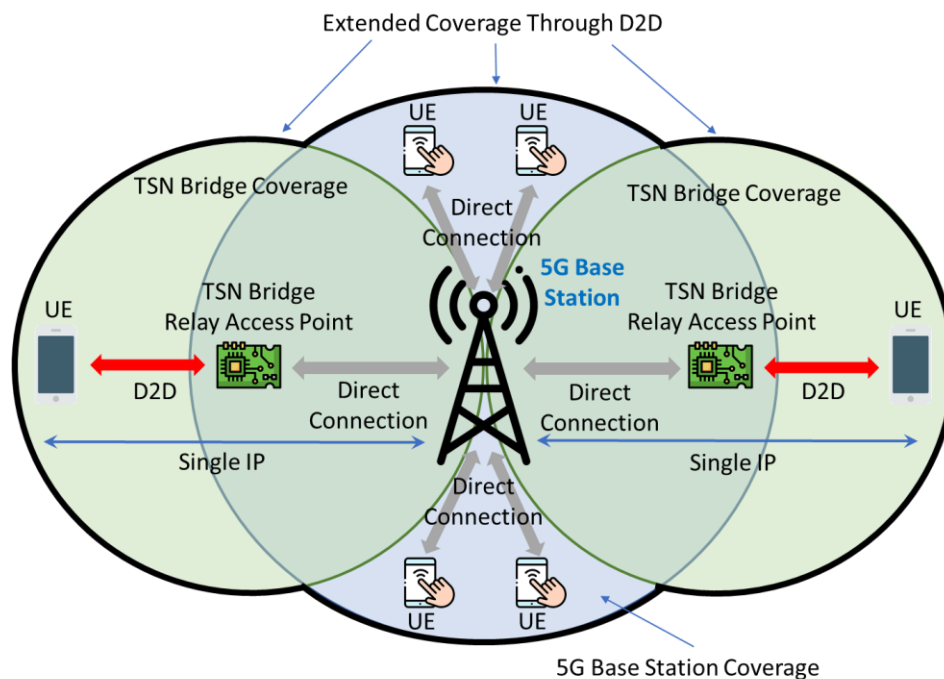


Figure 3-1 Concept of 5G coverage extension through D2D using the TSN Bridge.

D2.2 – Enhancing IoT Underlying Technology

Although this is the final implementation of the technology, at this stage, and until the TSN bridge is completed, the coverage extension will be achieved using smartphones as the relay nodes, as shown in Figure 3-2, while still achieving the coverage extension.

Another challenge to consider is security. In fact, given the type of installation D2D requires, it is prone to man-in-the-middle attacks via a malicious relay. Here, a malicious relay can be discovered by a discoverer, which allows it to capture each victim's session, encryption, and mac. As a result, the attacker could decrypt, observe, manipulate and encrypt the application layer packets. Therefore, it is essential to take user authenticity into account and keep traffic encrypted.

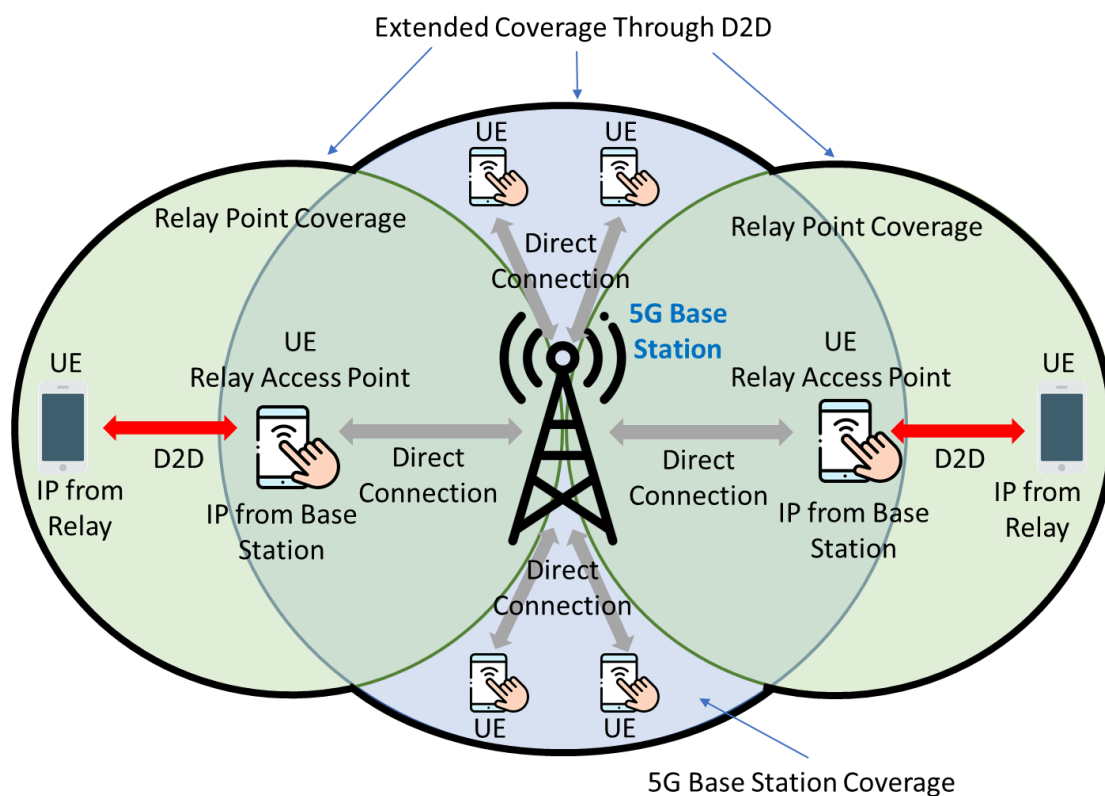


Figure 3-2 Concept of 5G coverage extension through D2D using smartphones as relays

Since the beginning of the project, we have focused on two metrics. The first one, described in the deliverable D2.1, was the throughput of D2D links once the connection was established (Figure 3-3).

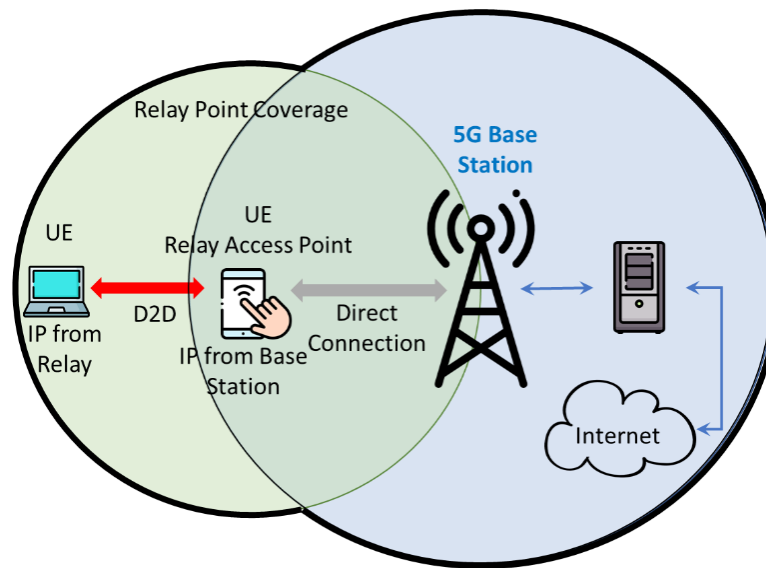


Figure 3-3 Architecture on experimental setup for tests on throughput

In this deliverable, we focus on the latency for connection establishment. In cases where multiple relay nodes are candidates to help reach out-of-range nodes, it is fundamental to introduce relay selection strategies that consider the characteristics of the network at the time of communication. To this end, we need to propose techniques and methods to measure the quality of the links in real situations, especially the D2D links.

In the following chapter, we will describe our advances with regard to the characterization of the connection latency.

We consider a scenario based on Android smartphones. Although the development is dedicated to one specific platform, the methodology is general and can be extended to other types of nodes. Furthermore, because the Android operation system does not provide explicit measurements over D2D links, we developed AtomD, a tool to measure specifically the D2D link. Its goal is to perform some actions over a link and come up with characteristics such as throughput, latency, energy consumption, and stability, to cite a few.

The problem of connection latency is illustrated in Figure 3-4. As depicted in the left-hand side of the Figure, nodes A and B move towards each other. When they enter within communication range, they trigger a discovery mechanism, which takes some time to complete. The consequence of such a delay is shown in the illustration on the right side of the figure. The shaded area indicates the resources that the nodes could have used to transfer data, but that remains unused because the connection between the nodes has not been established yet.

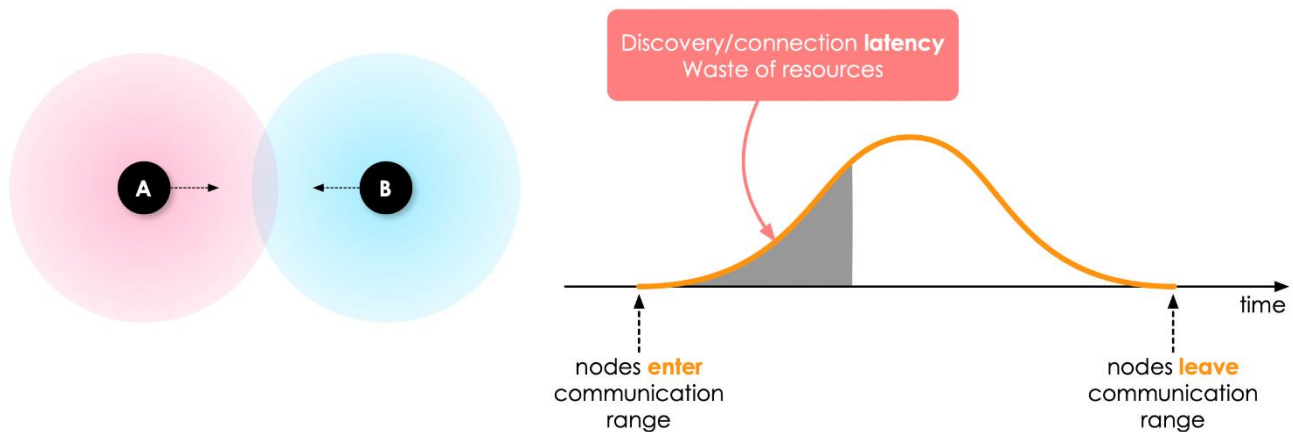


Figure 3-4 When A and B get sufficiently close to each other, it takes some time for the discovery mechanisms of the two devices to detect the presence of the respective neighbour. The consequence is the waste of communication resources during the discovery period.

In the remainder of this chapter, we will give details about the methodology, the experimental approach, and the results obtained.

3.3 Characterization of connection latency

For a D2D network to take place, the devices involved must perform neighbour discovery mechanisms to discover each other and subsequently reach an agreement to establish a direct link between them. Each device must be configured to play the role of discoverer or advertiser to achieve this. Thus, an advertiser device broadcasts beacon packets to be found, while a discoverer device sends a connection request packet upon reception of one of these beacons. Unfortunately, this procedure can harm the availability of the D2D link since the devices involved are not stationary and need some time to find each other, resulting in a shortening of the link availability time.

Consider the simple situation where a discover device D_d and an advertiser device D_a are moving with a speed V linearly towards each other. The neighbour discovery will take place at the moment that D_d enter into the wireless coverage area (C) of D_a . Thus, we obtain that the total interaction time T_{tot} is given by the following equation:

$$T_{tot} = \frac{2}{V_{D_d} - V_{D_a}} * \sqrt{\frac{C_{D_a}}{\pi}}.$$

In addition, if we consider T_d as the time in which at least one of the devices finds the other and T_c as the synchronization time to perform a communication, then we obtain that the window of communication opportunity T_{up} is given by the following equation:

$$T_{up} = T_{tot} - (T_d + T_c).$$

As a result, the window of communication opportunity is directly affected by the time that D2D takes to establish connectivity. Therefore, it is crucial to understand this in practice and consider it when making relay selections for coverage extension.

3.3.1 Measurement methodology

As shown in Figure 3-5, to establish a D2D connection, two roles are required to perform neighbour discovery. On the one hand, we have the Advertiser, which is in charge of advertising by emitting beacons. On the other hand, we have the Discoverer, which maintains a monitoring state to detect any Advertiser in its area.

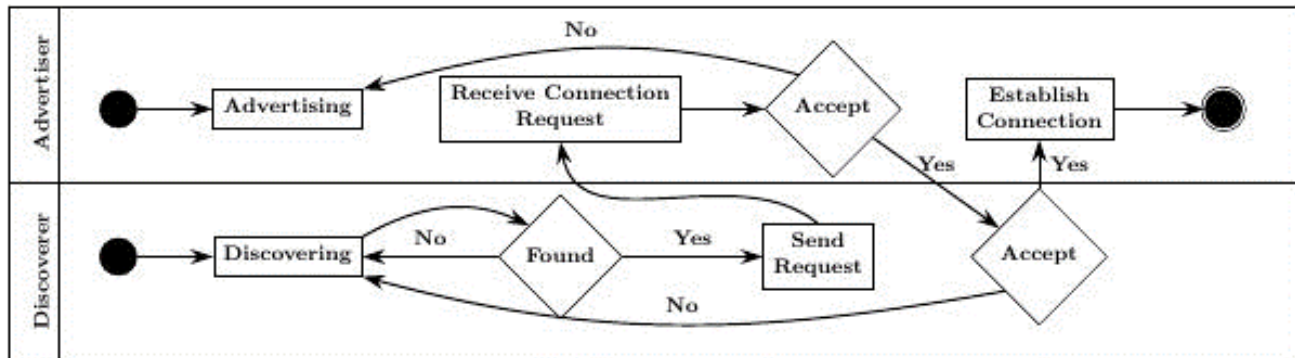


Figure 3-5 D2D communication connectivity flow diagram

As soon as a Discoverer identifies an Advertiser, it sends a connection request, which triggers an asymmetric authentication flow in which each device decides whether or not to accept the connection request. Once both agree to establish connectivity, they establish a P2P link to exchange data.

In addition to the per-device role, a topological layout is required to establish the network distribution policy, being the star, mesh, or point-to-point topologies supported by D2D. Consequently, we focus on the point-to-point topology given the requirements of this project.

Our work aims to capture and analyse the different stages of the connectivity process using Bluetooth Classic and BLE technologies. To do so, we have measured the delay of a Discoverer device in detecting an Advertiser device since it starts its beacon tracking until it successfully finds a device. Likewise, we have measured the connection request delay from when an Advertiser has sent a connection request until both devices establish a connection. For this purpose, we have developed a set of new features in our AtomD measurement application that includes capturing the time instants of each stage of connectivity.

3.3.2 Experimental setup

We conducted a set of experiments to evaluate the time it takes for two devices to detect each other and perform a D2D communication using Bluetooth Classic and BLE. Specifically, we evaluate the time it takes for a discoverer device to discover and connect to an announcer device for distances of 0 m, 20 m, 40 m, 60 m, 80 m, and 100 m. To this end, we have developed functions within the AtomD application that allow us to repeat and automate the neighbour discovery procedure and to log the discovery and connectivity request processes.

In fact, AtomD was configured to allow devices to establish point-to-point P2P links. In addition, we configured the application to perform a set of experiments. This set consisted of 25 connection attempts per set of two devices, one of which assumed the role of

Discoverer and the other the role of Advertiser. It is worth noting that each of these sets takes an average of 2.5s with Bluetooth Classic and 9.5s with BLE.

Table 3-1 Location of the devices under study

Coordinates	Latitude	Longitude	Distance from A
A	48.119562	-1.629856	0 m
B	48.119551	-1.629581	20 m
C	48.119540	-1.629316	40 m
D	48.119532	-1.628996	60 m
E	48.119519	-1.628721	80 m
F	48.119514	-1.628512	100 m

We set up a specific spacing between devices for each set of experiments, as indicated by the coordinates in Table 3-1. Here, one device is consistently placed at coordinate A, while another device is placed at the coordinate according to the distance to coordinate A.

From the selected devices, we choose the following:

- One plus 5t: Android 10, Qualcomm MSM8998 Snapdragon 835, Bluetooth - (5.0, A2DP, LE, aptX HD),
- Samsung S20 FE 5G: Android 10, Qualcomm SM8250 Snapdragon 865 5G, Bluetooth - (5.0, A2DP, LE),
- Samsung S8: Android 9, Exynos 8895 - EMEA, Bluetooth - (5.0, A2DP, LE, aptX), and
- Xiaomi Redmi 9T: Android 10, Qualcomm SM6115 Snapdragon 662, Bluetooth - (5.0, A2DP, LE).

As a result, we get a total of 12 possible combinations for both Bluetooth Classic and BLE. However, given the limitations of the Samsung S8, we set it as Discover node only, resulting in 9 available combinations for BLE.

3.3.3 Results

We have considered data with a z-score of 3, which we represented by a boxplot as shown in -Figure 3-6 , for Bluetooth Classic, and Figure 3-7 for BLE. The x-axis represents the distance between devices, as discussed in Chapter 3.3.2. In the case of Bluetooth Classic, the experiments were performed up to 100 m, while with BLE, they were performed up to 20 m (beyond this distance, the nodes cannot discover each other anymore). On the y-axis, we set the delay in seconds for both the search procedure. In addition, we present the mean values of each box in the header of the graph and show them as a green triangle in each set. Finally, we take in consideration the Samsung S20 as a Discovery device whose name is shown at the top left of the graph, while the scenario studied is specified at the top right.

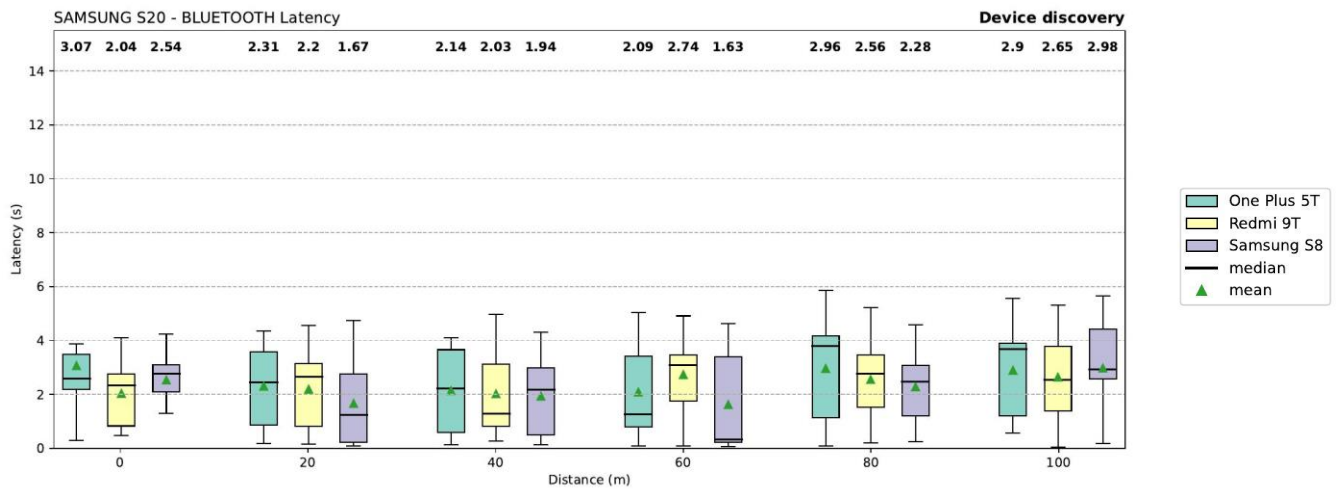


Figure 3-6 Bluetooth Classic latency using the Samsung S20 as a discovery device.

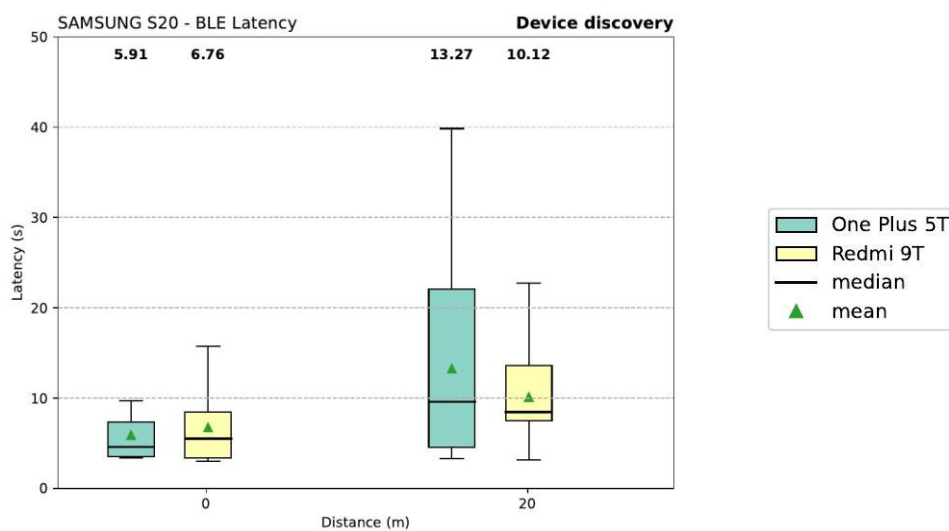


Figure 3-7 BLE discovery latency using the Samsung S20 as a discovery device.

From the results obtained with the Bluetooth Classic, we can observe that 75% of the device discovery latency taken by the Samsung S20 is below 4 seconds. Furthermore, its average values range from 1.67 s to 3.07 s.

With BLE, on the other hand, we can observe that at 0 m, the search for an Advertiser is 2.4 times slower than with Bluetooth Classic, while at 20 m, it is 7.2 times slower. Additionally, we can see that with BLE, the values are more spread out as the devices get farther away from each other.

In synthesis, if we compare the results obtained during this experimentation, we can state that a Discoverer device takes approximately 2.37 s to find an Advertiser device using Bluetooth Classic and 9.02 s with BLE.

In terms of performance, we can observe that Bluetooth Classic performs better than BLE, as it can perform discoveries within 100 m and is 3.8 times faster than BLE.

Finally, we can observe that as devices move away from each other, the average latency with Bluetooth Classic remains constant.

3.4 Merging elements: 5G core and D2D extension

As explained in the previous chapter, there is a significant delay in switching from one relay to the other. This can cause serious problems in time-critical applications and user discomfort in non-time-critical. In order to decrease the switching latency, another approach is to implement a 5GLAN. The 5GLAN will consist of a LAN network where all connected devices to the same network use the same 5G Core and slice. As a result, the devices are visible to each other. The concept here is that each device and relay connected to the 5G network will also be registered in a 5GLAN created by the 5G Core. Several 5GLANs can be implemented either for different slices or based on the properties of each registered device, e.g., smartphones with relay capabilities, IoT sensors, or autonomous robots. In the scenario that an out-of-range device connects to a relay and is eventually registered to the same 5GLAN with all the available relay nodes, it will be able to see which relays are available on that LAN and choose the best relay. This approach increases speed and reliability. Furthermore, if the out-of-range device moves away from the relay that is currently connected but gets closer to another relay, the handover between the relays will be much faster since they will both belong to the same LAN, and authorization will be faster. An example of the described scenario is shown in Figure 3-8.

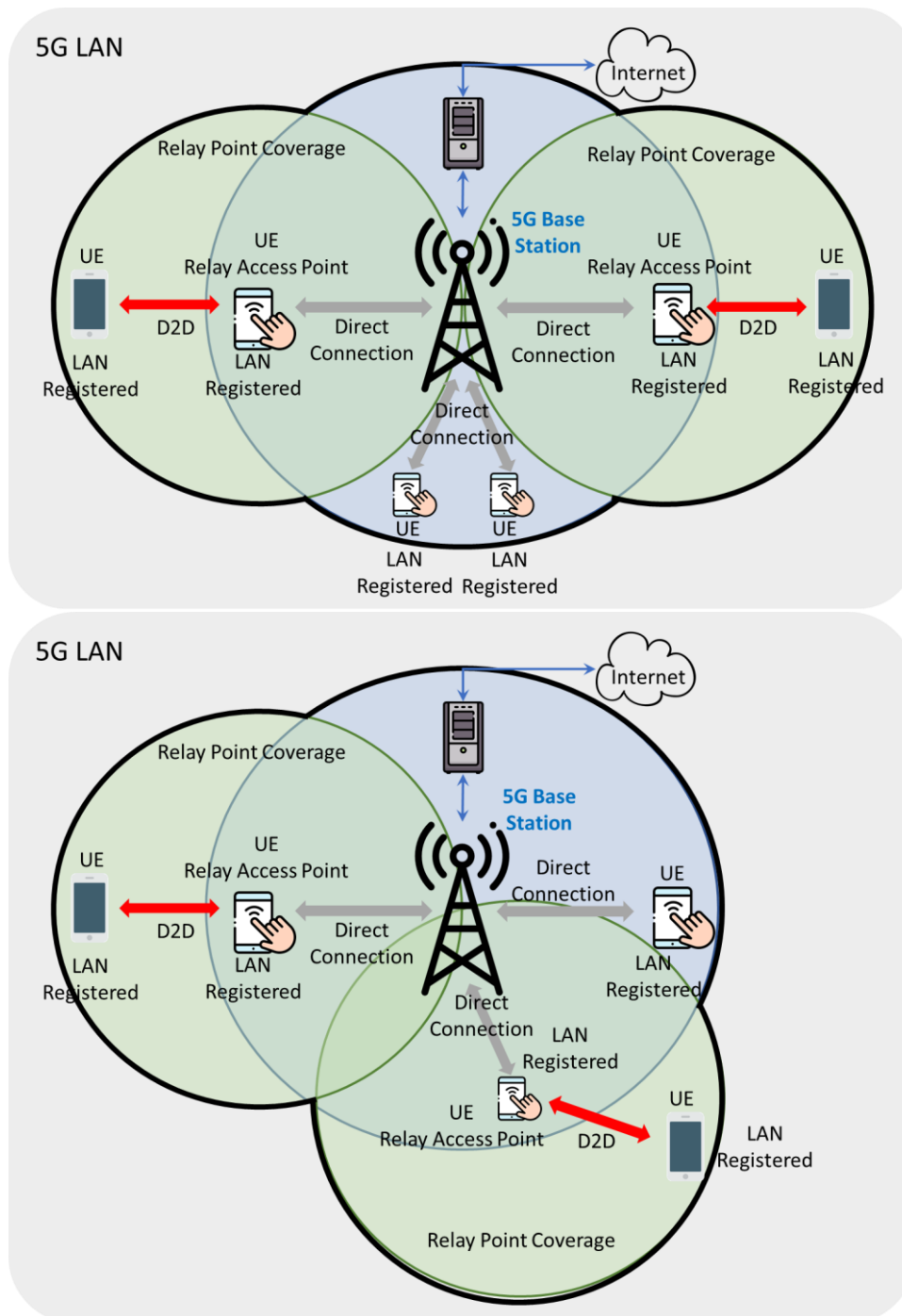


Figure 3-8 Scenario on the use of 5GLAN starting from position 1 (up) moving towards position 2 (down)

For this to be achieved, the 5G Core should be capable of providing this LAN creation and device registration service for all of the connected devices. Hence, a new 5G Core with this feature has been developed in Task 2.2. This is a novel approach to address the relay switching latencies due to relay discovery and authorisation.

At this stage, experiments have not yet been conducted, but the plan is to run tests both without the 5GLAN, i.e., only measuring throughput, latencies, and jitter but also using the 5GLAN and measuring latencies in connection establishment, jitter, and throughput between devices on the LAN as shown in Figure 3-9. The development of the 5G Core with LAN features is described in the next chapter.

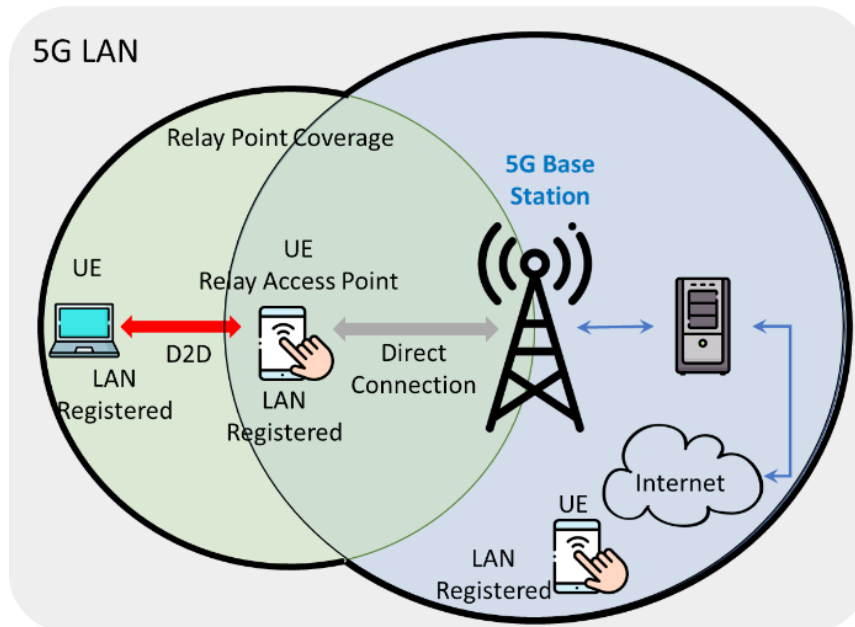


Figure 3-9 Architecture of experimental setups for future tests.

3.5 Next steps in this work

In the second reporting period, we will set up a 5G laboratory testbed and we will install a new 5G core. Using this testbed, further D2D configurations can be deployed and tested providing the basis for further development of our results.

3.6 Conclusions

In this chapter, we have presented the activities performed for the study of the latency characteristics of D2D connectivity using Bluetooth Classic and BLE. In addition, we have presented a physical architecture necessary for the D2D to work as an extension of a 5G core to satisfy the solutions required by the IoT-NGIN.

In terms of performing latency analysis of D2D connectivity, we have extended our D2D measurement tool, called AtomD, with the necessary functionalities to perform a set of automated iterations to perform the D2D connectivity process. In addition, we have added functionality for tracking the time instants of the processes performed by a discoverer device to detect an advertiser device.

Our preliminary results indicate that the discovery process with Bluetooth Classic allows devices to be within 100 m at most. However, with BLE, a radius of 20 m is not exceeded. Moreover, in terms of latency, we observed that with Bluetooth Classic, the devices are found more quickly, with an average delay of 2.37 s. Therefore, we can state that Bluetooth Classic meets the requirements of this project better than BLE.

As for the network design required for D2D to scale with 5G, we have provided a scheme that leads UE devices connected directly to the base station to act as relay access points, providing a bridge between the 5G core and the D2D link. Moreover, with the use of a new lab test-bed and installing a new 5G core, further configurations and results are expected shortly.

4 Enabling 5G to support protocols for deterministic communications

4.1 Introduction

Owners of Industry 4.0 use cases are becoming more and more interested in using private LTE and 5G networks to increase their flexibility to configure devices and communications without the need for cabling and its maintenance. The market for such private mobile networks is growing rapidly.

As the use of private mobile networks in an Industry 4.0 setting picks up pace, numerous new use cases which require close to deterministic wireless communications are emerging. In this context, mobile system developers are investigating techniques based on time sensitive networking as a solution to this challenge.

In this chapter, we describe our work on the concepts and prototypes under development in IoT-NGIN addressing time sensitive communications in mobile networks through integrating implementations of an existing Ethernet time sensitive standard into 5G networks. The work has focused on architectures and prototype implementations of this functionality, which has already been defined in 3GPP standards for 5G networks. Investigations of the use of this new prototype functionality in the IoT-NGIN project use cases on powertrains (use case 8 with ABB) and on robots including both stationary robots and moving robots, such as Autonomous Guided Vehicles (AGVs) in use case 6 together with BOSCH are progressing. These ongoing investigations have focused on both the theoretical use of the functionality and the potential deployment of the functionality in a live LTE or 5G network at trial sites.

4.2 TSN communications in the IoT-NGIN context

Industry 4.0 is setting new requirements for 5G networks to deliver Time Sensitive Network (TSN) communications. Current industrial IoT and machine process control is based on wired automation where data of devices and sensors is collected to the edge servers and a Data Base (DB). In a basic setup, devices (Powertrains, robots, etc) and sensors are connected to an edge server via private 5G networks. To add new devices into the process and support their mobility through 5G networks the current transport and protocols used over the wired connections need to be maintained. Thus, existing industrial protocols e.g., OPC-UA [1], Profinet [2], etc. needs to be used both over the wired and wireless connections as shown in Figure 4-1.

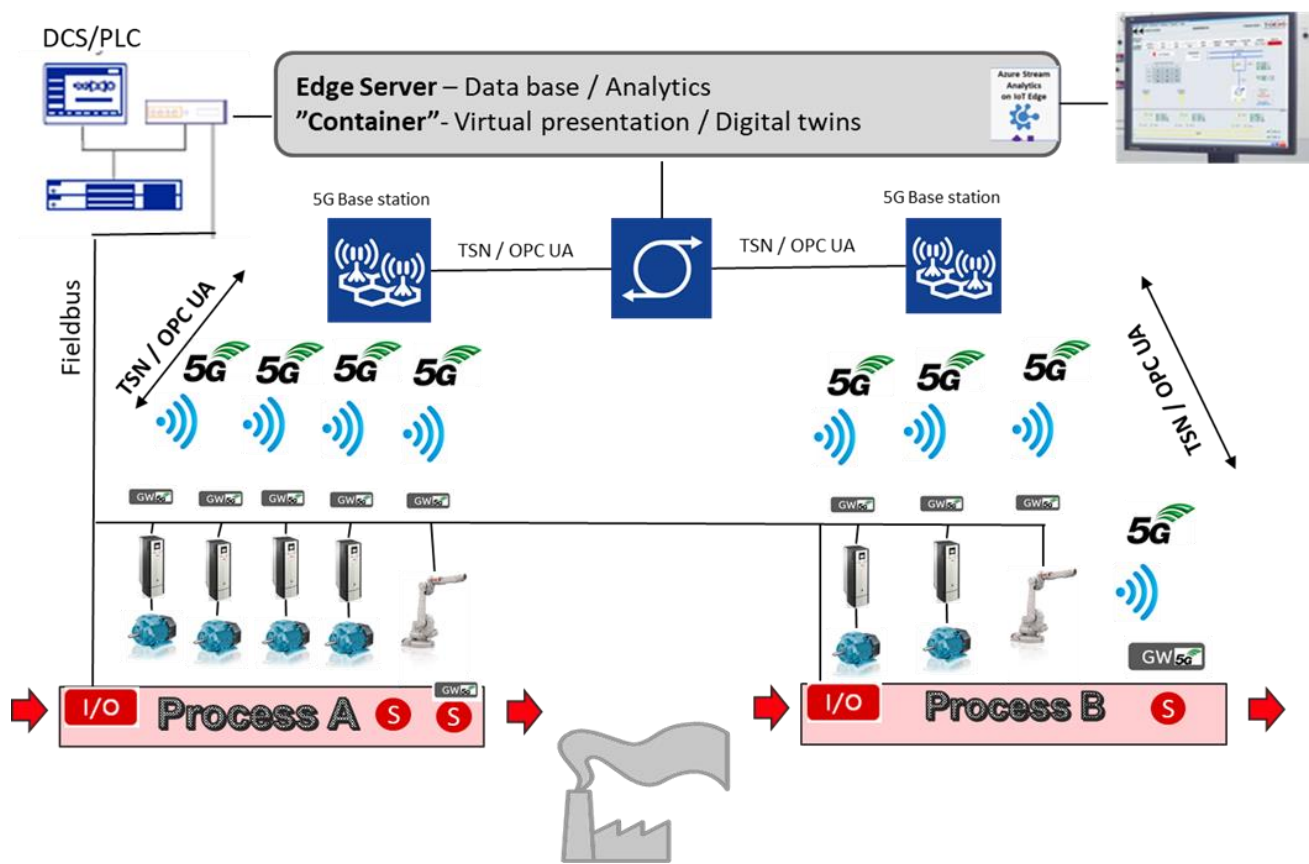


Figure 4-1 ABB TSN network topology

TSN has been used in fixed network to provide accurate and high reliable networks used in industrial environment for controlling high precision timely operated machinery.

The objective of the first prototype of the TSN developed is to synchronize wireless devices with the fixed reference master clock to support deterministic communication between wireless and fixed industry devices. A prototype was developed and tested in the lab. Initial test results on synchronization of the prototype on synchronization show a synchronization accuracy of 30-80 μ s. This accuracy is not sufficient. The specifications for TSN communications (standard IEEE 802.1AS) state that synchronization accuracies should be better than 900 ns.

In the current prototype, we have developed two new functional units for the time synchronization towards fixed devices (Device-Side TSN Translator (DS-TT)) and towards a fixed network deployed in a factory (Network-Side Translator (NW-TT)). DS-TT is deployed in the User Equipment (UE). NW-TT unit is deployed in the User Plane Function (UPF) of the 5G network. The prototype is standardized following specifications in 3GPP Rel-16, specification 23.501, clause 5.27. The prototype is relevant for smart industries and smart agriculture.

The unbundling and extraction of the prototype can be achieved with very little effort. The prototype could be deployed and tested on the standard 5G network in any industry field trial if the 5G frequencies are available. Discussions are ongoing with ABB who manage an IoT-NGIN industrial Living Lab and with the mobile operator in Helsinki (DNA) to provide the 5G frequencies for a live 5G test in the ABB factory.

Before deploying the TSN modules in the 5G network, the 5GLAN functionality has to be implemented for creating a Private Virtual Network (PVN) that will connect only devices which are part of the TSN network. The 5GLAN functionality is used for connecting fixed devices together with the 5G devices that will be part of TSN group for exchanging time synchronization. 5GLAN is defined in 3GPP technical specifications 23.501 and a user interface for creating the 5GLAN group has been developed. The 5GLAN API prototype is being evaluated in the lab and the possibility to test it in an ABB lab together with the TSN modules NW-TT and DS-TT, is being discussed.

Therefore, 5GLAN is an enabler providing PVN functionality on top of which Time Sensitive Network (TSN) components can be deployed.

4.3 IoT MCM communication optimization

In the project, we have implemented the API available in CMC Graphical Interface (GUI) that allows to create the 5GLAN group. This API allows external Application Function (AF) to create 5GLAN groups. We have also implemented the AF as Graphical User Interface (GUI). This implementation is based on a 5G core which includes the 3GPP defined network functions required for 5GLAN to connect mobile devices with fixed LAN and TSN network functions.

Using network slicing, the traffic can be isolated according to the slice service type (sst). Thus, sst=1 corresponds to broadband traffic while sst=2 is IoT traffic. Moreover, using the same service type, multiple slices can be created each having different requirements. Using 5GLAN, a virtual network connection between a fixed LAN and a group of 5G devices can be created within an existing slice.

The 5GLAN feature enables the integration of mobile networks as a part of existing IT infrastructure. Connectivity based on 5GLAN reduces the use of Ethernet cables and provides similar connectivity to fixed industrial devices.

For traditional Ethernet communication, a device needs to determine the MAC address of its peer device. The device, according to the destination IP address XXX derived from the IP packet that needs to be delivered, would initiate an enquiry “who has the IP address XXX”, and this enquiry is broadcasted to all the devices belonging to the same LAN. The device who has this IP address will respond and provide its MAC address to the requesting device.

In the case of 5GLAN, it is essential to allow a UE to obtain the identifiers of other UEs in the same private 5GLAN-type service. In LAN networks, devices make use of discovery mechanism (e.g., Bonjour, UPnP) to discover other devices and their characteristics online. This discovery mechanism makes use of the multicast capabilities of the network. Therefore, it is important that 5GLAN supports the same discovery mechanisms.

The 5G network shall support the routing of non-IP messages (e.g., Ethernet packets) efficiently for private communication between UEs and the TSN Controller. Moreover, the 5G network should also support on-demand establishment of a multicast communications within a subset of UEs that are members of the 5G Private Virtual Network (PVN) for devices and service discovery using protocols such as UPnP, Bonjour or LLDP.

The following text describes the design of the 5GLAN functionality and the prototype implementation. The 5GLAN functionality includes the API for creating 5GLAN groups. The API can be used by the mobile operator to dynamically create the 5GLAN groups or the API is available through a Network Exposure Function (NEF) that allows external applications to

create the 5GLAN groups. To create the 5GLAN groups through the NEF, the 5G devices have to expose non-public network identities (NPN-ID) to avoid any security or privacy issues when using 5G internal identities. The assignment of this public identities i.e., NPN-ID can be done in two ways, either locally or universally managed.

Locally managed NPN-IDs are assumed to be chosen randomly at deployment time to avoid collisions (and may therefore not be unique in all scenarios). Universally managed NPN-ID are managed by a central entity and are therefore assumed to be unique. The ultimate goal of these IDs is to be used by external applications to create the 5GLAN groups without disclosing 5G internal identities.

The Figure 4-2 below shows the modules implemented as part of the 5GLAN prototype following the 3GPP specifications. Those modules are part of the standard 5G network. The first module is the 5GLAN Application Function (5GLAN AF) as defined in 3GPP that implements the 5GLAN API developed by CMC. The 5GLAN API provides the GUI to the user to create the group of devices that will be part of the 5GLAN group. The 5GLAN AF sends the information entered by the user to the NEF which is the interface with 5G internal functions. The NEF will store the 5GLAN group information into the User Data Repository (UDR) that is 3GPP defined module for storing user information. The NEF implements the Group Management Function (GMF) that interacts with the Session Management Function (SMF) to create the session for communication within the 5GLAN Private Virtual Network (PVN) group.

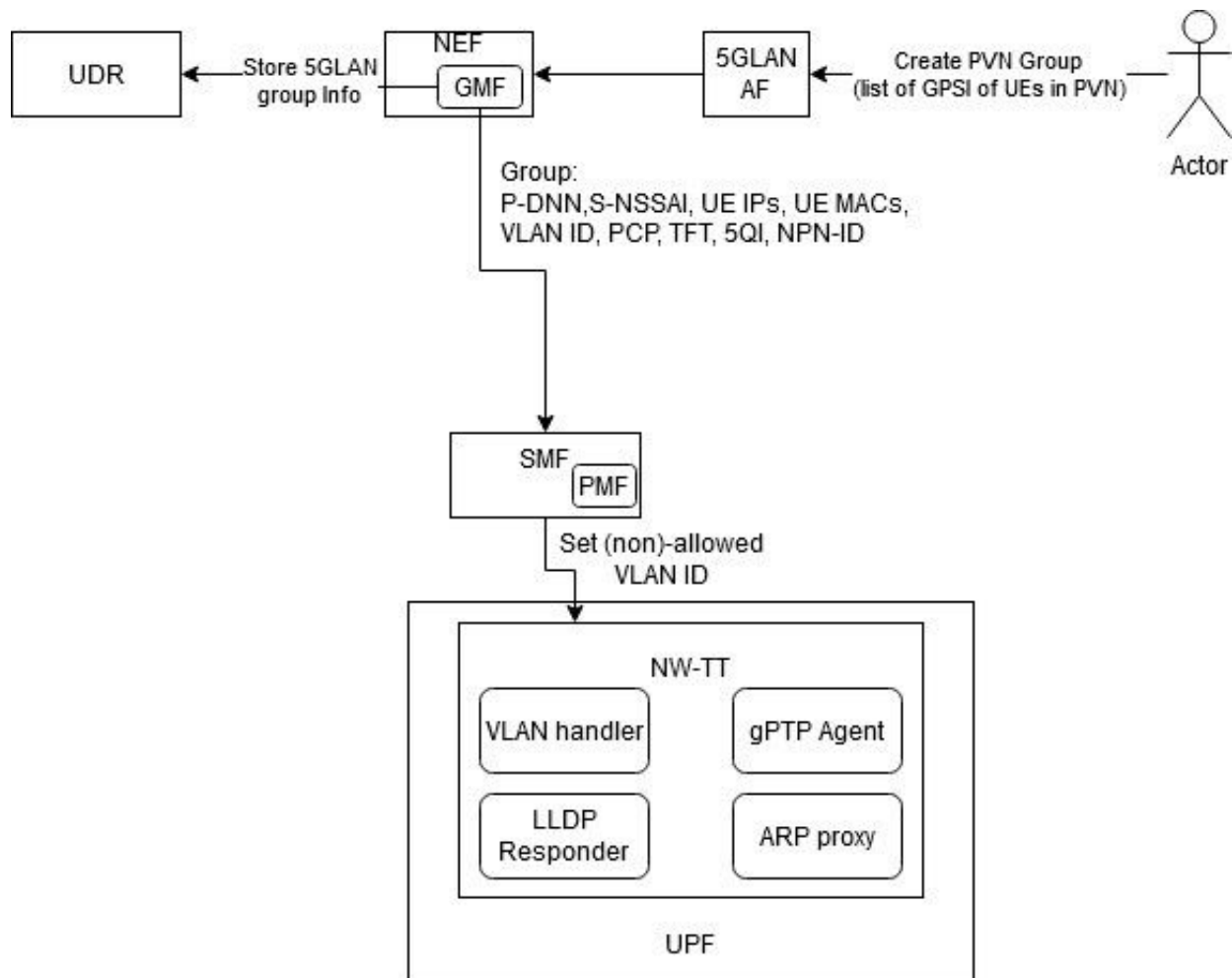


Figure 4-2 5GLAN architecture and network functions

4.4 TSN based IoT communication

The TSN architecture is defined in 3GPP and the NW-TT, DS-TT modules and their functionality are introduced in the 3GPP specifications as shown in Figure 4-3 to integrate fixed TSN devices with 5G networks for enabling wired-wireless TSN communications.

TSN features enable synchronizing mobile network UEs. TSN can be used to synchronize mobile network with existing IT infrastructure.

The industrial LAN may also consist of TSN-enabled Ethernet bridges. The latest release of 5G specification supports the fully centralized TSN configuration model, where a central controller should be able to configure both Ethernet and 5G System (5GS) bridges as a unified network. The 5GS supports the whole industrial network, both Medium Access Control (MAC) learning and flooding-based forwarding as well as the static forwarding configured by the central controller need to be supported. 3GPP has defined that a 5GS can be modelled as one or more virtual TSN bridges.

The Centralised Network Configuration (CNC) is the entity defined in the IEEE TSN specifications that has complete knowledge of the network topology and is responsible for configuring the bridges to enable transmission of TSN streams from source to destination. The 5G control plane is interacting with CNC via the TSN Application Function (AF) which maps

between the TSN parameters and the 5GS parameters. The TSN AF reports the 5GS bridge capabilities such as minimum and maximum delays between every port pair and per traffic class, including the residence time within the UE and DS-TT via TSN-AF to CNC.

Topology discovery information based on the widely adopted standard IEEE 802.1AB Link Layer Discovery Protocol (LLDP) is also exposed. The TSN AF also exposes its TSN capabilities like the support for scheduled traffic and per-stream filtering and policing (PSFP) as specified in IEEE 802.1Q-2018, in case that they are supported by all of the ports. The CNC obtains the 5GS bridge VLAN configuration from TSN AF according to IEEE Std 802.1Q

The TSN AF shall be pre-configured (e.g., via OAM) with a mapping table. The mapping table contains TSN traffic classes, pre-configured bridge delays (i.e., the preconfigured delay between UE and UPF/NW-TT) and priority levels. The CNC reads the capabilities of all bridges and calculates the traffic paths and schedules in the network. The CNC then provides the bridge configuration to the 5GS through the TSN AF, which contains, e.g., scheduled traffic, PSFP, and traffic forwarding information. In order to support QoS for Ethernet and TSN traffic, the traffic flows are mapped to 5G QoS flows. The CNC configures the traffic handling in the 5GS bridge for the different traffic classes according to the capabilities that have previously been reported by the 5GS bridge. The 5GS maps the Ethernet/TSN traffic classes or TSN traffic streams to the corresponding 5G QoS flows.

3GPP has defined 5G Virtual Network (VN) groups consisting of a set of UEs using private communication for 5GLAN type services. A 5G VN group can be utilized for IP or Ethernet based services. A specific data network, identified by a Data Network Name (DNN), is one of the possibilities to realize a 5G VN group, where the VN group can be either provided by Operation and Management (O&M) or by an TSN-AF.

For a centrally managed Ethernet network, it is required that the NMS/CNC can configure the VLAN handling for all bridges and all ports, including the 5GS bridge, as specified in IEEE 802.1Q.

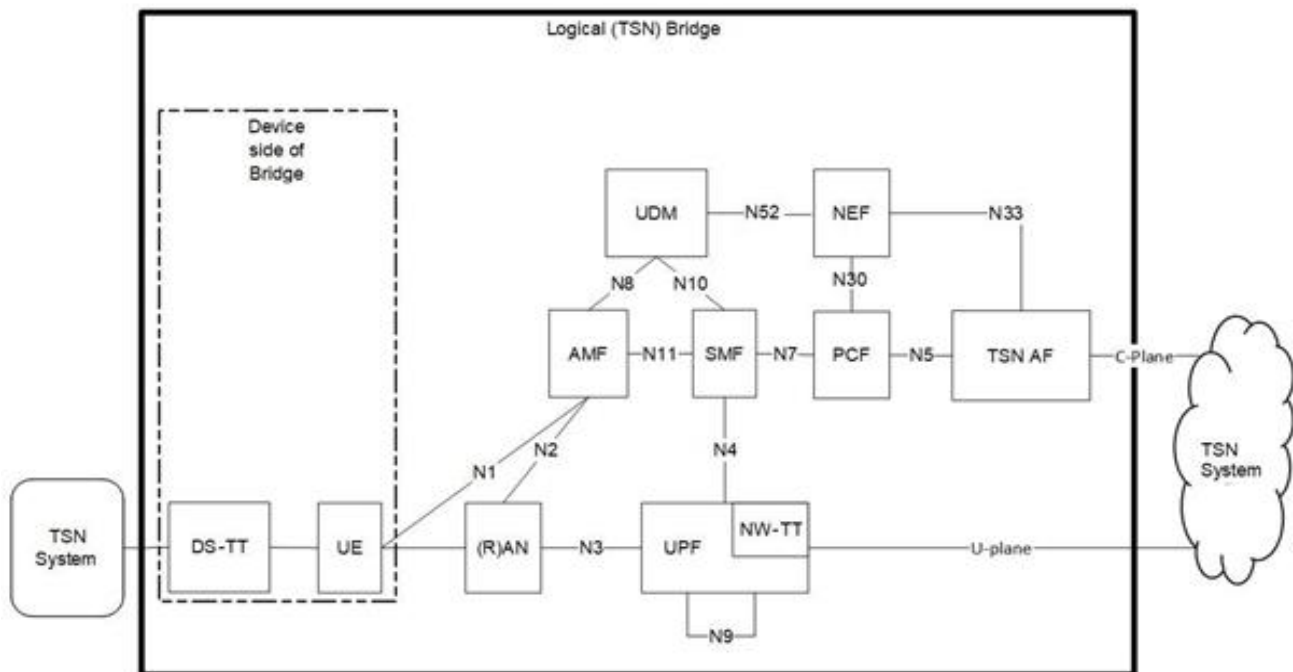


Figure 4-3 3GPP TSN architecture

Furthermore, a UE is defined to have attached functions of time stamping in data frames, using the device side of the TSN translator (DS-TT). On this Network Function, a step of time synchronization is implemented using the Tsi from the Suffix field of the gTP messages (Sync or Follow Up messages), as it has been defined on 24.535 from 3GPP.

In order to achieve this function a 5G-Modem is integrated in a NPN following an inherent structure based on a 5G component, a TSN packet handler and wired network components. These last two components may be integrated in the same hardware such as a microprocessor but following the TSN standards defined on IEEE 802.1 Qbv, 802.1 CB, 802.1 As and 802.1 Qbu.

Following the structure of the last figure, it is possible to get a UE that may perform as a bridge between a 5G NPN and an industrial wired network in addition to implementing TSN.

As it points out, TSN main characteristics are performed: Seamless Redundancy that permits ultra-reliability due to the duplicity of buffers, Frame Preemption that includes a priority of the information that produces, and time critical communication. In addition to the Time Aware Shaper that allows scheduling of the outputs and Time Synchronization.

The development builds on the excellent security features already standardised in 5G and reference to Annex 6 of this document.

4.5 Results

The expected result of this work is to improve the level of accuracy of synchronisation between clocks in fixed network and UEs. This preciseness is not possible in the current implementation based on current standards.

In the laboratory test results, CMC showed the first results of time synchronization in TSN lab setup shown in Figure 4-4 below were completed with following results. The current level of

accuracy achieved is about 20 μ s but it does not meet the demands of TSN networks of an accuracy of <900 ns but CMC is still working on the optimization to reach better performance.

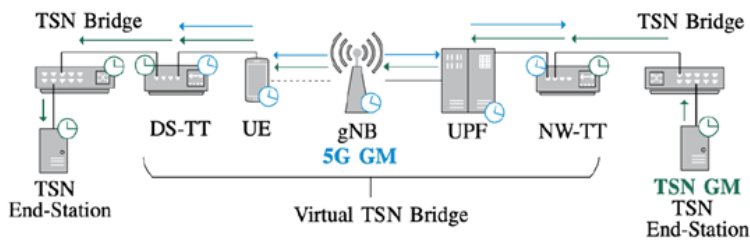


Figure 4-4 Lab measurements TSN synchronization

4.6 Next steps in this work

The next steps would be mainly to improve the accuracy of the time synchronization modules and proceed with the implementation of the TSN AF. In order to improve accuracy, the software has to be improved to manage the time synchronization information rapidly and check access to radio modem to obtain reference time from the base station

4.7 Conclusions

Future industry 4.0 Process and application control could be based on edge cloud and Edge controllers, which are connected to devices such as frequency converters and motors (Variable speed drives) using private 5G networks. Combining control, data collection, data storage and analytics functionality will improve the flexibility to change configurations and enable new features and benefits in an industrial control. These benefits include cost savings, optimization and modular automation. The deployment of 5G network functions required to support the connectivity of industrial wireless with fixed devices is under development and preliminary tests have already been performed. However, some limitations have been identified which will be addressed through optimizing implementations of network functions and further validation of the performance of the optimised implementations.

5 Enhancing 5G ease of use through improved 5G APIs

5.1 Introduction

Modern 5G has enabled services which can play an important role in future smart industry, smart agriculture, smart cities and smart energy deployments. To exploit the full potential of these services and functionalities, interfaces between the IoT and 5G infrastructure are required. These Interfaces are defined by APIs and are exposed to the IoT applications and customer.

There are already a variety of APIs integrated into 5G products and services available on the market. A key challenge associated with these APIs is that they can be hard to understand for developers not familiar with 5G infrastructures and the mobile network domain.

To overcome this difficulty, a generic 5G resource management API is developed in the scope of this project. This API will provide a more generic interface and simplify the usage of 5G resources in IoT applications. The specification of the IoT-NGIN 5G resource management API is an ambitious undertaking. The IoT-NGIN 5G resource management API is expected to reduce the time and cost of implementing new services and IoT applications using 5G communications by reducing the need for the involvement of 5G experts.

In this chapter, we provide an overview of the 5G capabilities exposure APIs considered in the IoT-NGIN project and the IoT-NGIN 5G resource management API in terms of their features, specifications, sequence diagrams and implementations. Initial feedback has been requested and received from the Living Labs of IoT-NGIN on the functionality under investigation for the IoT-NGIN 5G resource management API. These interactions with the Living Labs are continuing in the second reporting period.

The results of the work on the IoT-NGIN 5G resource management API will be published as OpenAPI specifications in GitLab. An open-source implementation of the IoT-NGIN 5G resource management API is under development and will be made publicly available on the project open-source repository in GitLab. The relationship of the IoT-NGIN 5G resource management API to existing standards and the requirements it addresses have been described in chapter 4 of D2.1 [3].

Investigations are on-going to prepare the standardisation of the 5G device management API and the 5GLAN API. These APIs belong to the set of new 5G capabilities exposure APIs in the IoT-NGIN project. The IoT-NGIN 5G resource management API will be promoted at the EuCNC & 6G Summit taking place in Grenoble on 7-10 June.

In this chapter, we focus on describing examples of features, specifications, sequence diagrams and implementations of the IoT-NGIN 5G resource management API. In Annexes to this document, we include further detailed information on aspects of features, specifications, sequence diagrams and implementations which were too long to include directly in the text of the chapter.

5.2 IoT-NGIN 5G resource management API capabilities and architecture

Based on research and exchanges between IoT-NGIN project partners active in the 5G domain, and different use case owners represented in the project Living Labs, three different groups of capabilities were identified as focus points for project work. There are the areas of:

- 5G Connectivity and Device Management,
- Network Slice Management, and
- Microservice Lifecycle Management.

In Figure 5-1 below, the overall structure of the proposed IoT-NGIN 5G resource management API is depicted. In the **top layer** of the diagram, the customers application running on devices in the field as well as on virtualized infrastructure in the 5G network is depicted. These applications interface, via the unified and simplified IoT-NGIN 5G resource management API, with the network infrastructure. It should be noted here that this API is used solely for parameterizing and controlling the infrastructure but not for data transmission between devices, microservices nor with the Internet.

In the **middle layer** of Figure 5-1, the different modules are depicted as listed above (5G Connectivity and Device Management, Network Slice Management, and Microservice Lifecycle Management). The modules ensure the translation from generic operations to the standardised and vendor specific APIs, which is achieved using adapters.

The **lower layer** of Figure 5-1 graphically depicts the 5G core and edge cloud infrastructure which the IoT-NGIN 5G resource management API can interface with. The IoT-NGIN 5G resource management API is planned to be interfaced with 5G capabilities exposure APIs. The following are examples of the interfacing work ongoing in IoT-NGIN: the Virtualized Infrastructure Managers (VIMs) including OpenStack, Kubernetes as well as a variety of available 3GPP open APIs, are all considered in current implementation work. Currently, two of the IoT-NGIN 5G infrastructure vendors are investigating the standardisation of the 5G device management API and the 5G TSN interface in 3GPP.

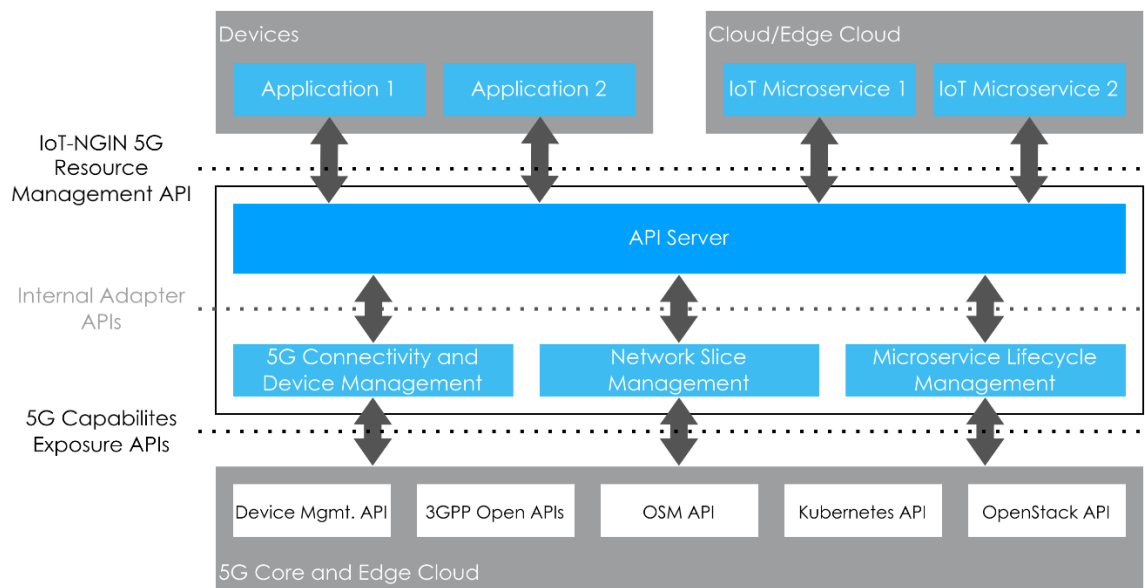


Figure 5-1 IoT-NGIN 5G resource management API

The following sub-chapters provide more details about the different elements depicted in Figure 5-1.

5.3 The 5G capabilities exposure APIs

This sub-chapter describes 5G capabilities exposure APIs used by the IoT-NGIN 5G resource management API. The descriptions start with an overview of the different 5G capabilities exposure APIs provided in sub-chapter 5.3. Furthermore, the description of specific APIs and available functionalities is provided. In sub-chapter 5.4, a more detailed explanation of the IoT-NGIN 5G resource management API is provided. This includes first revisions of sequence diagrams and API operations as well as an overview of a project internal living lab survey, which assists in identifying the most important API functionalities.

This work builds on our earlier work on exposure categories described in D2.1 [3].

5.3.1 Features of the 5G capabilities exposure APIs

In this sub-chapter, we describe the functional requirements of the IoT-NGIN 5G resource management API mapped onto three exposure categories defined in D2.1 [3]. We describe the features of these exposure categories in detail and explain their benefit for IoT-NGIN applications and use cases. We refer to the collection of three exposure categories as 5G capabilities exposure APIs.

The requirements are not only defined based on the Task 2.3 description, but also based on questionnaires prepared for LL owners and their use cases, various internal meetings and discussions with LL owners and other project partners.

Features of the 5GLAN API capability

The 5GLAN API allows the creation of device groups, to be connected to a fixed LAN network.

To start the process of creation of device groups, the 5GLAN administrator goes to the operator's portal and makes a request for 5GLAN-type service. The request includes the GPSI (General Public Subscription Identifier) or SUPI (Subscriber Permanent Identity) of all UEs that plan to use this 5GLAN-type service for private communications, and it also includes the type of communication (IP or Ethernet). In addition, the 5GLAN administrator may indicate any of the following additional information:

- Requested QoS,
- IPv4 or IPv6 communication,
- Static or dynamic IP address,
- Additional IP services (e.g., DNS, Dynamic DNS, DHCP, IMS, egress to Internet), and
- Additional Ethernet services (e.g., multiple IEEE 802.1Q VLANs).

A Private DNN uniquely identifies a 5GLAN group and all the member UEs of the same group need to establish a PDU Session towards the same Private DNN for 5GLAN group communication.

Features of the network slice management API capability

The network slice management API features, relevant for IoT-NGIN applications, are defined in this chapter. This network slice management API provides various features in three different scopes:

1. Management of infrastructure resources from different domains such as registration of cloud/edge and RAN resources,
2. Slice management through which reserved resources per slice are registered and configured. These reserved resources are composed of a) cloud/edge compute chunks, b) RAN chunks, and c) 5GCN/Cloud/edge network chunks, and
3. Network services management, which not only includes service instantiation, termination and migration, but also service update, and recovery.

Features of the 5G device management API capability

The features of the 5G device management API capability relevant for the IoT-NGIN applications are defined and a summary is given below in Table 5-1 [4]. The 5G device management API and its services are being continuously defined and developed by the 3GPP SA6 group under the study called Service Enabler Architecture Layer for Verticals (SEAL) [5]. In addition, the detailed descriptions of these API services and the potential uses of these services, especially in the smart agriculture use cases, are described and given in Annex 1.2. The exemplary use cases defined for the smart agriculture domain can be used as basis for defining and supporting several other use cases of the vertical sectors studied in IoT-NGIN.

Table 5-1 Description of the 5G device management API features

5G device management API feature	Description of the feature	5G API service according to 5G-ACIA
Provisioning and onboarding devices	This feature allows users to provide the unique identifiers of their 5G devices to the 5G network, so that these devices are accepted by and onboarded to the 5G network.	Device Provisioning and Onboarding

Getting a list of devices	This feature allows users to get a list of devices that are provisioned to the 5G network. A list of provisioned devices is requested from the Device Provisioning and Onboarding Service, whereas a list of connected devices in a group is requested from Device Group Management Service.	Device Provisioning and Onboarding & Device Group Management
Creating, modifying and removing device groups	This feature allows users to create many device groups with same or different purposes, group some devices in the same group and remove a device from a group.	Device Group Management
Monitoring the quality of the communication links of devices	This feature allows users to monitor the quality of the communication links (e.g., signal strength, packet loss, latency) of their 5G devices.	Device Connectivity Monitoring
Defining and changing QoS parameters of individual device connections	This feature allows users to define and change the values of Quality-of-Service (QoS) parameters for communication links of devices.	Device Connectivity Management
Getting location information of devices	This feature allows users to get the geographical location of their connected devices to track their mobility.	Device Location Information

5.3.2 OpenAPI specifications of the 5G capabilities exposure APIs

In this sub-chapter, we provide some examples of the API specifications of the 5G capabilities exposure API features that are described in Chapter 5.3.1.

Examples of the 5GLAN API

POST {gui-cnc}

Creates 5GLAN group configuration. It stores the configuration in Mongo DB. This operation allows the GUI to send a request to create a 5GLAN group. The request is sent to the module that provides service endpoint for storing into DB the 5GLAN group metadata.

GET {gui-cnc}

This operation is used to request the 5GLAN group metadata. This transaction retrieves 5GLAN group configuration which includes the groupid, sharedDataId, members etc.

PUT {gui-cnc/extGroupId}

This operation modifies group configuration, the input parameter consists of the 5GLAN group ID and the transaction will include the new group metadata to replace existing one.

DELETE {gui-cnc/extGroupId}

This operation removes group configuration from the system so the metadata associated to the group ID will be deleted from the database.

Examples of operations of the network slice management API

i2CAT's tool, Slicing & Orchestration Engine (SOE), offers a set of APIs, which are not only able to register and manage the edge resources, but also perform the tasks related to service instantiation and management, as described in D2.1 [3]. In the present chapter, the southbound API of the SOE is elaborated.

This set of APIs enables the communication between the resource managers registered and managed by the Slice Manager (SM) of the SOE. This interface consists of data related to the validation, reservation, and configuration of the radio, cloud, and network resources allocated to a specific slice. Examples of operations used as REST-based calls within SOE:

POST {RAN/Compute} Resources Validation

The purpose of this operation is to validate the integrity of the infrastructure in terms of the requested resources. In other words, the SM checks whether resources can be assigned to a certain slice through the RAN Controllers and VIMs. This operation returns a confirmation or an error message.

POST {RAN/Compute} Slice Chunk Creation

The operation allows reservation and registration of the specified RAN and compute resources needed by a slice if the previous verification is successful. A confirmation or error message is returned by the operation.

POST {RAN/Compute} Slice Chunk Activation

Once the RAN and compute resources are reserved and stored on the SM, this operation allows the radio components and cloud domains to be activated to begin the deployment of the requested network service by device. An informational message or an error message will be returned regarding the reserved radio/compute resources.

Examples of operations of the 5G device management API

In here, some examples of the OpenAPI specifications for the 5G device management API are given. The 5G device management API operations described here are studied and defined by the 3GPP SA6 group [5] [6]. We are in close contact with 3GPP SA6 standardisation activities and investigating the potential to contribute.

Provisioning and onboarding devices

This operation is not defined in standards. However, example operations are defined in Annex 2.2 according to the 5G-ACIA descriptions.

GET {/ss-gm/v1/group-documents/{groupDocId}} - Getting a list of connected devices in a group

This operation allows the forwarding of the requests to the appropriate 5G network functions related to getting a list of connected devices added as members to a group requested by the IoT-NGIN 5G resource management API Adapter.

POST {/ss-gm/v1/group-documents} - Creating device groups

This operation allows the forwarding of the requests to the appropriate 5G network functions and perform the creation of a group requested by the IoT-NGIN 5G resource management API Adapter.

POST {/ss_events/v1/subscriptions} - Monitoring the quality of the communication link of a device

This operation allows the forwarding of the requests to the appropriate 5G network functions and perform the subscription to the SEAL events to get the information on the quality of the communication link of a device requested by the IoT-NGIN 5G resource management API Adapter.

Changing a QoS parameter of an individual device connection

The path for this operation is not defined in standards, however the messages, parameters and example operations are defined in the standards and provided in Annex 2.2.

POST {/ss_events/v1/subscriptions} - Getting location information of a device

This operation allows the forwarding of the requests to the appropriate 5G network functions and perform the subscription to the SEAL events to get device location information requested by the IoT-NGIN 5G resource management API Adapter.

5.3.3 Sequence Diagrams of the 5G capabilities exposure APIs

In this sub-chapter we illustrate, using sequence diagrams, the features of the 5GLAN API, the network slice management API and of the 5G device management API. For simplicity, we provide some sequence diagrams as examples in this chapter.

As the complete sequence diagrams including the request and response messages of each 5G API feature are extensive, we included them in Annex 5.

Sequence diagrams for the 5GLAN API

The user can create a 5GLAN group after selecting the SIM information of the User Equipment to be part of the group. As depicted in Table 5-2, the 5GLAN API is implemented in the 5GLAN Application Function and will interact with the NEF to store the group information in the 5G UDR. The NEF will interact with the SMF to set the policies to make the UE discoverable to

other fixed devices through the UPF. After the group and policies have been set the UE part of the 5GLAN group would be able to communicate with fixed devices.

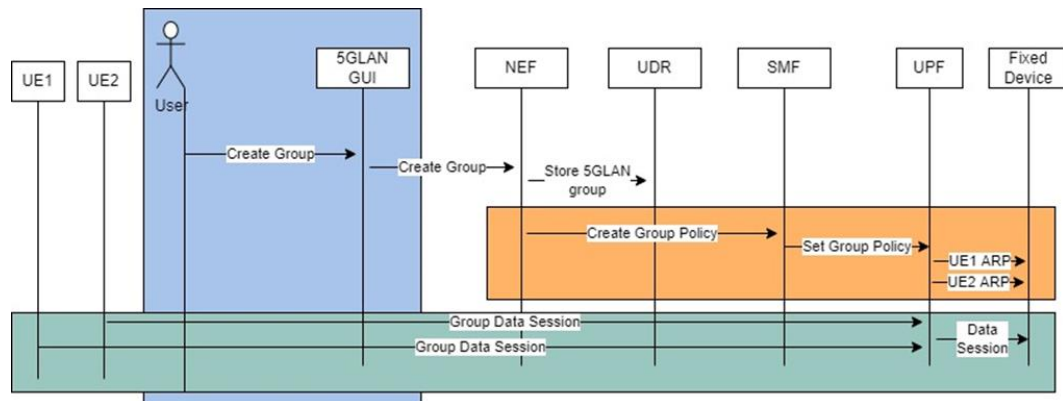


Figure 5-2 Sequence diagram for 5GLAN

Sequence diagrams for the network slice management API

This chapter describes the sequence diagram for the network slice management API.

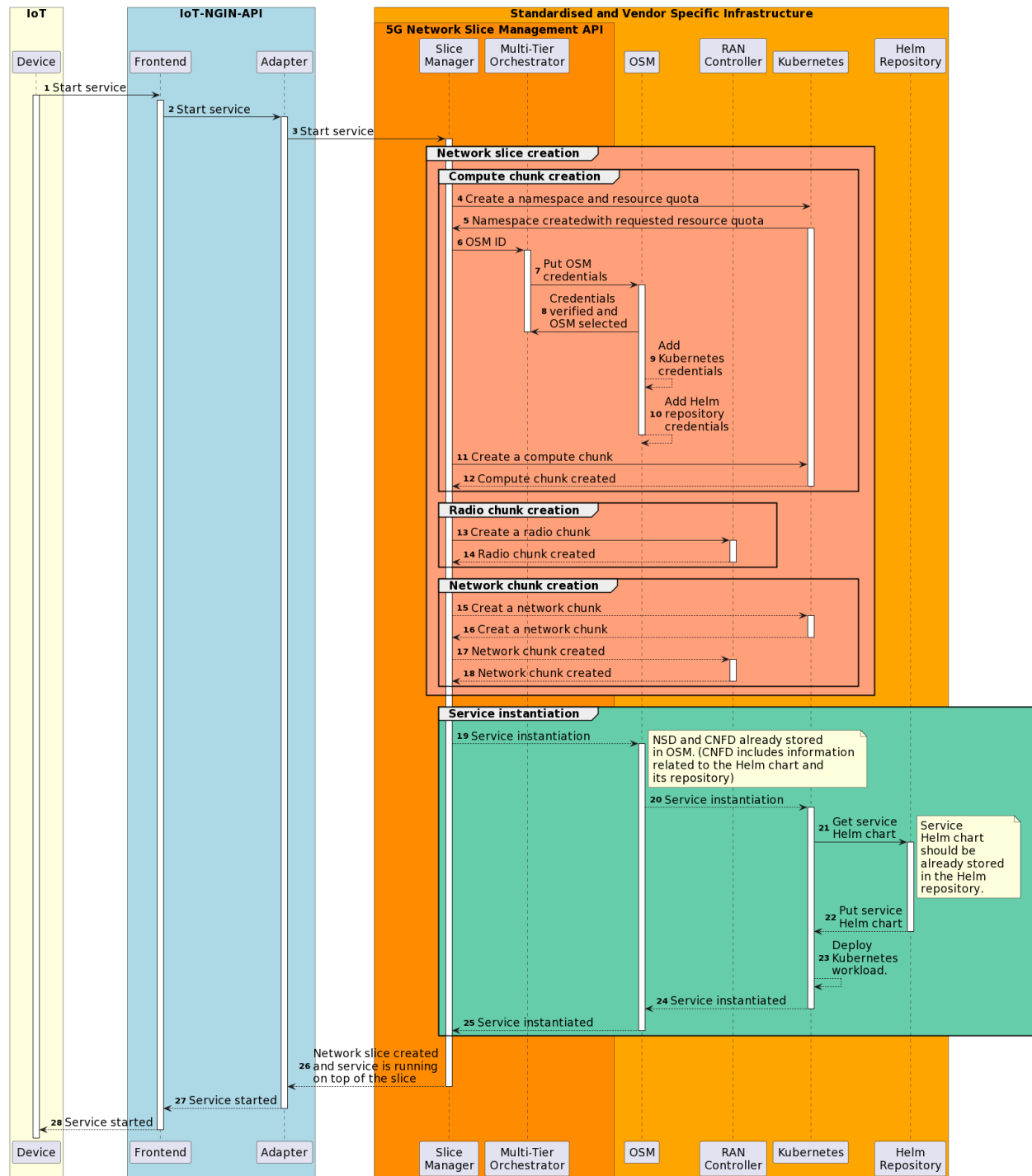


Figure 5-3 Workflow example of the service instantiation using i2CAT's tool, SOE, over edge domain

Figure 5-3 illustrates an example of the interactions conducted during the service instantiation at management and orchestration level. By receiving the start service request triggered by the IoT device, IoT-NGIN-API's frontend forwards this request to IoT-NGIN's adapter. Then the adapter communicates with SM through the SOE's NB-API (steps 1-3). The network slice management API, developed by i2CAT, is divided into two main parts: a) Network slice creation (steps 4-18) and b) Service instantiation (steps 19-25).

Once the SM receives the start service operation from the IoT-NGIN-API's adapter, it asks Kubernetes to create a new namespace with a specific resource quota for that namespace and Kubernetes sends an acknowledgment to the SM once the namespace is created with the specified resource quota (steps 4-5). Then, the SM selects the ETSI Open-Source MANO (OSM) as the NFV Orchestrator (NFVO), by communicating with the Multi-Tier Orchestrator (MTO) (steps 6-8). Next, OSM adds the credentials of Kubernetes and Helm repository (steps 9-10) and then asks Kubernetes to create the compute chunk of the network slice (step 11). Once the compute chunk formed of CPU, memory, and storage resources is created, Kubernetes sends the acknowledgement to the SM in this regard (step 12). Similarly, the radio chunk of the slice is created through the RAN controller (steps 13-14). The last chunk created is the network chunk, which mainly describes the VLAN configurations both in the compute and radio parts. The network chunk creation is triggered by the SM and handled by Kubernetes and the RAN controller (steps 15-18).

Prior to performing the service instantiation process, a) both the Network Service Descriptor (NSD) and Cloud-Native Network Function Descriptor (CNFD) should be stored in OSM, and b) OSM should create the Helm repository for the service, while CNFD contains the information related to this repository. The service instantiation process is triggered by OSM and then OSM sends the service instantiation request to Kubernetes (steps 19-20). Then, Kubernetes gets the application's Helm chart from the Helm repository and deploys the Kubernetes workload (steps 21-23) and then the corresponding acknowledgements are sent accordingly (steps 24-25). Once the network service is running, the IoT-NGIN-API's adapter is updated by the SM (step 26). In the end, the IoT-NGIN-API updates the IoT device through its frontend (steps 27-28).

Sequence diagrams for the 5G device management API

This chapter describes a sequence diagram for the 5G device management API, which is the complementary diagram provided in Chapter 5.4.3.

D2.2 – Enhancing IoT Underlying Technology

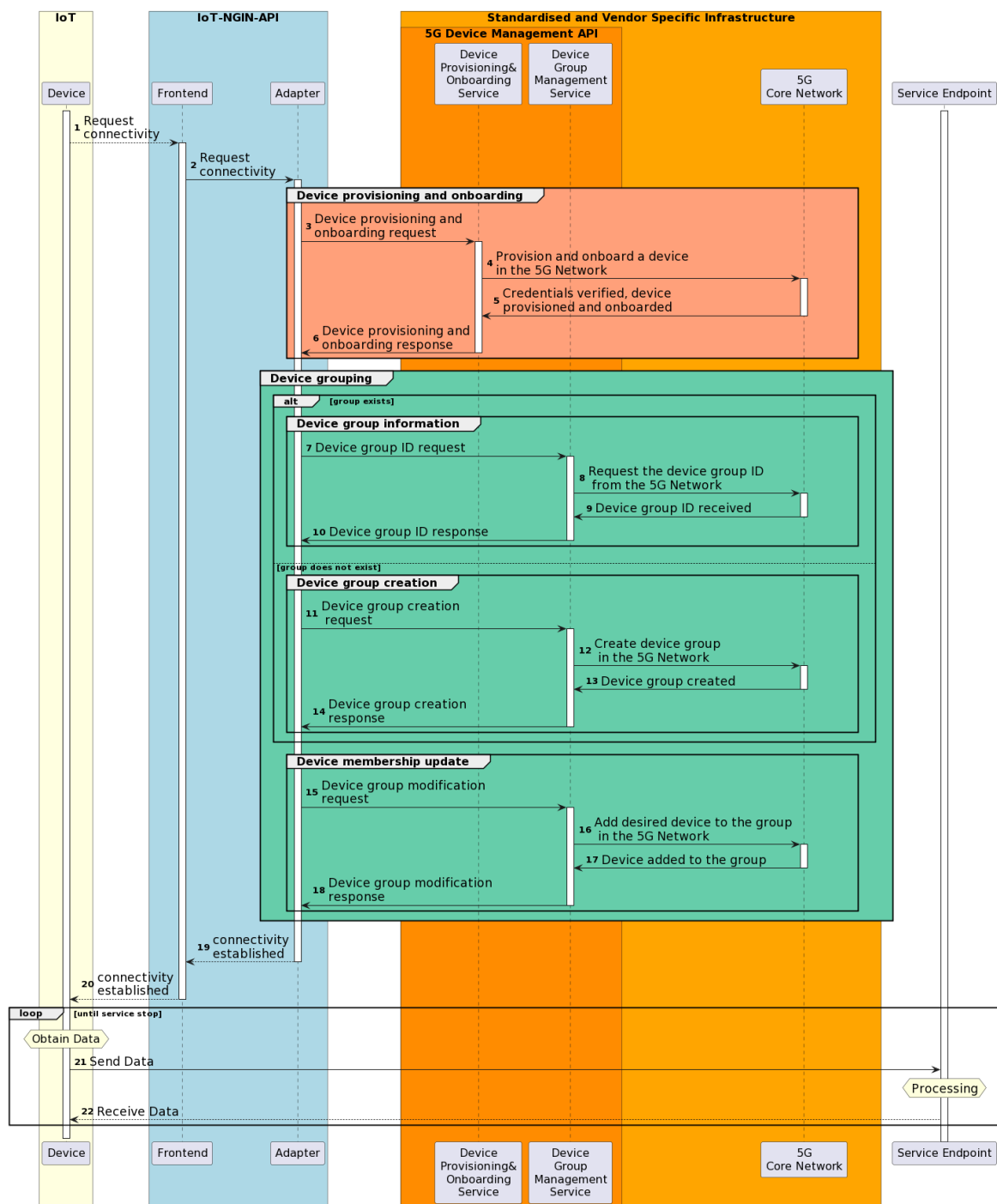


Figure 5-4 Workflow example of device management using the 5G device management API

Figure 5-4 provides an overall picture on how the IoT-NGIN 5G resource management API Implementation can communicate with an authorized IoT device, the 5G device management API and its services. Here, we can see that the IoT-NGIN-API's frontend is the one interacting with the device (steps 1-2, 19-20) and then, the IoT-NGIN-API's adapter translates and forwards the requests coming from the device to the 5G device management API for further processes (steps 3, 7, 11, 15). After the corresponding 5G device management API service receives the requests through the Internal API operations, it further translates and forwards the requests to the corresponding 5G core network functions to execute the desired operation (steps 4-5, 8-9, 12-13, 16-17). The diagram above shows an overall use case of

“Adding a device to a particular group as a member”. Although this use case may appear to be only one operation, it includes three different sub-operations to be executed by the 5G device management API.

In Figure 5-4, firstly, the IoT-NGIN-API's adapter asks the 5G device management API to provision and onboard the device to the 5G network (step 3). Secondly, in case the group with the requested name exists, the IoT-NGIN-API's adapter asks the 5G device management API to provide the group ID, so that it can add the device to that particular group in the next step (steps 7-10). In case the group does not exist, then first, the IoT-NGIN-API's adapter asks that a group be created with the requested name in the 5G network (steps 11-14). Thirdly, the device is added as a member to this group (steps 15-18). Finally, after the end-to-end connections are established via the 5G device management API (steps 1-20), the data transfer from the IoT device to the service endpoint and vice versa is performed (steps 21-22).

5.3.4 Implementation descriptions of the 5G capabilities exposure APIs

In this sub-chapter, we describe in detail how the 5G capabilities exposure APIs can be implemented in 5G systems, edge cloud or central cloud environments and can be accessed by authorized users or authorized applications.

The implementation of the 5GLAN API

The configuration of the 5GLAN group through the API is as described in the Figure 5-5 below. The CumuCore Network Controller (CNC) API includes the 5GLAN and TSN API that allows external Application Function (AF) to configure those services. In Figure 5-5 the external AF consists of a Graphical user interface (GUI) accessing the 5GLAN API provided by the CNC.

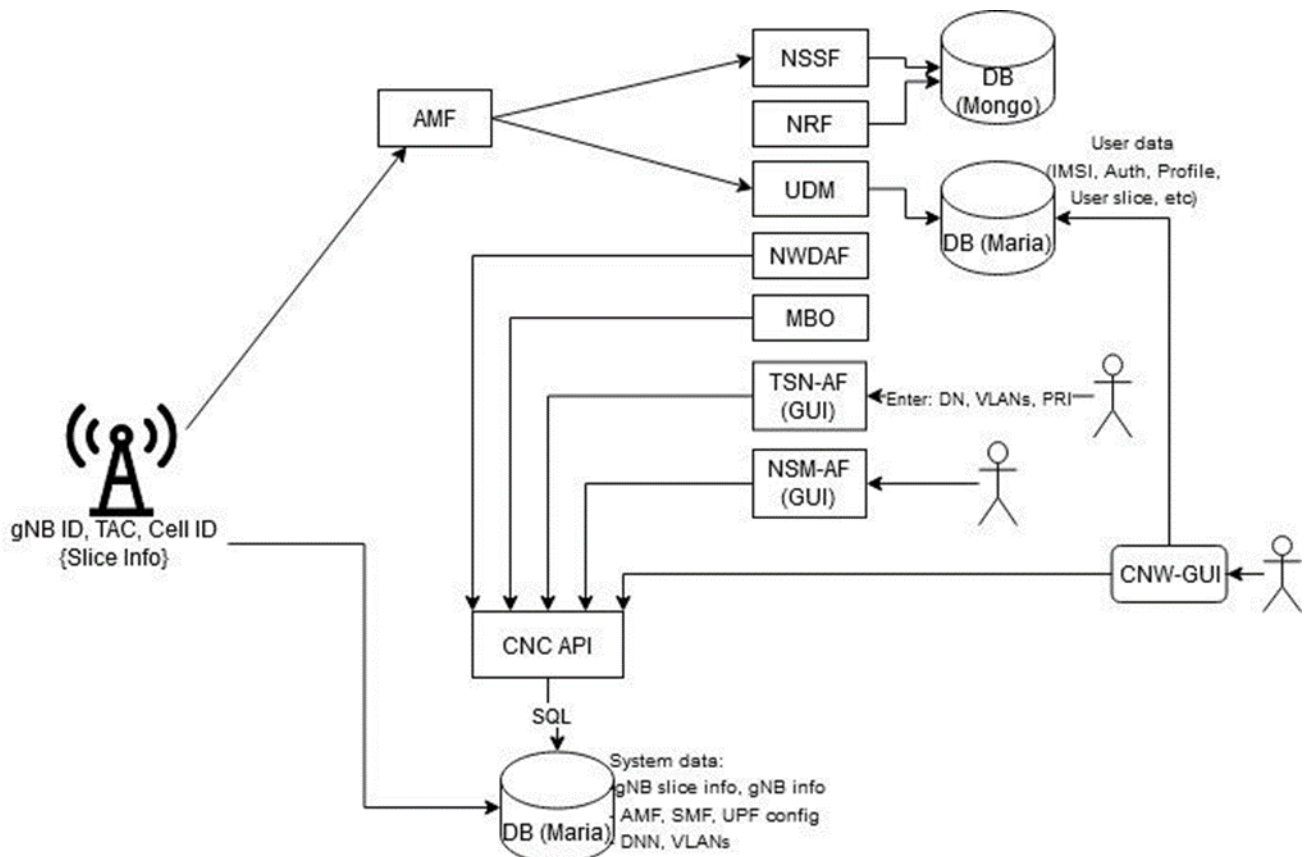


Figure 5-5 5GLAN and network configuration through the CumuCore Network Controller (CNC) API

The implementation of the network slice management API

The SM, which forms part of i2CAT's SOE tool, executes steps for handling the preparation, fault, instantiation, configuration, service provisioning, service update, and service recovery procedures per network slice. The following list summarizes the operational flows carried out by the SM:

- Management of infrastructure resources, which includes Cloud and Edge Compute Registration: Performed through the interaction with VIMs.
 - Compute resource object stored in the SM.
 - Compute resource information to be reached from SM.
- RAN Registration: Performed through the interactions with RAN controller.
 - Cell Creation and configuration
 - Devices and interfaces configuration
 - RF and ports configuration
- Management of reserved infrastructure resources (chunks), which include three parts: Cloud and Edge Compute Chunks:
 - Assigned Compute chunk stored in SM.
 - Related-slice users and project created in OSM.
 - VIM account registered in OSM.
- RAN Chunks:
 - Retrieval information of RAN topology configured previously.
 - Validation of RAN resources.
 - Radio chunk creation delegated to RAN Controller.
 - Radio chunk stored in SM.

- (5GCN, Cloud or edge) Network Chunks:
 - Retrieve information of pool of addresses per slice.
 - Network chunks creation delegated to VIMs and 5GCN Controller.
 - Network Chunk stored in SM.
- Management of network slices and radio services including following parts:
- Network Slice Instance:
 - Create network slice instance in terms of reserved infrastructure resources.
 - Network slice instance information is stored in the SM as collection of chunks.
- RAN Service:
 - Radio service object updated and/or configured and stored in the SM.
 - Linking radio access nodes, belonging to the slice, with 5G core network deployed if it is required.
- Management of network service instances as Container Network Functions (CNFs):
- Vertical Services:
 - Instantiation or termination of the IoT device service.
 - The IoT device service objects deployed and stored in SM.

Security handling of the IoT-NGIN enhanced network slice management API

In this project, the SOE is not the entry point for IoT devices, hence it does not implement security methods within its internal components. However, the interaction between the IoT-NGIN 5G resource management API adapter and the SOE is expected to take place in a secure network. Furthermore, the southbound client entities of the SOE use security credentials for interaction. The SM, for instance, communicates with Kubernetes, OSM, and the RAN controller using security credentials, preventing unauthorized access. In addition, since each network service runs on its own isolated network slice, if a security attack occurs in one of the network slices, the other network services remain unaffected as they are running on different network slices. The Network slice management API developed by i2CAT applies only to local area private networks; therefore, it could be deployed to increase the security of smart agriculture and smart factory LL use cases using multiple network services.

The implementation of 5G device management API in 5G systems

This chapter describes the implementation in 5G systems of the 5G device management API features described in Chapter 5.3.1.

The concepts and implementation of the 5G device management API features are derived from the SEAL services described in 3GPP standard TS 23.434 [5] and based on the 5G-ACIA white paper [4]. SEAL supports vertical applications by providing a common set of 5G services such as group management, configuration management and location management. The implementation of the 5G device management API features is based on and described in further detail in the following 3GPP standards: SEAL standard TS 23.434 [5], TS 23.501 [7] and in TS 23.502 [12].

Several features are introduced to ensure the security of the 5G device management API. 5G exposure points must support a means for mutual authentication between the 5G network and an IoT-NGIN application, confidentiality, and integrity of communication between the 5G network and an IoT-NGIN application, and a means for authorization of an IoT-NGIN application to use exposed capabilities [4]. IoT-NGIN applications will be able to interact with the 5G network via the 5G device management API only using the device 5G Generic Public Subscription Identifier (GPSI).

The Subscription Permanent Identifier (SUPI) is a globally unique identifier that is allocated to each device by 5G network, and it is only used inside of 5G network. The Subscription Concealed Identifier (SUCI) is a privacy related identifier that maintains the concealed SUPI. Within the 5G network, the SUPI and the SUCI are used. The mappings between the GPSI and the SUPI are maintained by the 5G network. The GPSI is either Mobile Station International Subscriber Directory Number (MSISDN) or an external identifier in the form of username@realm. The SUPI is in the format of IMSI (International Mobile Subscriber Identity), or NAI (Network Access Identifier) as specified in RFC 7542.

IoT-NGIN applications interact with the Network Exposure Function (NEF) of the 5G network via the N33 reference point for each device management related operations. For example, NEF will be used for handling of 5G virtual network groups, setting of the QoS of the connection to the device and gathering of the device location information. The information handled during the device management operations, such as external group IDs, are stored in the 5G network in the Unified Data Management (UDM) function.

Apart from the 5G functions described above, further 5G functions can be called depending on the need. For example, for the purpose of gathering device location information, the Access and Mobility Management Function (AMF) and the Gateway Mobile Location Centre (GMLC) can be called.

Detailed implementation descriptions of each of the 5G device management API features in the 5G system are provided in Annex 1.2.

5.4 IoT-NGIN 5G resource management API

In this chapter the interface between the IoT-NGIN 5G resource management API and the IoT device or client application is described. This part of the API will be implemented in Rust and will provide a REST API for us by the different services.

5.4.1 IoT-NGIN 5G resource management API features

In this sub-chapter, four features of the API are listed. The list is preliminary and might be extended if further features are identified. With these four operations a first prototype covering a majority of identified use cases can be developed. A list with brief descriptions of the selected and discussed use cases can be found in Table 5-2. More detailed information about the functionalities is given in chapter 5.4.3.

Table 5-2 List of features

Feature	Description
Start stop service	The interaction needed to start and stop a server
Resource allocation	The interaction needed for resource reallocation in case of changed resource requirements by the service

Service migration	The interaction needed for service migration triggered by the supervisory service or the IoT device.
Connectivity management	The interaction needed to providing connectivity either between services in a virtual network within the cloud or with a service outside the cloud.

5.4.2 OpenAPI Specifications of the IoT-NGIN 5G resource management API

In this chapter, the first version of the simplified API is briefly described with a focus on specific features. The specification itself can be found on the project's Gitlab repository¹. A screenshot of the Gitlab repository is shown in Figure 5-6. Currently, the repository is not yet publicly available. After internal sanity checks, the repository will be opened for the public.

The API operations are defined in OpenAPI format. At the time of creation of this document, the defined API calls listed in Table 5-3. These API operations represent the interface for the application developer or IoT device user. These are simplified API operations which are translated to standardised and vendors specific API operations listed and described in chapter 5.3. The translation is achieved by implementing an adapter using standardised and vendor specific APIs in the south-bound interface.

Table 5-3 List of 5G IoT-NGIN Resource Management API operations

Operation	Short description
Device control API	
/device	Register device with API.
/device/{device-id}	Query device information or unregister device.
/device/{device-id}/qos	Query device quality-of-service parameters or change QoS Parameters for Device.
/device/{device-id}/location	Query device location.
Service control API	
/service/start/	Start a new service.
/service/{service-id}	Get Service information or terminate a service.
/service/{service-id}/load	Get the CPU load produced by the service.
Network control API	

¹ https://gitlab.com/h2020-iot-ngin/enhancing_iot_underlying_technology/5g-api

/virtual-network	Create new virtual network.
/virtual-network/{network-id}	Get virtual network configuration or remove virtual network.
/virtual-network/{network-id}/device	Add a device to a virtual network or list all devices in a virtual network.
/virtual-network/{network-id}/device/{device-id}	Remove device from virtual network.

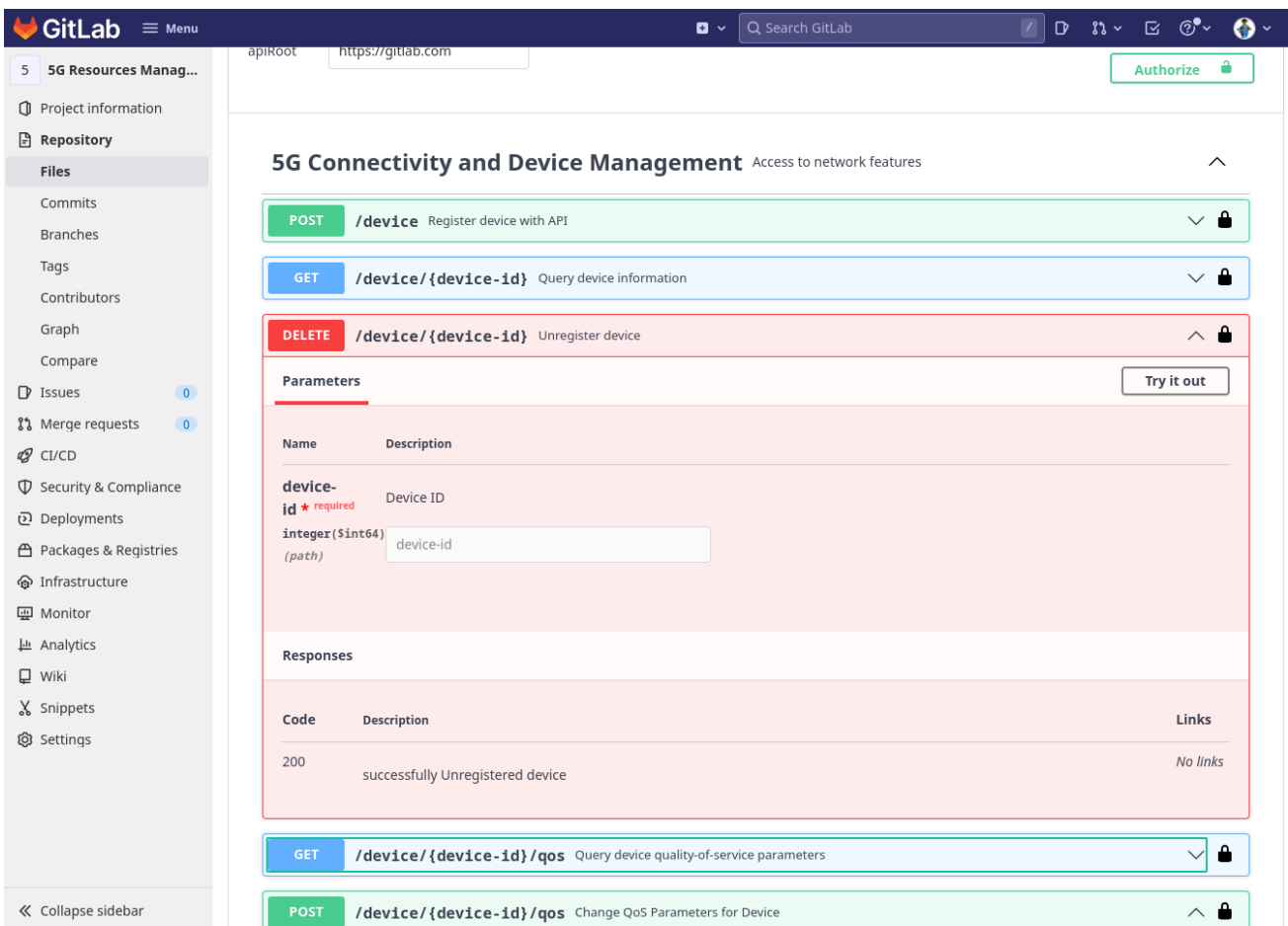


Figure 5-6 Swagger IoT-NGIN 5G resource management API definition

5.4.3 Sequence Diagrams of the IoT-NGIN 5G resource management API

This chapter provides an exemplary set of sequence diagrams. Additional sequence diagrams can be found in Annex 3 of this document. Each sequence diagram illustrates a single case that is going to be implemented in the IoT-NGIN 5G resource management API. In general, all sequence diagrams are defined from an IoT device viewpoint. Therefore, the **IoT device** is placed on the very **left** of each diagram. For the following sequence diagrams, three domains are identified and illustrated with backgrounds of different colours.

In the **sequence diagrams in the sub-chapters below**, the leftmost domain in **light yellow** is the IoT device domain representing one or multiple devices deployed by the customer. An example for an IoT device could be a Raspberry PI that is connected via 5G to a base station.

The IoT-NGIN-API Implementation **in light blue** is placed next to the IoT device domain. This domain represents the components needed to implement the IoT-NGIN 5G resource management API. This API implementation can either run independently of, or within, the edge cloud. All components needed to translate a simplified API operation from the IoT device to the infrastructure manager API operation are included here. The infrastructure manager represents the endpoint of standardised or the vendor specific 5G infrastructure. The IoT-NGIN-API implements the generalized side of the IoT-NGIN 5G resource management API as well as the customizations needed, depending on the specific 5G use case.

The rightmost domain **in orange** depicts the standardised and vendor-specific 5G infrastructure in orange and dark orange. This domain represents the custom use case-specific infrastructure, for example OpenStack or Kubernetes.

On the **rightmost outer side** of the diagrams, **actors**, which could either run inside the 5G infrastructure or outside in a customer owned infrastructure are not grouped into a box and are shown on a white background. Such actors (service endpoints) could be an IoT service micro-service or IoT applications. A power grid infrastructure control centre could be an example of a specific actor.

5.4.3.1 Service migration

The sequence diagram depicted in Figure 5-7 shows the case of service migration. The shown implementation assumes a stateless service that can be stopped and restarted. In this example, the service is running on Network Slice 1 and is migrated to Network Slice 2. Here, the service migration is triggered by an assumed supervisory service from the customer running in the IoT domain as well. However, neither a separate supervisory service is required for migration operations, nor does a supervisory service have to be separated from the IoT device.

D2.2 – Enhancing IoT Underlying Technology

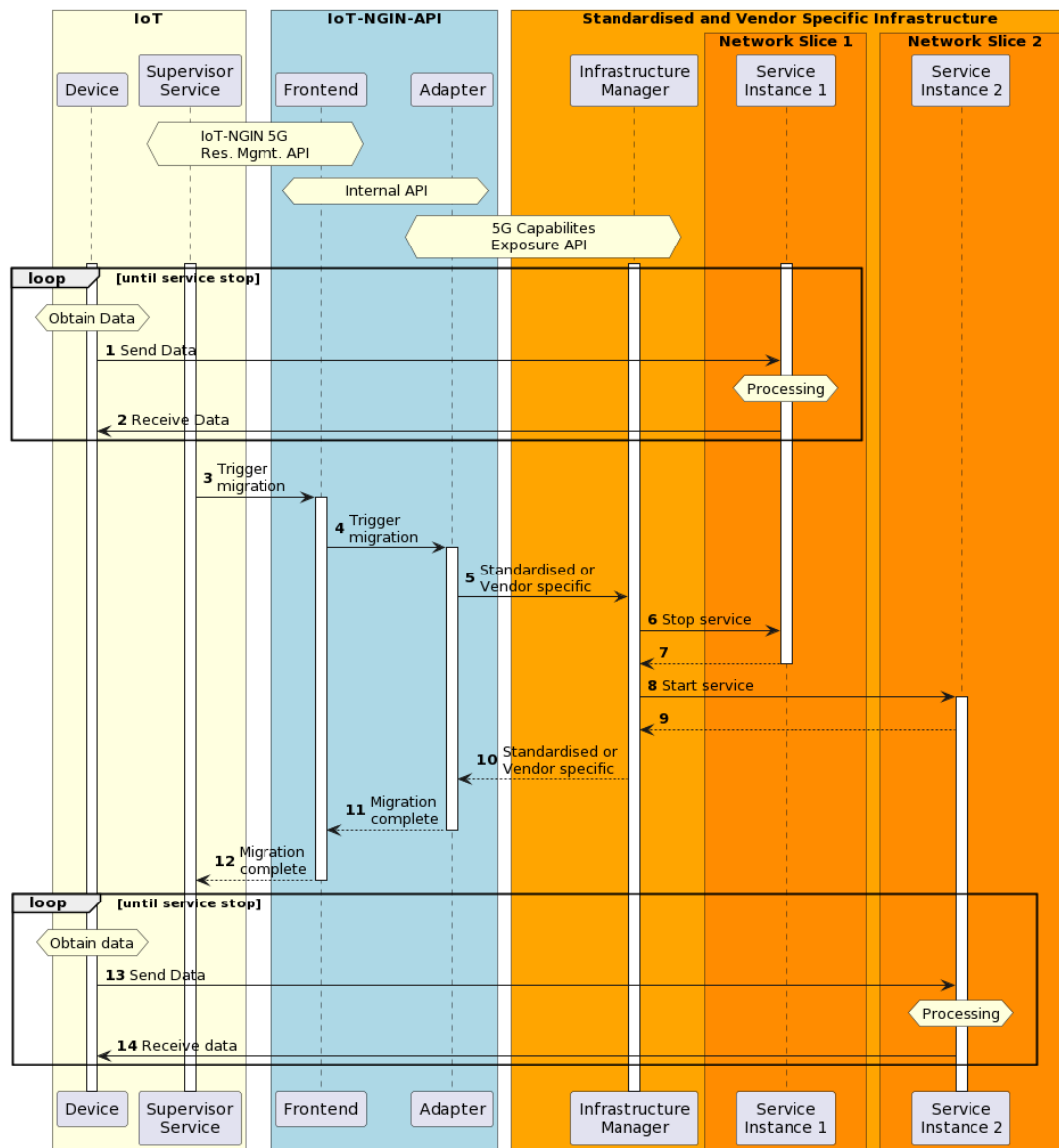


Figure 5-7 Service migration Connectivity management

The sequence diagram depicted in Figure 5-8 shows the case of enabling connectivity between an IoT device and a Service Endpoint. The IoT device as well as the service is in operation. First, the IoT device requests connectivity to the service. This request is processed and then forwarded to the infrastructure manager. After the connection has been established, the device can start sending data to the service endpoint.

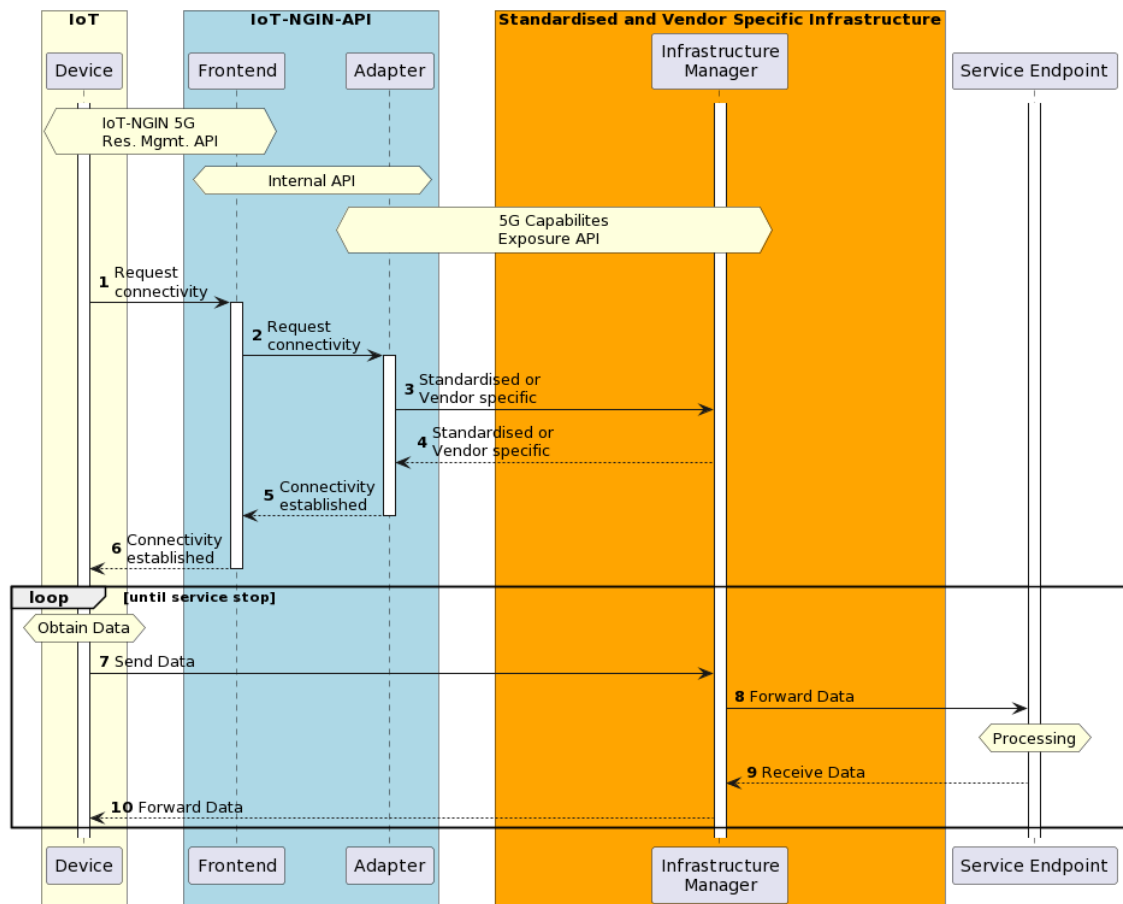
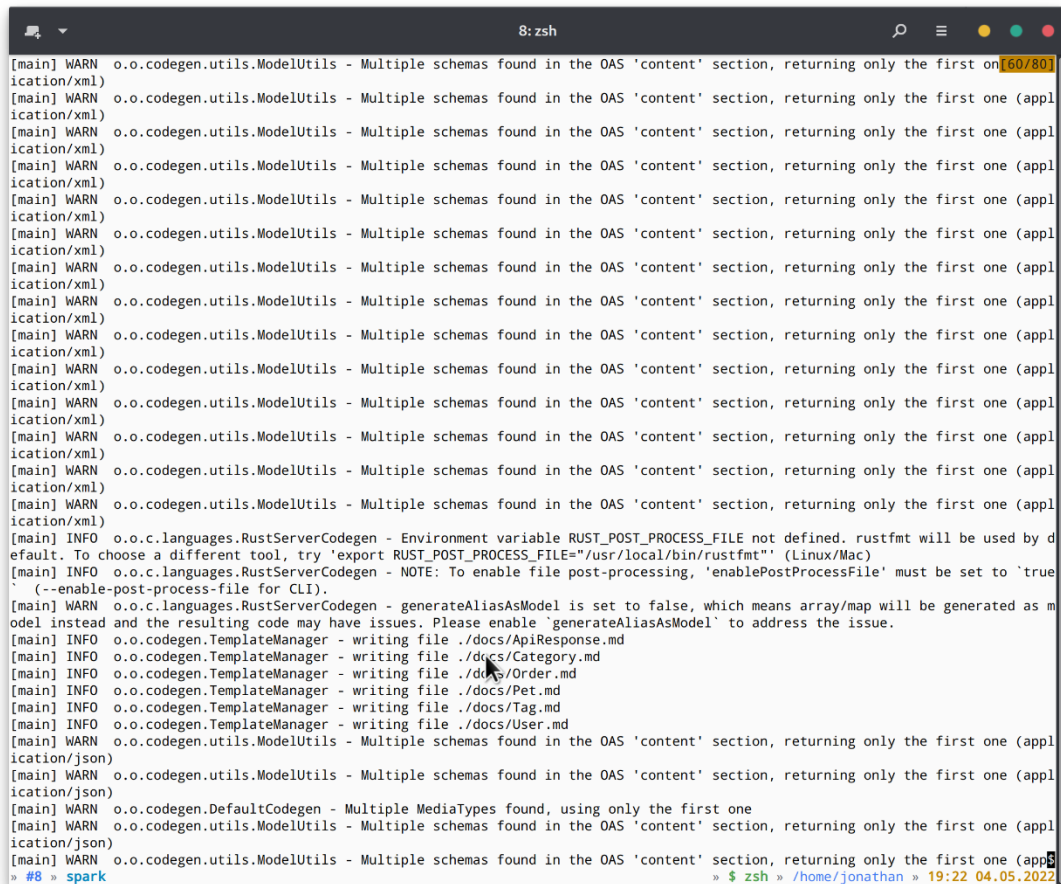


Figure 5-8 Connectivity management

5.4.4 Implementation descriptions of the IoT-NGIN 5G resource management API

For the implementation of the frontend application providing the API, a Rust based web server will be developed based on the OpenAPI specification mentioned in Chapter 5.4.2. For rapid development, the OpenAPI generator tools will be used to generate both stubs for the server as well as for unit tests to ensure functional validation. An example of this process is shown in Figure 5-9.



```
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/xml)
[main] INFO o.o.c.languages.RustServerCodegen - Environment variable RUST_POST_PROCESS_FILE not defined. rustfmt will be used by default. To choose a different tool, try 'export RUST_POST_PROCESS_FILE="/usr/local/bin/rustfmt"' (Linux/Mac)
[main] INFO o.o.c.languages.RustServerCodegen - NOTE: To enable file post-processing, 'enablePostProcessFile' must be set to 'true' (--enable-post-process-file for CLI).
[main] WARN o.o.c.languages.RustServerCodegen - generateAliasAsModel is set to false, which means array/map will be generated as model instead and the resulting code may have issues. Please enable 'generateAliasAsModel' to address the issue.
[main] INFO o.o.codegen.TemplateManager - writing file ./docs/ApiResponse.md
[main] INFO o.o.codegen.TemplateManager - writing file ./docs/Category.md
[main] INFO o.o.codegen.TemplateManager - writing file ./docs/Order.md
[main] INFO o.o.codegen.TemplateManager - writing file ./docs/Pet.md
[main] INFO o.o.codegen.TemplateManager - writing file ./docs/Tag.md
[main] INFO o.o.codegen.TemplateManager - writing file ./docs/User.md
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/json)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/json)
[main] WARN o.o.codegen.DefaultCodegen - Multiple MediaTypes found, using only the first one
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/json)
[main] WARN o.o.codegen.utils.ModelUtils - Multiple schemas found in the OAS 'content' section, returning only the first one (application/json)
» #8 > spark » $ zsh » /home/jonathan » 19:22 04.05.2022
```

Figure 5-9 Example code generation utilizing the OpenAPI tools

Depending on the utilized underlying infrastructure, different security concepts are possible for access control. On the one hand, the security could be provided by the underlying infrastructure (5G standard security features could be used), on the other hand, the frontend of the API implementation could provide security features. This needs to be decided case by case.

The access control security options described above do not necessarily take care of the security of the application itself. In IoT-NGIN, we utilize more secure programming languages such as Rust and modern secure programming paradigms to increase application security.

5.4.5 Living Lab API feedback

To validate the selected set of features of the IoT-NGIN 5G resource management API, general feedback was acquired from the Living Lab owners. The major challenges in defining a questionnaire to be completed by Living Lab owners included explaining the API topic which is abstract in nature, as well as the difficulty of defining the questions in a non-technical manner, while maintaining a clear relationship between the questions and the underlying problem. The first feedback round provided good initial feedback and provided reassurance of the appropriateness of the approach taken in WP2, as well as on the relevance of the functionalities we defined.

5.5 Next steps

In the next phase, the implementation of the 5G resource management API will be matured and updated. Based on the matured specification of the API and on a practical implementation of the IoT-NGIN 5G resource management API, we plan a further feedback round with the Living Labs to gather more input to finalise our specifications and implementations. This second feedback round will verify that we have overcome the challenges identified in the first feedback round and, as a side effect, it will also update the Living Labs on the outcomes of the API work.

The next step for 5GLAN API is to complete the Application Function (i.e., TSN-AF) for interacting with the fixed TSN controller. The TSN API does not exist at present and needs to be implemented. The TSN-AF will then include a TSN API to be used by the TSN controller.

As the next step for the network slice management API, the network service migration will be implemented. The service migration scenario could be used for allocating either more resources to an existing network service, which is running on a specific network slice or creating new network slice, if needed, and then run a new instance of the network service on top of the new slice with more resource allocated to it.

As next steps for device management API, a demonstration of the 5G device management API supporting a smart industry use case of IoT-NGIN project (Use Case 6 - Human-centred safety in a self-aware indoor factory environment) will be investigated. The scenario being investigated for the lab testing and demonstration is that of illustrating how the functionality of the 5G device management API can provide easy to use 5G functionality to smart industry use case owners. The 5G device management functionality will improve the ease of use of 5G features, remove the need to interact with mobile network operators or network infrastructure vendors, and reduce the time and therefore cost associated with the configuration and reconfiguration of devices. Furthermore, investigations of the possibilities to contribute to 3GPP SA6 based on the 5G device management API will continue.

5.6 Conclusions

In this chapter an overview of the novel IoT-NGIN 5G resource management API approach to improving the ease of use of 5G resource management was provided. This is a major step towards defining and providing an easy-to-use 5G resource management API for application developers of IoT applications using private and public 5G networks.

A subset of the features of the API has been described. In addition to descriptions of the features prototyped, the API operations, sequence diagrams, implementation details (describing how the implementation should be realised based on the normal 3GPP format of implementation descriptions) of the prototypes as well as interactions between the north-bound interface to the applications and the south-bound interfaces of the 5G Infrastructure are described. Aspects of these 5G capabilities exposure API features are being prototyped in the live 5G laboratory networks of project partners.

Based on the detailed description of the 5G capabilities exposure features provided in this deliverable, a simplified north-bound interface for end-users, software developers and other customers is proposed as part of the IoT-NGIN 5G resource management API. This north-

bound interface is described as an OpenAPI specification and backed up with multiple sequence diagrams describing examples of the interaction between customers and the API.

In general, significant progress has been made towards achieving the goal of defining the IoT-NGIN 5G resource management API in sequence diagrams. An early version of the OpenAPI specification of the IoT-NGIN 5G resource management API have been developed and described.

6 Enhancing 5G and IoT applications security with a secure edge cloud micro-services execution framework

6.1 Approaches to edge cloud infrastructure

Typical cloud and edge infrastructure is often based on containers, which allows the existence of multiple isolated user space instances. Such instances can be used as building blocks to deploy, maintain, and scale applications on cloud and edge infrastructure. Kubernetes² and its lightweight version k3s³ are typical orchestration tools to manage such building blocks. In order to secure such an infrastructure, the complete infrastructure must be hardened against attacks. Excellent guidelines to harden compute infrastructure are published by the National Security Agency (NSA) [8] and the German Federal Office for Information Security (BSI) [9]. The NSA also points out the following:

Some platforms and container engines provide additional options to harden the containerized environments. A powerful example is the use of hypervisors to provide container isolation. Hypervisors rely on hardware to enforce the virtualization boundary rather than the operating system. Hypervisor isolation is more secure than traditional container isolation. ... Some security focused container engines natively deploy each container within a lightweight hypervisor for defense-in-depth. Hypervisor-backed containers mitigate container breakouts.

The reason for the above-mentioned suggestion is that containers are traditionally based on OS-level virtualization, where the applications are bundled into logical namespaces. The left side of Figure 6-1 describes this traditional way to handle containers. The disadvantage of this technique is the lack of isolation between the host operating system and the containerized application. A security vulnerability in the container runtime directly exposes the host kernel for attacks or the other containers, which are managed by the runtime. To increase the security of such an environment, a second layer, as proposed by the above-mentioned citation, can be added. This is achieved by running a virtual machine instance inside the container as depicted on the right-hand side of Figure 6-1.

² <https://kubernetes.io/>

³ <https://k3s.io/>

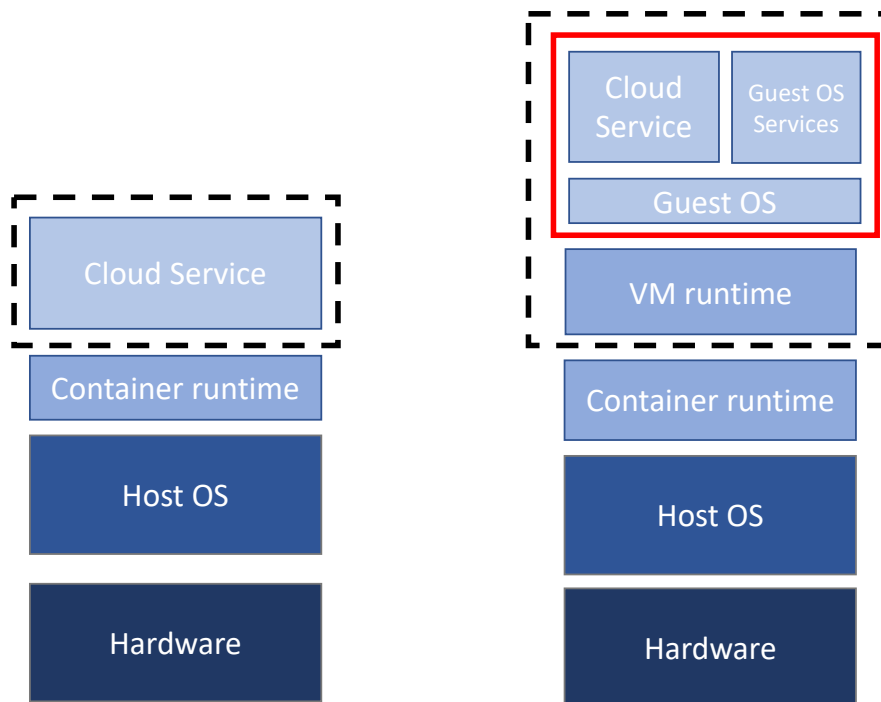


Figure 6-1 Comparing container technology-based OS-level virtualization and virtual machines

There are several existing techniques for realize containers. The Open Container Initiative⁴ (OCI) specified a standard interface for the creation of containers and in general the OCI also offers containers based on virtual machines. These containers provide a stronger isolation between deployed services and the host system, thus enhancing 5G and IoT applications security. With Kube⁵⁶ multiple open-source projects exist to realize VM-based containers.

6.2 Unikernel and microVMs

The usage of virtual machines provides stronger isolation, but increases the overhead needed to run an application. To use such virtualization technologies in IoT environments, it is important to reduce such an overhead created by the virtualization. One solution is the use of microVMs. Instead of creating a virtual machine emulating a real computer for running traditional general-purpose operating systems, these VMs are optimized and shrunk down to only contain the functionality necessary to run a specific application within a hypervisor. With Solo⁵⁷, Firecracker⁸, uhyve⁹ and Qemu's microVMs¹⁰, several already established solutions exist, which are able to reduce the memory footprint and the boot time significantly.

⁴ <https://opencontainers.org/>

⁵ <https://kubevirt.io/>

⁶ <https://katacontainers.io/>

⁷ <https://github.com/Solo5/solo5>

⁸ <https://firecracker-microvm.github.io/>

⁹ <https://github.com/hermitcore/uhyve>

¹⁰ <https://qemu.readthedocs.io/en/latest/system/i386/microvm.html>

IoT-NGIN applications will be based on a set of micro-services. Consequently, each container deploys usually just one application, mainly using the network interface and CPU to handle requests. For each container, traditional multi-processor, multi-tasking, multi-user operating system such as Linux as guest, would create a lot of overhead for small IoT-NGIN applications. Library Operating Systems (also known as Unikernels) are an attractive solution decreasing this overhead. The basic idea is to bundle the kernel with the application by linking them together and transforming the application and the kernel into a bootable application. Consequently, this realizes a single-address-space machine image only containing the necessary code for the application, thus reducing the memory footprint and boot time. In addition, the complete software stack from the kernel through the IP stack to the application itself can be analysed and optimized. Besides performance improvements, this reduces the attack surface of an application thus improving the security. However, in comparison to traditional operating systems such as Linux, unikernels are a relatively young technology and still have to prove their robustness. MirageOS¹¹ and RustHermit¹² are two established approaches in this new area. In IoT-NGIN, RustyHermit will be used to show the robustness of the unikernel approach. As Figure 6-2 shows, the IoT-NGIN approach combines unikernels with a microVM to minimize the overhead. The setup is similar to the right-hand side of Figure 6-1, but instead of first running a full virtual machine, a lightweight unikernel is used.

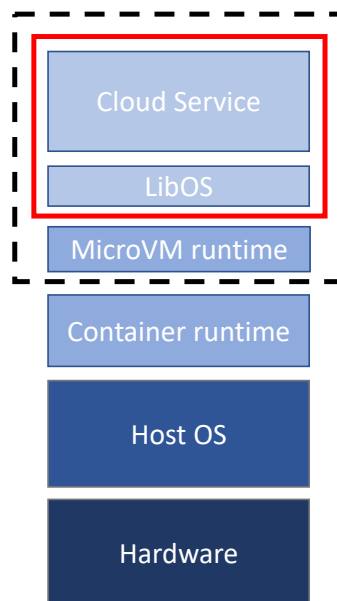


Figure 6-2 Container runtime based on a microVM

6.3 The IoT-NGIN Secure Edge Cloud Framework

From a virtualisation point of view, two main technologies for realizing an edge-cloud exist today. One is containerization and the other one is virtualization. Both have their advantages and disadvantages. The advantages include the lightness of containers and the increased security of virtual machines. The disadvantages include the reduced isolation of containers

¹¹ <https://mirage.io/>

¹² <https://github.com/hermitcore/rusty-hermit>

and the higher overhead related to the virtual machines. This is described in Chapter 6.1 in more detail. The approach described in Chapter 6.2 combines the advantages of the two mentioned technologies to achieve a lightweight and fast application in a secure virtual environment.

IoT-NGIN is developing a flexible edge-cloud framework with security being one of its key paradigms. For this reason, we investigate and further develop the described microVM with a unikernel approach as a central component of the framework.

Nevertheless, the framework should support classical containers as well. This lowers the burden for adoption and solves the chicken and egg problem of infrastructure deployment and application adoption. An integration into common and existing orchestration tools such as Kubernetes is a key requirement. Figure 6-3 shows, the creation of containers utilizing the standardized OCI interface. OCI defines a runtime interface to manage containers, including spawning new containers. Typically, Kubernetes and Docker utilising OS-level virtualization use runc. In IoT-NGIN, a new container spawner runh that is not only able to spawn common containers but also the previously introduced containers based on a microVM and the unikernel RustyHermit is developed. This runh spawner prototype developed in IoT-NGIN is available in GitHub¹³.

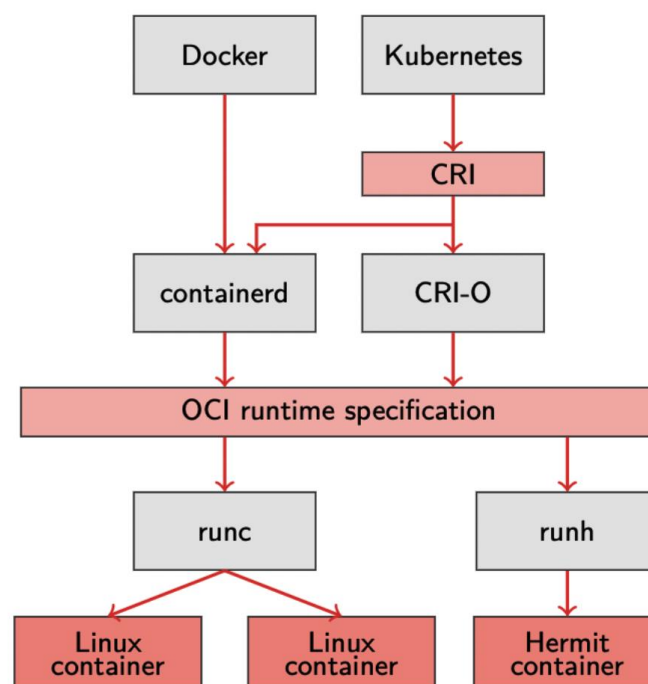


Figure 6-3 Comparing container runtimes based on the OCI runtime specification

Since Kubernetes 1.2, the simultaneous use of different runtimes is supported by means of a selector in the container description file, which defines the runtime used to spawn a container. This allows mixing of different runtimes, thus being a core component for the IoT-NGIN framework.

¹³ <https://github.com/hermitcore/runh>

The performance of this new **runh** container runtime can be evaluated by running benchmarking tests, which start a small demo application within a container that permanently consumed computation cycles thus stressing the system, on a range of possible implementations. Our benchmarking set of tests was run for runc, and runh with and without microVMs.

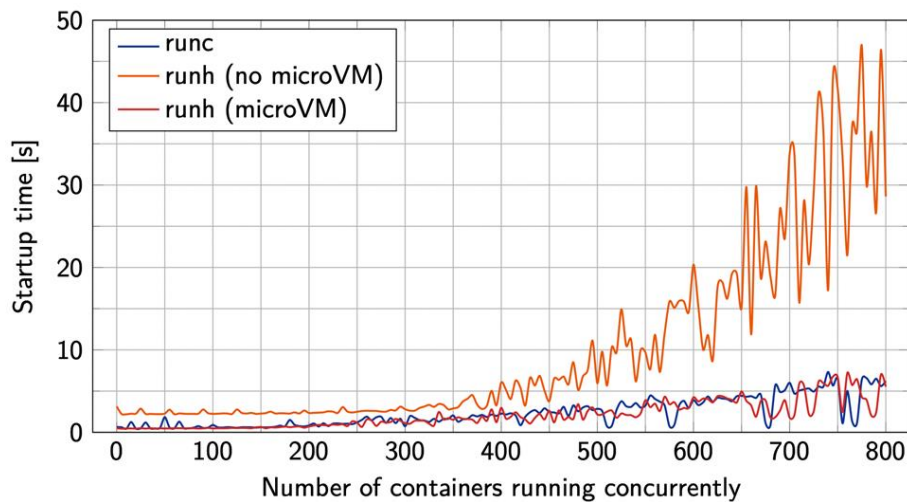


Figure 6-4 Startup time for Hermit and Linux containers under increasing system load

The benchmarking test set starts up to 800 containers. Figure 6-4 shows that containers based on OS-level virtualization (**blue line**) basically provide the same performance as containers based on microVMs (**red line**). Consequently, a stronger isolation based on a unikernel and a microVM has very low impact on the overall performance. In contrast, the usage of a unikernel within a common virtual machine clearly results in a larger overhead (**orange line**). This clearly shows the benefits of microVMs and is a first indicator for their excellent scalability of this approach.

In IoT-NGIN, a strong emphasis is put on new machine learning technologies and approaches. This challenges the suggested microVM+unikernel approach, as machine learning often involves hardware accelerators such as GPUs but neither do today's microVMs support PCI pass-through, nor does any unikernel include drivers for this hardware accelerators. Eiling et al. [10] have presented a technology to circumvent this problem. Their software Cricket provides a server to relay calls utilising the prominent CUDA framework¹⁴ via the network by means of remote procedure calls (RPC). As RPCs are also available for unikernels, this technology can be combined with the microVM+unikernel approach as depicted in Figure 6-5. The feasibility has already been demonstrated with a working prototype, developed within the IoT-NGIN project and available in the IoT-NGIN Gitlab. Such remote GPU access could, for instance, be provided by a separate VM on the same host as the microVMs+unikernels, or by a dedicated GPU server in the cluster.

¹⁴ <https://developer.nvidia.com/cuda-toolkit>

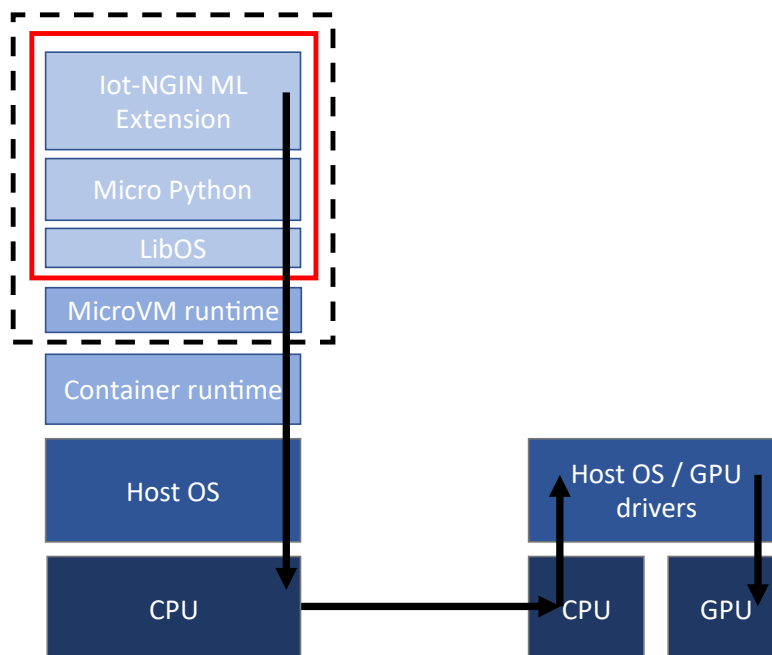


Figure 6-5 Enhancing 5G and IoT applications security with a secure edge cloud micro-services execution framework

6.4 Next steps of this work

A lot of effort was put into researching the different technologies and comparing the different solutions. In the next steps the different solutions, e.g., the runh spawner and Cricket, will be extended and integrated into the IoT-NGIN secure edge cloud framework. This framework will then be validated in terms of not only functionality, but also scalability and compatibility with existing solutions. The evaluation will then be extended for the use cases of the living labs, in particular the integration with the agriculture use case will be investigated. In addition, it will be shown that trained machine learning models can be deployed with the secure edge framework approach, to support major use cases in IoT-NGIN.

6.5 Conclusions

In conclusion, this chapter provides a comparison of today's and a possible tomorrow's approach to the deployment of cloud services. The proposed approach is superior, especially from a security perspective as it improves security by design. By reducing the attack surface, the number of possible attack vectors is reduced, and the severity of a possible breach is decreased.

Furthermore, it is shown that the impact on response time of a unikernel, running inside microVM, is minimal to non-existing compared to the conventional approach of just containerizing the application. By utilizing RustyHermit as an open-source project, contributions from different vendors and developer groups are possible. Additionally, the support for GPU hardware can easily be added via remote procedure (without the need to implement pass-through support or new GPU drivers). This exposes the GPU functionalities for machine learning applications to the RustyHermit running inside a microVM, thus enabling use cases of IoT-NGIN living labs.

7 Relationship of the 5G enhancements to markets and LL use cases

This chapter describes the relationship of the 5G enhancements developed in the IoT-NGIN project to the target markets and the Living Lab use cases.

7.1 The target markets for IoT-NGIN 5G enhancements

Our 5G enhancements target two market segments, namely the market for wide area public networks and the market for local area networks. Table 7-1 illustrates the target markets for the 5G enhancements being developed in the project.

Table 7-1 Target 5G markets addressed by IoT-NGIN 5G-Enhancements

5G enhancement	Potential target markets addressed by the 5G enhancement
Coverage extension using D2D optimisation	Wide area 5G public networks and local area 5G networks can be addressed by this enhancement. The markets addressed are not limited to those for 5G networks. This enhancement can be used together with almost any communications network to optimise its use of D2D techniques to extend its coverage.
Network slice management API	This enhancement, offering improved network slice management, targets private 5G local area networks .
TSN	The target markets of the Time Sensitive Networking (TSN) are private 5G local area networks in an industrial context , and following standardisation, wide area 5G public networks markets can be addressed.
5GLAN API	The target markets addressed by the 5GLAN API are the private 5G local area networks market in industrial contexts , and following standardisation, wide area 5G public networks markets can be addressed.
5G device management API	The target market addressed by the 5G device management API is the market for private local area 5G networks in an industrial context . The standardisation of the functionality is being investigated so that other vendors of private local area 5G networks could implement the same functionality and interworking between the implementations of different vendors can be ensured.
IoT-NGIN 5G resource management API	The IoT-NGIN 5G resource management API, considered in its broadest scope, can target the market for private local area 5G networks for any use case sector if extensive further development work is undertaken on the specification. Further extensive

Secure Edge Framework	<p>standardisation effort would be needed before public wide area 5G networks could be addressed as a market. Investigations of standardisation of aspects of the IoT-NGIN 5G resource management API are already on-going.</p> <p>The target markets addressed by the Secure Edge Framework includes both the markets for both public and private 5G wide and local area networks. This functionality could be integrated in both local and wide area 5G networks.</p> <p>The Secure Edge Framework does not need to be associated with a 5G network. It can be deployed in any IT infrastructure cloud and hence addresses a very broad set of markets.</p>
-----------------------	---

7.2 The relationship of the 5G enhancements to the LL use cases

Table 7-2 shows the type of geographical wireless network coverage required by each IoT-NGIN Living Lab use case. Some use cases require wide area public network coverage while others require local area coverage. In the table, use cases requiring **local area wireless network coverage** are marked in **Green**. We are investigating deploying several of the project 5G enhancements targeting local area private 5G networks at the Living Lab field trial sites of these use cases.

The use cases requiring **wide area 5G coverage** are shown in **Gold** (use cases 1,2,3,4,5 and 9) in the table below. We have been investigating the availability of public 5G networks at Living Lab sites and we continue these investigations.

The deployment of the prototype IoT-NGIN 5G enhancements in 5G public networks is not possible before standardisation and productisation of these features has taken place. The deployment of prototype services could interfere with the correct working of public networks and could lead to disruptions of service for their customers in mission critical infrastructures as well as for private users.

Table 7-2 Living Lab field trial site use of Public 5G or Private 5G with project 5G enhancements

Use case/ Geographical coverage requirement	UC 1	UC 2	UC 3	UC 4	UC 5	UC 6	UC 7	UC 8	UC 9	UC 10
Local area coverage needed by UC										
Wide area coverage needed by UC										

In recent months, we have undertaken extensive investigations of the possibilities for the laboratory testing and deploying of 5G enhancements in support of Living Lab field trial use cases, taking into account the need to focus project resources to maximise the impact of our results, and the technical, standardisation, financial and operational implications of organising individual laboratory test sets and field deployments.

7.3 Conclusions

Our 5G enhancements target two market segments: the markets for wide area 5G public networks and for local area 5G networks and we have related the capabilities of our 5G enhancements to the needs of the IoT-NGIN use cases. Six of the use cases require the support of wide area mobile networks while four use cases require local area network support. All seven 5G enhancements target local private 5G networks while the D2D and secure edge framework enhancements additionally target wide area 5G networks.

The relationship of the 5G enhancements to the Living Lab field trials have been developed and work on this topic continues.

8 Conclusions

A mutual understanding by all partners of the relationship between the 5G enhancements, the IoT development topics and the Living Lab use cases in the project was achieved. The activities of developing 5G enhancements have been closely integrated with all project work packages through an organised dialogue.

For each enhancement, we are working on implementations and specifications of the developed functionality and have provided specifications of the IoT-NGIN resource management API. The tool needed for the coverage extension with relay functionality has been implemented. The 5GLAN API implementation was completed and the TSN implementation is on-going, as is the implementation of the Secure Edge Framework. Laboratory testing of much of the functionality developed has already been undertaken and further tests are on-going.

We have undertaken extensive investigations of the actions needed to bring our enhancements to market, defined the target markets for each enhancement, defined the testing of enhancements in the laboratory and are continuing our investigations of the possibilities for deployment of 5G enhancements in the Living Labs use cases.

The key takeaways from our work so far can be summarised as follows:

- Standard LTE and 5G networks offer good support for the current requirements of many of the Living Lab small scale implementations of the use cases studied in IoT-NGIN, contributing to the digitalisation of many industrial sectors, and providing them with flexible and mobile communications options.
- The specific 5G enhancements being developed in IoT-NGIN have the potential to make a strong contribution to the large-scale commercial deployment of applications implementing the IoT-NGIN use cases by extending the capabilities of existing standard 5G networks with features enabling optimised performance of the network and increased ease of use for industrial sector IoT applications and users of 5G. Our enhancements will optimise options for network coverage extension, improve support for deterministic communications, simplify the interactions of IoT applications and industrial users with the 5G network through the introduction of the IoT-NGIN 5G resource management API functionality, and improve security and flexibility with the Secure Edge Framework.
- Additionally, through discussions with the Living Lab partners, we have learned that our IoT-NGIN 5G enhancements can enable new use cases for 5G in industrial sectors.

In summary, we have made very good progress towards achieving our objective of exploring implementing and promoting the synergies between enhanced 5G technologies and IoT technologies to the benefit of the IoT-NGIN vertical application sectors of smart cities, smart agriculture, smart manufacturing and smart energy.

9 References

- [1] "OPC Unified Architecture (UA)," 2008. [Online]. Available: <https://opcfoundation.org/about/opc-technologies/opc-ua/>.
- [2] "IEC 61784-2: Real-time Ethernet".
- [3] "IoT-NGIN D2.1; Enhancing IoT M2M/MCM Communications," 2021.
- [4] 5GACIA, "Exposure of 5G Capabilities for Connected Industries and Automation Applications," February 2021. [Online]. Available: https://5g-acia.org/wp-content/uploads/WP_039_Network-Exposure-Interface_single-pages.pdf. [Accessed February 2022].
- [5] "3GPP TS 23.434; Service Enabler Architecture Layer for Verticals (SEAL); Functional architecture and information flows; Release 18," 2022.
- [6] "3GPP TS 29.549; Service Enabler Architecture Layer for Verticals (SEAL); Application Programming Interface (API) specification; Stage 3 (Release 17)," 2022.
- [7] "3GPP TS 24.545, Technical Specification Group Core Network and Terminals; Location Management - Service Enabler Architecture Layer for Verticals (SEAL); Protocol Specification; (Release 17)," 2022.
- [8] IoT-NGIN, D9.1 - Project Handbook, H2020-957246 IoT-NGIN Deliverable Report, 2020.
- [9] B. f. S. i. d. Informationstechnik, SYS.1.6 Containerisierung, 2022.
- [10] N. Eiling, J. Baude, S. Lankes, A. monti and Cricket, "A virtualization layer for distributed execution of CUDA applications with checkpoint/restart support".
- [11] "3GPP TS 24.544; Technical Specification Group Core Network and Terminals; Group Management - Service Enabler Architecture Layer for Verticals (SEAL); Protocol Specification; Release 17," 2022.
- [12] "3GPP TS 24.548, Technical Specification Group Core Network and Terminals; Network Resource Management - Service Enabler Architecture Layer for Verticals (SEAL); Protocol Specification; (Release 17)," 2021.
- [13] "3GPP TS 23.501; System Architecture for the 5G System, Stage 2 (Release 17)," 2022.
- [14] "3GPP TS 23.502; Procedures for the 5G System, Stage 2 (Release 17)".
- [15] "New Appendix on shared E.212 Mobile Country Code (MCC) 999 for internal use within a private network".

Annex 1 Features of the 5G capabilities exposure APIs

Annex 1.1 Features of the network slice management API

The Northbound API of SOE enables the SM to receive resource reservation and configuration requests from IoT-NGIN-API's adapter, which are related to network slices management tasks that SM performs. The features that are enabled through SOE's northbound APIs include exposing information about the registered infrastructure, configurable RAN and compute resources as well as creation/deletion of network slices. The methods enabling the mentioned features will be explained in more details in Annex 2.1.

Annex 1.2 Features of the 5G device management API

In this chapter, the detailed descriptions of the 5G device management API features considered in the 5G-ACIA white paper [4] as well as the potential uses of these features, especially in the smart agriculture use cases, are described. The exemplary use cases defined specifically for smart agriculture domain are considered as basis to define and support several other use cases from vertical sectors studied in IoT-NGIN.

Table 9-1 The description of various 5G device management API features relevant for IoT-NGIN applications

API Feature	Feature Description	Exemplary Use Case from Agriculture Domain
Provisioning and onboarding devices	This feature allows users to provide the unique identifiers of their 5G devices deployed in a field to the 5G network so that these devices are accepted by the network. The provisioning process is done through providing the Subscriber Identity Module (SIM) card or eSIM credentials of the devices to the API. After a successful provisioning, devices are onboarded to the 5G network, and they can start communicating and sending their data. This feature is not yet standardized and therefore it does not have API specifications.	Devices that are used in the smart agriculture use cases can be onboarded to the 5G network easily by a farmer. In addition, as each of these devices will have a unique IDs, these unique IDs can be used to list devices and to further analysis per device, such as vulnerability check.
Getting a list of devices	This feature allows users to get a list of devices provisioned to the 5G network. As the 5G devices have their unique	IoT vulnerability crawler being developed in WP5 in IoT-NGIN uses "list of

	<p>SIM/eSIM credentials that are provisioned to the 5G network by a user, they are stored in the network with their unique identities and this information can be requested by a user. The list of devices could be used by other tools/services to perform further operations such as sending a message from a service to a specific device.</p>	<p>devices" as a component for security check. In case 5G can provide this information to the vulnerability tool, then the integration of new devices to the IoT-NGIN platform would be easier, digital twins can be created quicker and vulnerability crawling process can start earlier.</p>
Creating, modifying and removing device groups	<p>This feature allows users to create many device groups with same or different purposes, group some devices in the same group and remove a device from a group. In case there are more than one 5G devices available in a field, some of these devices would need to be grouped in order to manage the data transmission or 5G devices easier. For instance, a group could be created if the 5G devices collect the same data from the field and send this data to the same service in an edge cloud.</p>	<p>Creating device groups per specific purpose would be relevant for devices used in the smart agriculture use cases. For example, groups of devices could indicate e.g., devices monitoring a certain crop type or devices belonging to a single farmer.</p>
Monitoring the quality of the communication links of devices	<p>This feature allows users to monitor the quality of the communication links of 5G devices. For some use cases, it would be necessary to know the current quality of the communication links of devices. According to the values received, prompt actions would need to be made. Monitoring can be performed by checking the current QoS parameters of a communication link, current reference signals received power values, current packet loss value or current latency value etc.</p>	<p>If the data rate of the communication link drops due to bad radio conditions, an alert could be sent to the drone inspecting the crop field. Based on the alert, the drone could change to a lower resolution codec and thereby adapt the video streaming quality to the link capacity.</p>
Defining and changing QoS parameters of individual device connections	<p>This feature allows users to define and change the values of Quality-of-Service (QoS) parameters for communication links of devices. By default, after 5G devices are onboarded to the 5G network, their communication links are established according to the default QoS parameters. However, higher values for e.g., bit rate would be required due to the stringent</p>	<p>In case the QoS parameters for the communication links of drones are low, the resolution of the images that are taken by the camera on the drones would also be low. By changing/increasing the relevant QoS parameters of</p>

<p>Getting location information of devices</p>	<p>throughput requirements of some use cases. In that case, this feature can be used to define new values and change the communication links of 5G devices without interrupting the communication. As a result, the values for QoS parameters of communication links are optimized for the requirements of use case.</p> <p>This feature allows users to get the location of their connected devices to track their mobility. For some use cases, it would be required to know the location of a device or a group of devices to see whether they are operating in the right area or not. According to the current location, some measures would need to be taken by a service, e.g., sending a control message back to the mobile devices with the correct coordinates that they should be.</p>	<p>that communication link, the resolution of the image/video could be increased and the diseased crop can be seen more properly.</p> <p>For future use cases, receiving the coordinates of the drone location could be relevant, e.g., the drone can be automatically moved from one location to another to perform aerial spraying to specific area with diseased crops.</p>
--	--	--

Annex 2 OpenAPI Specifications of the 5G capabilities exposure APIs

Annex 2.1 OpenAPI specifications for the network slice management API

In this chapter, some examples of the OpenAPI specifications of the Network Slice Management API features are given as follows:

- **GET {RAN/Compute} Infrastructure:**

This method provides the information related to the radio devices and compute domains previously registered in SM. This method allows different information about an instance to be exposed, such as its location, its status, and the URL through which it can be accessed.

- **GET {RAN/Compute} Topology:**

This is a purely informative method that provides information about RAN and compute resources, which can be configured properly and reserved for any network slice. The exchanged data by this operation includes the name of the radio devices and the cloud domains, as well as their location, vendor, and possible configuration.

- **POST {slice_name} Slice:**

By using this method, the user can request a set of radio, compute, and network resources for the creation of a new network slice. Moreover, the method enables specifying VLAN information used for service communication that describes an isolated end-to-end slice of the service network. The returned value could be the reserved radio/compute/network resources or an error message.

- **DELETE {slice_name} Slice:**

It is possible to perform the decommissioning procedure over a specific network slice using this method as well as to remove the previously reserved infrastructure resources so that the entire service communication associated with that network slice is disabled. Either an error message or a confirmation could be returned.

Annex 2.2 OpenAPI specifications for the 5G device management

In this chapter, the OpenAPI specifications of the 5G device management API features are given in detail. Some of these features do not have a standardized OpenAPI specification yet and they are being described and implemented as proprietary solutions. However, we included detailed descriptions of some messages that are being standardized and we have defined some example methods from those messages. The following request and response bodies are based on the information flow given in 3GPP standards 23.434 [5], TS 24.544 [11], 24.548 [12] and 24.545 [7]. The parameters are adapted and merged with the corresponding

yaml files provided in 3GPP TS 29.549 [6] and data types provided in 3GPP TS 23.434 [5] to give the exact parameter names and a consistent overview. For all methods, it holds:

- (Requester) identity of the user to verify if the user is authorized.
- Result corresponds to the HTTP status given as response.

The HTTP commands and paths are taken from 3GPP TS 29.549 [6].

Provisioning and Onboarding Devices

Process: Provision and onboard a device

This method is not yet defined in standards.

Process: Get a list of devices

This method is not yet defined in standards.

Process: Delete a device

This method is not yet defined in standards.

Getting a list of devices

Process: Get a list of devices

This method is not yet defined in standards. Therefore, we provide the method of getting a list of devices in a group in the next process.

Process: Get a list of device group members

GET {{apiRoot}}/ss-gm/v1/group-documents/{valGroupId}

The method to request a list of devices in the 5G network from the "Device Provisioning and Onboarding service" is not yet defined in 3GPP standards. However, the method to request a list of members in a device group from the "Device Group Management service" exists. Here, the "/ss_gm/v1" describes the URL to be used to send requests to the "SEAL Service for Group Management (ss_gm)", the "/group-documents" describes the path required to get the list of devices in a group and the "{valGroupId}" describes the path and unique ID of the created group that needs to be provided to request the information on group members for that specific group (see 3GPP TS 23.434 [5] and TS 29.549 [6]).

Query parameters:

Name	Type	Description
Identity	String	The identity of the VAL user or VAL UE performing the query.
valGroupId (groupDocId)	String	The identity of the VAL group to be queried.
group-members (Query type)	Boolean	When set to true indicates the group management server to send the members list information of the VAL group.
group-configuration (Query Type)	Boolean	When set to true indicates the group management server to send the group configuration information of the VAL group.

Response body:

Name	Type	Description
valGroupId (groupDocId)	String	The identity of the VAL group to be queried.
group-members (Query type)	Boolean	When set to true indicates the group management server to send the members list information of the VAL group.
Members (Query result)	Array	The list of VAL User IDs or VAL UE IDs, which are members of the VAL group. Given when group-members are set to true.
group-configuration (Query type)	Boolean	When set to true indicates the group management server to send the group configuration information of the VAL group.
valGrpConf (Query result)	String	Configuration data for the VAL group. Given when group-configuration is set to true.
Result	String	Indicates the success or failure for the operation

Creating, modifying and removing device groups

Process: Group creation

POST {{apiRoot}}/ss-gm/v1/group-documents}

This method allows to forward the requests to the appropriate 5G network functions and perform the creation of a group requested by the IoT-NGIN 5G resource management API, indirectly by the user, in the 5G network. Here, the “/ss_gm/v1” describes the URL to be used to send requests to the “SEAL Service for Group Management (ss_gm)”, whereas the “/group-documents” describes the path required to be used in order to perform the group creation operation (see 3GPP TS 23.434 [5], TS 29.549 [6] and TS 24.544 [11]).

Request body:

Name	Type	Description
Requester Identity	String	The identity of the group management client performing the request.
Members (Identity list)	Array	List of VAL user IDs or VAL UE IDs that are part of the group to be created corresponding to the list of the configured services
valServiceIds (VAL service ID list (see NOTE 1))	Array	List of VAL services whose service communications are to be enabled on the group. Optional.
VAL service specific information (NOTE 2)	String	Placeholder for VAL service specific information. Optional.

groupName	String	A human readable name of the VAL group.
Category	String	Indicates the category of the group, e.g. "normal" or "location-based".
memberDetails	Array	List of detailed member information for each member of the VAL group including memberId and memberType.

NOTE 1: This information element shall be included in the message for creating a group configured for multiple VAL services.

NOTE 2: The details of this information element are specified in VAL service specific specification and are out of scope of the present document.

Response body:

Name	Type	Description
valGroupId (groupDocId)	String	This is VAL group identity (VAL group ID) as per TS 23.434, which is a unique identifier within the VAL service that represents a VAL group, set of VAL users or VAL UEs according to the VAL service.
Result	String	Indicates the success or failure for the operation

Process: Group membership update

PUT {{apiRoot}}/ss-gm/v1/group-documents/{valGroupId}

This method allows to change the group membership by adding or deleting a device. The request is forwarded to the appropriate 5G network functions requested by the IoT-NGIN 5G resource management API, indirectly by the user, in the 5G network. Here, the "/ss-gm/v1" describes the URL to be used to send requests to the "SEAL Service for Group Management (ss-gm)", the "/group-documents" describes the path required to be used in order to get the list of devices and the "/groupDocId" describes the path and unique ID of the created group that needs to be provided to request the information for that specific group (see 3GPP TS 23.434 [5], TS 29.549 [6] and TS 24.544 [11]).

Query parameters:

Name	Type	Description
valGroupId (groupDocId)	String	Identity of the VAL group

Request body:

Name	Type	Description
Requester Identity	String	The identity of the group management client performing the request.
valGroupId	String	Identity of the VAL group

Members (Identity)	String	List of identities of the VAL users and VAL UEs affected by this operation
Operations	String	Add to or delete from the group
VAL service specific information (NOTE)	String	Placeholder for VAL service specific information. Optional
memberDetails	Array	List of detailed member information for each member of the VAL group including memberId and memberType.

NOTE: The details of this information element are specified in VAL service specific specification and are out of scope of the present document.

Response body:

Name	Type	Description
valGroupId (groupDocId)	String	The identity of the VAL group to be queried.
Result	String	Indicates the success or failure for the operation

Process: Get a list of groups and members in the groups

GET {{apiRoot}}/ss-gm/v1/group-documents/{valGroupId}

This method allows to forward the requests related to the listing the groups and listing connected devices in a group to the appropriate 5G network functions requested by the IoT-NGIN 5G resource management API, indirectly by the user, in the 5G network. Here, the “/ss_gm/v1” describes the URL to be used to send requests to the “SEAL Service for Group Management (ss_gm)”, the “/group-documents” describes the path required to be used in order to get the list of devices and the “/{valGroupId}” describes the path and unique ID of the created group that needs to be provided to request the information for that specific group(see 3GPP TS 23.434 [5], TS 29.549 [6] and TS 24.544 [11]).

Query parameters:

Name	Type	Description
Identity	String	The identity of the VAL user or VAL UE performing the query.
valGroupId (groupDocId)	String	The identity of the VAL group to be queried.
group-members (Query type)	Boolean	When set to true indicates the group management server to send the members list information of the VAL group.
group-configuration (Query Type)	Boolean	When set to true indicates the group management server to send the group configuration information of the VAL group.

Response body:

Name	Type	Description
valGroupId (groupDocId)	String	The identity of the VAL group to be queried.
group-members (Query type)	Boolean	When set to true indicates the group management server to send the members list information of the VAL group.
Members (Query result)	Array	The list of VAL User IDs or VAL UE IDs, which are members of the VAL group. Given when group-members are set to true.
group-configuration (Query type)	Boolean	When set to true indicates the group management server to send the group configuration information of the VAL group.
valGrpConf (Query result)	String	Configuration data for the VAL group. Given when group-configuration is set to true.
Result	String	Indicates the success or failure for the operation

Process: Delete a group

DELETE {{apiRoot}}/ss-gm/v1/group-documents/{valGroupId}

This method allows to delete a group. The request is forwarded to the appropriate 5G network functions requested by the IoT-NGIN 5G resource management API, indirectly by the user, in the 5G network. Here, the “/ss_gm/v1” describes the URL to be used to send requests to the “SEAL Service for Group Management (ss_gm)”, the “/group-documents” describes the path required to be used in order to get the list of devices and the “/{valGroupId}” describes the path and unique ID of the created group that needs to be provided to request the information for that specific group (see 3GPP TS 23.434 [5], TS 29.549 [6] and TS 24.544 [11]).

Query parameters:

Name	Type	Description
Requester Identity	String	The identity of the group management client performing the request.
valGroupId (groupDocId)	String	Identity of the VAL group

Response body:

Name	Type	Description
valGroupId (groupDocId)	String	Identity of the VAL group requested to be deleted

groupDocId	String	Indicates success (group no longer exists), or failure (group deletion did not occur, e.g. authorization failure).
------------	--------	--

Monitoring the quality of the communication links of devices

Process: Subscribe to device connectivity monitoring

POST {{apiRoot}}/ss_events/v1/subscriptions}

Here, the “/ss_events/v1” describes the URL to be used to send requests to the “SEAL Service for Events Management (ss_events)”, whereas the “/subscriptions” describes the path required to be used in order to perform the event subscription operation. Depending on the parameters related to the communication link monitoring included in the request body, the notifications body provides information relevant for the communication link quality (see 3GPP TS 23.434 [5], TS 29.549 [6] and TS 24.548 [12]).

Request body:

Name	Type	Description
idnts (Identities list (Note 1))	Array	Identities of the VAL Users or UEs whose events monitoring is requested. Either VAL users/UEs or a VAL group identifying VAL UEs shall be present.
valGrpId (VAL group ID (Note 1))	String	Identity of the VAL group of the target UEs whose events monitoring is requested. Either VAL users/UEs or a VAL group identifying VAL UEs shall be present.
valSvcId (VAL service ID)	String	Identity of the VAL service.
valCnds (Validity conditions)	Array	The temporal and/or spatial conditions applied for the events to be considered as valid.
profId (Monitoring profile ID (Note 2))	String	The monitoring profile ID identifying a list of monitoring and/or analytics events. Either event details or monitoring profile ID shall be present in the subscription request. The monitoring profile ID shall present in the subscription response when event details are provided in the subscription request.
evntDets (Event Details (Note 2))	Array	List of monitoring and/or analytics events that the VAL server is interested in. Either event details or monitoring profile ID shall be present in the subscription request. The monitoring profile ID shall present in the subscription response when event details are provided in the subscription request.

eventId	String	Should be NRM_EventMonitor: Monitoring and analytic events related to VAL UEs, users or VAL group from the Network Resource Management Server.
eventReq	Object	Indicates reporting information.
notificationDestination	String	The URI which is notified about changes.

Response body:

Name	Type	Description
Subscription status	String	It indicates the subscription result
subscriberId	String	String identifying the subscriber of the event.
eventId	String	Should be NRM_EventMonitor: Monitoring and analytic events related to VAL UEs, users or VAL group from the Network Resource Management Server.
eventReq	Object	Indicates reporting information.
notificationDestination	String	The URI which is notified about changes.
profId (Monitoring profile ID (Note 2))	String	The monitoring profile ID identifying a list of monitoring and/or analytics events. Either event details or monitoring profile ID shall be present in the subscription request. The monitoring profile ID shall present in the subscription response when event details are provided in the subscription request. Optional.

Callback URI {.../notificationDestination}

event notification is "NRM_MONITOR_UE_USER_EVENTS"

Notification body:

Name	Type	Description
subscriptionId	String	Identifier of the subscription resource.
tgtUe (identity)	String	VAL UE for which the events are related.
evnts (events)	String	List of monitoring and analytics events related to VAL UE.
Timestamp	String	The timestamp for the monitoring and analytics events

NOTE 1: For identifying the target UE(s), either a list of VAL users/UEs or a group of VAL UEs shall be provided.

NOTE 2: Either Event Details or Monitoring profile ID is present.

Process: Unsubscribe to device connectivity monitoring

This method is not defined in 3GPP TS 23.434 [5] so that no request or response body can be given. However, the path to unsubscribe of SEAL events is standardized in 3GPP TS 29.549 [6].

DELETE {{apiRoot}} /ss-events/v1/subscriptions/{subscriptionId}

Path parameter:

Name	Type	Description
subscriptionId	String	Identifier of the subscription resource.

Defining and changing QoS parameters of individual device connections

Process: Change QoS parameters of a device

The method for changing the QoS parameters has not yet defined in standards, however example messages defined in 3GPP TS 23.434 [5] are given in here.

Request body:

Name	Type	Description
List of VAL UEs	Array	List of VAL UEs for whom the end-to-end QoS management occurs
VAL UE/user ID	Array	Identity of the VAL UE
IP address	String	IP address of the VAL UE
VAL service ID	String	The VAL service identity for whom the end-to-end QoS management occurs. Optional.
End-to-end QoS requirements	String	<p>The application QoS requirements / KPIs (latency, error rate, ...) for the end-to-end session.</p> <p>This may optionally include information which will support the NRM server to identify the per session QoS requirements (e.g., a flag indicating the use of HD video for assisting the end-to-end session, a video resolution/encoding required for the HD video).</p>
Service area	String	The area where the QoS management request applies. This can be geographical area, or topological area. Optional.

Time validity	String	The time of validity of the requirement. Optional.
---------------	--------	--

Response body:

Name	Type	Description
Result	String	The positive or negative result of the end-to-end QoS management request.
QoS report configuration	String	The configuration of the NRM client's report triggering by NRM server (e.g., setting thresholds for reporting a QoS downgrade / notifications based on channel loss great than threshold value)

Getting location information of devices

Process: Subscribe to the location information of a device

POST {{apiRoot}}/ss_events/v1/subscriptions}

Here, the “/ss_events/v1” describes the URL to be used to send requests to the “SEAL Service for Events Management (ss_events)”, whereas the “/subscriptions” describes the path required to be used in order to perform the event subscription operation. Depending on the parameters related to the location information included in the request body, the notifications body provides information relevant for the location (see 3GPP TS 23.434 [5], TS 29.549 [6] and TS 24.545 [7]).

Request body:

Name	Type	Description
Identity	String	Identity of the requesting VAL server/VAL user or VAL UE
subscriberId	String	String identifying the subscriber of the event.
valTgtUes (Identities list)	Array	VAL User IDs or VAL UE IDs that the event subscriber wants to know in the interested event. Optional.
valSvcId (VAL service ID)	Array	Each element of the array represents the VAL User / UE IDs of a VAL service that the event subscriber wants to know in the interested event. Optional.
eventReq (Time between consecutive reports)	Object	Represents the type of reporting that the subscription requires.
eventId	String	Should be LM_LOCATION_INFO_CHANGE: Events related to the location information of VAL Users or VAL UEs from the Location Management Server.

notificationDestination	String	The URI which is notified about changes.
expiry-time	String	Element specifying the time when the VAL server wants to receive the current status and later notification

Response body:

Name	Type	Description
Identity	String	Identity of the requesting VAL server/VAL user or VAL UE
eventId	String	Should be LM_LOCATION_INFO_CHANGE: Events related to the location information of VAL Users or VAL UEs from the Location Management Server.
subscriberId	String	String identifying the subscriber of the event.
eventReq (Time between consecutive reports)	Object	Represents the type of reporting that the subscription requires.
notificationDestination	String	The URI which is notified about changes.
Subscription status	String	Result of the subscription. It indicates the subscription result.
expiry-time	String	Element specifying the time when the VAL server wants to receive the current status and later notification

Callback URI {.../notificationDestination}

Event notification is "LM_LOCATION_INFO_CHANGE"

Notification body:

Name	Type	Description
Identity	String	Identity of the VAL user or VAL UE subscribed to location of another VAL user or VAL UE (NOTE)
valTgtUe (Identities list)	Array	VAL User ID or UE ID that the event subscriber wants to know in the interested event.
ImInfos (Location Information)	Object	Represents the location information for a VAL User ID or a VAL UE ID.
Timestamp	String	Timestamp of the location report. Optional.
subscriptionId	String	Identifier of the subscription resource.

eventId (Triggering event)	String	Should be LM_LOCATION_INFO_CHANGE: Events related to the location information of VAL Users or VAL UEs from the Location Management Server.
----------------------------	--------	--

NOTE: This is only used for location management server sends location information notification to the VAL user or VAL UE who has subscribed the location.

Process: Unsubscribe to the location information of a device

This method is not defined in standards in TS 23.434, therefore no request or response body can be given. However, the path to unsubscribe of SEAL events is standardized in 3GPP TS 29.549 [6].

DELETE {{apiRoot}}/ss-events/v1/subscriptions/{subscriptionId}

Path parameter:

Name	Type	Description
subscriptionId	String	Identifier of the subscription resource.
expiry-time	String	Set to zero.

Annex 3 Sequence diagrams of the 5G device management API

In this chapter, we will explain the 5G device management API features that can be used in IoT-NGIN use cases. For each feature, we have drawn a sequence diagram and explain the procedures to be followed. In these sequence diagrams, we provided an end-to-end lifecycle of the specific API operations initiated by the authorized user/device and performed in the 5G network. Therefore, these sequence diagrams include both the IoT-NGIN 5G resource management API as an Adapter and the corresponding 5G device management API services.

Provisioning and Onboarding Devices

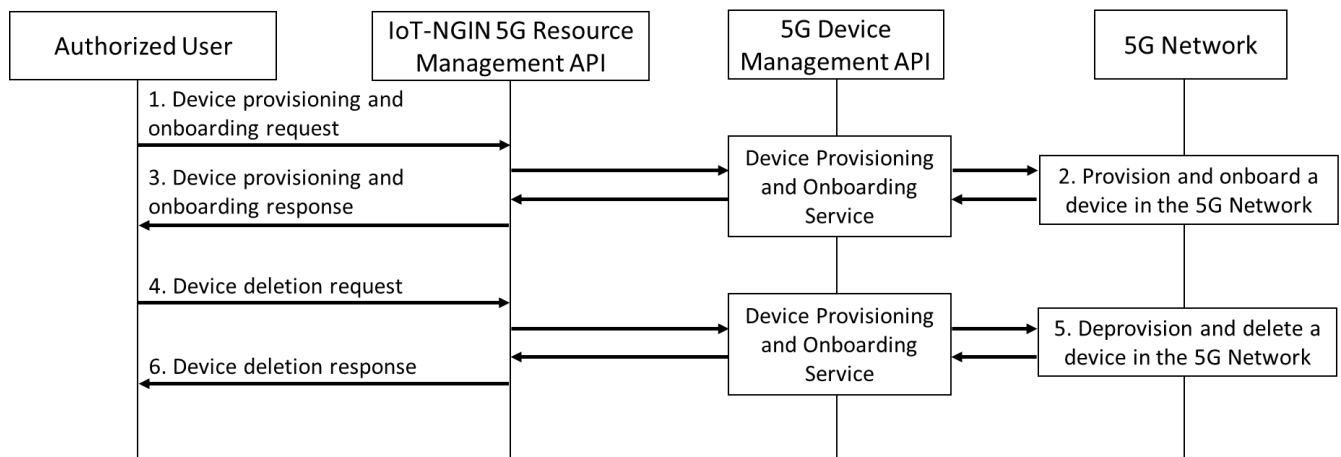
The procedure has not been standardised yet. The implementation is based on 5G ACIA paper Annex B.

Actors:

- A User that is authorized to use the 5G device management API
- A 5G-capable device or a device connected to a 5G user equipment
- IoT-NGIN 5G resource management API
- 5G network
- Device provisioning and onboarding service of the 5G device management API

Process:

1. The authorized user registers to device onboarding and provisioning service to provision and onboard a device to the 5G network.
2. The device provisioning and onboarding service forwards the request to the 5G network.
3. The 5G network onboards the device.
4. The device onboarding and provisioning service sends a response to the authorized user indicating success or failure.
5. The authorized user registers to device onboarding and provisioning service to delete a device from the 5G network.
6. The device provisioning and onboarding service forwards the request to the 5G network.
7. The 5G network deprovisions the device and removes it from the network.
8. The device onboarding and provisioning service sends a response to the authorized user indicating success or failure.



Getting a list of devices

This procedure is standardized in 3GPP TS 23.434 [5] and modified here to include IoT-NGIN 5G resource management API.

Actors:

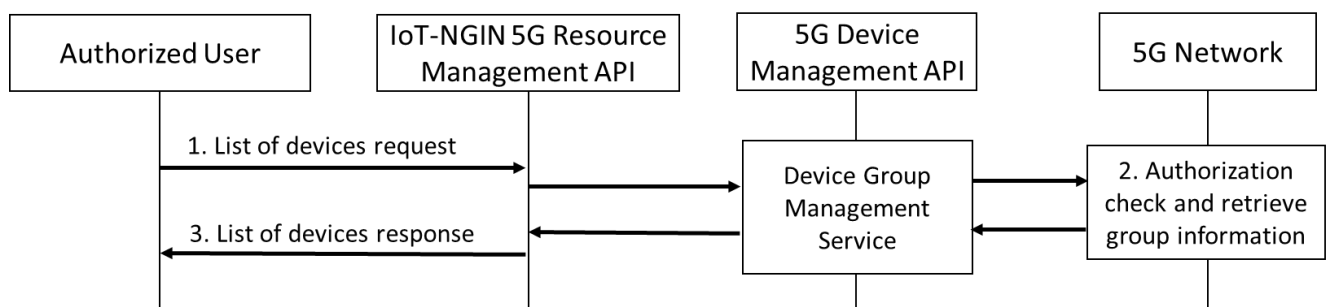
- A User that is authorized to use the 5G device management API
- A 5G-capable device or a device connected to a 5G user equipment
- IoT-NGIN 5G resource management API
- 5G network
- Device group management service of the 5G device management API

Precondition:

List of groups to which a UE or User belongs to is known to the 5G network for each of the UE or User.

Process:

1. The authorized user requests the list of devices from the list of devices service by sending the unique group identification.
2. The list of devices service checks whether the user/UE is authorized to perform the request. If authorized, then the group management server retrieves the requested members of the group.
3. The list of devices service sends a group information query response including the retrieved group information to the authorized user.



Creating, modifying and removing device groups

This procedure is standardized in 3GPP TS 23.434 [5] and modified here to include IoT-NGIN 5G resource management API.

Actors:

- A User that is authorized to use the 5G device management API
- A 5G-capable device or a device connected to a 5G user equipment
- IoT-NGIN 5G resource management API
- 5G network
- Device group management service of the 5G device management API

Preconditions:

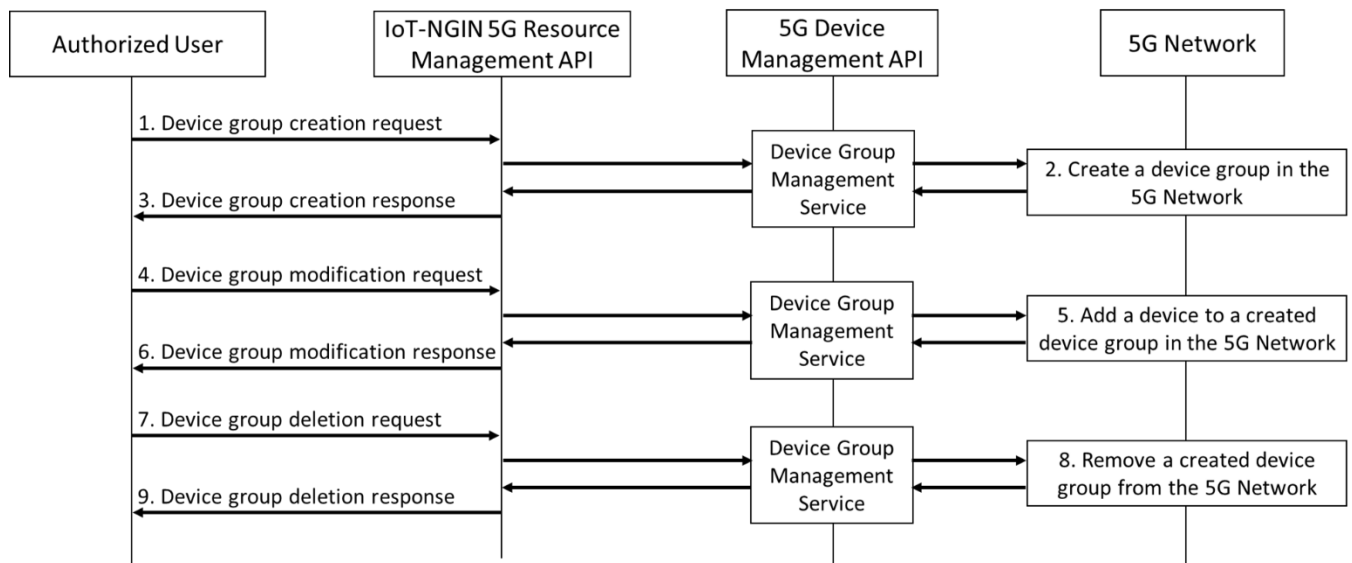
The authorized user is aware of the users' identities which will be combined to form the group, and which will be needed later to delete the group.

Process:

1. The authorized user sends a request to the device group management service to create a device group.
2. The device group management service provides the device group data provided by the user to the 5G network.
3. The device group management service sends a reply to the user indicating the success or failure of the device group creation operation. In case of a success, the response will contain a unique device group identifier.
4. The authorized user sends a request to the device group management service to modify a device group.
5. In case the request is about adding a device to a group, the user request will contain the device identifier and the unique device group identifier of the group to which the device will be added as a member. The device group management service forwards this modification request to the 5G network.
6. The device group management service sends a reply to the user indicating the success or failure of the operation of adding a device to a group.
7. The authorized user sends a request to the device group management service to delete a device group.
8. The device group management service forwards this deletion request to the 5G network.
9. The device group management service sends a reply to the user indicating the success or failure of the operation of deleting the device group from the 5G network.

Result:

- The device group is defined and created in the 5G network.
- The device is added as a member to the device group.
- The device group is removed from the 5G network.



Monitoring the quality of the communication links of devices

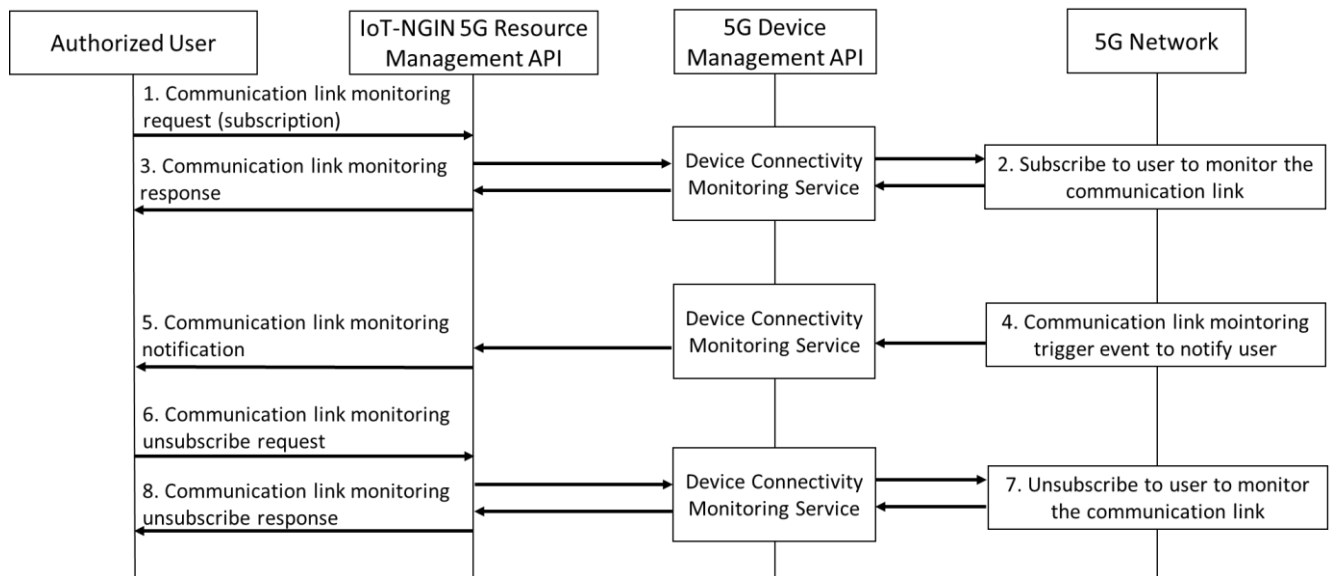
This procedure is standardized in 3GPP TS 23.434 [5] and modified here to include IoT-NGIN 5G resource management API.

Actors:

- A User that is authorized to use the 5G device management API
- A 5G-capable device or a device connected to a 5G user equipment
- IoT-NGIN 5G resource management API
- 5G network
- Device connectivity monitoring service of the 5G device management API

Process:

1. The authorized user sends a link monitoring subscription request to the device connectivity monitoring service to subscribe to link information of the device.
2. The device connectivity monitoring service subscribes for communication link information from 5G network for the device.
3. The device connectivity monitoring service replies with a link monitoring subscription response indicating the subscription status and if immediate reporting was requested, the link information of the device.
4. Based on the configurations, e.g., subscription, periodical location information timer, device connectivity monitoring service is triggered to report the latest communication link information to the authorized user. It receives the link information of the device from 5G network.
5. The device connectivity monitoring service sends the link monitoring report including the latest communication link information of the device to the authorized user that has previously configured.
6. The authorized user sends a unsubscribe request if the subscription is still valid and not terminated e.g., by the maximal duration of subscription.
7. The device connectivity monitoring service unsubscribes for communication link information from 5G network for the device.
8. The device connectivity monitoring service replies with a link monitoring unsubscribe response indicating the subscription status.



Defining and changing QoS parameters of individual device connections

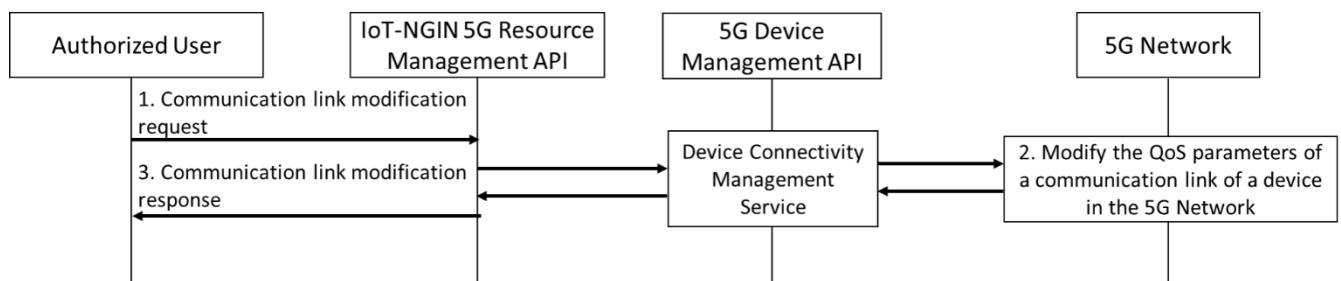
This procedure is standardized in 3GPP TS 23.434 [5] and modified here to include IoT-NGIN 5G resource management API.

Actors:

- A User that is authorized to use the 5G device management API
- A 5G-capable device or a device connected to a 5G user equipment
- IoT-NGIN 5G resource management API
- 5G network
- Device connectivity management service of the 5G device management API

Process:

1. The authorized user sends to the device connectivity management service an end-to-end QoS management request for managing the QoS.
2. The device connectivity management service provides the QoS parameter data provided by the user to the 5G network. The 5G networks modifies the QoS parameters of a communication link of the requested device.
3. The service sends to the user a QoS management response with a positive or negative acknowledgement of the request.



Getting location information of devices

This procedure is standardized in 3GPP TS 23.434 [5] and modified here to include IoT-NGIN 5G resource management API.

Actors:

D2.2 – Enhancing IoT Underlying Technology

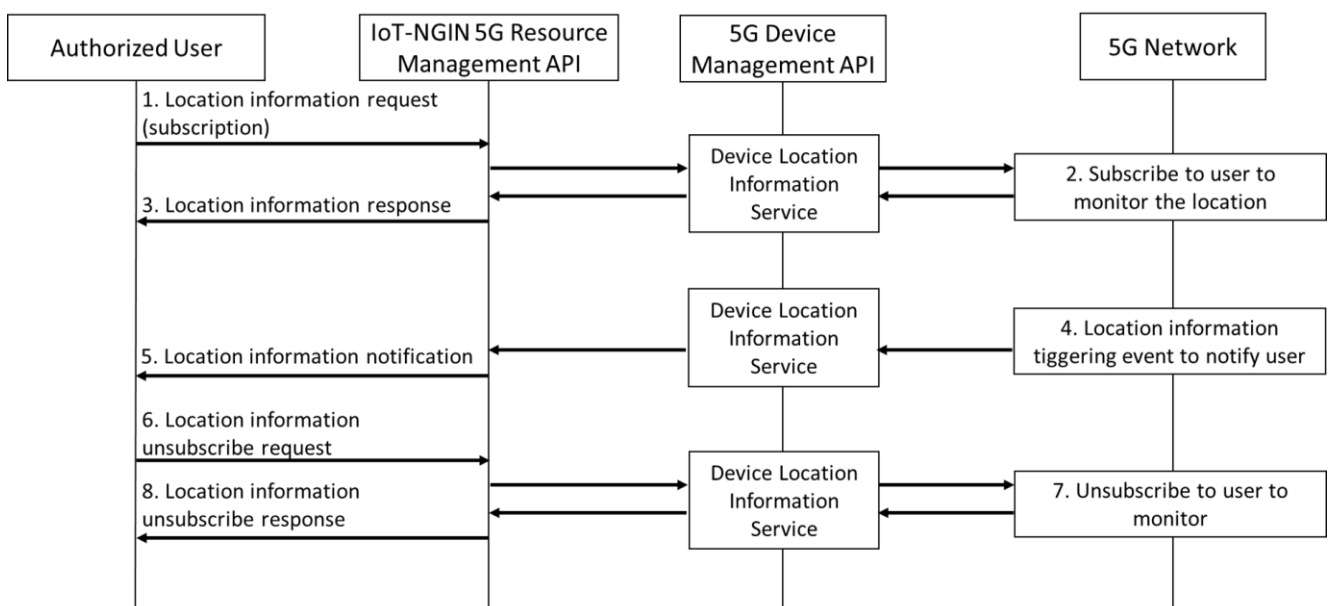
- A User that is authorized to use the 5G device management API
- A 5G-capable device or a device connected to a 5G user equipment
- IoT-NGIN 5G resource management API
- 5G network
- Device location information service of the 5G device management API

Preconditions:

- The 5G network has a jurisdiction over a geographical area for which the device location information service is configured to operate.
- The location information of devices is obtained from the 5G network.

Process:

1. The authorized user sends a location information subscription request to the device location service to subscribe location information of the device.
2. The device location service subscribes for device location information from 5G network.
3. The device location service replies with a location information subscription response indicating the subscription status and if immediate reporting was requested, the location information of the device.
4. Based on the configurations, e.g., subscription, periodical location information timer, device location service is triggered to report the latest device location information to authorized user.
5. The device location service sends the location information report including the latest location information of the device to the authorized user that has previously configured.
6. The authorized user sends a location information unsubscribe request to the device location service if the subscription is still valid. The subscription may be terminated earlier e.g., when the maximum duration of subscription is overstepped.
7. The device location service unsubscribes for device location information from 5G network.
8. The device location service replies with a location information unsubscribe response indicating the subscription status.



Annex 4 Implementation description of the 5G device management API

This chapter describes implementation in 5G network of the 5G device management API features listed in 5.3.1.

Provisioning and onboarding devices

This feature is not yet defined in standards, therefore the implementation of this feature in the 5G network cannot be provided at this stage. During device provisioning, standard 5G security checks take place as described in Chapter 5.3.4.

Getting a list of connected devices

The feature 'Getting list of connected devices', when devices do not belong to a group, is not yet defined in 3GPP standards. However, getting list of members of a device group is part of to the SEAL Group Management Service as defined in the 3GPP SEAL standard 23.434 [5]. In the project, we have used the implementation of the SEAL standard which is described in the feature 'Creating, modifying and removing devices groups' below.

Creating, modifying and removing device groups

The feature 'Creating, modifying and removing device groups' enables IoT-NGIN applications to conduct different operations towards 5G network such as querying group information, modifying the device group membership, creating and deleting the device group. These operations are described in the SEAL standard 23.434 clause 10.3 [5]. The operations can be executed only by authorised users or applications. Policies check applies where applicable, e.g., maximum limit of the total number of the group members.

For group management operations applicable to a 5G virtual network group, the external application interacts with the Network Exposure Function (NEF) of the underlying 5G network using the dynamic 5G Virtual Network group management procedures exposed by the NEF via the N33 reference point, as specified in TS 23.501 [13] and in TS 23.502 [14]. 5G Virtual Network group represents a set of devices using private IP and/or non-IP type communications.

5G network supports dynamic management of the 5G Virtual Network group identification and membership, and the 5G Virtual Network group data as described in the 3GPP 23.501 standard, Clause 5.29.2 [13]. Furthermore, 5G Virtual Network group is depicted by the following:

- 5G Virtual Network group identities: External Group ID and Internal Group ID are used to identify the 5G Virtual Network group.
- 5G Virtual Network group membership: The 5G Virtual Network group members are uniquely identified by GPSI.
- 5G Virtual Network group data such as protocol data unit session type and data network name. Data network name is equivalent to an Access Point Name (APN) in LTE network.

In order to support dynamic management of 5G Virtual Network group identification and membership, and 5G Virtual Network group data, the NEF exposes a set of services to

manage 5G Virtual Network groups and members such as create, delete and modify groups and members.

External Group ID is used for identification of a 5G Virtual Network group. The NEF stores the External Group ID into Unified Data Management (UDM). The UDM maps the External Group ID to Internal Group ID. If the new 5G Virtual Network group is created, the UDM will allocate an Internal Group ID. The external application can update the 5G Virtual Network group members identities at any time after the initial provisioning.

Monitoring the quality of the communication links of devices

The feature 'Monitoring the quality of the communication links of devices' is partly standardized. Implementation of this feature in the project is based on the description provided in the 5G-ACIA paper, Annex B [4] and the SEAL service 'Network Resources Management' described in the 3GPP standard 23.434, Clause 14.3.2.17-19 [5]. The implementation description of the SEAL service 'Network Resources Management' is provided in the feature 'Defining and changing QoS parameters of individual device connections' description below.

Defining and changing QoS parameters of individual device connections

The feature 'Defining and changing QoS parameters of individual device connections' enables the external application to set the Quality of Service (QoS) of the connection towards the device. The feature is standardised and described in the SEAL standard 23.434, Clause 14.3.2.13-14 [5]. The feature is a part of the SEAL service 'Network Resource Management'. The IoT-NGIN application communicates with the NEF function in the 5G network via N33 reference point to set the QoS of the connection to the device. The NEF supports external exposure of 5G network functions capabilities. External exposure is categorised in different categories. The category that supports QoS handling is the Policy/charging. Setting and update of the connection with required QoS is described in detail in the 3GPP standard 23.502, in the clause 4.15.6.6 [14].

Getting location information of devices

In order to gather device location information, the IoT-NGIN application communicates with the Network Exposure Function (NEF) in the 5G network via N33 reference point as described in the 3GPP standard 23.434, Clause 9.2.2 [5]. NEF collects data from the 5G network and provide them to the IoT-NGIN application in secured manner. In purpose of gathering of the device location information, the Monitoring Events feature of the 5G network is used as described in the clause 4.15.3.1 of the 3GPP standard 23.502 [14]. Specifically, the Location Reporting event applies in the case of device location information gathering. The following network functions detect the Location Reporting event: Access and Mobility Management Function (AMF) and Gateway Mobile Location Centre (GMLC).

Annex 5 Sequence diagrams for the IoT-NGIN 5G resource management API

Annex 5.1 Start and stop service

The sequence diagram depicted in Figure 9-1 shows the start and stop procedure for a service running in a cloud/edge-cloud. Call 1 to 8 define the sequence for starting a service, call 11 to 18 the stop service sequence. In between start and stop service procedures, the actual service is running. The depicted service instance is a generic service in which some processing is done on the data sent from the IoT device, and then a result is sent back to the IoT device.

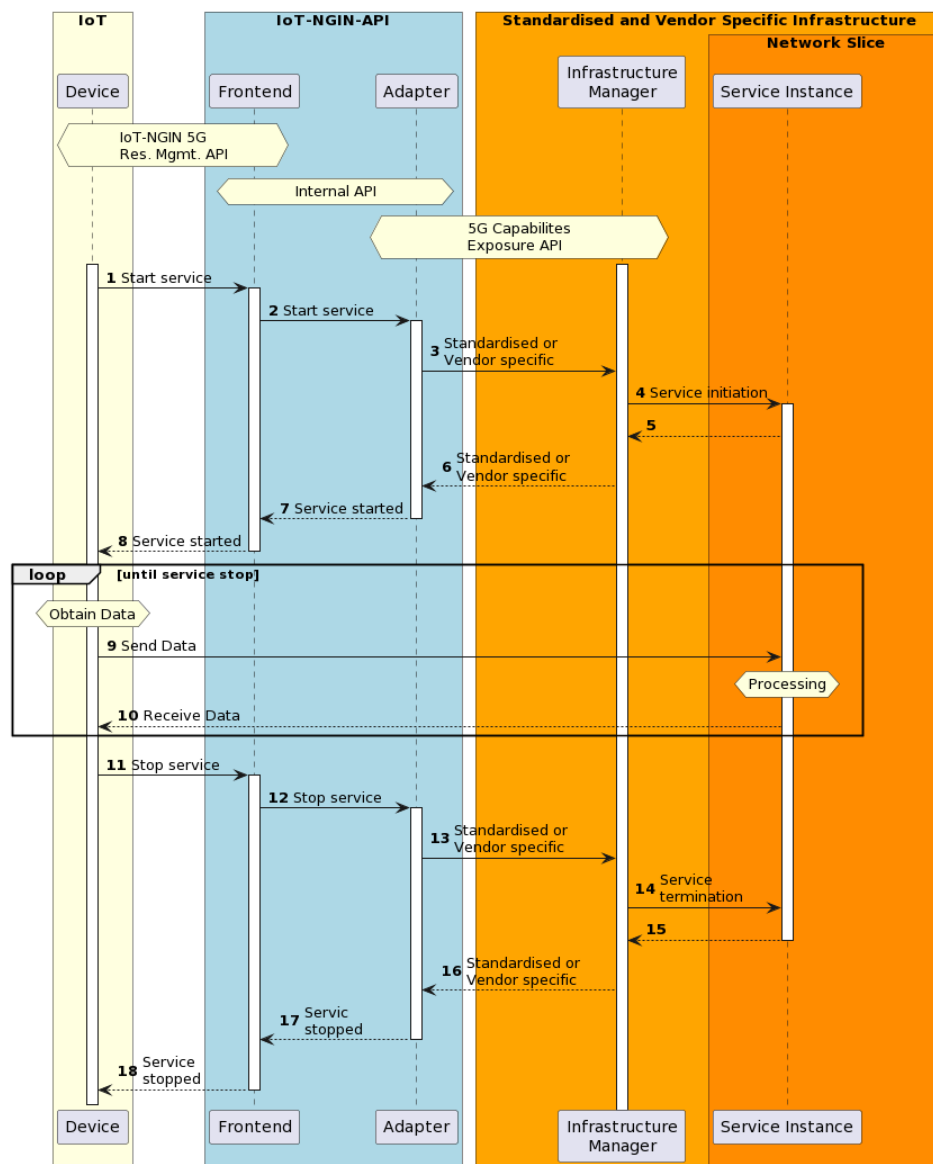


Figure 9-1 Start and stop sequence

Annex 5.2 Resource allocation

The sequence diagram depicted in Figure 9-2 shows the case of resource modification. This could be necessary e.g., if the algorithm running on the service instance changes. This change might result in requiring more CPU cores or RAM by the service. To solve that, two approaches are possible: a) the required resources could be reallocated dynamically, b) a new network slice with more reserved resources is created and a new version of the service on top of this slice is instantiated. The first case is called *dynamic resource allocation* and case two is called *static resource allocation*. Which of the two cases is used depends on the underlying infrastructure and is part of the 5G capabilities exposure described in Chapter 5.3.

D2.2 – Enhancing IoT Underlying Technology

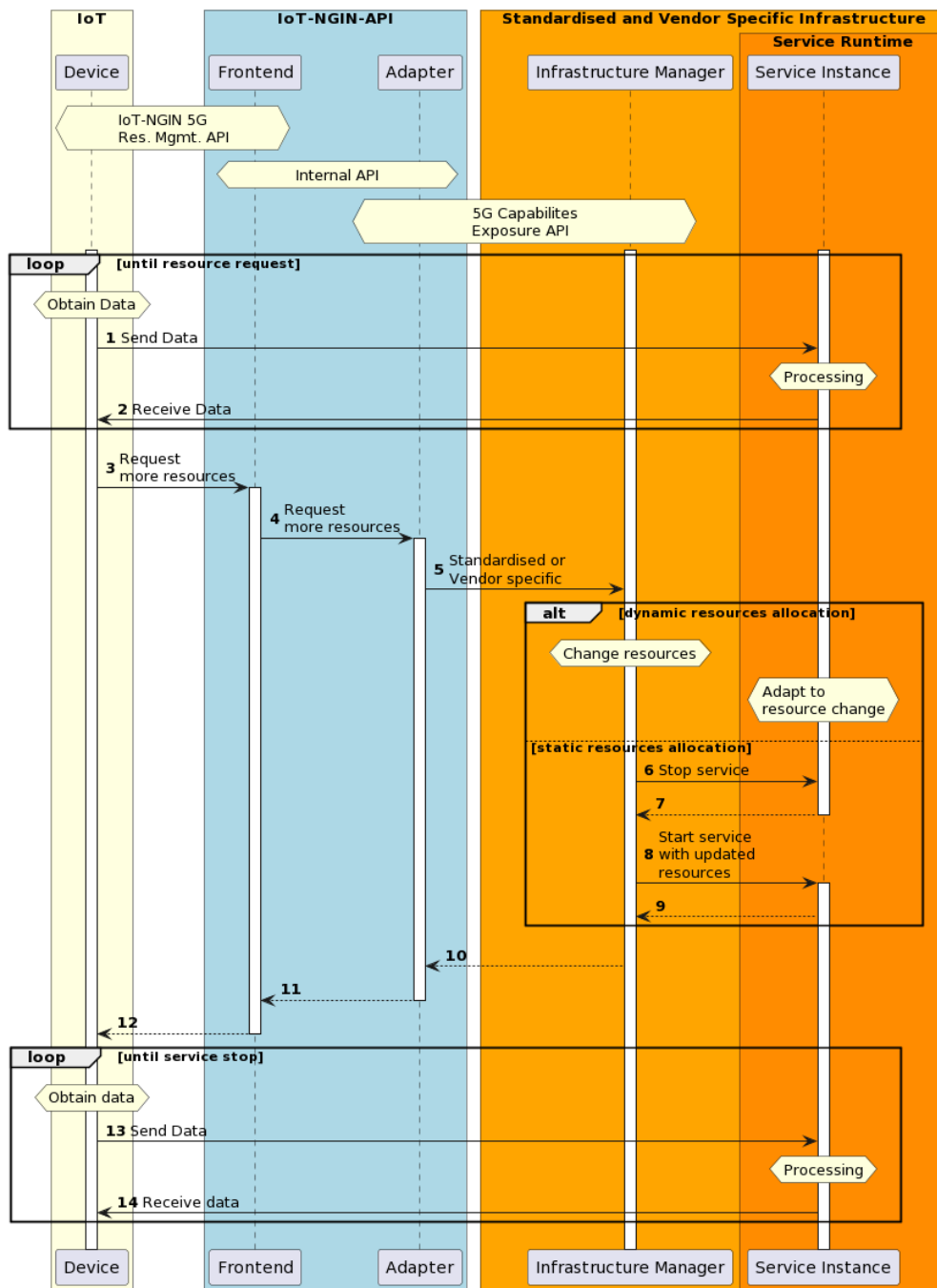


Figure 9-2 Resource allocation sequence

Annex 6 Security features of 5G today

Annex 6.1 Introduction to 5G security in IoT-NGIN

In IoT-NGIN, we are developing a novel approach to securing edge cloud infrastructure which relates directly to the security needs of all of the IoT-NGIN Living Lab use cases. In the development of our IoT-NGIN 5G resource management API, we consider the security implications of the functionality. As our focus is on developing ease-of-use functionality for 5G, we build on the high level of security inherent in the 3GPP standards for 5G. The 3GPP standards for 5G include a range of enhancements compared to those of LTE networks. In this annex, we provide an overview of the security features standardised for 5G in 3GPP for readers interested in understanding the security context of 5G networks.

Annex 6.2 Introduction to 5G security features

Security was and is critical for all generations of mobile networks and they are perceived in that way. Security is improved in 5G network comparing to 4G. The most important security enhancements are summarized in the following.

First, the security mechanisms can be divided into two groups. On the one hand, **Network access security mechanisms**¹⁵ provide users secure access to services through the device (typically a phone) and protect against attacks on the air interface between the device and the 5G network e.g., by using

- flexible authentication framework enabling different types of credentials besides the SIM card (primary and secondary authentication)
- IMSI encryption and enhanced subscriber privacy features
- general framework for detecting false base stations
- encryption and integrity protection of the user plane (UP) between the device and the gNB (5G base stations).

On the other hand, **Network domain security mechanisms**¹⁶ enable nodes to exchange securely signaling and user data for example between radio nodes and core network nodes for instance by using

- service-based architecture (SBA) supporting authorization framework as well as authentication and transport protection between network functions
- improved interconnect security (i.e., security between different operator networks) consisting of three building blocks:
 - o security edge protection proxy (SEPP): All signaling traffic across operator networks is expected to transit through these security proxies.
 - o authentication between SEPPs: Traffic coming from the interconnect is filtered effectively.

¹⁵ <https://www.ericsson.com/en/blog/2019/7/3gpp-5g-security-overview>

¹⁶ <https://www.ericsson.com/en/blog/2019/7/3gpp-5g-security-overview>

- o a new application layer security solution on the N32 interface between the SEPPs: Sensitive data attributes are protected while still allowing mediation services throughout the interconnect.

The basis of all 5G security is given by **standardisation**. Some parts are mandatory to implement and use e.g., the mutual authentication, which was defined already for 3G, making the network authenticate the device and the device authenticate the network or integrity protection of signaling. SEPP is another example which is mandatory to implement between 5G cores. Some other parts are mandatory or optional to implement and optional to use to allow flexible deployment options. By allowing the vendor and the operator to decide on the level of security, they choose the mechanisms how to reach that level. An example of security that is mandatory to implement but optional to use is network domain security (IPsec, for example) between the nodes of mobile network or IMSI encryption. Since the standards need to be implemented, deployed, and operated by vendors and operators, there comes responsibility. Besides, many other security aspects follow that are not in scope of the standards so that the right balance between standardized and non-standardized aspects is beneficial to allow freedom for innovation and differentiation.¹⁷

A critical security aspect which has no standard rules or guidelines is the **separation of RAN and 5G core** because gNBs terminate the encryption of user data. This means that there is no end-to-end security of the data. End-to-end security can be provided only on application layer, but this is not guaranteed by 3GPP 5G standards and is beyond the control of an operator's 5G network.¹⁸

¹⁷ <https://www.ericsson.com/en/blog/2020/6/security-standards-role-in-5g>

¹⁸ <https://www.ericsson.com/en/reports-and-papers/white-papers/security-in-5g-ran-and-core-deployments>