

---

# OAI Webinar Chapter Two

Walkthrough 3GPP  
Specifications



---

Feb 21, 2022

**Rohan Kharade**

# Agenda

- **Part 1: Overview 3GPP specifications**
  - Introduction, challenges, recommendations
- **Part 2: Implementing procedure/protocols from 3gpp specification**
  - Call flows, UML diagrams, struct/class (IEs)
- **Part 3: Some use cases**
  - Example case of pfcf, gtp1u etc.

---

## **Part 1: Overview 3GPP specifications**

**Introduction, challenges, recommendations**



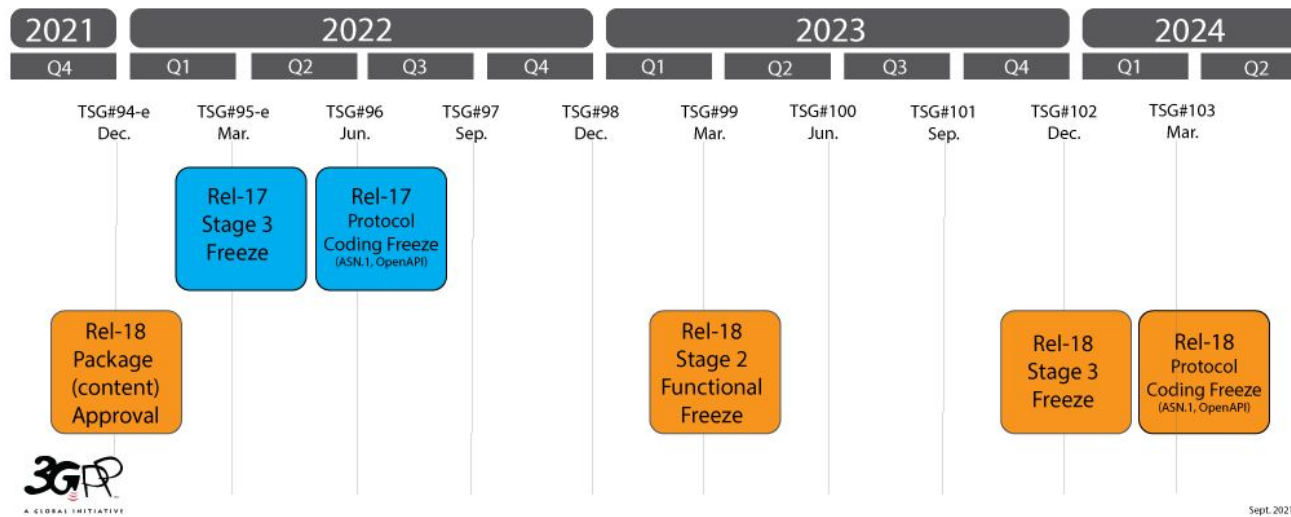
# 3GPP - 3rd Generation Partnership Project

- Project to produce the reports and specifications for 3GPP technologies.
- Seven SDOs worldwide (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC)
- Project Co-ordination Groups-
  - Radio Access Network
  - Service & System Aspects
  - Core Network & Terminals



# 3GPP - 3rd Generation Partnership Project

- Releases -> functionally frozen and ready for implementation



# 3GPP - 3rd Generation Partnership Project

- Organised into organized into 16 specialized Working Groups (WGs)

TSG Structure

Project Co-ordination Group (PCG)		
TSG RAN Radio Access Network	TSG SA Service & System Aspects	TSG CT Core Network & Terminals
RAN WG1 Radio Layer 1 (Physical layer)	SA WG1 Services	CT WG1 User Equipment - Core Network protocols
RAN WG2 Radio layer 2 and Radio layer 3 Radio Resource Control	SA WG2 System Architecture and Services	CT WG3 Interworking with External Networks & Policy and Charging Control
RAN WG3 UTRAN/E-UTRAN/NG-RAN architecture and related network interfaces	SA WG3 Security and Privacy	CT WG4 Core Network Protocols
RAN WG4 Radio Performance and Protocol Aspects	SA WG4 Multimedia Codecs, Systems and Services	CT WG6 Smart Card Application Aspects
RAN WG5 Mobile Terminal Conformance Testing	SA WG5 Management, Orchestration and Charging	
	SA WG6 Application Enablement and Critical Communication Applications	
RAN AH1 RAN ad hoc group on ITU-R		

# 3GPP - 3rd Generation Partnership Project

- Organised into organized into 16 specialized Working Groups (WGs)

overall architecture and service capabilities

specifying terminal interfaces, terminal capabilities

TSG Structure

Project Co-ordination Group (PCG)		
<b>TSG RAN</b> Radio Access Network	<b>TSG SA</b> Service & System Aspects	<b>TSG CT</b> Core Network & Terminals
<b>RAN WG1</b> Radio Layer 1 (Physical layer)	<b>SA WG1</b> Services	<b>CT WG1</b> User Equipment - Core Network protocols
<b>RAN WG2</b> Radio layer 2 and Radio layer 3 Radio Resource Control	<b>SA WG2</b> System Architecture and Services	<b>CT WG3</b> Interworking with External Networks & Policy and Charging Control
<b>RAN WG3</b> UTRAN/E-UTRAN/NG-RAN architecture and related network interfaces	<b>SA WG3</b> Security and Privacy	<b>CT WG4</b> Core Network Protocols
<b>RAN WG4</b> Radio Performance and Protocol Aspects	<b>SA WG4</b> Multimedia Codecs, Systems and Services	<b>CT WG6</b> Smart Card Application Aspects
<b>RAN WG5</b> Mobile Terminal Conformance Testing	<b>SA WG5</b> Management, Orchestration and Charging	
	<b>SA WG6</b> Application Enablement and Critical Communication Applications	
<b>RAN AH1</b> RAN ad hoc group on ITU-R		

---

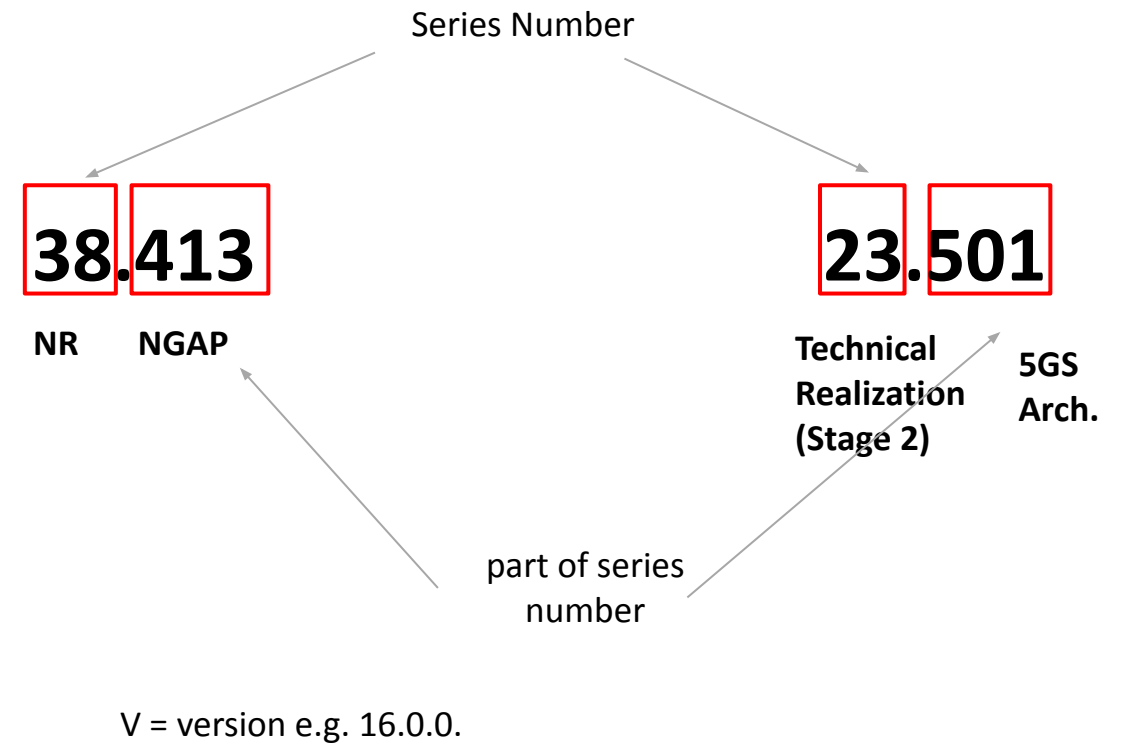
# Why reading & understanding 3GPP specs is challenging ?

- So many technologies
- Choosing right specification
- Some specifications are very big
- They are interlinked
- Lack of normative references

# Recommended Practice

- Know what to read

General information (not used)			00 series
Requirements	21 series	41 series	01 series
Service aspects ("stage 1")	22 series	42 series	02 series
Technical realization ("stage 2")	23 series	43 series	03 series
Signalling protocols ("stage 3") - user equipment to network	24 series	44 series	04 series
Radio aspects	25 series	45 series	05 series
CODECs	26 series	46 series	06 series
Data	27 series	47 series (none exists)	07 series
Signalling protocols ("stage 3") - (RSS-CN) and OAM&P and Charging (overflow from 32.- range)	28 series	48 series	08 series
Signalling protocols ("stage 3") - intra-fixed-network	29 series	49 series	09 series
Programme management	30 series	50 series	10 series
Subscriber Identity Module (SIM / USIM), IC Cards. Test specs.	31 series	51 series	11 series
OAM&P and Charging	32 series	52 series	12 series
Access requirements and test specifications		13 series (1)	13 series (1)
Security aspects	33 series	(2)	(2)
UE and (U)SIM test specifications	34 series	(2)	11 series



# Recommended Practice



- **Know what to read**

## 5G System

- TS 23.501: System Architecture for the 5G System, Stage 2, Rel 16.0.0, 2019-03
- TS 23.502: Procedures for 5G System, Stage 2, Rel 16.0.0, 2019-03
- TS 29.500: 5G System, Technical Realization of Service Based Architecture, Stage 3, Rel 16.0.0, 2019-03
- TS 29.501: 5G System, Principles and Guidelines for Services Definition, Stage 3, Rel 16.0.0, 2019-03
- TS 29.571 - Common Data Types for Service Based Interfaces, Stage 3, v16.0.0, 2019-03

## Security

- TS 33.102 - Security architecture, v16.0.0, 2020-07-10

## NAS:

- TS 24.501: Non-Access-Stratum (NAS) protocol for 5G System, Stage 3, v16.0.0, 2019-03

## NGAP

- TS 38.413, NG Application Protocol (NGAP), v16.0.0, 2019-12

## PFCP

- TS 29.244, Interface between the Control Plane and the User Plane nodes, v16.0.0, 2019-06-13
- TS 23.503, Policy and Charging Control Framework for the 5G System, Stage 2, v16.0.0, 2019-03

## SMF

- TS 29.502 - Session Management Services, Stage 3, v16.0.0, 2019-03
- TS 29.508 - Session Management Event Exposure Service, Stage 3, v16.0.0, 2019-03

## AMF

- TS 29.518 - Access and Mobility Management Services, Stage 3, v16.0.0, 2019-03

## UDM/AUSF

- 3GPP TS 33.501 V16.0.0 (2019-09): Security architecture and procedures for 5G system
- UDM: TS 29.503 - Unified Data Management Services, Stage 3, v16.0.0, 2018-09
- AUSF: TS 29.509 - Authentication Server Services, Stage 3, v16.0.0, 2019-03: 29509-v200.yaml

## NRF

- TS 29.510 - Network Function Repository Services, Stage 3, v16.0.0, 2019-03

## PCF

- TS 29.507 - Access and Mobility Policy Control Service, Stage 3, v16.0.0, 2019-03
- TS 29.512 - Session Management Policy Control Service, Stage 3, v16.0.0, 2019-03

## NSSF

- TS 29.531 - Network Slice Selection Services; Stage 3, v16.0.0 2019-03, 2019-09-23

# Recommended Practice

- Read respective specification first then references

---

## 2 References

The following documents contain provisions which, through reference in this text, constitute provisions of the present document.

- References are either specific (identified by date of publication, edition number, version number, etc.) or non-specific.
- For a specific reference, subsequent revisions do not apply.
- For a non-specific reference, the latest version applies. In the case of a reference to a 3GPP document (including a GSM document), a non-specific reference implicitly refers to the latest version of that document *in the same Release as the present document*.

- [1] 3GPP TR 21.905: "Vocabulary for 3GPP Specifications".
- [2] 3GPP TS 23.501: "System Architecture for the 5G System; Stage 2".
- [3] IETF RFC 7296: "Internet Key Exchange Protocol Version 2 (IKEv2)".
- [4] IETF RFC 5998: "An Extension for EAP-Only Authentication in IKEv2".
- [5] IETF RFC 4282: "The Network Access Identifier".
- [6] IETF RFC 4861: "Neighbor Discovery for IP version 6 (IPv6)".
- [7] 3GPP TS 23.040: "Technical realization of the Short Message Service (SMS)".
- [8] IETF RFC 4862: "IPv6 Stateless Address Autoconfiguration".

# Recommended Practice

- Search for normative references
  - Blogs, tutorials, use cases scenarios
  - Whitepapers .....!!!

---

## Recommended Practice ..... !!!

- Take printout **when necessary**

# Recommended Practice ..... !!!

- Take printout **when necessary**
- Always good to ask

When life suddenly starts going too well



---

## **Part 2: Implementing procedure/protocols from 3gpp specification**

**Call flows => UML diagrams => IEs etc.**



# 5CN Communication Mechanism

- **Communication within NFs (SBIs) -**
  - Transport layer -> TCP
  - Application layer -> http (serialization with json)
  - Openapi -> interface definitions
- **Communication with User Plane (Protocols)-**
  - NAS
  - NGAP (serialization with ASN1.c)
  - SCTP (at lower layer)
  - PFCP
  - gtpv1u

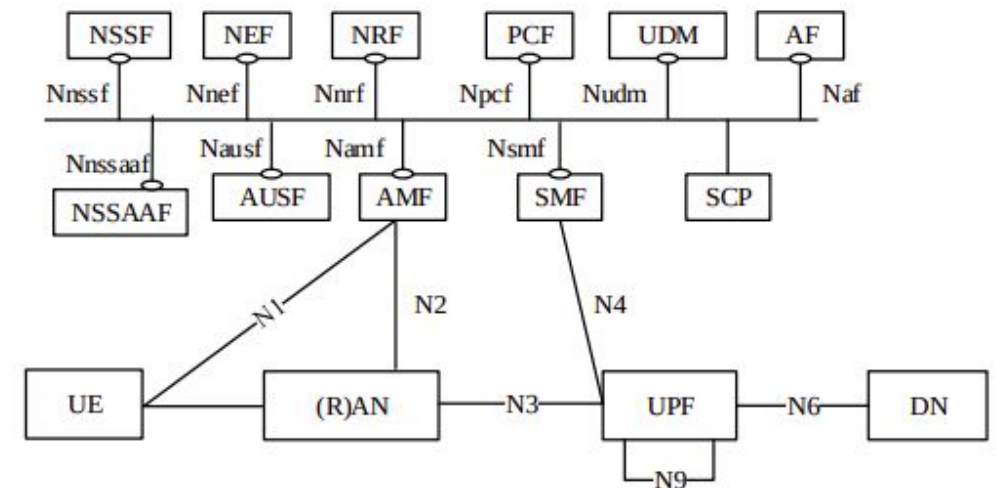
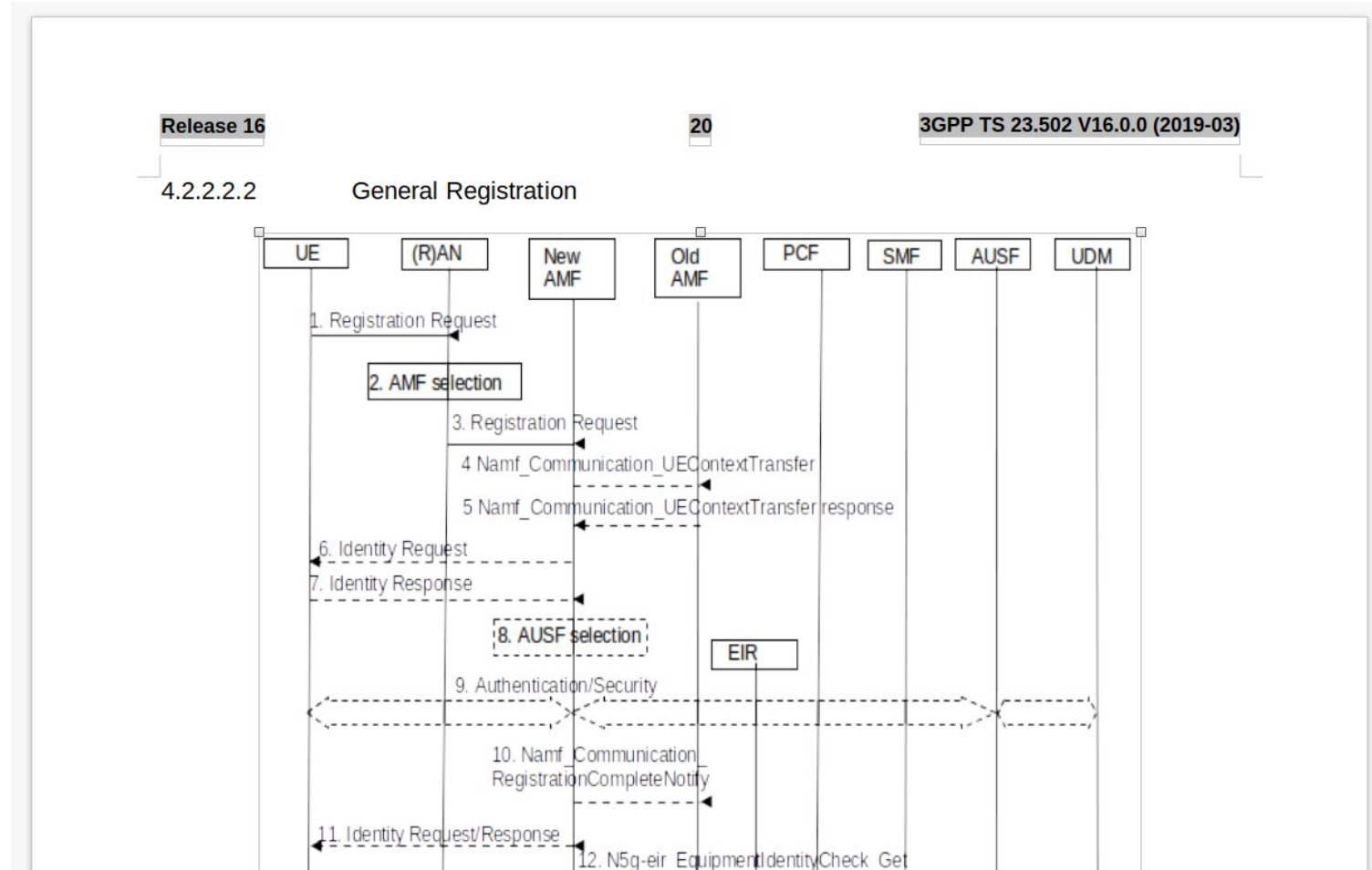


Figure 4.2.3-1: 5G System architecture

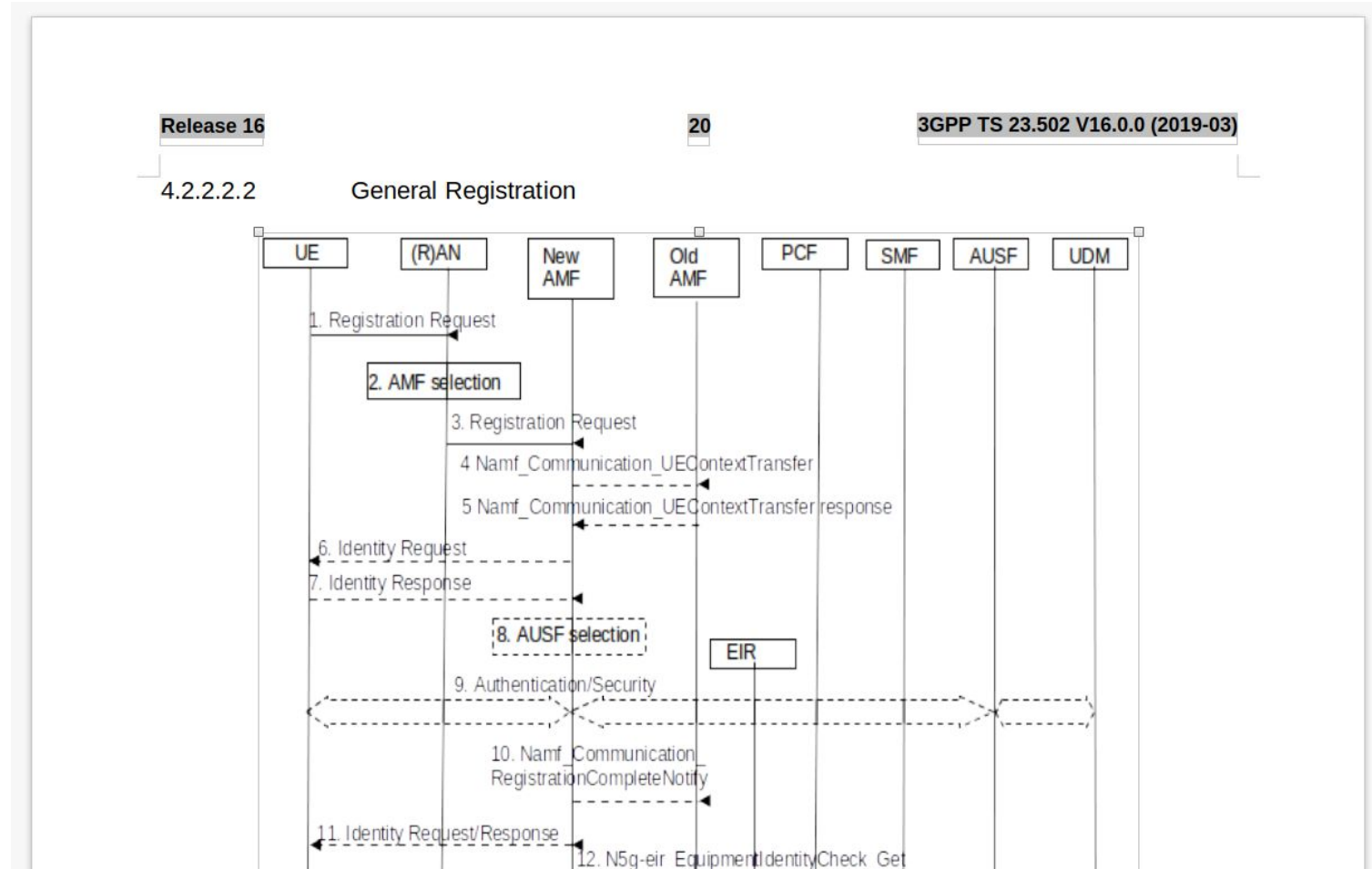
# 3GPP Procedures (Call Flows)

- 3GPP TS 23.502 -> Procedures for the 5G System



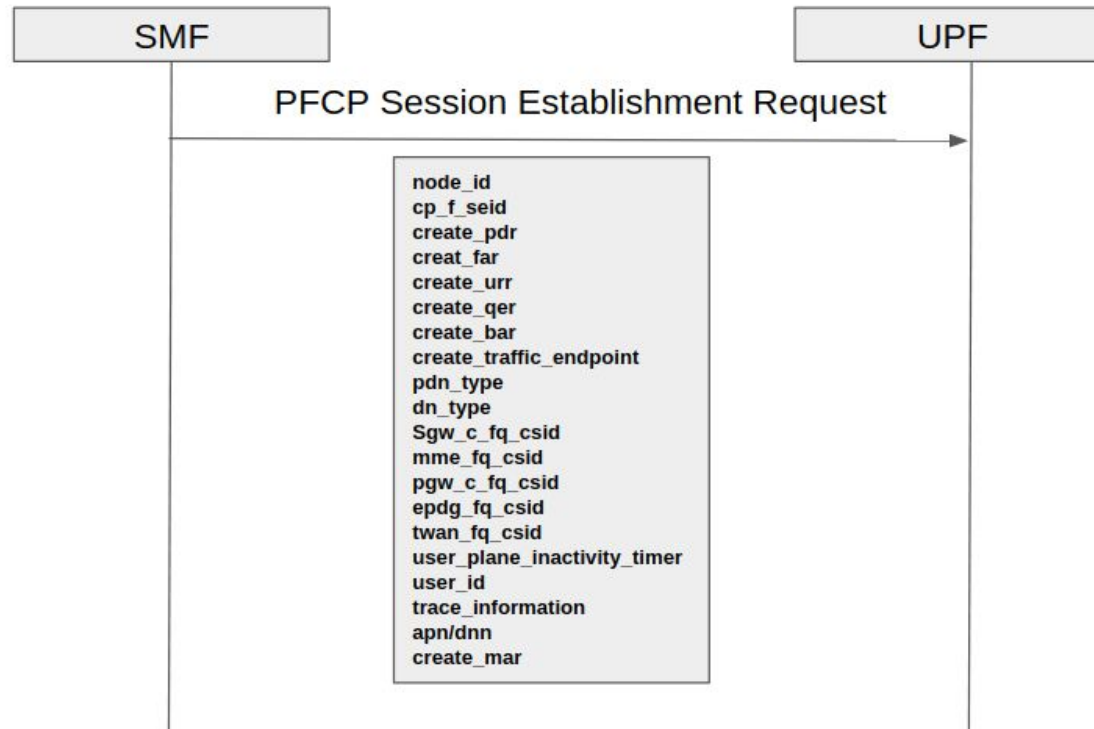
# 3GPP Procedures (Call Flows)

- 3GPP TS 23.502 -> Procedures for the 5G System



# 3GPP Procedures (Call Flows - Create detailed call flow)

- 3GPP TS 23.502 -> Procedures for the 5G System



Release 16

90

3GPP TS 29.244 V16.0.0 (2019-06)

Table 7.5.2.1-1: Information Elements in an PFCP Session Establishment Request

Information elements	P	Condition / Comment	Appl.				IE Type
			Sx a	Sx b	Sx c	N4	
Node ID	M	This IE shall contain the unique identifier of the sending Node.	X	X	X	X	Node ID
CP F-SEID	M	This IE shall contain the unique identifier allocated by the CP function identifying the session.	X	X	X	X	F-SEID
Create PDR	M	This IE shall be present for at least one PDR to be associated to the PFCP session.  Several IEs with the same IE type may be present to represent multiple PDRs. See Table 7.5.2.2-1.	X	X	X	X	Create PDR
Create FAR	M	This IE shall be present for at least one FAR to be associated to the PFCP session.  Several IEs with the same IE type may be present to represent multiple FARs. See Table 7.5.2.3-1.	X	X	X	X	Create FAR
Create URR	C	This IE shall be present if a measurement action shall be applied to packets matching one or more PDR(s) of this PFCP session. Several IEs within the same IE type may be present to represent multiple URRs.	X	X	X	X	Create URR

# 3GPP Procedures (Call Flows - UML Class Diagram)

Class Name

Class Attributes

Class Methods

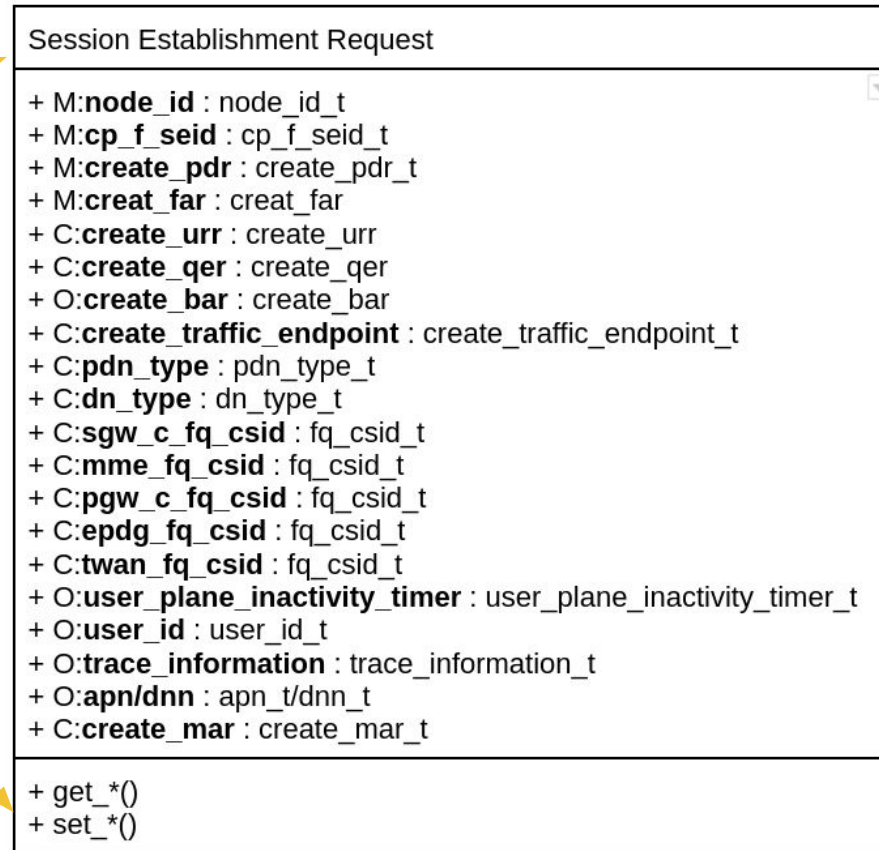


Table 7.5.2.1-1: Information Elements in an PFCP Session Establishment Request

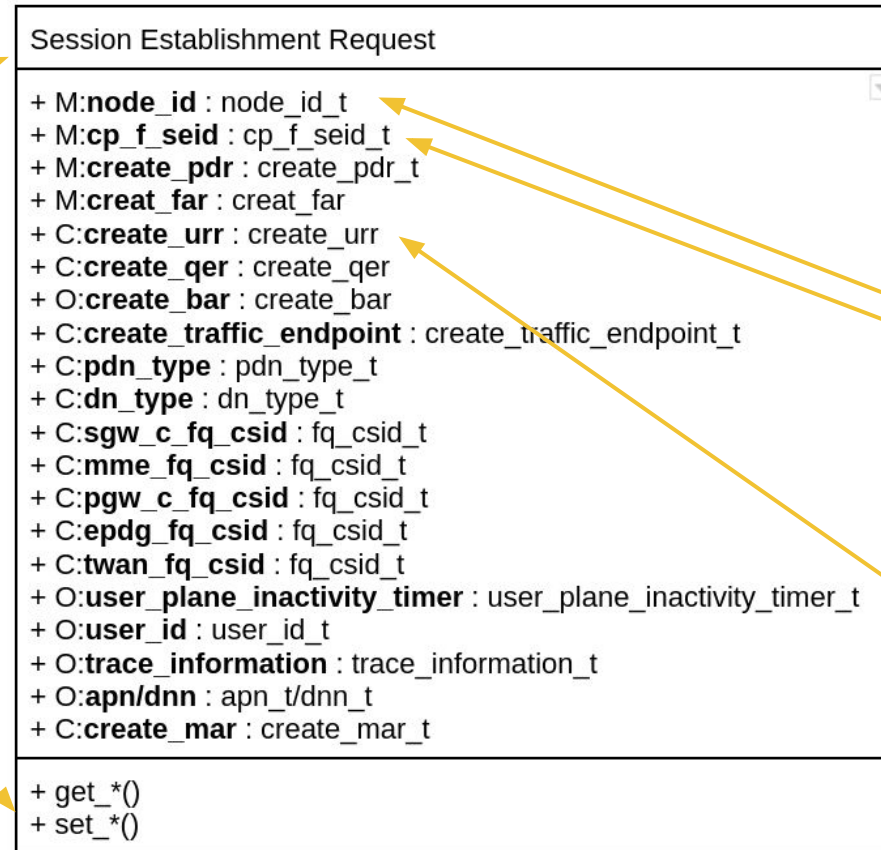
Information elements	P	Condition / Comment	Appl.				IE Type
			Sx a	Sx b	Sx c	N4	
Node ID	M	This IE shall contain the unique identifier of the sending Node.	X	X	X	X	Node ID
CP F-SEID	M	This IE shall contain the unique identifier allocated by the CP function identifying the session.	X	X	X	X	F-SEID
Create PDR	M	This IE shall be present for at least one PDR to be associated to the PFCP session.  Several IEs with the same IE type may be present to represent multiple PDRs. See Table 7.5.2.2-1.	X	X	X	X	Create PDR
Create FAR	M	This IE shall be present for at least one FAR to be associated to the PFCP session.  Several IEs with the same IE type may be present to represent multiple FARs. See Table 7.5.2.3-1.	X	X	X	X	Create FAR
Create URR	C	This IE shall be present if a measurement action shall be applied to packets matching one or more PDR(s) of this PFCP session.  Several IEs within the same IE type may be present to represent multiple URRs.	X	X	X	X	Create URR

# 3GPP Procedures (Call Flows - UML Class Diagram)

Class Name

Class Attributes

Class Methods



Release 16

90

3GPP TS 29.244 V16.0.0 (2019-06)

Table 7.5.2.1-1: Information Elements in an PFCP Session Establishment Request

Information elements	P	Condition / Comment	Appl.				IE Type
			Sx a	Sx b	Sx c	N4	
Node ID	M	This IE shall contain the unique identifier of the sending Node.	X	X	X	X	Node ID
CP F-SEID	M	This IE shall contain the unique identifier allocated by the CP function identifying the session.	X	X	X	X	F-SEID
Create PDR	M	This IE shall be present for at least one PDR to be associated to the PFCP session.  Several IEs with the same IE type may be present to represent multiple PDRs. See Table 7.5.2.2-1.	X	X	X	X	Create PDR
Create FAR	M	This IE shall be present for at least one FAR to be associated to the PFCP session.  Several IEs with the same IE type may be present to represent multiple FARs. See Table 7.5.2.3-1.	X	X	X	X	Create FAR
Create URR	C	This IE shall be present if a measurement action shall be applied to packets matching one or more PDR(s) of this PFCP session.  Several IEs within the same IE type may be present to represent multiple URRs.	X	X	X	X	Create URR

# OAI CN Repository Structure

```
oai-cn5g-<network_function>$ tree -L 1
├── build
├── ci-scripts
├── docker
├── docs
├── etc
├── scripts
├── src
├── CHANGELOG.md
├── CONTRIBUTING.md
├── LICENSE
└── README.md

7 directories, 4 files
```

Cmake build directory, BM install scripts

Scripts used for CI/CD validation

Dockerfiles for Ubuntu and RHEL

Documentation (feature set)

Config files

Some init scripts

Implementation of 3gpp procedures and protocols

# OAI CN Repository Structure

**/src**

3gpp protocols

```
oai-cn5g-amf/src/$ tree -L 1
├── amf-app
├── common
├── contexts
├── itti
├── nas
├── ngap
├── oai-amf
├── sbi
├── sctp
├── secu_algorithms
└── utils

1 directory, 15 files
```

```
oai-cn5g-smf/src/$ tree -L 1
├── api-server
├── common
├── itti
├── nas
├── ngap
├── oai_smf
├── pfcf
├── smf_app
└── udp

9 directories, 0 files
```

```
openair-spgwu-tiny/src/$ tree -L 1
├── common
├── gtpv1u
├── gtpv2c
├── itti
├── oai_spgwu
├── pfcf
├── spgwu
└── udp

8 directories, 0 files
```

# OAI CN Repository Structure

**/src/<nf\_name>\_app/**

3gpp procedures

```
oai-cn5g-amf/src/amf_app$ tree -L 1
├── amf_app.cpp
├── amf_app.hpp
├── amf_config.cpp
├── amf_config.hpp
├── amf_event.cpp
├── amf_event.hpp
├── amf_event_sig.hpp
├── amf_module_from_config.cpp
├── amf_module_from_config.hpp
├── amf_msg.cpp
├── amf_msg.hpp
├── amf_n11.cpp
├── amf_n11.hpp
├── amf_n1.cpp
├── amf_n1.hpp
├── amf_n2.cpp
├── amf_n2.hpp
├── amf_profile.cpp
├── amf_profile.hpp
├── amf_statistics.cpp
├── amf_statistics.hpp
├── amf_subscription.cpp
├── amf_subscription.hpp
├── CMakeLists.txt
├── mysql_db.cpp
└── mysql_db.hpp

0 directories, 26 files
```

```
oai-cn5g-smf/src/smf_app$ tree -L 1
├── CMakeLists.txt
├── smf_app.cpp
├── smf_app.hpp
├── smf_config.cpp
├── smf_config.hpp
├── smf_context.cpp
├── smf_context.hpp
├── smf_event.cpp
├── smf_event.hpp
├── smf_event_sig.hpp
├── smf_msg.cpp
├── smf_msg.hpp
├── smf_n1.cpp
├── smf_n1.hpp
├── smf_n2.cpp
├── smf_n2.hpp
├── smf_n4.cpp
├── smf_n4.hpp
├── smf_paa_dynamic.hpp
├── smf_pco.cpp
├── smf_pco.hpp
├── smf_pfcf_association.cpp
├── smf_pfcf_association.hpp
├── smf_procedure.cpp
├── smf_procedure.hpp
├── smf_profile.cpp
├── smf_profile.hpp
├── smf_sbi.cpp
├── smf_sbi.hpp
├── smf_subscription.cpp
└── smf_subscription.hpp

0 directories, 31 files
```

```
oai-cn5g-ausf/src/ausf_app$ tree -L 1
├── ausf_app.cpp
├── ausf_app.hpp
├── ausf_client.cpp
├── ausf_client.hpp
├── ausf_config.cpp
├── ausf_config.hpp
├── ausf_nrf.cpp
├── ausf_nrf.hpp
├── ausf_profile.cpp
├── ausf_profile.hpp
└── CMakeLists.txt

0 directories, 11 files
```

# OAI CN Repository Structure

## /src/common

```
oai-cn5g-amf/src/common$ tree -L 1
├── 3gpp_23.003.h
├── 3gpp_24.501.h
├── 3gpp_29.500.h
├── 3gpp_29.502.h
├── 3gpp_29.510.h
├── 3gpp_29.518.h
├── amf.hpp
├── CMakeLists.txt
├── comUt.cpp
├── comUt.hpp
├── conversions.cpp
├── conversions.hpp
├── endpoint.hpp
├── logger.cpp
├── logger.hpp
├── unicode
└──
```

1 directory, 15 files

```
oai-cn5g-nrf/src/common$ tree -L 1
├── 3gpp_23.003.h
├── 3gpp_29.500.h
├── 3gpp_29.510.h
├── CMakeLists.txt
├── common_defs.h
├── dynamic_memory_check.c
├── dynamic_memory_check.h
├── logger.cpp
├── logger.hpp
├── nrf.h
├── utils
└──
```

1 directory, 10 files

3gpp common  
data types,  
enumerations,  
constants

# OAI CN Repository Structure

**/src/oai\_<nf\_name>/**

Parse config parameters

Initiate application layer

Initiate thread manager (ITTI or Task Manager)

Initiate HTTPv1 & HTTPv2 server

**generic main.cpp**

```
oai-cn5g-nrf/src/oai_nrf$ cat main.cpp
int main(int argc, char** argv) {

    // Config
    nrf_cfg.load(Options::getlibconfigConfig());
    nrf_cfg.display();

    // Event subsystem
    nrf_event ev;

    // NRF application layer
    nrf_app_inst = new nrf_app(Options::getlibconfigConfig(), ev);

    // Task Manager
    task_manager tm(ev);
    std::thread task_manager_thread(&task_manager::run, &tm);

    // PID file
    // Currently hard-coded value. TODO: add as config option.
    string pid_file_name = get_exe_absolute_path("/var/run", nrf_cfg.instance);
    if (!is_pid_file_lock_success(pid_file_name.c_str())) {
        Logger::nrf_app().error("Lock PID file %s failed\n", pid_file_name.c_str());
        exit(-EDEADLK);
    }

    // NRF Pistache API server (HTTP1)
    Pistache::Address addr(
        std::string(inet_ntoa(*(struct in_addr*) &nrf_cfg.sbi.addr4)),
        Pistache::Port(nrf_cfg.sbi.port));
    nrf_api_server = new NRFApiServer(addr, nrf_app_inst);
    nrf_api_server->init(2);
    std::thread nrf_manager(&NRFApiServer::start, nrf_api_server);

    // NRF NGHTTP API server (HTTP2)
    nrf_api_server_2 = new nrf_http2_server(
        conv::toString(nrf_cfg.sbi.addr4), nrf_cfg.sbi_http2_port, nrf_app_inst);
    std::thread nrf_http2_manager(&nrf_http2_server::start, nrf_api_server_2);

    nrf_manager.join();
    nrf_http2_manager.join();
}
```

---

## **Part 3: Use cases**

**Example case of pfcP, gtp1u etc.**

# Example case of PFCP

- TS 29.244 - Interface between the Control Plane and the User Plane nodes
- Sx/N4 interface
- UDP based, Port 8805

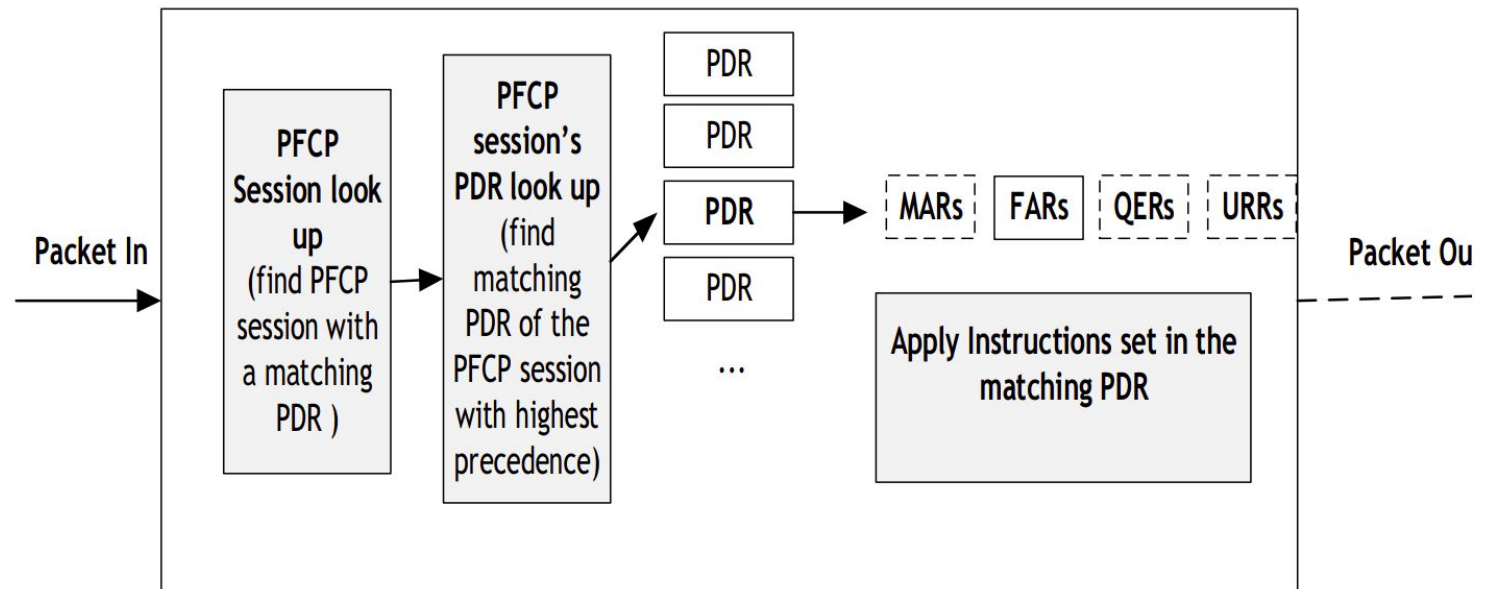


Figure 5.2.1-1: Packet processing flow in the UP function

# Example case of PFCP

- TS 29.244 - Interface between the Control Plane and the User Plane nodes
- Sx/N4 interface
- UDP based, Port 8805

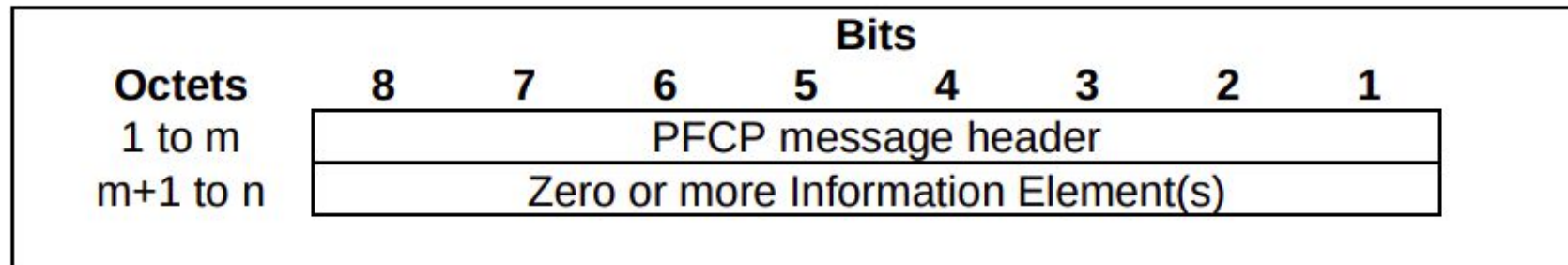


Figure 7.2.1-1: PFCP Message Format

# Example case of PFCP

Node Related Messages

Session Related Messages

```
namespace pfc {
#define PFCP_MESSAGE_RESERVED (0)
// PFCP NODE RELATED MESSAGES
#define PFCP_HEARTBEAT_REQUEST (1)
#define PFCP_HEARTBEAT_RESPONSE (2)
#define PFCP_PFCP_PFD_MANAGEMENT_REQUEST (3)
#define PFCP_PFCP_PFD_MANAGEMENT_RESPONSE (4)
#define PFCP_ASSOCIATION_SETUP_REQUEST (5)
#define PFCP_ASSOCIATION_SETUP_RESPONSE (6)
#define PFCP_ASSOCIATION_UPDATE_REQUEST (7)
#define PFCP_ASSOCIATION_UPDATE_RESPONSE (8)
#define PFCP_ASSOCIATION_RELEASE_REQUEST (9)
#define PFCP_ASSOCIATION_RELEASE_RESPONSE (10)
#define PFCP_VERSION_NOT_SUPPORTED_RESPONSE (11)
#define PFCP_NODE_REPORT_REQUEST (12)
#define PFCP_NODE_REPORT_RESPONSE (13)
#define PFCP_SESSION_SET_DELETION_REQUEST (14)
#define PFCP_SESSION_SET_DELETION_RESPONSE (15)

// PFCP SESSION RELATED MESSAGES
#define PFCP_SESSION_ESTABLISHMENT_REQUEST (50)
#define PFCP_SESSION_ESTABLISHMENT_RESPONSE (51)
#define PFCP_SESSION_MODIFICATION_REQUEST (52)
#define PFCP_SESSION_MODIFICATION_RESPONSE (53)
#define PFCP_SESSION_DELETION_REQUEST (54)
#define PFCP_SESSION_DELETION_RESPONSE (55)
#define PFCP_SESSION_REPORT_REQUEST (56)
#define PFCP_SESSION_REPORT_RESPONSE (57)
} // namespace pfc
```

# Example case of PFCP

Octets	Bits							
	8	7	6	5	4	3	2	1
1	Version		Spare	Spare	FO=0	MP=0	S=0	
2	Message Type							
3	Message Length (1 <sup>st</sup> Octet)							
4	Message Length (2 <sup>nd</sup> Octet)							
5	Sequence Number (1 <sup>st</sup> Octet)							
6	Sequence Number (2 <sup>nd</sup> Octet)							
7	Sequence Number (3 <sup>rd</sup> Octet)							
8	Spare							

Figure 7.2.2.2-1: PFCP Message Header for node related messages

Octets	Bits							
	8	7	6	5	4	3	2	1
1	Version		Spare	Spare	FO	MP	S=1	
2	Message Type							
3	Message Length (1 <sup>st</sup> Octet)							
4	Message Length (2 <sup>nd</sup> Octet)							
5	Session Endpoint Identifier (1 <sup>st</sup> Octet)							
6	Session Endpoint Identifier (2 <sup>nd</sup> Octet)							
7	Session Endpoint Identifier (3 <sup>rd</sup> Octet)							
8	Session Endpoint Identifier (4 <sup>th</sup> Octet)							
9	Session Endpoint Identifier (5 <sup>th</sup> Octet)							
10	Session Endpoint Identifier (6 <sup>th</sup> Octet)							
11	Session Endpoint Identifier (7 <sup>th</sup> Octet)							
12	Session Endpoint Identifier (8 <sup>th</sup> Octet)							
13	Sequence Number (1 <sup>st</sup> Octet)							
14	Sequence Number (2 <sup>nd</sup> Octet)							
15	Sequence Number (3 <sup>rd</sup> Octet)							
16	Message Priority			Spare				

Figure 7.2.2.3-1: PFCP message Header for session related messages

## Generic PFCP header

# Example case of PFCP

Parent class for pfcf header - **Members**

```

class pfcf_msg_header : public stream_serializable {
private:
#define PFCF_MSG_HEADER_MIN_SIZE 8
    union {
        struct {
            uint8_t s : 1;
            uint8_t mp : 1;
            uint8_t spare : 3;
            uint8_t version : 3;
        } bf;
        uint8_t b;
    } u1;
    uint8_t message_type;
    uint16_t message_length;
    uint64_t seid;
    uint32_t sequence_number;
    union {
        struct {
            uint8_t spare : 4;
            uint8_t message_priority : 4;
        } bf;
        uint8_t b;
    } u2;
}
    
```

flag  
type  
len  
seid  
seq num  
MP

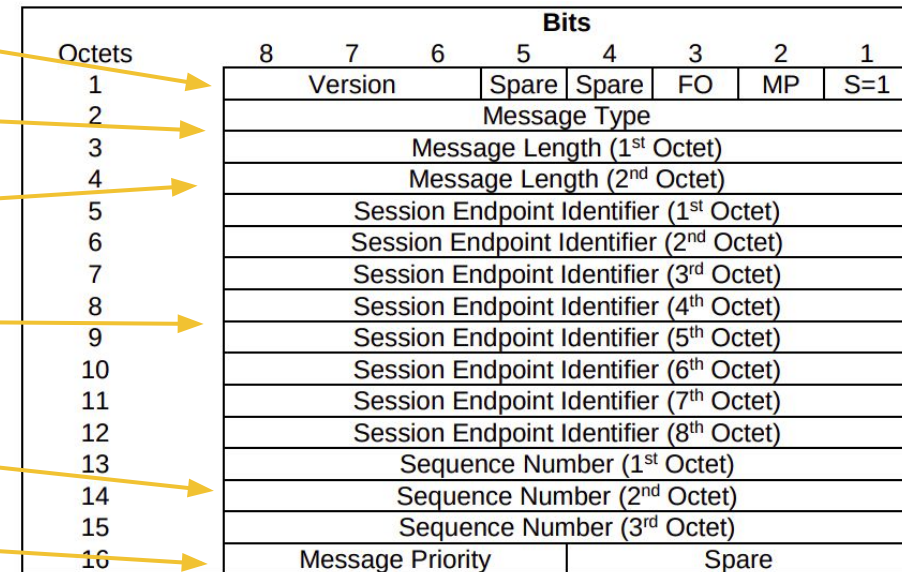


Figure 7.2.3-1: PFCP message Header for session related messages

# Example case of PFCP

Parent class for pfcf header - **Methods (get/set)**

```
void set_seid(const uint64_t& s) {
    seid = s;
    if (u1.bf.s == 0) {
        u1.bf.s = 1;
        message_length += 8;
    }
}

bool has_seid() const { return (u1.bf.s == 1); }

uint64_t get_seid() const { return seid; }

void set_message_type(const uint8_t& t) { message_type = t; }

uint8_t get_message_type() const { return message_type; }

void set_message_length(const uint16_t& l) { message_length = l; }

uint16_t get_message_length() const { return message_length; }

void set_sequence_number(const uint32_t& s) {
    sequence_number = s & 0x00FFFFFF;
}
```



# Example case of PFCP

Parent class for pfcf header - **Methods (serialization/deserialization)**

- Deserialization (*is.read()*)

```
virtual void load_from(std::istream& is) {
    is.read(reinterpret_cast<char*>(&u1.b), sizeof(u1.b));
    is.read(reinterpret_cast<char*>(&message_type), sizeof(message_type));
    is.read(reinterpret_cast<char*>(&message_length), sizeof(message_length));
    message_length = be16toh(message_length);
    if (u1.b.f.s) {
        is.read(reinterpret_cast<char*>(&seid), sizeof(seid));
        seid = be64toh(seid);
    } else {
        seid = 0;
    }
    uint8_t sn[3];
    is.read(reinterpret_cast<char*>(sn), 3);
    sequence_number =
        (((uint32_t) sn[0]) << 16) | (((uint32_t) sn[1]) << 8) | sn[2];
    is.read(reinterpret_cast<char*>(&u2.b), sizeof(u2.b));
}
```

- Serialization (*is.write()*)

```
virtual void dump_to(std::ostream& os)
```

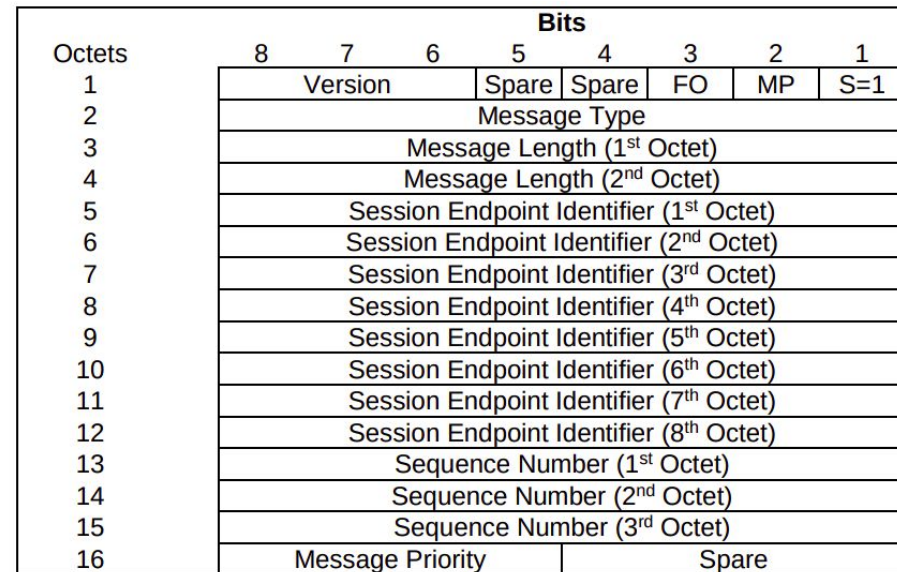


Figure 7.2.2.3-1: PFCP message Header for session related messages

# Example case of PFCP

- IE NODE ID - ID of PFCP peer endpoints

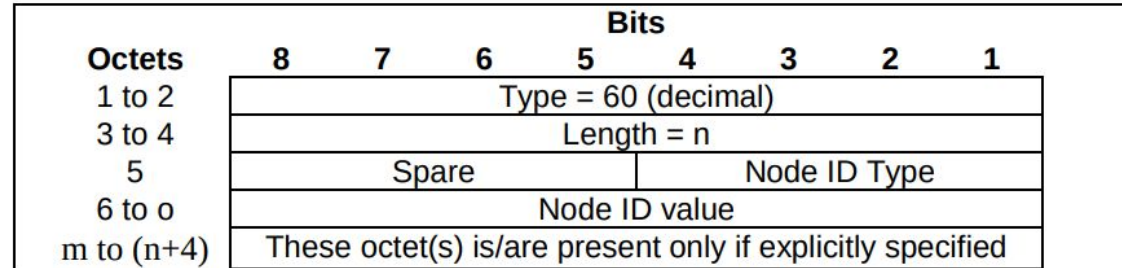


Figure 8.2.38-1: Node ID

Table 8.2.38-2: Node ID Type

Node ID Type Value (Decimal)	Node ID Type
0	IPv4 address
1	IPv6 address
2	FQDN
3 to 15	Spare, for future use.

# Example case of PFCP

## IE NODE ID

```

// IE NODE_ID
class pfcpc_node_id_ie : public pfcpc_ie {
public:
    union {
        struct {
            uint8_t node_id_type : 4;
            uint8_t spare1 : 4;
        } bf;
        uint8_t b;
    } u1;

    struct in_addr ipv4_address;
    struct in6_addr ipv6_address;
    std::string fqdn;
}
    
```

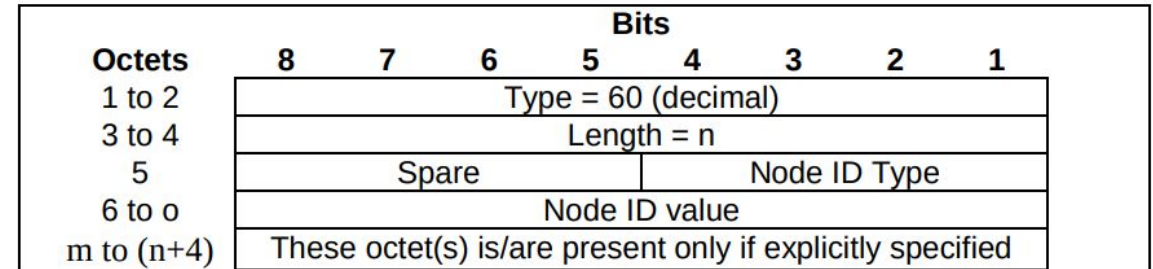


Figure 8.2.38-1: Node ID

Table 8.2.38-2: Node ID Type

Node ID Type Value (Decimal)	Node ID Type
0	IPv4 address
1	IPv6 address
2	FQDN
3 to 15	Spare, for future use.

IE included in the message

# Example case of PFCP

## Deserialization

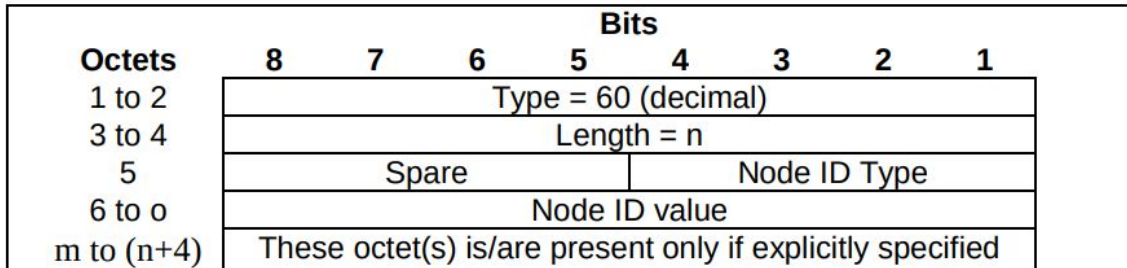


Figure 8.2.38-1: Node ID

Table 8.2.38-2: Node ID Type

Node ID Type Value (Decimal)	Node ID Type
0	IPv4 address
1	IPv6 address
2	FQDN
3 to 15	Spare, for future use.

```

}
//-----
void load_from(std::istream& is) {
    // tlv.load_from(is);
    is.read(reinterpret_cast<char*>(&u1.b), sizeof(u1.b));
    uint16_t check_length = tlv.get_length() - 1;
    switch (u1.bf.node_id_type) {
        case pfcpc::NODE_ID_TYPE_IPV4_ADDRESS:
            if (check_length != 4) {
                throw pfcpc_tlv_bad_length_exception(
                    tlv.type, tlv.get_length(), __FILE__, __LINE__);
            }
            ipv4_address_load_from(is, ipv4_address);
            break;
        case pfcpc::NODE_ID_TYPE_IPV6_ADDRESS:
            if (check_length != 16) {
                throw pfcpc_tlv_bad_length_exception(
                    tlv.type, tlv.get_length(), __FILE__, __LINE__);
            }
            ipv6_address_load_from(is, ipv6_address);
            break;
        case pfcpc::NODE_ID_TYPE_FQDN: {
            if (check_length == 0) {
                throw pfcpc_tlv_bad_length_exception(
                    tlv.type, tlv.get_length(), __FILE__, __LINE__);
            }
            char e[check_length];
            is.read(e, check_length);
            std::string dot = {};
            dot.assign(e, check_length);
            pfcpc_ie::dotted_to_string(dot, fqdn);
        } break;
        default:;
    }
}

```

flag  
len

node id  
type

# Example case of PFCP

Sample node related message

## 7.4.4.5 PFCP Association Release Request

Table 7.4.4.5-1: Information Elements in a PFCP Association Release Request

Information elements	P	Condition / Comment	IE Type
Node ID	M	This IE shall contain the unique identifier of the sending Node.	Node ID

**PFCP Message**

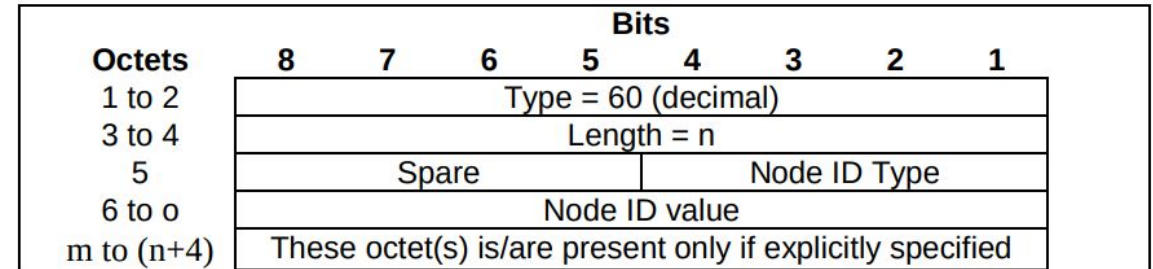


Figure 8.2.38-1: Node ID

Table 8.2.38-2: Node ID Type

Node ID Type Value (Decimal)	Node ID Type
0	IPv4 address
1	IPv6 address
2	FQDN
3 to 15	Spare, for future use.

**IE included in the message**

# Example case of PFCP

Sample node related message

- IE Association Release request -

```
pfcp_msg::pfcp_msg(const pfcp_association_release_request& pfcp_ies)
: pfcp_msg_header() {
  ies = {};
  set_message_type(PFCP_ASSOCIATION_RELEASE_REQUEST);
  if (pfcp_ies.node_id.first) {
    std::shared_ptr<pfcp_node_id_ie> sie(
      new pfcp_node_id_ie(pfcp_ies.node_id.second));
    add_ie(sie);
  }
}
```



Adding IE

# Example case of PFCP

Sample node related message

- IE Association Release request -

```
pfcp_msg::pfcp_msg(const pfcp_association_release_request& pfcp_ies)
: pfcp_msg_header() {
  ies = {};
  set_message_type(PFCP_ASSOCIATION_RELEASE_REQUEST);
  if (pfcp_ies.node_id.first) {
    std::shared_ptr<pfcp_node_id_ie> sie(
      new pfcp_node_id_ie(pfcp_ies.node_id.second));
    add_ie(sie);
  }
}
```

Adding IE

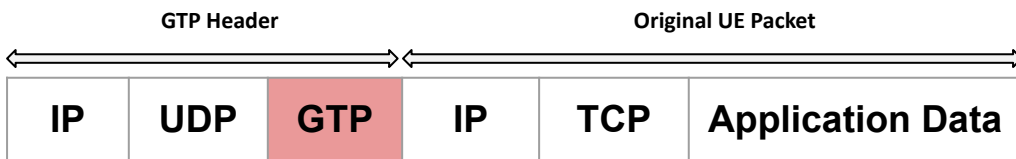
Grouped IE ??

# Summary

- Short overview of 3GPP specifications for 5G CN
- Recommended practice for studying specs
- Implementation in OAI CN stack
- Examples of encoding/decoding IEs

# Example case of gtp1u

- GTP - GPRS Tunneling Protocol
- 3GPP **TS 29.281** - User Plane (GTPv1-U)
- S1-U, S11-U, S2a, S2b, X2, S4, S5, S8, S12 (EPS)
- F1-U, Xn, N3, N9 and N19 (5GS)



Octets	Bits							
	8	7	6	5	4	3	2	1
1	Version			PT	(*)	E	S	PN
2	Message Type							
3	Length (1 <sup>st</sup> Octet)							
4	Length (2 <sup>nd</sup> Octet)							
5	Tunnel Endpoint Identifier (1 <sup>st</sup> Octet)							
6	Tunnel Endpoint Identifier (2 <sup>nd</sup> Octet)							
7	Tunnel Endpoint Identifier (3 <sup>rd</sup> Octet)							
8	Tunnel Endpoint Identifier (4 <sup>th</sup> Octet)							
9	Sequence Number (1 <sup>st</sup> Octet) <sup>1) 4)</sup>							
10	Sequence Number (2 <sup>nd</sup> Octet) <sup>1) 4)</sup>							
11	N-PDU Number <sup>2) 4)</sup>							
12	Next Extension Header Type <sup>3) 4)</sup>							

- NOTE 0: (\*) This bit is a spare bit. It shall be sent as '0'. The receiver shall not evaluate this bit.  
 NOTE 1: 1) This field shall only be evaluated when indicated by the S flag set to 1.  
 NOTE 2: 2) This field shall only be evaluated when indicated by the PN flag set to 1.  
 NOTE 3: 3) This field shall only be evaluated when indicated by the E flag set to 1.  
 NOTE 4: 4) This field shall be present if and only if any one or more of the S, PN and E flags are set.

Figure 5.1-1: Outline of the GTP-U Header



# Example case of gtp1u

## /src/gtpv1u/3gpp\_29.281.hpp

```

class gtpv1u_msg_header : public stream_serializable {
private:
#define GTPV1U_MSG_HEADER_MIN_SIZE 8
union {
    struct {
        uint8_t pn      : 1;
        uint8_t s      : 1;
        uint8_t e      : 1;
        uint8_t spare  : 1;
        uint8_t pt     : 1;
        uint8_t version : 3;
    } bf;
    uint8_t b;
} u1;
uint8_t message_type;
uint16_t message_length;
uint32_t teid;
uint16_t sequence_number;
uint8_t npdu_number;
uint8_t next_extension_header_type;

```

Octets	Bits							
	8	7	6	5	4	3	2	1
1	Version			PT	(*)	E	S	PN
2	Message Type							
3	Length (1 <sup>st</sup> Octet)							
4	Length (2 <sup>nd</sup> Octet)							
5	Tunnel Endpoint Identifier (1 <sup>st</sup> Octet)							
6	Tunnel Endpoint Identifier (2 <sup>nd</sup> Octet)							
7	Tunnel Endpoint Identifier (3 <sup>rd</sup> Octet)							
8	Tunnel Endpoint Identifier (4 <sup>th</sup> Octet)							
9	Sequence Number (1 <sup>st</sup> Octet) <sup>1) 4)</sup>							
10	Sequence Number (2 <sup>nd</sup> Octet) <sup>1) 4)</sup>							
11	N-PDU Number <sup>2) 4)</sup>							
12	Next Extension Header Type <sup>3) 4)</sup>							

- NOTE 0: (\*) This bit is a spare bit. It shall be sent as '0'. The receiver shall not evaluate this bit.
- NOTE 1: 1) This field shall only be evaluated when indicated by the S flag set to 1.
- NOTE 2: 2) This field shall only be evaluated when indicated by the PN flag set to 1.
- NOTE 3: 3) This field shall only be evaluated when indicated by the E flag set to 1.
- NOTE 4: 4) This field shall be present if and only if any one or more of the S, PN and E flags are set.

Figure 5.1-1: Outline of the GTP-U Header

# Example case of gtp1u

## /src/gtpv1u/3gpp\_29.281.hpp

- Serialize stream

```

virtual void dump_to (std::ostream& os) {
    u1.bf.spare = 0;
    os.write(reinterpret_cast<const char*>(&u1.b), sizeof(u1.b));

    os.write( reinterpret_cast<const char*>(&message_type), sizeof(message_type));
    auto be_message_length = htobe16(message_length);
    os.write(reinterpret_cast<const char*>(&be_message_length), sizeof(be_message_length));
    auto be_teid = htobe32(teid);
    os.write(reinterpret_cast<const char*>(&be_teid), sizeof(be_teid));

    if (u1.bf.s) {
        auto be_sequence_number = htobe16(sequence_number);
        os.write(reinterpret_cast<const char*>(&be_sequence_number), sizeof(be_sequence_number));
    }
    if (u1.b & 0x05) {
        os.write(reinterpret_cast<const char*>(&npdu_number), sizeof(npdu_number));
        os.write(reinterpret_cast<const char*>(&next_extension_header_type),
            sizeof(next_extension_header_type));
    }
}

```

- Deserialize stream

```
virtual void load from(std::istream& is)
```



flag  
message type  
message len  
sequence no  
npdu  
ext hdr

Octets	Bits								
	8	7	6	5	4	3	2	1	
1	Version		PT	(*)	E	S	PN		
2	Message Type								
3	Length (1 <sup>st</sup> Octet)								
4	Length (2 <sup>nd</sup> Octet)								
5	Tunnel Endpoint Identifier (1 <sup>st</sup> Octet)								
6	Tunnel Endpoint Identifier (2 <sup>nd</sup> Octet)								
7	Tunnel Endpoint Identifier (3 <sup>rd</sup> Octet)								
8	Tunnel Endpoint Identifier (4 <sup>th</sup> Octet)								
9	Sequence Number (1 <sup>st</sup> Octet) <sup>1) 4)</sup>								
10	Sequence Number (2 <sup>nd</sup> Octet) <sup>1) 4)</sup>								
11	N-PDU Number <sup>2) 4)</sup>								
12	Next Extension Header Type <sup>3) 4)</sup>								

# Example case of AMF Reselection

3GPP TS 23.502, R16.0.0., Section 4.2.2.2.3

Slice Selection during Registration with AMF re-allocation  
[When requested NSSAI not present in subscribed NSSAI]

3GPP TS 23.502, R16.0.0., Section 4.2.2.2.3

Slice Selection during registration  
[AMF Reselection- When requested NSSAI is not supported by AMF]

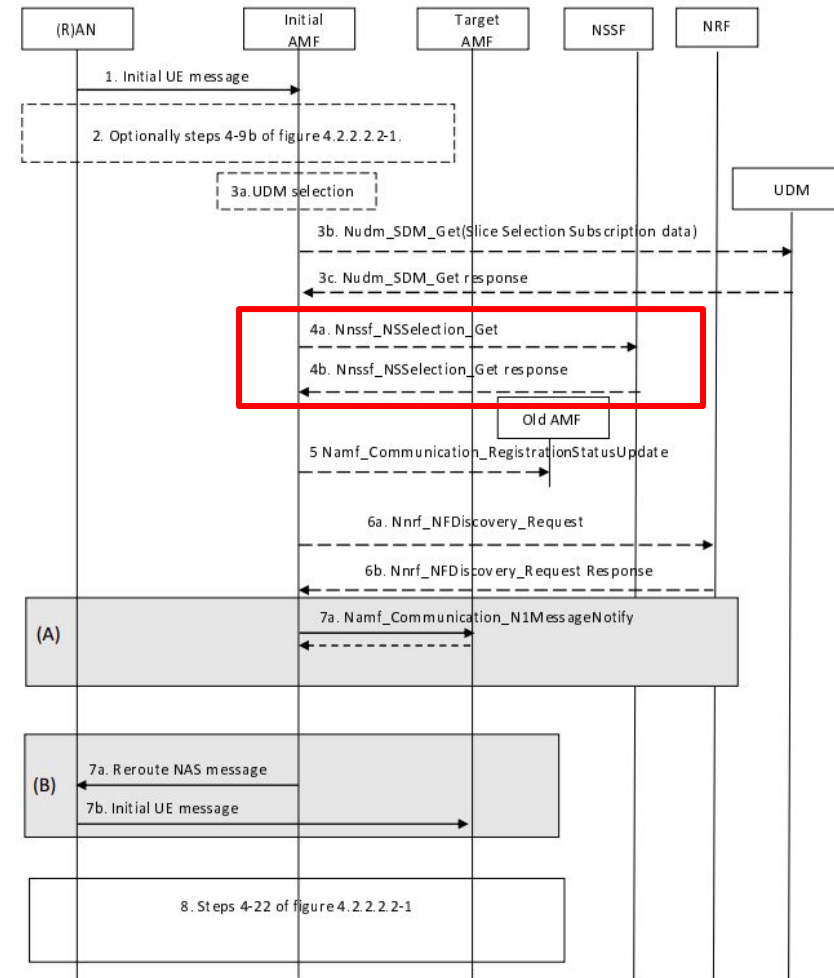


Figure 4.2.2.2.3-1: Registration with AMF re-allocation procedure

# Call Flow - AMF Reselection

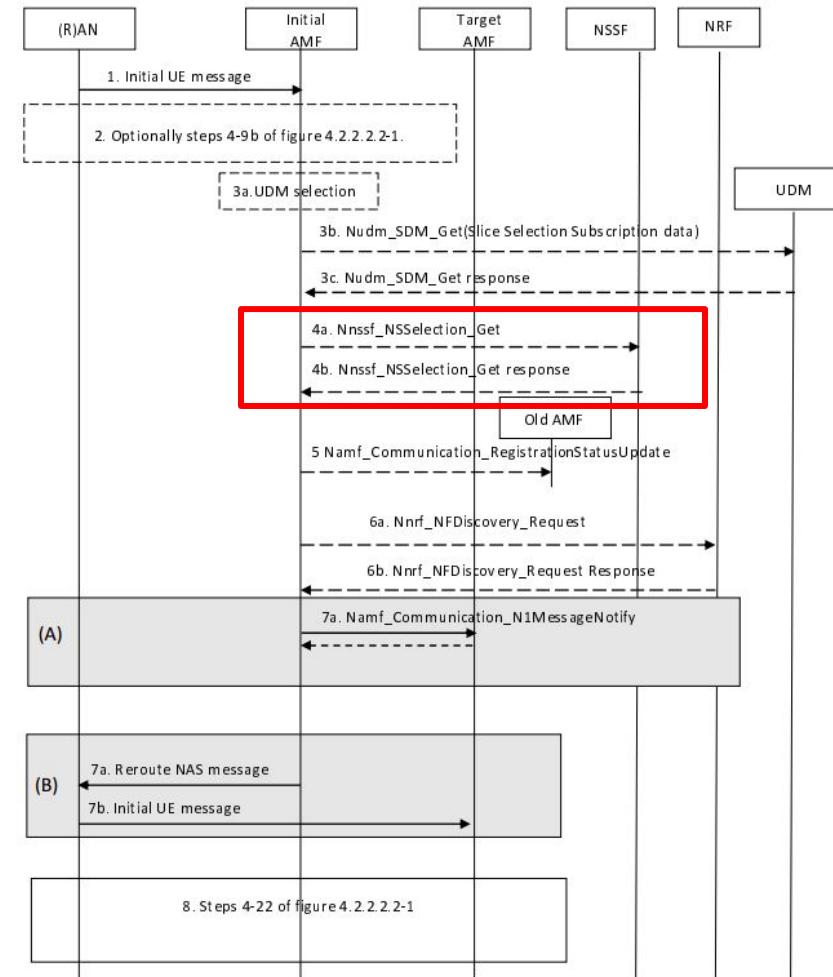
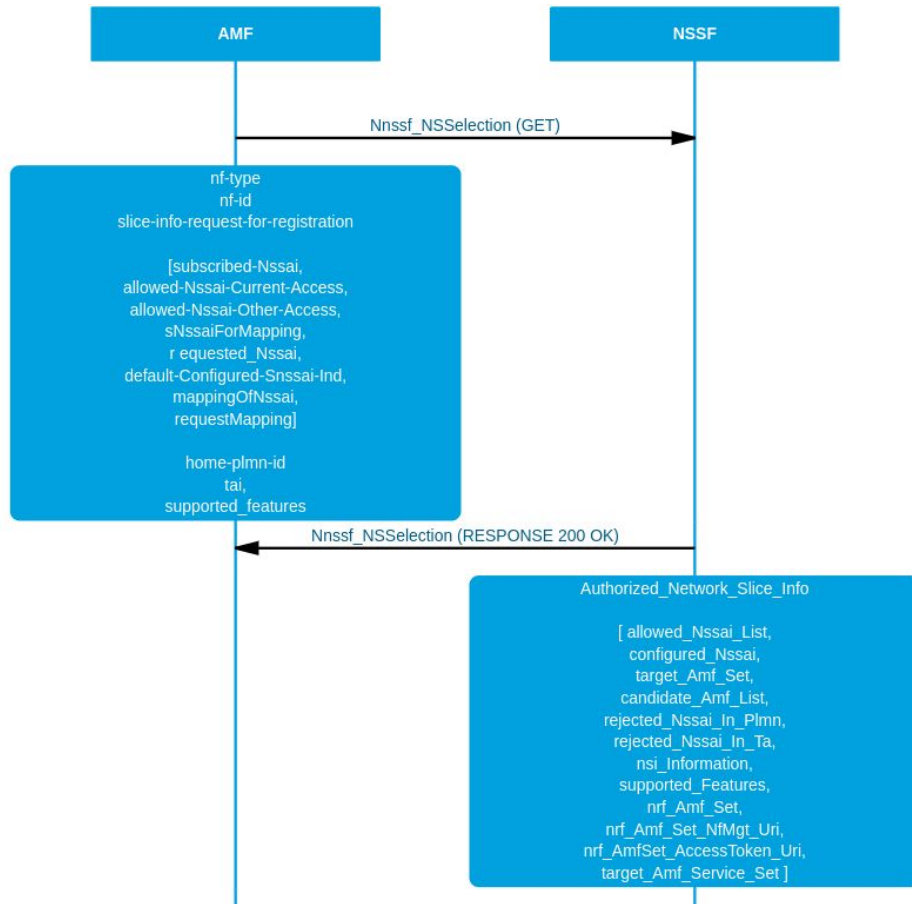


Figure 4.2.2.3-1: Registration with AMF re-allocation procedure

# UML diagram - AMF Reselection (nnnsf\_get)

