|  |  |
| --- | --- |
| 3GPP TR 33.700-22 V0.6.0 (2025-02) | |
| Technical Report | |
| 3rd Generation Partnership Project;  Technical Specification Group Services and System Aspects;  Study on security aspects of Common API Framework (CAPIF)Phase3  (Release 19) | |
|  | |
|  | 3GPP-logo_web |
|  | |
| The present document has been developed within the 3rd Generation Partnership Project (3GPP TM) and may be further elaborated for the purposes of 3GPP. The present document has not been subject to any approval process by the 3GPPOrganizational Partners and shall not be implemented. This Specification is provided for future development work within 3GPPonly. The Organizational Partners accept no liability for any use of this Specification. Specifications and Reports for implementation of the 3GPP TM system should be obtained via the 3GPP Organizational Partners' Publications Offices. | |

|  |  |
| --- | --- |
|  |  |
| ***3GPP***  Postal address  3GPP support office address  650 Route des Lucioles - Sophia Antipolis  Valbonne - FRANCE  Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16  Internet  http://www.3gpp.org |  |
| ***Copyright Notification***  No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.  © 2024, 3GPP Organizational Partners (ARIB, ATIS, CCSA, ETSI, TSDSI, TTA, TTC).  All rights reserved.  UMTS™ is a Trade Mark of ETSI registered for the benefit of its members  3GPP™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners LTE™ is a Trade Mark of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners  GSM® and the GSM logo are registered and owned by the GSM Association |  |

Contents

Foreword 12

1 Scope 14

2 References 14

3 Definitions of terms, symbols and abbreviations 15

3.1 Terms 15

3.2 Symbols 15

3.3 Abbreviations 15

4 High-level architectures 15

4.1 High-level architecture for RNAA 16

4.2 High-level architecture for CAPIF interconnection 16

5 Key issues 18

5.1 Key Issue #1: Security of resource owner authorization management and CAPIF-8 reference point 18

5.1.0 Introduction 18

5.1.1 Key Issue #1.1: CAPIF-8 reference point 18

5.1.1.1 Key issue details 18

5.1.1.2 Security threats 18

5.1.1.3 Potential Security Requirement 18

5.1.2 Key Issue #1.2: Resource owner authorization management 18

5.1.2.1 Key issue details 18

5.1.2.2 Security threats 19

5.1.2.3 Potential Security Requirement 19

5.1.3 Key Issue #1.3: Finer granular authorization 19

5.1.3.1 Key issue details 19

5.1.3.2 Security threats 19

5.1.3.3 Potential Security Requirement 19

5.2 Key issue #2: CAPIF interconnection security 20

5.2.1 Key issue details 20

5.2.2 Security threats 20

5.2.3 Potential security requirements 20

5.3 Key Issue #3: Authorizing API invoker on one UE accessing resources related to another UE 21

5.3.1 Key issue details 21

5.3.2 Security threats 21

5.3.3 Potential security requirements 21

5.4 Key issue #4: Nested API invocation 21

5.4.1 Key issue details 21

5.4.2 Security threats 22

5.4.3 Potential security requirements 22

5.5 Key Issue KI#5: Authenticating multiple API invokers of the same Resource Owner 22

5.5.1 Key Issue details 22

5.5.2 Security threats 22

5.5.3 Security requirements 22

5.6 Key Issue KI#6: Onboarding security issues 22

5.6.1 Key issue details 22

5.6.2 Threats 23

5.6.3 Potential requirements 23

6 Proposed solutions 24

6.0 Mapping of solutions to key issues 24

6.1 Solution #1: Security protection mechanism for CAPIF-8 reference point 26

6.1.1 Introduction 26

6.1.2 Solution details 26

6.1.3 Evaluation 26

6.2 Solution #2: CAPIF-8 reference point security 26

6.2.1 Introduction 26

6.2.2 Solution details 26

6.2.2.1 Mutual authentication 26

6.2.2.2 Protection of messages between ROF –AZF/CCF 27

6.2.3 Evaluation 27

6.3 Solution #3: Security procedures for CAPIF-8 reference points 27

6.3.1 Introduction 27

6.3.2 Solution details 27

6.3.3 Evaluation 27

6.4 Solution #4: resource owner authorized revocation 28

6.4.1 Introduction 28

6.4.2 Solution details 28

6.4.2.1 Authorization procedure 28

6.4.2.2 Revocation procedure 28

6.4.3 Evaluation 29

6.5 Solution #5: Security of resource owner authorization management and CAPIF-8 29

6.5.1 Introduction 29

6.5.2 Solution details 29

6.5.3 Evaluation 31

6.6 Solution #6: Security procedures for resource owner authorization management 31

6.6.1 Introduction 31

6.6.2 Solution details 31

6.6.2.1 Security procedure for obtaining resource owner authorization 31

6.6.2.2 Security procedure for authorizing the API invoker in RNAA 32

6.6.2.3 Security procedure for revoking resource owner authorization 33

6.6.3 Evaluation 33

6.7 Solution #7: RO permission/ management 34

6.7.1 Introduction 34

6.7.2 Solution details 34

6.7.2.1 Notifications for permissions / wakeup 34

6.7.3 Evaluation 36

6.8 Solution #8: Resource owner triggered revocation procedure 36

6.8.1 Introduction 36

6.8.2 Solution details 36

6.8.3 Evaluation 37

6.9 Solution #9: Resource owner authentication and authorization mechanism 38

6.9.1 Introduction 38

6.9.2 Solution details 38

6.9.3 Evaluation 39

6.10 Solution #10: resource-level and/or API-level authorization and revocation 39

6.10.1 Introduction 39

6.10.2 Solution details 40

6.10.2.1 Service operation/resource level authorization 40

6.10.2.2 Service operation/resource level revocation 40

6.10.3 Evaluation 40

6.11 Solution #11: Client initiated backchannel authorization (CIBA) based solution 40

6.11.1 Introduction 40

6.11.2 Solution details 41

6.11.3 Evaluation 42

6.12 Solution #12: Security method retrieval in CAPIF interconnect 42

6.12.1 Introduction 42

6.12.2 Solution details 42

6.12.2.1 Summary 42

6.12.2.2 Information flow 43

6.12.3 Evaluation 43

6.13 Solution #13: Requesting security information from another CCF in order to authenticate using TLS-PSK in CAPIF interconnect 44

6.13.1 Introduction 44

6.13.2 Solution details 44

6.13.2.1 Summary 44

6.13.2.2 Information flow 44

6.13.3 Evaluation 45

6.14 Solution #14: Authentication aspect in CAPIF interconnect when API invoker has not included CCF information 45

6.14.1 Introduction 45

6.14.2 Solution details 46

6.14.2.1 Summary 46

6.14.2.2 Information flow 46

6.14.3 Evaluation 47

6.15 Solution #15: Authorization token request handling in CAPIF interconnect 47

6.15.1 Introduction 47

6.15.2 Solution details 47

6.15.2.1 Summary 47

6.15.2.2 Information flow 48

6.15.3 Evaluation 50

6.16 Solution #16: Mapping an API invoker authorization request to the correct CCF in CAPIF interconnect 50

6.16.1 Introduction 50

6.16.2 Solution details 50

6.16.2.1 Introduction 50

6.16.2.2 Summary 51

6.16.2.3 Information flow 52

6.16.3 Evaluation 52

6.17 Solution #17: Security procedures for CAPIF interconnection 53

6.17.1 Introduction 53

6.17.2 Solution details 53

6.17.2.1 Security procedure for API invoker authentication and authorization using Method 3 in CAPIF interconnection 53

6.17.2.2 Security procedure for API invoker offboarding in CAPIF interconnection 55

6.17.3 Evaluation 55

6.18 Solution #18: API invoker authentication mechanism in CAPIF interconnection scenarios 56

6.18.1 Introduction 56

6.18.2 Solution details 56

6.18.2.1 TLS-PSK based authentication mechanism for CCF interconnection scenarios 56

6.18.2.2 TLS-PKI based authentication mechanism for CCF interconnection scenarios 57

6.18.3 Evaluation 58

6.19 Solution #19: API invoker authorization mechanism in CAPIF interconnection scenarios 58

6.19.1 Introduction 58

6.19.2 Solution details 58

6.19.3 Evaluation 60

6.20 Solution #20: Security method negotiation mechanism in CAPIF interconnection scenarios 61

6.20.1 Introduction 61

6.20.2 Solution details 61

6.20.3 Evaluation 62

6.21 Solution #21: Solution for CAPIF interconnection security 62

6.21.1 Introduction 62

6.21.2 Solution details 62

6.21.3 Evaluation 67

6.22 Solution #22: CAPIF interconnection 67

6.22.1 Introduction 67

6.22.2 Solution details 68

6.22.3 Evaluation 69

6.23 Solution #23: Security protection mechanism for CAPIF-6 and CAPIF-6e reference points 69

6.23.1 Introduction 69

6.23.2 Solution details 69

6.23.3 Evaluation 69

6.24 Solution #24: Security procedure for CAPIF interconnection 69

6.24.1 Introduction 69

6.24.2 Solution details 70

6.24.2.1 Security method negotiation in CAPIF interconnection 70

6.24.2.2 Authentication and authorization 71

6.24.2.2.1 Authentication and authorization with security method TLS-PSK or PKI 71

6.24.2.2.2 Authentication and authorization with security method TLS with OAuth token 72

6.24.3 Evaluation 74

6.25 Solution #25: Backend based solution for UE-deployed API invoker accessing resources not owned by that UE 74

6.25.1 Introduction 74

6.25.2 Solution details 74

6.25.3 Evaluation 76

6.26 Solution #26: Nested API invocation 76

6.26.1 Introduction 76

6.26.2 Solution details 76

6.26.3 Evaluation 77

6.27 Solution #27: Authorization for nested API invocation 78

6.27.1 Introduction 78

6.27.2 Solution details 78

6.27.2.3 Access token claims 80

6.27.3 Evaluation 80

6.28 Solution #28: Authenticating multiple API invokers of the same RO 80

6.28.1 Introduction 80

6.28.2 Solution Details 80

6.28.3 Evaluation 82

6.29 Solution #29: Enhancing authorization through finer granularity access token. 83

6.29.1 Introduction 83

6.29.2 Solution details 83

6.29.2.1 Summary 83

6.29.2.2 Information flow 84

6.29.3 Evaluation 85

6.30 Solution #30: Authentication of the origin API invoker in nested API invocation 85

6.30.1 Introduction 85

6.30.2 Solution Details 85

6.30.3 Evaluation 86

6.31 Solution#31: Authorization mechanism for nested API invocation 86

6.31.1 Introduction 86

6.31.2 Solution details 87

6.31.3 Evaluation 88

6.32 Solution #32: Validation of correct GPSI in API invoker information 89

6.32.1 Introduction 89

6.32.2 Solution Details 89

6.32.3 Evaluation 89

6.33 Solution #33: Onboarding of API Invoker residing in UE 89

6.33.1 Introduction 89

6.33.2 Solution details 90

6.33.3 Evaluation 91

6.34 Solution #34: UE-deployed API invoker accessing resources not owned by that UE 91

6.34.1 Introduction 91

6.34.2 Solution details 92

6.34.2.1 Obtaining resource owner authorization and authorization revocation information 92

6.34.2.2 UE-deployed API invoker accessing resources not owned by that UE 93

6.34.3 Solution evaluation 94

6.35 Solution #35: Onboarding of UE-hosted API invoker 94

6.35.1 Introduction 94

6.35.2 Solution details 94

6.35.3 Solution evaluation 95

6.36 Solution #36: Reusing existing mechanism to enable cross-UE authorization 96

6.36.1 Introduction 96

6.36.2 Solution details 96

6.36.3 Evaluation 96

6.37 Solution #37: Enabling mTLS between ROF and CCF using AKMA 96

6.37.1 Introduction 96

6.37.2 Solution details 96

6.37.3 Evaluation 98

6.38 Solution #38: Renewal of onboarding 98

6.38.1 Introduction 98

6.38.2 Solution details 98

6.38.3 Evaluation 100

6.39 Solution #39: ROF certificate generation 100

6.39.1 Introduction 100

6.39.2 Solution details 101

6.39.2 Evaluation 102

6.Y Solution #Y: <Title> 102

6.Y.1 Introduction 102

6.Y.2 Solution details 102

6.Y.3 Evaluation 102

7 Conclusions 103

7.1.1 Conclusions for KI#1.1 CAPIF-8 reference point 103

7.1.2 Conclusions for KI#1.2 Resource owner authorization management 103

7.1.2.1 Authentication and authorization of the end points and security of transferred authorization data 103

7.1.2.2 Resource owner authorization data 103

7.1.2.3 Revocation 103

7.1.3 Conclusions for KI#1.3 Finer granular authorization 103

7.2 Conclusion for KI #2: CAPIF interconnection security 104

7.2.0 General 104

7.2.1 Conclusion for CAPIF 6/6e security 104

7.2.2 Conclusion for security method negotiation 104

7.2.3 Conclusion for API invoker authentication and authorization mechanism 104

7.3 Conclusion for KI #3: Authorizing API invoker on one UE accessing resources related to another UE 105

7.4 Conclusion for KI #4: Nested API Invocation 105

7.5 Conclusion for KI #5: Authenticating multiple API invokers of the same Resource Owner 105

7.6 Conclusion for KI #6: Onboarding security issues 105

Annex <X>: Change history 106

# Foreword

This Technical Report has been produced by the 3rd Generation Partnership Project (3GPP).

The contents of the present document are subject to continuing work within the TSG and may change following formal TSG approval. Should the TSG modify the contents of the present document, it will be re-released by the TSG with an identifying change of release date and an increase in version number as follows:

Version x.y.z

where:

x the first digit:

1 presented to TSG for information;

2 presented to TSG for approval;

3 or greater indicates TSG approved document under change control.

y the second digit is incremented for all changes of substance, i.e. technical enhancements, corrections, updates, etc.

z the third digit is incremented when editorial only changes have been incorporated in the document.

In the present document, modal verbs have the following meanings:

**shall** indicates a mandatory requirement to do something

**shall not** indicates an interdiction (prohibition) to do something

The constructions "shall" and "shall not" are confined to the context of normative provisions, and do not appear in Technical Reports.

The constructions "must" and "must not" are not used as substitutes for "shall" and "shall not". Their use is avoided insofar as possible, and they are not used in a normative context except in a direct citation from an external, referenced, non-3GPP document, or so as to maintain continuity of style when extending or modifying the provisions of such a referenced document.

**should** indicates a recommendation to do something

**should not** indicates a recommendation not to do something

**may** indicates permission to do something

**need not** indicates permission not to do something

The construction "may not" is ambiguous and is not used in normative elements. The unambiguous constructions "might not" or "shall not" are used instead, depending upon the meaning intended.

**can** indicates that something is possible

**cannot** indicates that something is impossible

The constructions "can" and "cannot" are not substitutes for "may" and "need not".

**will** indicates that something is certain or expected to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**will not** indicates that something is certain or expected not to happen as a result of action taken by an agency the behaviour of which is outside the scope of the present document

**might** indicates a likelihood that something will happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

**might not** indicates a likelihood that something will not happen as a result of action taken by some agency the behaviour of which is outside the scope of the present document

In addition:

**is** (or any other verb in the indicative mood) indicates a statement of fact

**is not** (or any other negative verb in the indicative mood) indicates a statement of fact

The constructions "is" and "is not" do not indicate requirements.

# 1 Scope

The present document investigates the security and privacy impacts of the procedures introduced in the study on CAPIF Phase 3. Specifically, it covers the following:

- Resource owner authorization management

- CAPIF interconnection security

- Authorizing API invoker on one UE accessing resources related to another UE

- Nested API invocation

- Authentication and authorization of multiple API invokers

Note: More security and privacy aspects are up to the study progress in SA6.

# 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 23.222: "Common API Framework for 3GPP Northbound APIs".

[3] 3GPP TR 23.700-22: "Study on CAPIF Phase 3".

[4] 3GPP TS 33.122: "Security aspects of Common API Framework (CAPIF) for 3GPP northbound APIs".

[5] IETF RFC 8693: “OAuth 2.0 Token Exchange”

[6] 3GPP TS 33.310: "Network Domain Security (NDS); Authentication Framework (AF)".

[7] 3GPP TS 33.210: "Network Domain Security (NDS); IP network layer security".

[8] 3GPP TS 33.220: " Generic Authentication Architecture (GAA); Generic Bootstrapping Architecture (GBA)".

[9] 3GPP TS 33.535: " Authentication and Key Management for Applications (AKMA)based on 3GPP credentials in the 5G System (5GS)".

[10] IETF RFC 6749: "The OAuth 2.0 Authorization Framework".

[11] OpenID Connect Client-Initiated Backchannel Authentication Flow 1.0.

[12] IETF RFC 7515: “JSON Web Signature (JWS)”

[13] 3GPP 23.700-95: "Study on application enablement aspects for subscriber-aware northbound API access"

# 3 Definitions of terms, symbols and abbreviations

## 3.1 Terms

For the purposes of the present document, the terms given in 3GPP TR 21.905 [1] and the following apply. A term defined in the present document takes precedence over the definition of the same term, if any, in 3GPP TR 21.905 [1].

example: text used to clarify abstract rules by applying them literally.

**Resource owner authorization:** The permission by the resource owner to allow the API invoker to access the resource owner’s resource via the northbound API.

## 3.2 Symbols

For the purposes of the present document, the following symbols apply:

<symbol> <Explanation>

## 3.3 Abbreviations

For the purposes of the present document, the abbreviations given in 3GPP TR 21.905 [1] and the following apply. An abbreviation defined in the present document takes precedence over the definition of the same abbreviation, if any, in 3GPP TR 21.905 [1].

<ABBREVIATION> <Expansion>

# 4 High-level architectures

TS 33.122 [4] provides the security architecture for RNAA based on the architecture specified in TS 23.222 [2].

## 4.1 High-level architecture for RNAA



Figure 4.1-1: High level functional architecture for CAPIF supporting RNAA

According to TS 23.222[2], the authorization function is an internal entity of the CAPIF core function.

The resource owner function (ROF) interacts with the authorization function in the CAPIF core function (CCF) via CAPIF-8. The resource owner function communicates with the authorization function in the CAPIF core function to manage resource owner consent.

## 4.2 High-level architecture for CAPIF interconnection

In line with TS 23.222 [2], figure 4.2-1 in the present document shows the architectural model for the CAPIF interconnection which allows API invokers of a CAPIF provider to utilize the service APIs from the 3rd party CAPIF provider.

The API invoker within the trust domain of CAPIF provider A onboads in CCF of CAPIF provider A.

The API invoker within the trust domain of CAPIF provider A interacts with the CAPIF core function of the CAPIF provider A via CAPIF-1 and discovers the service APIs of both CAPIF providers, and invokes the service APIs in the trust domain of CAPIF provider A via CAPIF-2 and invokes the service APIs in the trust domain of CAPIF provider B via CAPIF-2e.



Figure 4.2-1: High level functional architecture for CAPIF interconnection with multiple CAPIF provider domains

In line with TS 23.222 [2], figure 4.2-2 in the present document shows the architectural model for the CAPIF interconnection within the same CAPIF provider domain, which allows API invokers of CAPIF core function 1 to utilize the service APIs from CAPIF core function 2, where both CAPIF core function 1 and CAPIF core function 2 are hosted within the trust domain of the CAPIF provider A.

The API invokers of CAPIF core function 1 indicates that API invoker onboards in CAPIF core function 1.



Figure 4.2-2: High level functional architecture for CAPIF interconnection within a CAPIF provider domain

# 5 Key issues

## 5.1 Key Issue #1: Security of resource owner authorization management and CAPIF-8 reference point

### 5.1.0 Introduction

The key issue is addressing KI#1 and KI#3 of TR 23.700-22 [3] and consists of three sub-key issues for security of CAPIF-8 reference point, resource owner authorization management and finer granular authorization.

This key issue identifies the security aspects of resource owner authorization management and enhancements to the CAPIF architecture considering the Resource Owner Function (ROF) functionalities and its interactions with the CAPIF entities (e.g., CAPIF-8 related interactions) studied in TR 23.700-22 [3].

NOTE: Coordination with SA6 is needed.

### 5.1.1 Key Issue #1.1: CAPIF-8 reference point

#### 5.1.1.1 Key issue details

The security requirements, the security models, and the baseline security procedures for the CAPIF have been specified in TS 33.122 [4]. Based on CAPIF RNAA architecture specified in TS 23.222 [2], the CAPIF allows the resource owner to provide authorization to the API invocation for resource access. For that purpose, CAPIF-8 reference point was introduced to CAPIF RNAA. However, how to secure the transport of messages over CAPIF-8 was not specified in TS 33.122 [4] Rel-18, and part of the security procedures between the ROF and the authorization function/CCF supporting the Resource owner-aware Northbound API Access (RNAA) are left open in Release 18, as stated in the authorization procedures in the clause 6.5.3 of TS 33.122 [4]. It becomes apparent that the security aspects for the architecture enhancements are open issues in the Release 19 study, as also stated in the TR 23.700-22 [3].

#### 5.1.1.2 Security threats

Without integrity protection for CAPIF-8 reference point, messages over the CAPIF-8 reference point can be modified by attackers.

Without confidentiality protection for CAPIF-8 reference point, messages over the CAPIF-8 reference point can be sniffed by attackers.

Without the anti-replay attack mechanism for CAPIF-8 reference point, messages over the CAPIF-8 reference point can be replayed by attackers.

#### 5.1.1.3 Potential Security Requirement

The transport of messages over the CAPIF-8 reference point should be integrity protected.

The transport of messages over the CAPIF-8 reference point should be protected from replay attacks.

The transport of messages over the CAPIF-8 reference point should be confidentiality protected.

### 5.1.2 Key Issue #1.2: Resource owner authorization management

#### 5.1.2.1 Key issue details

KI#1 of TR 23.700-22 [3] is studying resource owner authorization management (e.g., authorizing access to the resource owner's resource or revoking the authorization of access to the resource owner's resource). There is a NOTE in TR 23.700-22 [3]:

NOTE: Aspects pertaining to the definition of resource owner consent/authorization over CAPIF-8 are in the scope of SA3, noting that the R18 security aspects of CAPIF supporting RNAA are specified in TS 33.122 [4].

This key issue studies how to authenticate and authorize the resource owner to provide resource owner authorization.

#### 5.1.2.2 Security threats

Without the authentication between resource owner and authorization server, malicious resource owner can impersonate victim resource owner to do resource owner authorization management.

#### 5.1.2.3 Potential security requirement

Mutual authentication between the authorization server and the resource owner should be supported.

CAPIF RNAA should support to authorize the resource owner to provide resource owner authorization.

CAPIF RNAA should support authorization of API invoker based on resource owner authorization and should support revocation of the resource owner authorization.

### 5.1.3 Key Issue #1.3: Finer granular authorization

#### 5.1.3.1 Key issue details

One additional aspect regarding the resource owner authorization management is about the granularity of the authorization information. According to TR 23.700-22 [3], one of the open issues is

How to align and manage access control that is more granular than simply granted/denied for service API (e.g., service operation level, resource level, service API originator/requestor details) with the provided resource owner consent to ensure appropriate usage of resource owner consent at the enabler layer.

It has been stated in the TR 23.700-22 [3] that the corresponding security aspects are in the scope of SA3. The objective of this sub-key issue is also to study how to secure authorization procedures with finer granularity.

#### 5.1.3.2 Security threats

Without finer granular authorization and revocation, the system can allow resource access more than necessary. This can cause service resources being abused.

#### 5.1.3.3 Potential Security Requirement

CAPIF RNAA should support finer granular authorization and revocation when API invoker access resource(s) of the resource owner provided by the service API.

## 5.2 Key issue #2: CAPIF interconnection security

### 5.2.1 Key issue details

The architectural model for the CAPIF interconnection has been introduced TS 23.222 [2] (see also clause 4.2 in the present document). It allows API invokers of a CAPIF provider to utilize the service API(s) from the 3rd party CAPIF provider and other CAPIF core function within the same CAPIF provider.

Figure 4.2-1 describes the CAPIF interconnection framework that connects CCFs in two different CAPIF provider domains.The API provider domain function (AEF) of one domain only communicates with the CCF in CAPIF provider domain B (CCF-B), where it is registered. It does not communicate with the interconnected CCF in CAPIF provider domain (CCF-A), but still must be able to provide AEF service APIs to an API invoker onboarded at CCF-A. Therefore, one target of this key issue is to study how the API invoker onboarded to CCF-A is autheticated and authorized to access API services of the AEF registered to CCF-B.

Figure 4.2-2 describes the CAPIF interconnection framework that connects CCFs in the same CAPIF provider domains. Another target of this key issue is study how one API invoker onboarded with CAPIF core function 1 (CCF-1) is authenticated and authorized to access AEF registered in CAPIF core function 2 (CCF-2).

### 5.2.2 Security threats

Without integrity protection for CAPIF-6/6e reference points, messages over the CAPIF-6 and CAPIF-6e reference points can be modified by attackers.

Without confidentiality protection for CAPIF-6/6e reference points, messages over the CAPIF-6 and CAPIF-6e reference points can be sniffed by attackers.

Without the anti-replay attacks mechanism for CAPIF-6/6e reference points, messages over the CAPIF-6 and CAPIF-6e reference points can be replayed by the attackers.

Without the API invoker authentication mechanism in CAPIF interconnection scenarios, a malicious API invoker can impersonate another victim API invoker to access service API(s) registered in the other CCFs.

Even if the API invoker is authorized by the CCF which it’s onboarded with, if there is no sufficient API service authorization and verification in CAPIF interconnection scenarios, this API invoker can still invoke AEF's service APIs registered in the other CCFs and get sensitive information (e.g., user's location information) without authorization.

Without the API invoker authorization revocation mechanism in CAPIF interconnection scenarios, the CAPIF system cannot revoke the authorization for API invoker accessing service API(s) registered in the other CCFs.

### 5.2.3 Potential security requirements

Potential security requirements for CAPIF interconnection are as followed:

1. The CAPIF should support mutual authentication between API invoker and AEF when AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios.

2. The API invoker should support retrieval of the security method needed for accessing service APIs when these AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios.

3. The CAPIF should support authorization and revocation of the API invoker in CAPIF interconnection scenarios.

4. The transport of messages over the CAPIF-6 and CAPIF-6e reference points should be integrity protected.

5. The transport of messages over the CAPIF-6 and CAPIF-6e reference points should be protected from replay attacks.

6. The transport of messages over the CAPIF-6 and CAPIF-6e reference points should be confidentiality protected.

7. The CAPIF should support mechanisms for mutual authentication between CCFs over the CAPIF-6/6e reference point.

NOTE: Coordination with SA6 is needed.

## 5.3 Key Issue #3: Authorizing API invoker on one UE accessing resources related to another UE

### 5.3.1 Key issue details

This key issue addresses the security aspects of TR 23.700-22 [3] KI #6.

It studies the security aspects for the case that API invoker(s) are deployed on one UE and requests to access resources (hosted in the network) related to another UE (e.g., application client on UE is fetching location of another UE or setting QoS for PDU sessions of another UE).

As specified in 3GPP TS 23.222 [2], the API invoker may be deployed in any of the following ways:

a. API invoker may be deployed as AF on the UE (i.e. 3rd party application).

b. API invoker may be deployed as AF on the UE supporting several other 3rd party applications deployed on the UE.

c. API invoker may be deployed on the network as AF.

So far, only a UE accessing its own resources is considered if the API invoker is on a UE. Resource owner-aware northbound API access (RNAA) defined in TS 33.122 [4] only supports authorizing API invoker on one UE to request resources related to the same UE.

Therefore, it is proposed to study how to authorize an API invoker on one UE to access resources related to another UE.

The use case in Annex A.2 of TR 23.700-95 [13] is UE-originated API invocation (Location tracking) where the application running on a UE initiates the procedure and the resource owner can be not using the application at the API invocation time. In CAPIF RNAA, only client credential and authorization code flow (w/o PKCE) are supported. In the client credential flow, it is assumed that the resource owner authorization information is available. In the authorization code flow, it is assumed that the client (API invoker) can access to the user agent of the resource owner and direct the resource owner to the authorization server.

### 5.3.2 Security threats

RNAA only supports authorizing API invoker on one UE to request resources related to the same UE. Hence, the CAPIF can only select non-RNAA (i.e., authorization procedure without resource owner involvement) based authorization mechanism for API invoker on one UE request to access resources related to another UE. This may lead to information leakage of the resource owner if the API invoker is authorized without engaging of the resource owner.

Without a proper security mechanism, unauthorized API invokers can access to resources related to a UE, which potentially results in sensitive information leakage and unauthorized modification to the resources accessed by northbound APIs.

### 5.3.3 Potential security requirements

CAPIF should support a mechanism for authorization of the API invoker on one UE to access resources related to another UE.

## 5.4 Key issue #4: Nested API invocation

### 5.4.1 Key issue details

In nested API invocation, the API exposing function (AEF) invokes API service(s) of another AEF which is in the same API provider domain as the first AEF. The procedure specified in clause 8.32 of TS 23.222 [2] in Rel-18 optimized authorization information query. The security aspect of that procedure is left to SA3 with the following note:

NOTE: The security aspects of this procedure are specified in TS 33.122 [4].

To provide security protection for the optimization procedure, the key issue derives a security requirement to mitigate potential security threats.

### 5.4.2 Security threats

If there is a vulnerability in the optimized authorization procedure in nested API invocation, an unauthorized API invoker can consume the API services, resulting in information leakage and unauthorized modification to the resources of the resource owner.

### 5.4.3 Potential security requirements

The AEF (destination AEF handling service API) should be able to authorize the AEF, requesting the API service, in an optimized way.

## 5.5 Key Issue KI#5: Authenticating multiple API invokers of the same Resource Owner

### 5.5.1 Key Issue details

This key issue addresses the security aspects of 29.700-22 KI #2, how multiple API invokers can use one or more AEFs exposing resources related to the same Resource Owner (RO) providing the credentials.

For example, in CAPIF RNAA context, this can enable a Resource Owner to allow one or several API invokers (e.g. gaming apps) running on the same UE to securely authenticate with one or multiple services provided by the AEF (e.g. location and/or QoS). In more detail, a gaming app wants to access the location. RO provides the security information to the gaming app to access the location service. Another API invoker on the same UE, e.g. the weather app, wants to access the location as well. The RO can provide the security information to the weather app to access the location service, without the need for both apps to request separate credentials from the CCF.

This key issue seeks to reducing the process of authenticating several API invokers of the same RO without introducing overhead.

### 5.5.2 Security threats

The same threats as for authentication and authorization in general apply, i.e. an unauthenticated and/or unauthorized API invoker can access to the AEF.

### 5.5.3 Security requirements

AEF should be able to authenticate and authorize multiple API invoker of the same RO.

## 5.6 Key Issue KI#6: Onboarding security issues

### 5.6.1 Key issue details

Onboarding is the one-time registration process to the CCF for enrolment of a new network application such that the application becomes a recognized API invoker of the CAPIF. After onboarding, the API invoker is enabled to subsequently access the CAPIF and the service APIs, i.e. to consume other APIs.

Key issue #5 of TR 23.700-22 [3] enhanced the support for API Invoker on-boarding/off-boarding and is normatively adopted in TS 23.222 [2], clause 8.1. Additional security concerns need to be checked.

In RNAA scenarios, API invoker residing in UE is introduced in CAPIF system. The CCF needs to be enabled to support the secure onboarding/offboarding of the API invoker residing in UE.

Whether the CCF can trust in the APIinvoker provided information details by the trust relationship established by mutual authentication will be studied in this key issue.

### 5.6.2 Threats

Malicious API invoker may impersonate victim API invoker to do the onboarding/offboarding.

### 5.6.3 Potential requirements

The CCF shall be able to support onboarding/offboarding of the API invoker residing in the UE.

The CCF shall be able to authenticate the API invoker residing in the UE.

# 6 Proposed solutions

## 6.0 Mapping of solutions to key issues

Table 6.0-1: Mapping of solutions to key issues

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Solutions | KI#1 | KI#1.1 | KI#1.2 | KI#1.3 | KI#2 | KI#3 | KI#4 | KI#5 | KI#6 |
| Solution #1: Security protection mechanism for CAPIF-8 reference point |  | X |  |  |  |  |  |  |  |
| Solution #2: CAPIF-8 reference point security |  | X |  |  |  |  |  |  |  |
| Solution #3: Security procedures for CAPIF-8 reference points |  | X |  |  |  |  |  |  |  |
| Solution #4: resource owner authorized revocation |  |  | X |  |  |  |  |  |  |
| Solution #5: Security of resource owner authorization management and CAPIF-8 | X |  |  |  |  |  |  |  |  |
| Solution #6: Security procedures for resource owner authorization management |  |  | X |  |  |  |  |  |  |
| Solution #7: RO permission/authorization management |  |  | X |  |  |  |  |  |  |
| Solution #8: Resource owner triggered revocation procedure |  |  | X |  |  |  |  |  |  |
| Solution #9: Resource owner authentication and authorization mechanism |  |  | X |  |  |  |  |  |  |
| Solution #10: resource-level and/or api-level authorization and revocation |  |  |  | X |  |  |  |  |  |
| Solution #11: Client initiated backchannel authorization (CIBA) based solution |  |  |  |  |  | X |  |  |  |
| Solution #12: Security method retrieval in CAPIF interconnect |  |  |  |  | X |  |  |  |  |
| Solution #13: Requesting security information from another CCF in order to authenticate using TLS-PSK in CAPIF interconnect |  |  |  |  | X |  |  |  |  |
| Solution #14: Authentication aspect in CAPIF interconnect when API invoker has not included CCF information |  |  |  |  | X |  |  |  |  |
| Solution #15: Authorization token request handling in CAPIF interconnect |  |  |  |  | X |  |  |  |  |
| Solution #16: Mapping an API invoker authorization request to the correct CCF in CAPIF interconnect |  |  |  |  | X |  |  |  |  |
| Solution #17: Security procedures for CAPIF interconnection |  |  |  |  | X |  |  |  |  |
| Solution #18: API invoker authentication mechanism in CAPIF interconnection scenarios |  |  |  |  | X |  |  |  |  |
| Solution #19: API invoker authorization mechanism in CAPIF interconnection scenarios |  |  |  |  | X |  |  |  |  |
| Solution #20: Security method negotiation mechanism in CAPIF interconnection scenarios |  |  |  |  | X |  |  |  |  |
| Solution #21: Solution for CAPIF interconnection security |  |  |  |  | X |  |  |  |  |
| Solution #22: CAPIF interconnection |  |  |  |  | X |  |  |  |  |
| Solution #23: Security protection mechanism for CAPIF-6 and CAPIF-6e reference points |  |  |  |  | X |  |  |  |  |
| Solution #24: Security procedure for CAPIF interconnection |  |  |  |  | X |  |  |  |  |
| Solution #25: Backend based solution for UE-deployed API invoker accessing resources not owned by that UE |  |  |  |  |  | X |  |  |  |
| Solution #26: Nested API invocation |  |  |  |  |  |  | X |  |  |
| Solution #27: Authorization for nested API invocation |  |  |  |  |  |  | X |  |  |
| Solution #28: Authenticating multiple API invokers of the same RO |  |  |  |  |  |  |  | X |  |
| Solution #29: Enhancing authorization through finer granularity access token |  |  |  | X |  |  |  |  |  |
| Solution #30: Authentication of the origin API invoker in nested API invocation |  |  |  |  |  |  | X |  |  |
| Solution #31: Authorization mechanism for nested API invocation |  |  |  |  |  |  | X |  |  |
| Solution #32: Validation of correct GPSI in API invoker information |  |  |  |  |  |  |  |  | X |
| Solution #33: Onboarding of API Invoker residing in UE |  |  |  |  |  |  |  |  | X |
| Solution #34: UE-deployed API invoker accessing resources not owned by that UE |  |  |  |  |  | X |  |  |  |
| Solution #35: Onboarding of UE-hosted API invoker |  |  |  |  |  |  |  |  | X |
| Solution #36: Reusing existing mechanism to enable cross-UE authorization |  |  |  |  |  | X |  |  |  |
| Solution #37: Enabling mTLS between ROF and CCF using AKMA |  |  | X |  |  |  |  |  |  |
| Solution #38: Renewal of onboarding |  |  |  |  |  |  |  |  | X |
| Solution #39: ROF certificate generation | X |  |  |  |  |  |  |  |  |

## 6.1 Solution #1: Security protection mechanism for CAPIF-8 reference point

### 6.1.1 Introduction

This solution is for KI #1.1 and addresses the security requirements for protecting CAPIF-8 reference point. This solution proposes to use TLS to protect CAPIF-8 reference points.

### 6.1.2 Solution details

TLS is to be used to provide integrity protection, replay protection and confidentiality protection. Security profiles for TLS implementation and usage follows the provisions given in TS 33.310 [6], Annex E.

### 6.1.3 Evaluation

This solution proposes to use TLS to provide security protection mechanism for CAPIF-8 reference point. No new security protection mechanism needed to be defined.

## 6.2 Solution #2: CAPIF-8 reference point security

### 6.2.1 Introduction

This solution addresses "Key Issue #1.1: CAPIF-8 reference point ".

The resource owner function (ROF) interacts with the authorization function (AZF) in the CAPIF core function (CCF) through the CAPIF-8 reference point. This solution proposes mutual authentication between the ROF and AZF. Besides, the messages exchanged between them are protected with integrity protection, replay protection and confidentiality protection.

NOTE: The AZF is part of the CCF and is used interchangeably with the CCF.

### 6.2.2 Solution details

#### 6.2.2.1 Mutual authentication

For authentication between a ROF and an AZF/CCF, mutual authentication based on TLS is proposed. The CCF is authenticated by the CCF certificates. The certificate profiles follow the TS 33.310 [6], clause 6.1.3a. The ROF authentication can be based on the ROF certificate, the pre-shared key or password etc., and is left for implementation.

NOTE: The structure of the PKI used for the certificate is out of scope of the present document.

#### 6.2.2.2 Protection of messages between ROF –AZF/CCF

TLS is used to provide integrity protection, replay protection and confidentiality protection for the CAPIF-8 interface.

The security profiles for TLS implementation and usage follow the provisions given in the clause 6.2 of the TS 33.210 [7].

### 6.2.3 Evaluation

The solution addresses the requirements of Key Issue #1.1.

TLS based mutual authentication is performed between the ROF and the CCF to establish a secure channel. The security protections for all messages transmitted through the channel include integrity protection, confidentiality protection and. replay protection.

The solution assumes the ROF can handle the credentials. It does not cover recovery in case that the ROF loses its credentials.

## 6.3 Solution #3: Security procedures for CAPIF-8 reference points

### 6.3.1 Introduction

This solution addresses the requirements identified in key issue #1.1.

It’s proposed to reuse Generic Bootstrapping Architecture (GBA) specified in TS 33.220 [8], or Authentication and Key Management for Applications (AKMA) specified in TS 33.535 [9] or TLS to perform mutual authentication and establish secure session between resource owner client (ROC) and CAPIF core function (CCF). It is up to the CAPIF provider domain administrator's policy to decide which method to use.

### 6.3.2 Solution details

If the CAPIF provider domain administrator decides to use GBA, procedures for mutual authentication and secure session establishment between ROF and CCF follow the clause 4 in TS 23.220 [8].

If the CAPIF provider domain administrator decides to use AKMA, procedures for mutual authentication and secure session establishment between ROF and CCF follow the clause 6 in TS 33.535 [9].

If the CAPIF provider domain administrator decides to use TLS, certificate based mutual authentication is performed between the ROF and CCF using TLS. Certificate based authentication shall follow the profiles given in TS 33.310 [6], subclauses 6.1.3a and 6.1.4a. The structure of the PKI used for the certificate is out of scope of the present document. TLS shall be used to provide integrity protection, replay protection and confidentiality protection. Security profiles for TLS implementation and usage shall follow the provisions given in TS 33.310 [6], Annex E.

GBA, AKMA and TLS can be simultaneously supported.

If the ROF is a web browser, ROF needs first to receive, generate or request a certificate. The certificate needs to be stored in the browser's certificate store to allow for mutual authentication between ROF and CCF.

### 6.3.3 Evaluation

Evaluation has not been provided.

## 6.4 Solution #4: resource owner authorized revocation

### 6.4.1 Introduction

This solution addresses the "Key Issue #1.2: Resource owner authorization management ". The resource owner (RO) authorization is based on the RNAA procedure specified in TS 33.122 [4]. For the RO revocation, this solution extends the procedure in the

TS 33.122 [4], where it is stated

*CCF can receive a revocation request message from the resource owner via the UE, resource owner function, web page etc.*

Specifically, in this solution, the revocation request message is described to complete the revocation procedure, given the CAPIF-8 reference point and relevant procedure is specified in the present document.

### 6.4.2 Solution details

#### 6.4.2.1 Authorization procedure

The procedure for authorization of resource owner follows the RNAA procedure as specified in clause 6.5.3 of TS 33.122 [4]. Specifically, the API invoker sends an access token request message to the CCF and the CCF issues the token with GPSI after checking specific to the authorization flow used. For example, the CCF may request RO authorization based on RFC 6794 if the authorization code flow is used.

#### 6.4.2.2 Revocation procedure

The procedure for revoking API invoker authorization initiated by the resource owner through the resource owner function (ROF) is given below extended from the procedure specified in clause 6.5.3.4 of TS 33.122 [4]:

1. The resource owner may trigger token revocation through the ROF. The ROF sends an authorization revocation request message to the CCF. The message shall include the service API ID, the GPSI, and other information related to the revoked token (e.g., the scope info).

NOTE: the GPSI is the identifier of the resource owner. . It is stored at the ROF through configuration or the authorization procedure.

2. With reference to step 2 in clause 8.23.4 of TS 23.222 [3], the request message includes in addition the GPSI of the UE, on which the ROF resides.

3. With reference to step 3 in clause 8.23.4 of TS 23.222 [3], the AEF may additionally determine whether to update the resource due to revocation, e.g., the API invoker is using the resource (i.e., QoS) that should be revoked after token revocation for the QoS service API, the AEF may inform PCF/SMF to modify the QoS level of corresponding PDU sessions after revocation.

4. The same as the step 4 in clause 8.23.4 of TS 23.222 [3].

5. Similar to the step 5 in clause 8.23.4 of TS 23.222 [3], the difference is invalidated authorization here is API invoker authorization for the resource owner/UE corresponding to the GPSI.

6. Similar to the step 6 in clause 8.23.4 of TS 23.222 [3], the difference is that the message is sent to the ROF.



Figure 6.4.2-1: Revocation procedure for initiated through ROF

### 6.4.3 Evaluation

This solution addresses the requirements of Key Issue #1.2.

This solution has two separate procedures for authorization and revocation.

This solution reuses the authorization procedure specified in clause 6.5.3 in the TS 33.122 [4] and extends the revocation procedure in clause 6.5.3.4 in TS 33.122 [4] to include steps related to CAPIF-8 interface. Therefore no impact is introduced by the authorization procedure. As to the revocation procedure, the ROF will send a revocation request message to the CCF and receives the corresponding revocation response message. In addition, AEF may determine whether to update the resource due to revocation, if yes, the AEF may send a message to the PCF/SMF to modify the QoS level. The impact to the AEF in this aspect is that the AEF needs to store information of original resources in order to update related operations.

The revocation procedure is to prevent token being misused by an attacker. Although a short-lived token can potentially mitigate the issue, it has restrictions for its applicability or use cases. For example, setting token lifetime may be challenging in many scenarios. On the one hand, a large number of tokens or frequent token refreshing may be needed if the token lifetime is short, which constrains the system with complexity and overhead. On the other hand, if the token lifetime is short, the token may be misused or compromised.

A short-lived token can be an alternative solution. As comparison with the alternative solution, this solution can provide revocation immediately and avoid frequent refreshing of token.

## 6.5 Solution #5: Security of resource owner authorization management and CAPIF-8

### 6.5.1 Introduction

The solution address KI#1 (i.e., KI#1.1, 1.2, 1.3 related to CAPIF-8 security, resource owner authorization management along with finer granular authorization aspects respectively).

### 6.5.2 Solution details

The solution proposes to use TLS based mutual authentication between the authorization server (i.e., CAPIF Core Function CCF) and the resource owner to secure the CAPIF-8 interface i.e., to provide confidentiality, integrity and replay protection (e.g., like TS 33.501 Clause 12.3). Specifically, for communication security over CAPIF-8 interface, TLS certificate based mutual authentication and secure session establishment can be performed. Certificate based authentication can follow the profiles given in TS 33.310 [6]. Alternatively, CAPIF-8 can be secured by reusing AKMA procedure described in TS 33.535[9] Clause 6, where the CCF takes the role of AKMA AF, and AKMA key can be used to derive a security key to protect CAPIF-8 interface i.e., to enable shared-key based mutual authentication and communication security establishment between the resource owner and the CCF. Figure 6.5.2-1 shows the resource authorization procedure to allow access to resources.

After CAPIF-8 interface security establishment in step 1, following steps (2-5) are performed to receive fine granular authorization and revoke authorization as required from an authenticated Resource owner to control access to resource(s) of a resource owner.

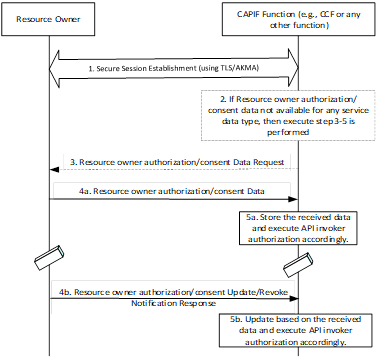


Figure 6.5.2-1 Resource authorization management procedure

1. Resource owner and the authorization function residing as part of CCF establishes secure session (using TLS/AKMA) as described above.

2. When the API invoker sends access token request related to service data relative to a resource owner and if no authorization/consent data is available, the CAPIF function may determine to fetch the related data from the resource owner as in step 3 or anytime the resource owner may perform step 1 and continue with step 4-5.

3. The CAPIF function sends a Resource owner authorization/consent data request along with UE ID (i.e. GPSI of the API Invoker), AF-ID, and Service data type IDs.

4a. The Resource owner can send to CAPIF Function, a Resource owner authorization/consent data notification, which can include Resource Owner ID (as GPSI), finer granular data Set (UE ID (i.e. GPSI) or API Invoker ID of the API Invoker as applicable), AF-ID, Service data type(s), and Consent Status (accept)).

5a. The CAPIF Function can store the received data Set locally along with GPSI. Further if any API invoker performs access token request, the CAPIF functions authorizes (e.g., can accept) the service access by considering the respective resource owner’s authorization/consent data and issuing the access token accordingly.

Resource owner authorization/consent data Revocation

4b. The Resource owner any time can send to CAPIF Function, a Resource owner authorization/consent data update/revoke notification, which can include Resource Owner ID (as GPSI), finer granular data Set (UE ID (i.e. GPSI of the API Invoker), AF-ID, Service data type(s), and Consent Status (update/revoke)).

5b. The CAPIF Function can store the received data Set locally along with GPSI. Further if any API invoker performs access token request, the CAPIF functions authorizes (e.g., can reject) the service access by considering the respective resource owner’s authorization/consent data accordingly.

### 6.5.3 Evaluation

The solution has the following impacts:

Resource owner and CCF need to use TLS based certificate or AKMA based shared key for mutual authentication and communication security establishment for CAPIF-8 interface.

CCF manages Resource owner authorization data along with UE ID (i.e. GPSI of the API Invoker), AF-ID, and Service data type IDs. Resource owner authorization data indicates whether the resource owner ‘allows/denys’ to expose its service data to any API Invoker. If not available, the CCF need to request and receive the Resource owner authorization data from the resource owner and perform finer granular authorization by considering the Resource owner authorization data such as the Resource Owner ID (as GPSI), finer granular data Set (UE ID (i.e. GPSI of the API Invoker), AF-ID, Service data type(s), and Status ‘allows (or) denys’ respectively.

The solution requires that the resource owner should be available and accessible by the CCF when the API invoker wants to invoke the API in case the Resource owner authorization data is not already available. The solution assumes that resource owner is accessible by the CCF when the CCF needs to interact with the resource owner.

## 6.6 Solution #6: Security procedures for resource owner authorization management

### 6.6.1 Introduction

This solution addresses the 2nd and 3rd requirements identified in key issue #1.2 resource owner authorization management.

The 6.6.2.1 enables the CCF to obtain the resource owner authorization from the ROF prior to API invoker invocation. The resource owner authorization received in 6.6.2.1 can be used to authorize the API invoker to access the resource owned by the resource owner, which is described in 6.6.2.2. The 6.6.2.3 enables the ROF to provide resource owner revocation information to the CCF, which may result in resource owner authorization change and API invoker authorization revocation.

### 6.6.2 Solution details

#### 6.6.2.1 Security procedure for obtaining resource owner authorization

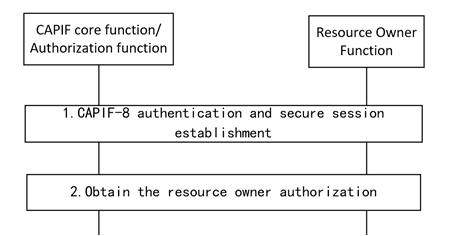


Figure 6.6.2.1-1: Procedure for obtaining resource owner authorization

1. The CCF and the ROF establish a secure session over CAPIF-8 reference point

NOTE: The details of secure session establishment is out of scope in this solution.

2. The resource owner provides the resource owner authorization information to the CCF via ROF. The CCF stores the resource owner authorization information along with the resource owner ID.

#### 6.6.2.2 Security procedure for authorizing the API invoker in RNAA

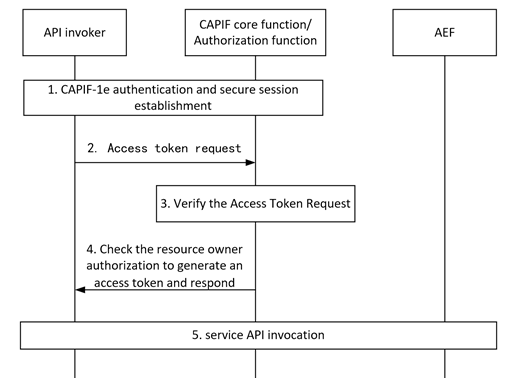


Figure 6.6.2.2-1: Procedure for authorizing the API invoker in RNAA

1. CAPIF-1e authentication and secure session establishment is performed as specified in subclause 6.3.1 in TS 33.122 [4].

2. The API invoker sends an Access token request message to the CCF with the resource owner ID.

3. The CCF shall verify the Access Token Request message per OAuth 2.0 [10] specification.

4. The CCF checks the resource owner authorization information and the resource owner ID received in step 2 to determine whether the API invoker is entitled to consume the API and allowed to access the resource owned by the resource owner. If it is, the CCF shall generate an access token specific to the API invoker and return it in an Access Token Response message. If there is no available specific resource owner authorization information, the CCF requests the resource owner authorization information from resource owner via ROF over secure CAPIF-8 connection.

NOTE: The CCF obtains the resource owner authorization information and may store it locally.

5. Authorization and authentication between the API invoker and the AEF in RNAA are performed as defined in 6.5..3 of TS 33.122 [4].

#### 6.6.2.3 Security procedure for revoking resource owner authorization

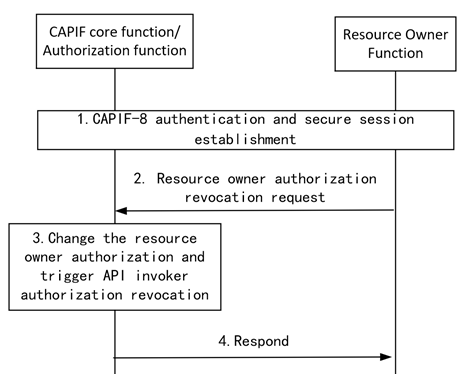


Figure 6.6.2.3-1: Procedure for revoking resource owner authorization

1. The CCF and the ROF establish a secure session over CAPIF-8 reference point

2. The resource owner sends a resource owner authorization revocation request to the CCF via ROF to provide resource owner authorization revocation information. The information includes the Resource Owner ID, service API information, API invoker ID to indicate which resource owner authorization is requested for revocation.

3. After receiving the resource owner authorization revocation information, the CCF updates the resource owner authorization information locally, and determines whether to trigger the revocation procedure as specified in clause 6.5.3.4 of TS 33.122 [4] to revoke the API invoker authorization. If it is, the CCF sends revoke API invoker authorization request to the AEF with the Resource Owner ID, service API information, API invoker ID.

4. The CCF sends a revocation response to the ROF.

### 6.6.3 Evaluation

This solution only addresses the 2nd and 3rd requirement of Key issue #1.2 and provide the security procedures for resource owner authorization management.

This solution propose that the CCF and the ROF establish a secure session over CAPIF-8 reference point . The resource owner provides the resource owner authorization information to the CCF via ROF. The CCF stores the resource owner authorization information along with the resource owner ID.

This solution proposes that the CCF check whether the API invoker is entitled to consume the API and allowed to access the resources of the resource owner, by using the resource owner authorization information.

This solution propose that the resource owner sends resource owner authorization revocation request to the CCF via ROF to trigger the revocation procedure as specified in clause 6.5.3.4 of TS 33.122 [4]. The resource owner authorization revocation information in request message includes the Resource Owner ID, service API information, API invoker ID.

## 6.7 Solution #7: RO permission/ management

### 6.7.1 Introduction

This solution is addressing KI#1.2.

CAPIF-8 interface between UE and CCF is being introduced by TS 23.122. ROF is handling this interface.

This solution addresses how to collect and manage permission notifications for resource owner authorization from the ROF over CAPIF-8 interface by subscribe to and unsubscribe from CAPIF events and by receive notifications from the CAPIF core function.

### 6.7.2 Solution details

RO authorization management needs a set of service operations to allow the ROF to collect and manage the permissions. The following sketches them and are additions to TS 23.222 clause 10.4.1 for CAPIF-8.

In general, the following can be added:

- Subscribe for permissions: allows ROF to subscribe for requests from Authorization Function (in CCF) to ask for permission on receiving RO details.

- Get pending permission: allows the ROF to obtain from the Authorization Function (in CCF) the pending permission requests associated with it, potentially based on being notified before and receiving information where to obtain it from.

- Notify event: allows the authorization function (in CCF) to notify the ROF about pending permissions.

- Wakeup: allows the Authorization Function (in CCF) to wake up the resource owner function (ROF) in the UE via AMF NAS operation which allows the UE to obtain pending permissions.

- Unsubscribe: allows a ROF to unsubscribe pending perrmissions events from Authorization function (in CCF).

- Report permission: allows ROF to post permissions to the Authorization Function URI based on permission requests received beforehand.

- Retrieve permission: allows ROF to retrieve the permission records that have been granted by it earlier.

#### 6.7.2.1 Notifications for permissions / wakeup

When ROF is waiting for notifications, different approaches can be envisioned.

**Triggering for RO permission via subscription notification**

- ROF subscribes for the pending permission requests with Authorization Function and waits.

- Upon Access Token Request from API Invoker (and based on optional notification criteria set by the resource owner), the Authorization Function (CCF) notifies ROF.

- The notification optionally includes the consent/permission information of API Invoker. If the notification does not contain this information, ROF explicitly fetches the details from Authorization Function and posts the consent responses to the URI provided by Authorization Function. CCF can store the information locally or at a repository service and then provide the access token to the API invoker.

The following provides a detailed flow:

Step 1: Resource Owner Function is successfully registered in CCF.

Step 2: API Invoker is onboarded successfully with CCF.

Step 3: Resource Owner Function subscribes with Authorization function (CCF) for resource owner consent requests notifications using SubscribeForConsent Request.

Step 4: Resource Owner Function receives 200 Ok and the subscriptionID.

Step 5: Resource Owner Function waits for the notifications from CCF.

Step 6: API Invoker sends access token request to CCF.

Step 7: CCF notifies the Resource Owner Function using Notify Event having only resourceOwnerID and subscriptionID. ROF gets the pending consent information from CCF.

Step 8: Alternatively, CCF sends notification to ROF with the consent information.

Step 9,10: Consent capture window is presented to the resource owner (via a web browser or application frontend) including the Purpose with other details and the resource owner provides the consent to ROF.

Step 11: ROF posts the consent to URI accessible to CCF.

Step 12,13: CCF stores the consent locally or at a repository service (e.g., UDM/UDR via NEF) and then provides the access token to API Invoker.



NOTE: CCF sends notification to the user agent of the application on the UE.

**Triggering for RO permission via NAS**

- service operation to allow the Authorization Function (in CCF) to wake up the resource owner function in the UE (taking into account notification restrictions provided by the resource owner during the subscription procedure) via AMF (NAS signalling). This service operation can be supported by AMF.

- ROF subscribes for the pending consent/permission request with Authorization Function.

- Upon Access Token Request from API Invoker, Authorization Function wakes up ROF via AMF using NAS signalling message, which can include the consent/pemission information of API Invoker. If the NAS message does not contain consent information, ROF explicitly fetches the consent requests from Authorization Function and posts the consent responses to the URI provided by Authorization Function.

### 6.7.3 Evaluation

This solution covers a management aspect necessary for the RO authorization/permission process, ie. how to ensure reachability of the ROF for authentication and authorization. In particular, how can the resource owner subscribe for the notifications.

In ROF and CCF new APIs need to be supported for subscription notifications.

For NAS based notifications, AMF and UE need to support new NAS IEs to support wakeup.

The NAS based solution has access network impact.

The solution needs to keep ROF (e.g., the web browser on UE) online to provide authorization information to CCF.

## 6.8 Solution #8: Resource owner triggered revocation procedure

### 6.8.1 Introduction

This solution addresses a part of KI#1.2 (i.e., CAPIF RNAA should support revocation of the resource owner authorization.).

### 6.8.2 Solution details

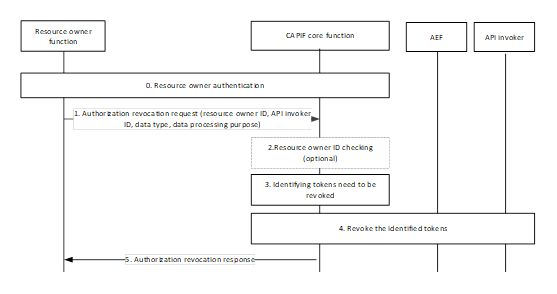


Figure 6.8.2 Resource owner triggered revocation procedure

1. The resource owner function obtains the address information of CCF from the network or API invoker. The ROF is not authenticated by the CCF. The ROF (e.g., the browser) can authenticate the CCF via the CCF’s certificate. Then the ROF can build TLS based on the CCF’s certificate (e.g., building the TLS tunnel based on a shared key negotiated via the public key of the CCF’s certificate). With the TLS tunnel, messages used to authenticate the resource owner can be securely delivered between ROF and the CCF. CCF authenticates the resource owner via interacting with resource owner function and gets the authenticated resource owner ID. The authentication of resource owner is based on application layer mechanism (e.g., password-based mechanism).

2. To revoke the resource owner related authorization, the resource owner function sends revocation related information to the CCF via CAPIF-8 reference point. The revocation related information may include the resource owner ID, the API invoker ID, the data type, and the data processing purpose.

3. The CCF checks the revocation related information against the authenticated resource owner ID.

If the CCF finds the authenticated resource owner ID is not identical to the resource owner ID in the revocation-related information, the CCF sends failure information to the resource owner function. The failure information indicates that the resource owner can only revoke authorization information related to the authenticated resource owner ID or it simply indicates that the revocation request fails.

4. If the resource owner ID in the revocation related information is identical to the authenticated one, the CCF identifies tokens need to be revoked using the resource owner ID.

The CCF identifies the tokens that need to be revoked by checking if the tokens contain the resource owner ID and the information (e.g., API invoker ID, data type, the data processing purpose) included in the revocation related information. For instance, if the resource owner needs to revoke the authorization for exposing UE ID, then the data type can be UE ID (e.g., GPSI), the data processing purpose can be exposing.

The CCF may map the data type and the data processing purpose to service information then the CCF checks if the token contains such service information.

The token contains the aforementioned information (e.g., resource owner ID, information included in the revocation related information, mapped service information) needs to be revoked.

The CCF interacts with AEF and API invoker to revoke the identified tokens. The CCF is required to start the revocation procedure as defined in clause 6.5.3.4 of TS 33.122 [4].

5. The CCF sends the revocation response to the resource owner function.

### 6.8.3 Evaluation

.

This solution addresses part of KI#1.2 (i.e., CAPIF RNAA should support revocation of the resource owner authorization).

This solution proposes that the resource owner to request CCF to revoke the authorization.

The benefit of this solution is enabling the resource owner to request CCF to revoke the authorization.

Impacts to CCF:

The CCF is required to receive revocation requests sent from the resource owner function. The revocation request includes the resource owner ID, the API invoker ID, the data type, and the data processing purpose.

This solution proposes an optional authorization revocation request checking mechanism that the CCF determine whether process the authorization revocation request by checking whether the resource owner ID in revocation related information is authenticated (i.e., The CCF is required to check the revocation related information against the authenticated resource owner ID.)

The CCF is required to identify the tokens that need to be revoked by checking if the tokens contain the resource owner ID and the information (e.g., API invoker ID, data type, the data processing purpose) included in the revocation related information.

The CCF is required to map the data type and the data processing purpose to service information then the CCF checks if the token contains such service information.

Impacts to ROF:

This solution proposes that the ROF sends authorization revocation request to the CCF to trigger the resource owner authorization revocation procedure. The ROF is required to send the revocation request to the CCF via CAPIF-8.

The revocation request includes the resource owner ID, the API invoker ID, the data type, and the data processing purpose.

## 6.9 Solution #9: Resource owner authentication and authorization mechanism

### 6.9.1 Introduction

This solution addresses a part of KI#1.2 (i.e., CAPIF RNAA should support authorizing the resource owner to provide resource owner authorization.).

### 6.9.2 Solution details

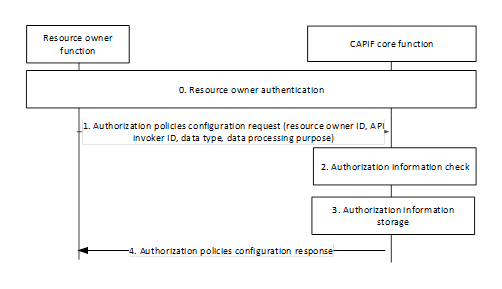


Figure 6.9.2 Resource-owner-related authorization

0. The resource owner function obtains the address information of CCF from the network or API invoker. The ROF authenticates CCF via the certificate of the CCF while the resource owner does not need to authenticate the CCF. ROF is not authenticated by the CCF. Specifically, the ROF (e.g., the browser) can authenticate the CCF via the CCF’s certificate. Then the ROF can build TLS based on the CCF’s certificate (e.g., building the TLS tunnel based on a shared key negotiated via the public key of the CCF’s certificate). With the TLS tunnel, messages used to authenticate the resource owner can be securely delivered between ROF and the CCF. The CCF authenticates the resource owner via interacting with the resource owner function and gets the authenticated resource owner ID. The authentication of the resource owner can be realized via application layer mechanisms (e.g., the resource owner can be authenticated via the password)

1. To configure authorization information, the resource owner function sends authorization-related information to the CCF. The authorization-related information may include the resource owner ID, the API invoker ID, the resource information (e.g., QoS, location), the data type, and the allowed data processing purpose. For instance, if the resource owner needs to authorize for exposing UE ID, then the data type can be UE ID (e.g., GPSI), the data processing purpose can be exposing.

2. The CCF checks the authorization-related information against the authenticated resource owner ID.

If the CCF finds the authenticated resource owner ID is not identical to the resource owner ID in the authorization-related information, the CCF sends failure information to the resource owner function. The failure information indicates that the resource owner can only configure authorization information related to the authenticated resource owner ID or it simply indicates that the authorization information configuration request fails.

3. If the resource owner ID in the authorization-related information is identical to the authenticated one, the CCF stores the authorization information related to the resource owner ID.

4. The CCF may map the data type and the data processing purpose to service information, then the CCF stores the authenticated resource owner ID, the authorization information (e.g., API invoker ID, resource, the mapped service information including service, service operation) included in the authorization-related information. The CCF informs the resource owner that the authorization information is stored.

### 6.9.3 Evaluation

This solution addresses part of KI#1.2 (i.e., CAPIF RNAA should support authorizing the resource owner to provide resource owner authorization.).

This solution proposes to use the authenticated resource owner ID to check if the resource owner is authorized to provide the corresponding authorization information in RNAA.

How a resource owner ID (i.e., GPSI) is considered authenticated and what is an authenticated resource owner ID referred in this solution is not addressed

Impacts to CCF:

The CCF is required to authenticate the resource owner via the application layer mechanism (e.g., password-based mechanism)

The CCF is required to check the authorization-related information against the authenticated resource owner ID.

If the CCF finds the authenticated resource owner ID is not identical to the resource owner ID in the authorization-related information, the CCF is required to send failure information to the resource owner function.

If the resource owner ID in the authorization-related information is identical to the authenticated one, the CCF is required to store the authorization information related to the resource owner ID.

The CCF is required to map the data type and the data processing purpose in the authorization-related information to service information (e.g., service, service operation)

Impacts to ROF:

The ROF is required to authenticate the CCF via the certificate while the resource owner does not need to authenticate the CCF. RO can obtain the authenticated CCF identity from the ROF.

The ROF is required to send authorization-related information to the CCF.

The ROF is required to receive the confirmation/failure information related to the configuration of authorization information from the CCF.

## 6.10 Solution #10: resource-level and/or API-level authorization and revocation

### 6.10.1 Introduction

This solution addresses "Key Issue #1.3: Finer granular authorization".

In this solution, authorization and/or revocation with finer granularity is proposed to support finer granularity of access control for service API, e.g., service operation level access and/or resource level access.

NOTE: the supported granularity at the service operation level or the resource level will be specified in TS 23.222 [3].

### 6.10.2 Solution details

#### 6.10.2.1 Service operation/resource level authorization

The procedure for authorizing an API invoker by the CCF/AZF is described as follows:

1. The API invoker sends an authorization request to the CCF/AZF. In addition to the API invoker ID, the authorization request includes the indication of the requested service operation/resource. For RNAA, the request also includes the GPSI of the UE.

2. The CCF/AZF verifies the API invoker ID and the requested service operation/resource, if available, match information, e.g., subscription information, stored at CCF/AZF.

3. The authorization result is sent to the API invoker if verification is successful. If a token is issued, the token claims include the granted service operation/resource, and GPSI for RNAA.

#### 6.10.2.2 Service operation/resource level revocation

The procedure for revoking authorization to an API invoker by the CCF/AZF is described as follows:

1. The API invoker/ROF/UE sends a revocation request to the CCF/AZF. In addition to the API invoker ID, the revocation request includes the indication of the requested service operation/resource. For RNAA, the request also includes the GPSI of the UE.

2. The CCF/AZF verifies the API invoker ID and, if successful, revokes the service operation/resource. The CCF/AZF updates its stored authorization information, e.g., updated service operation levels or resource levels.

3. The CCF/AZF responses to the API invoker/ROF/UE. The response includes indication whether revocation is successful, and the updated authorization information, e.g., updated service operation levels or resource levels. If a token is issued, the token claims include the updated service operation/resource levels, and GPSI for RNAA.

### 6.10.3 Evaluation

This solution addresses the requirements of the Key Issue #1.3.

This solution provides an authorization procedure and a revocation procedure with finer granularity. Both procedures are reusing the mechanisms specified in TS 323.222 [3]. The additions are IEs corresponding to the granularity levels included in the authorization/revocation request/response messages. The granularity levels of authorization and/or revocation will be aligned to TS 323.222 [3].

## 6.11 Solution #11: Client initiated backchannel authorization (CIBA) based solution

### 6.11.1 Introduction

This solution addresses the requirements identified in key issue #3 (Authorizing API invoker on one UE accessing resources related to another UE).

More precisely, this solution addresses the following requirements by allowing offline interaction with the resource owner, after triggering of API invocation by the API invoker.

In some cases, there can be no preconfigured authorization information and at the API invocation time the resource owner is not using the application which needs to direct the resource owner to the authorization server. UE-originated API invocation (Location tracking) captured in Annex A.2 of TR 23.700-95 [13] can be given as an example use case. . Client credentials (requiring a preconfigured authorization data) and authorization code flow (including PKCE) (requiring resource owner accessible by the API invoker at the API invocation time) are not enough to address these type of use cases. Utilization of client-initiated backchannel authorization (CIBA) [11] flow is proposed for such use cases.

### 6.11.2 Solution details

High-level steps of the solution are presented in Figure 6.11.2-1 and explained below. For details the CIBA flow [11] can be checked.

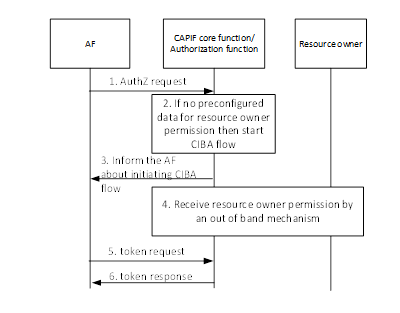


Figure 6.11.2-1: High-level procedure flow of the solution

1. The AF sends the authorization request to the CCF.

2. The CCF/ Authorization Function checks whether the request requires resource owner permission. If the request requires resource owner permission and there is no available resource owner permission data, then the CCF/ Authorization Function decides to return ACK to the AF.

3. The CCF/ Authorization Function return ACK to the AF about initiating the CIBA flow, so that the AF executes steps 5 and 6 regularly until the CCF/ Authorization Function obtains permission from the resource owner.

4. The CCF/ Authorization Function obtains permission from the resource owner via ROF . Note that this is an offline step, i.e., the execution of steps 5-6 does not have any impact on this step.

5-6. The AF regularly sends the token request to the CCF/ Authorization Function until the CCF/ Authorization Function obtains the permission from the resource owner. The CCF/ Authorization Function sends the response which depends on whether the CCF/ Authorization Function has obtained the permission from the resource owner.

### 6.11.3 Evaluation

The solution addresses the use cases where the API invoker and resource owner are residing on different UEs. The CCF/ Authorization Function needs to support the CIBA mechanism. There is no impact on the AEF.

Steps 5-6 are executed multiple times between the API invoker and CCF/ Authorization Function. These steps are existing steps in the CIBA specification.

The CIBA based solution especially is for the case that the API invoker and the resource owner are residing on different devices.

## 6.12 Solution #12: Security method retrieval in CAPIF interconnect

### 6.12.1 Introduction

This solution is addressing KI#2 on security aspects for CAPIF interconnect, specifically the security method retrieval to allow for authentication and authorization of the API invoker to the AEF.

An API invoker can fetch from CCF (via CAPIF-1/1e) information about the security method to access the API service (via CAPIF-2/2e). An API invoker can also request an API Service and the AEF is fetching the information about the API invoker from CCF (via CAPIF-3).

While this is specified for one security domain, in case of interconnect additional steps are needed because the API invoker is registered with one service provider's CCF while the AEF offering the API services is registered in another service provider’s CCF.

The solution describes how the API invoker is retrieving the supported security method of the exposing API in the CAPIF interconnect case, i.e. the API invoker wants to invoke the API from a different service provider than an AEF known at its local CCF.

### 6.12.2 Solution details

#### 6.12.2.1 Summary

The API invoker wants to consume APIs offered at an AEF outside of its trust domain. In its Security Method Request message to its local CCF (CCF-B) (via CAPIF-1/1e) it needs to include AEF details along with the security capability information to allow the local CCF to find the CCF where the AEF is registered. [With the current CAPIF framework the API invoker cannot get details from its own CCF (CCF-B) about the APIs offered at the requested AEF in the other security domain.]

Based on the AEF details received from the API invoker, CCF-B identifies the CCF responsible (i.e. CCF-A) where the AEF is registered and forwards the request to this CCF-A.

CCF-A selects a security method for the API service requested from its AEF, considering the information received from CCF-B, access scenarios and AEF capabilities. CCF-A sends a Security Method Response message back to CCF-B, providing the selected security method that allows to get authorization to access the AEF, and any security information related to the security method.

CCF-B sends the response to the API Invoker.

#### 6.12.2.2 Information flow

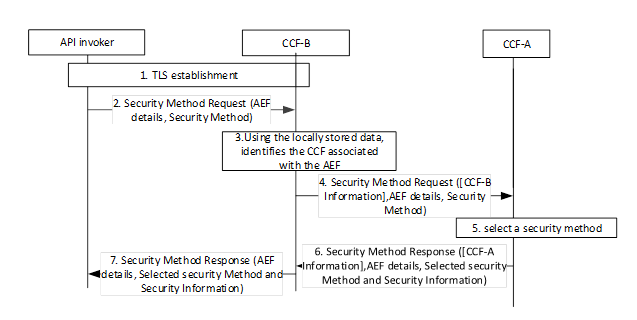


Figure 6.12.2.2-1: Information flow to retrieve security method in interconnect

Step 1: Mutual authentication based on client and server certificates shall be established using TLS between the API invoker and the CCF-B.

Step 2: The API invoker sends its CAPIF-2/2e security capability information along with AEF details to the CCF-B in the Security Method Request message. API Invoker security capability information is to indicate the list of security methods it supports and the AEF details from which the security method information retrieval is requested.

Step 3: CCF-B identifies th CCF (CCF-A) associated with AEF from its locally stored data.

Step 4: CCF-B sends the request to CCF-A, optionally appending CCF-B information.

Step 5: CCF-A shall select a security method to be used over CAPIF-2/2e reference point for requested AEF, considering the information received from CCF-B, access scenarios and AEF capabilities.

Step 6: The CCF-A shall send Security Method Response message to the CCF-B (optionally adding CCF-A information), indicating the selected security method for the AEF, any security information related to the security method.

Step 7: The CCF-B sends the Security Method Response to APIInvoker. Additionally, CCF-B also provides an indication to APIInvoker that AEF belongs to a different domain.

### 6.12.3 Evaluation

The solution address KI#2 by requesting the security method to be used with AEF directly to its responsible CCF (CFF-A in figure).

The solution affects CCFs functionalities and their inter-communication, by allowing one CCF to forward the security method request to a second CCF.

From key issue KI#2 CAPIF interconnect the aspect security method retrieval to allow for authentication and authorization of the API invoker to the AEF is addressed.

Additional communication between CCFs of different domains is needed.

Separate messages in Step 2 and 7 add new complexity at the API invoker side, such as indication to APIInvoker that AEF belongs to a different domain. According to TS 23.222, the other CCF can still belong to the same trust domain or different trust domain of the onboarded CCF. The solution addresses ‘different domain’ but can similarly be adapted to CCFs in the same domain.

## 6.13 Solution #13: Requesting security information from another CCF in order to authenticate using TLS-PSK in CAPIF interconnect

### 6.13.1 Introduction

This solution is addressing KI#2 on security aspects for CAPIF interconnect, specifically the authentication aspect between API invoker and AEF if in different security domains.

### 6.13.2 Solution details

#### 6.13.2.1 Summary

CCF-B and API invoker have obtained the security method that allows to authenticate to the AEF, and any security information related to the security method TLS-PSK. Hence, CCF-B and API invoker can derive AEFPSK based on the AEF’s API service details.

AEF receives an Authentication Initiation Request from APIInvoker, which includes the CCF-B information where the API invoker is registered. AEF requests security information of API invoker from the CCF-A it is registered with, mentioning the APIInvokerID and the CCF-B Information. CCF-A forwards the APIInvokerID to CCF-B which responds to CCF-A with the AEFPSK, which is forwarded to AEF.

API invoker and AEF authenticate using AEFPSK with the knowledge that CCF-B confirmed the APIInvokerID information.

#### 6.13.2.2 Information flow

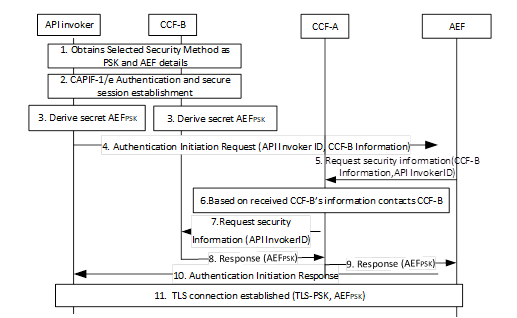


Figure 6.13.2.2-1: Information flow to allow authenticating API invoker to AEF in a different security domain

Step 1: APIInvoker gets the AEF details using Obtains\_Security method from CCF-B

Step 2: Mutual authentication based on client and server certificates shall be established using TLS between the API invoker and the CCF-B.

Step 3: APIinvoker and CCF-B derives AEF-PSK based on TLS master key used in step 2.

Step 4: APIInvoker sends Authentication Initiation Request to AEF based on AEF details received in step 1 and CCF-B information.

NOTE: The CCF-B information (address) is required so that CFF-A, when contacted by the AEF, can forward the Request Security information to the correct interconnected CCF (CCF-B). AEF needs to transmit CCF-B information to CCF-A to allow CCF-A to retrieve the PSK security information from CCF-B.

Step 5: AEF requests security information from CCF-A by passing the CCB’s information received in step 4 along with APIInvokerID.

Step 6,7: CCF-A based on CCF-B’s information received requests security information from CCF-B.

Step 8: CCF-B sends the response by providing AEF-PSK to CCF-A.

Step 9: CCF-A sends the response to AEF.

Step 10: AEF sends the Authentication Initiation Response to APIInvoker.

Step 11: TLS connection is established between APIInvoker and AEF using AEF-PSK.

Note: How to perform API invoker authentication using TLS-PKI is FFS.

### 6.13.3 Evaluation

The solution addresses the 1st requirement of KI#2 by enabling the AEF to request security information related to the security method TLS-PSK from the CCF-B via CCF-A based on the CCF-B’s information and the APIInvokerID. The AEF is registered with the CCF-A and the API invoker in onboard to the CCF-B.

The solution is providing a method for AEF in a second domain to verify the security information, e.g., AEFpsk, used by an API invoker from the first domain.

The solution introduces:

- minor changes in the communication between API Inovker and AEF.

- additional communication between interconnected CCFs to share security information.

## 6.14 Solution #14: Authentication aspect in CAPIF interconnect when API invoker has not included CCF information

### 6.14.1 Introduction

This solution is addressing KI#2 on security aspects for CAPIF interconnect, specifically the authentication and authorization aspect between API invoker and AEF if in different security domains, when API invoker does not provide its CCF details. In this case, AEF needs to first query CCF(s) to gain the necessary security details for authentication with the API invoker.

An API invoker registered at CCF-B wants to authenticate to an AEF (registered at CCF-A) to consume its API services. CCF-B and CCF-A are associated. Hence it knows the security method of AEF to enable an API invoker to access the API service (via CAPIF-1/e). The solution describes how the AEF in one security domain and the API invoker in the other security domain are enabled to establish a security session using TLS-PSK in the interconnect case. Specifically, it is addressed how to handle the case if the Authentication Initiation Request from APIInvoker does not include the source CCF details towards the AEF.

### 6.14.2 Solution details

#### 6.14.2.1 Summary

Prior to the authentication, CCF-B and API invoker have obtained the security method TLS-PSK that allows to authenticate to the AEF, and any security information related to this security method. Hence, CCF-B and API invoker can derive AEFPSK based on the AEF’s API service details.

AEF receives an Authentication Initiation Request from APIInvoker and requests security information of API invoker from the CCF-A where it is registered, mentioning the APIInvokerID. AEF needs to request security information of the APIInvoker from CCF-B. However, if CCF-B is connected to multiple CCFs and the AEF associated to CCF-A has no knowledge yet, that the APIInvoker is associated to CCF-B, it first needs to send an APIInvoker ownership query to the CCFs it is collaborating with.

Hence, since CCF-A does not know the API invoker, CCF-A needs to find the correct CCF (CCF-B) first. It therefore forwards the APIInvokerID to all interconnected CCFs. One of these CCFs responds (i.e. CCF-B) by confirming that it possesses information about the APIInvokerID. CCF-A gets the APIInvoker information including the AEFPSK from CCF-B that confirmed about the knowledge of APIInvokerID information and authenticates the API invoker to AEF.

#### 6.14.2.2 Information flow

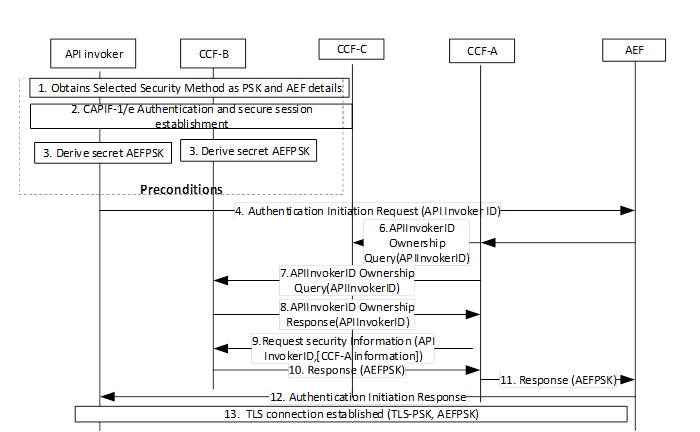


Figure 6.14.2.2-1: Information flow to allow establishment of TLS-PSK in interconnect

Step 1: APIInvoker gets the AEF details using Obtains\_Security method from CCF-B

Step 2: Mutual authentication based on client and server certificates shall be established using TLS between the API invoker and the CCF-B.

Step 3: APIinvoker and CCF-B derive AEF-PSK based on TLS master key used in step 2.

Step 4: APIInvoker sends Authentication Initiation Request to AEF based on AEF details received in step 1.

Step 5: AEF requests security information from CCF-A.

Step 6,7: CCF-A sends ApIInvokerID Ownership Query to all its interconnected CCFs (in the figure CCF-B and CCF-C).

Step 8: CCF-B responds to APIInvokerID Ownership Query request confirming that the APIInvokerID belongs to it.

Step 9: CCF-A requests the security information from CCF-B and optionally provides CCF-A ‘s information.

Step 10: CCF-B sends the response by providing AEF-PSK to CCF-A.

Step 11: CCF-A sends the response to AEF.

Step 12: AEF sends the Authentication Initiation Response to APIInvoker.

Step 13: TLS connection is established between APIInvoker and AEF using AEFPSK.

### 6.14.3 Evaluation

The solution is providing a method for AEF in a second domain to verify the security information, e.g., AEFpsk, used by an API invoker from the first domain.

The solution addresses the 1st requirement of KI#2 by enabling the AEF to gain security information for API invoker authentication through querying CCF(s) when API invoker does not provide its CCF details. In this case, AEF contacts the CCF-A it is registered with, CCF-A sends ApIInvokerID Ownership Query to all its interconnected CCFs. The CCF, to which the API Invoker belongs to responses (here CCF-B) and CCF-B onboard the API invoker responds security information related to TLS-PSK.

The CCF-A query is based on the API invoker ID received. Uniqueness of the API invoker ID in interconnection scenarios needs to be assured.

The solution introduces additional communication between interconnected CCFs to share security information.

## 6.15 Solution #15: Authorization token request handling in CAPIF interconnect

### 6.15.1 Introduction

This solution is addressing KI#2 on security aspects for CAPIF interconnect, specifically the authorization aspect that an access token needs to be requested by the CCF from the other CCF to which the AEF is associated.

### 6.15.2 Solution details

#### 6.15.2.1 Summary

An API invoker is onboarded at CCF-B, i.e. in the same trust domain, and wants to consume application services offered by an AEF outside of its trust domain. The AEF services are therefore not registered at the same CCF (CCF-B) as the API invoker but at a CCF-A. Hence, in interconnect an access token needs to be requested from CCF-A in a different trust domain. This is done by the CCF-B where the API invoker belongs to.

The API invoker includes onboarding secret and possibly also a client credential assertion (CCA) information into its access token request. CCF-B processes the information and forwards to CCF-A. Note, the onboard secret is not needed at CCF-A, hence it will be removed.

CCF-A verifies the access token request and CCA, if available. If successfully verified, CCF-A provides back to CCF-B an access token, with the client identifier in the access token claims set to the source API invoker ID as verified before.

Note: The assumption is that cross-domain certification is enabled, which allows CCF-A to verify the signature of the requesting API invoker known in CCF-B before creating the access token.

CCF-B provides the access token to the API invoker, which then can establish a TLS connection with AEF and invoke the northbound API with an OAuth 2.0 Access Token.

#### 6.15.2.2 Information flow

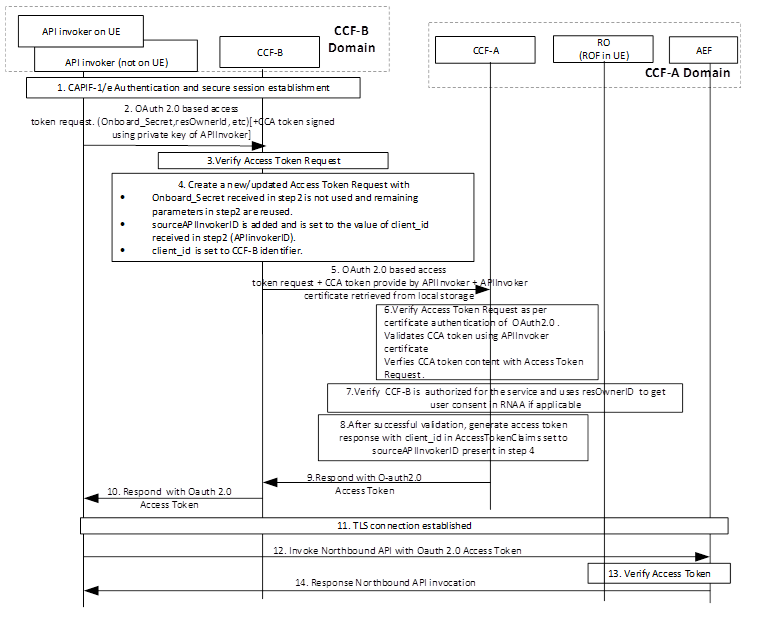


Figure 6.15.2.2-1: Information flow to retrieve security method in interconnect

Step 1: CAPIF-1/e authentication and secure session establishment is performed.

NOTE 0: APIInvoker can be residing on a UE or can be any outside the UE (e.g., an Application Function).

Step 2: After successful establishment of TLS session over CAPIF-1e, the API invoker sends to the CCF-B an Access Token Request message and optionally CCA token signed (using the APIInvoker’s private key) as per the OAuth 2.0 specification.

NOTE 1: The API invoker may include the CAPIF core function assigned API invoker ID and the Onboard\_Secret in the OAuth access token request message for the CAPIF core function to validate the access token request.

NOTE 2: The CCA token will be compliant to Json Web Token IETC RFC 7519 and includes the APIInvokerID, CCF ID, timestamp (iat) and the expiry time (exp). The lifetime of the CCA token is expected to be smaller than the expiry time associated with the OAuth 2.0 access token. The CCA token when received at CCF-A ensures the request is originated by APIInvoker.

Step 3: CCF-B verifies the Access Token Request message per OAuth 2.0 specification. In the above figure, Resource Owner is shown as part of CCF-A’s domain. Instead, if Resource Owner is part of CCF-B’s domain, then CCF-B can fetch the resource owner consent using RNAA (e.g. if consent is the applicable legal basis).

NOTE: How CCF-B gets resource owner authorization information using RNAA is part of solutions to KI#1.2.

Step 4: If CCF-B cannot successfully verify the Access Token Request message from APIInvoker, it provies an error message back.

If successfully verified, CCF-B creates a new Access Token request using the Access Token Request it received from APIInvoker. The new Access Token Request does not include the Onboard\_Secret as received in the access token request in step 2) anymore. The remaining parameters in step 2 are reused. In addition, the source, e.g. "sourceAPIInvokerID” is added and is set to the value of client\_id received in step 2 (APIinvokerID). client\_id is set to CCF-B identifier.

If the Resource Owner is part of CCF-B’s domain then and consent was retrieved, then CCF-B can also include the consent information.

If the Resource Owner is within CCF-A’s domain, and the consent needs to be captured, then CCF-B includes information about the APIInvoker identity (sourceAPIInvokerID) as well as consent-specific parameters (e.g., purpose of the data processing). By this, the RO can identify the party requesting access to the protected resources and the reason for that. The APIInvoker certificate, which is retrieved locally, is added as additional IE.

Step 5: CCF-B sends the newly generated OAuth access token request to CCF-A along with the CCA token, if available and as provided by the APIInvoker) and the APIInvoker certificate.

Step 6: CCF-A verifies the Access Token Request as per OAuth2.0 specification. (more details are below)

Step 7: CCF-A verifies if CCF-B is authorized for the service. It validates the CCA token, if available, with the received APIInvoker certificate. It validates whether the sourceAPIInvokerID and CCF-B id in CCA token are matching with the Access Token Request received. CCA token verification ensures the Access Token Request is originated from APIInvoker and also to authenticate APIInvoker at CCF-A.

CCF-A verifies if the APIInvoker is authorized for the service if RO is part of CCF-B and consent information if available in the token is valid, or if RO is part of CCF-A’s domain, and consent is applicable. In the latter case, it gets the consent from resource owner using RNAA.

Step 8: After successful validation, CCF-A generates an access token response with a token including client\_id in AccessTokenClaims set to sourceAPIInvokerID present in step 4.

Step 9: CCF-A sends the access token response to CCF-B.

Step 10: CCF-B forwards the access token response to APIInvoker.

Step 11-12: After successful authentication to the AEF on CAPIF-2e, the API invoker shall initiate invocation of a 3GPP northbound API with the AEF. The access token received in step 10 is included along with the northbound API invocation request.

Step 13: The API exposing function shall validate the access token. The AEF verifies the integrity of the access token by verifying the CAPIF core function signature. If validation of the access token is successful, the AEF verifies the API invoker's Northbound API invocation request against the authorization claims in access token, ensuring that the API Invoker has access permission for the requested service API.

Step 14: After successful verification of the access token and authorization claims of the API invoker, the requested northbound API is invoked, and the appropriate response is returned to the API invoker.

### 6.15.3 Evaluation

The solution partly addresses the 3rd requirement of KI#2 by enabling the API invoker to gain an access token issued by the CCF-A to which the AEF is associated from the CCF-B it registers with.

The solution is presenting a mechanism for API invoker to retrieve an access token which is understood by the AEF in a different domain.

The solution proposes that the CCF-B sends a new OAuth access token request to CCF-A along with the APIInvoker certificate and optional the CCA token. The new OAuth access token request is created using the Access Token Request received from APIInvoker, but with a new parameter indicating the API invoker ID and setting the client\_id to the CCF-B identifier.

The solution has an impact on the CCF functionalities of the Access token request procedure, and introduces additional communications between interconnected CCFs.

As for access token generation, firstly the solution recommends that the CCF-A validates the CCA token, if available, with the received APIInvoker certificate to ensure the Access Token Request is originated from APIInvoker and also to authenticate APIInvoker. Secondly, the solution proposes that the CCF-A verifies the consent information in the new OAuth access token request if the Resource Owner is part of CCF-B’s domain, or gets the consent information from the resource owner based on consent-specific parameters if the Resource Owner is within CCF-A’s domain.

The revocation of the API invoker in CAPIF interconnection scenarios is not addressed in the solution.

The solution has also impact on the API invoker for creating the CCA.

## 6.16 Solution #16: Mapping an API invoker authorization request to the correct CCF in CAPIF interconnect

### 6.16.1 Introduction

This solution is addressing KI#2 on security aspects for CAPIF interconnect for APIInvoker authentication and authorization using authorization code flow in CAPIF interconnection (CAPIF-6/6e).

The originator CCF of the API Invoker acts as client towards the interconnected CCF. The originator CCF serves several API invokers and can be connected to several CCFs. When using authorization code the originator CCF gets an authorization code from the CCF of another domain. The authorization code, when received from an API invoker in an access token request, needs to be correlated with the correct CCF. However, the CCF interconnecting with the API invoker’s CCF domain does only provide an authorization code.

The solution proposes how communication between CCF-B (originator CCF) and CCF-A (in the other domain) enables the originator CCF to identify in the access token request of the APIInvoker the correct CCF in the other domain, from which the APIInvoker received an authorization code before.

An example of ROF being in a different domain than API Invoker could be: A user registered with operator 1 is using the friend’s UE which is registered in a different domain (operator 2), to play a game. In this case the API Invoker, since it is installed on friend’s phone, would go to the CCF of operator 2, but to get the game user’s information, the API Invoker should access the information from operator 1, therefore we need the inter-communication between CCF-A and CCF-B.

### 6.16.2 Solution details

#### 6.16.2.1 Introduction

It is proposed that the originator CCF provides in the authorization code request information about the API invoker and its redirect URI. It receives from the CCF in the other domain the authorization code back and adds a CCF (CCF-A) identifier to the authorization code before sending it to the API invoker. Since the originator CCF needs to handle several API invoker requests, which may be redirected to different CCFs, this allows the originator CCF to send the authorization code together with an access token request from one API invoker to the correct CCF. Hence, the originator CCF is able to handle any subsequent access token request with an authorization code to the correct target CCF in the other domain.

NOTE 1: The following solution assumes that the API invoker and ROF, belongs to two different domains but use the same user-agent during the communication. In the case of API Invoker and ROF not being co-located, a CIBA flow could substitute the authorization code flow, which is however not in scope of this solution.

NOTE 2: Steps 5 and 6 are required only if the API Invoker did not previously granted authorization to the API Invoker.

#### 6.16.2.2 Summary

A CCF receiving from its API invoker a request for authorization code (step 2) adds the API invoker client identifier and its own identifier as it is now acting as client before forwardng the request towards the CCF in the other domain (step 4). Since the other CCF cannot redirect a response to the API invoker, the CCF URI needs to be added too.

Based on the redirect information, the CCF in the other domain provides the authorization code back to the originator CCF (step 7). The originator CCF processes the information before providing the authorization code within its authorization response towards the API invoker. I.e. upon receiving the authorization code, the originator CCF maintains a mapping of API invoker, authorization code and the CCF in the other domain. To map the authorization code (used between the API invoker and the CCF in the other domain, the originator CCF adds (step 8) within the authoriziation response an own identifier for the CCF towards the API invoker (as part of the authorization code). An example is given in the figure below.

In any (subsequent) access token request (step 10) from API invoker using the authorization code, the originator CCF then does the mapping (step11) to the actual identity of the CCF in the other domain and can forward the request via CAPF-6e.

Note: Figure update needed.

#### 6.16.2.3 Information flow

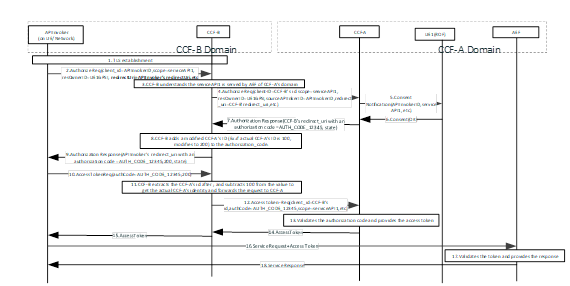


Figure 6.16.2.2-1: Identifying in an API access token request with authorization code the correct CCF in CAPIF interconnect

### 6.16.3 Evaluation

The solution addresses the KI#2 by extending already existing CAPIF authorization solutions to the interconnection scenario.

In particular, the solution enhances the CCF behaviour when receiving an Access Token Request. It introduces new communications between the interconnected CCFs but does not affect the API Invoker or the AEF.

This solution is specific to RNAA authorization code flow.

## 6.17 Solution #17: Security procedures for CAPIF interconnection

### 6.17.1 Introduction

This solution addresses the requirements identified in key issue#2 “CAPIF interconnection security”.

It is proposed to reuse clauses 6.6 and 6.10 of TS 33.122 [4] for securing CAPIF-6 and CAPIF-6e reference points respectively.

The solution 6.17.2.1 enhances the Authentication and Authorization using Method 3 specified in subclause 6.5.2.3 of TS 33.122 [4] for CAPIF interconnection scenario. It enables the API invoker onboarded to the CCF B to obtain the service API invocation authorization from the CCF A via CCF B in CAPIF interconnection. And the AEF registered to the CCF A can obtain security information from the CCF B via the CCF A to perform authentication and authorization with the API invoker. The enhancements are bolded..

If the API invoker previously accessed the AEF service APIs published by CCF A, when the API invoker is offboarding from the CCF B, the CCF B notifies the CCF A to delete the security information of the API invoker, which is mentioned in 6.Y.2.2. If the CCF A and the AEF do not delete the security information, the API invoker offboarded from the CCF B may access the AEF unexpectedly using the obtained security information.

### 6.17.2 Solution details

#### 6.17.2.1 Security procedure for API invoker authentication and authorization using Method 3 in CAPIF interconnection

Pre-condition:

1. The API invoker has onboarded to the CCF B.

2. The API invoker has discovered service APIs provided by an AEF using procedures specified in clause 8.25.3.2 and 8.25.3.3 of 3GPP TS 23.222 [i.2].

3. The AEF has registered to the CCF A.

4. The CCF A and the CCF B are connected to each other, and they have business agreement for service API authorization.

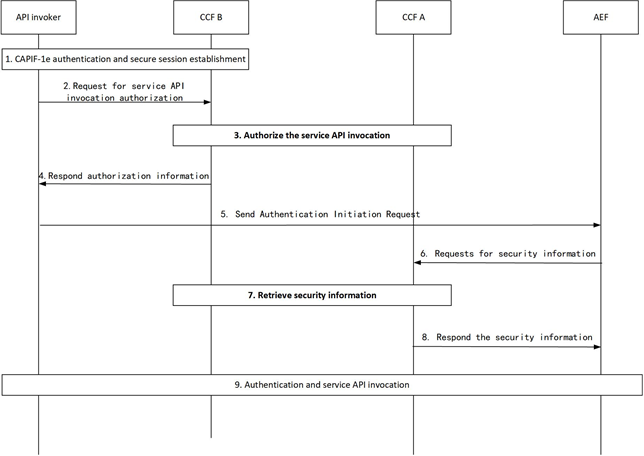


Figure 6.17.2.1-1: Procedure for API invoker authentication and authorization in CAPIF interconnection

1. CAPIF-1e authentication and secure session is established as specified in subclause 6.3.1 of TS 33.122 [4].

2. The API invoker sends an Access Token Request to the CCF B for obtaining authorization to access the service API published by CCF A via CAPIF-6/6e.

Note: In CAPIF interconnection scenarios, the “scope” in Access token request message is proposed to be REQUIRED.

3. The AEF service APIs information in “scope” indicate that these APIs are published by the CCF A, so the CCF B requests the CCF A to authorize the service API invocation of the API invoker. The request includes the API invoker ID and the AEF service APIs information. If the CCF A permits the service API invocation, the CCF A responds the CCF B with OAuth 2.0 access token defined in C.2.2 in TS 33.122 [4].

4. The CCF B returns Access Token Response message to the API invoker.

5. To invoke service APIs published by CCF A, the API invoker sends Authentication Initiation Request to the AEF, including API invoker ID and the CCF B ID.

6. The AEF requests for security information from the CCF A with the API invoker ID and the CCF B ID to perform authentication and secure interface establishment with the API invoker.

7. The CCF A retrieves security information based on API invoker ID and the CCF B ID. If it has no security information, the CCF A requests for security information from the CCF B with API invoker ID.

8. Receiving the security information from the CCF B, the CCF A responds the AEF. The security information is AEFPSK (TLS-PSK method) or root certificate of the API invoker (PKI method).

9. Authentication and secure interface establishment between the AEF and the API invoker is performed with the security information. And the AEF validates the access token following step 6~8 of clause 6.5.2.3 specified in TS 33.122 [4].

#### 6.17.2.2 Security procedure for API invoker offboarding in CAPIF interconnection

Pre-condition:

1. The API invoker previously accessed the AEF service APIs published by CCF A.

2. The CCF A and the AEF previously received security information of the API invoker from the CCF B.



Figure 6.17.2.2-1: Procedure for API invoker offboarding in CAPIF interconnection

1. Security procedure for API invoker offboarding specified in clause 6.8 specified in TS 33.122 [4] is performed.

2. The CCF B notifies the CCF A with the API invoker ID that this API invoker is no longer valid.

3. If the security information related to the API invoker has been stored, the CCF A and the AEF registered to the CCF A perform clause 6.8 specified in TS 33.122 [4] to delete the security information.

4. The CCF A sends an event notification acknowledge to the CCF B to indicate that the security related information associated with this API invoker is successfully deleted.

### 6.17.3 Evaluation

The solution partly addresses the requirements of KI#2.

For the 1st requirement, the solution proposes to enhance subclause 6.5.2.3 in TS 33.122 [4] that the AEF obtains the security information from the CCF B serving the API invoker via the CCF A serving the AEF based on the API invoker ID and the CCF B ID. The solution also proposes to enhance subclause 6.8 in TS 33.122 [4] to delete the security information stored in the CCF A and the AEF when the API invoker is offboarding.

The solution doesn’t address the 2nd requirement.

For the 3rd requirement, the solution proposes to enhance subclause 6.5.2.3 in TS 33.122 [4]. The main enhancements include:

- In step 2, the “scope” in Access token request message is proposed to be REQUIRED.

- In step 3, indicated by the AEF service APIs information in “scope”, the CCF B requests the CCF A to authorize the service API invocation including the API invoker ID and the AEF service APIs information.

For the 4th~7th requirements, the solution proposes to reuse clauses 6.6 and 6.10 of TS 33.122 [4] for securing CAPIF-6 and CAPIF-6e reference points respectively.

## 6.18 Solution #18: API invoker authentication mechanism in CAPIF interconnection scenarios

### 6.18.1 Introduction

This solution addresses part of KI#2 (i.e., The CAPIF should support mutual authentication between API invoker and AEF when AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios).

The existing API authentication mechanism defined in clause 6.5 of TS 33.122 [4] is enhanced to support the CAPIF inter-connection scenarios.

### 6.18.2 Solution details

#### 6.18.2.1 TLS-PSK based authentication mechanism for CCF interconnection scenarios

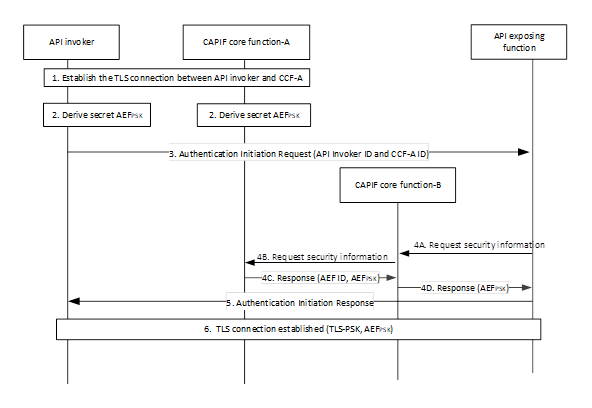


Figure 6.18.2.1 TLS-PSK based authentication mechanism for CCF interconnection scenarios

1. TLS connection is established between API invoker and CCF-A. API invoker sends the Service API interface information to the CCF to derive the AEFpsk.

2. After successful establishment of TLS between API invoker and the CCF-A, the API invoker and the CCF-A ID derive the key AEFPSK.

3. The API Invoker sends Authentication Initiation Request protected with the key AEFPSK to the AEF, including the CCF-A assigned API invoker ID and the CCF-A ID.

4. Based on the received CCF-A ID, the AEF requests for security information from CCF-A via the CCF-B to perform authentication and secure association establishment with the API invoker, if the AEF does not have a valid key. The request includes the AEF ID, API invoker ID and CCF-A ID (4A).

Upon receiving the request, CCF-B sends the request to the CCF-A, which is identified by the CCF-A ID in the request sent by AEF(4B).

The CCF-A sends the security information related to the chosen security method (TLS-PSK: AEFPSK) to the AEF by sending the AEFPSK and AEF ID to the CCF-B (4C and 4D).

5. After fetching the relevant security information (AEFPSK) for the authentication, the AEF sends Authentication Initiation Response message to API invoker to initiate the TLS session establishment.

6. The API Invoker and the AEF perform mutual authentication using the key AEFPSK and establish TLS session.

#### 6.18.2.2 TLS-PKI based authentication mechanism for CCF interconnection scenarios

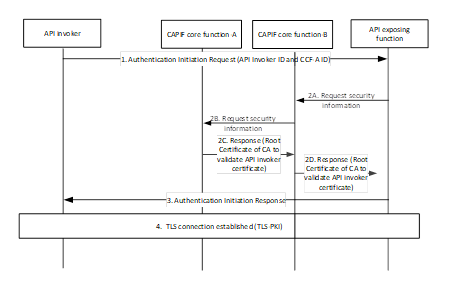


Figure 6.18.2.2: TLS-PKI based authentication mechanism for CCF interconnection scenarios

Authentication procedure is as follows.

1. The API invoker sends Authentication Initiation Request and its certificate to the AEF, including API invoker ID and CCF-A ID.

2A. The received CCF-A ID is an indication for the AEF not to use CCF-B certificate to validate the API invoker’s certificate but to request security information from CCF-A via the CCF-B to authenticate the API invoker. To request the root CA certificate related to the API invoker, the AEF sends CCF-A ID and API invoker ID to the CCF-B.

2B. To request the security information, the CCF-B sends the API invoker ID to the CCF identified by the CCF-A ID.

2C-2D. Based on the trusted business relationship between CCF-A and CCF-B, the CCF-A returns the API invoker's root CA certificate (e.g., CCF-A certificate) to the AEF via the CCF-B, which is used to validate the API invoker's certificate.

3. After fetching the root CA certificate, the AEF is able to authenticate the API invoker using the root CA certificate and send Authentication Initiation Response message to API invoker to initiate the TLS session establishment procedure.

4. The API Invoker and the AEF perform mutual authentication using the certificate and establish TLS session.

### 6.18.3 Evaluation

This solution addresses the following part of KI#2: the CAPIF should support mutual authentication between API invoker and AEF when AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios).

The benefit of this solution is that it realizes API invoker authentication via enhancing existing mechanisms.

Impacts to API invoker.

The API invoker is required to send CCF-A ID to the AEF.

Note: How API invoker can determine to send the CCF-A ID is not addressed.

Impacts to AEF:

AEF is required to send AEF ID, API invoker ID and CCF-A ID to the CCF-B.

The AEF ID and API invoker ID is used by the CCF-A to retrieve security information related to the specific AEF and API invoker. The CCF-A ID is used by the CCF-B to which the AEF registers to identify the CCF A and send the request to the CCF-A.

Impacts to CCF-B

CCF-B is required to request AEFPSK/root CA certificate from the CCF-A.

Impacts to CCF-A

CCF-A is required to send AEF the AEFPSK/root CA certificate via CCF-B.

## 6.19 Solution #19: API invoker authorization mechanism in CAPIF interconnection scenarios

### 6.19.1 Introduction

This solution addresses part of KI#2 (i.e., The CAPIF should support authorization of the API invoker in CAPIF interconnection scenarios).

In this solution, if the requested AEF is in the domain corresponding to the CCF-B while the API invoker 's onboarding is completed in CCF-A, CCF-B is used to generate token for the API invoker.

### 6.19.2 Solution details

Pre-conditions:

API invoker's onboarding is completed in CCF-A

The CCF-B is connected with the AEF

The authorization related request in this solution can be the enhancement of OAuth 2.0 based access token request defined in clause 6.5.2.3 of TS 33.122[4].

The authorization related response in this solution can be the enhancement of access token response defined in clause 6.5.2.3 of TS 33.122[4].

This solution reuses the service API discovery procedure defined in clause 8.25.3.3 of TS 23.222 [2]

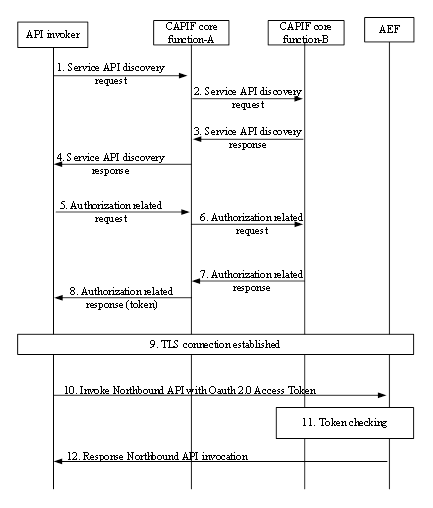


Figure 6.19.2 API invoker authorization mechanism in CAPIF interconnection scenarios

0. API invoker and CCF-A have established the TLS tunnel and completed the mutual authentication.

1. API invoker sends the service API discovery related request to the CCF-A.

2. CCF-A determines to send the service API discovery request to CCF-B based on local policy (e.g., A specific category of API needs to send the request to CCF-B).

3. CCF-B sends the discovered service API information to the CCF-A.

4. CCF-A sends the discovered service API information to the API invoker ID.

5. API invoker sends the authorization related request to the CCF-A. The request also includes the expected resource owner ID, expected service/service operation/service API, date type, and the corresponding data processing purpose.

6. To check if the authorization related request should be sent to the CCF-B, the following rules apply.

- If the CCF-A finds that the expected service/service operation/service API cannot be provided by AEF in its domain and finds that the expected service/service operation matches the discovered service API from CCA-B received in step 3, the CCF-A sends the authorization related request to the CCF-B.

- If the CCF-A finds that the expected service/service operation/service API is previously published by CCF-B, the CCF-A sends the authorization related request to the CCF-B.

The CCF-A may use the date type and the corresponding data processing purpose in addition to the expected service/service operation to identify the matched service API. Then the CCF-A can determine how to handle the authorization related request.

The request sent to CCF-B also includes the API invoker ID, expected resource owner ID, expected service/service API/service operation, date type, and the corresponding data processing purpose. Based on the trusted business relationship between CCF-A and CCF-B, the CCF-B trusts that the API invoker indicated by the API invoker ID is already authenticated and onboarded in the CCF-A.

7. If API invoker is authorized, CCF-B sends the token to the CCF-A. The token includes the API invoker ID, expected resource owner ID, expected service/service operation, date type, and the corresponding data processing purpose.

8. The CCF-A sends the token to the API invoker.

9. The API invoker authenticates to the AEF by establishing a TLS session with the AEF.

10. With successful authentication to the AEF, the API invoker shall initiate invocation of a 3GPP northbound API with the AEF by sending service operation request with the requested date type, and the corresponding data processing purpose. The access token received from the CCF-B is sent along with the northbound API invocation request as per OAuth 2.0.

11. The AEF validates the access token using CCF-B certificate or a key pre-shared with CCF-B.

12. After successful verification of the access token of the API invoker, the requested northbound API is invoked and the appropriate response shall be returned to the API invoker.

### 6.19.3 Evaluation

This solution addresses part of KI#2 (i.e., The CAPIF should support authorization of the API invoker in CAPIF interconnection scenarios).

The benefit of this solution is that it realizes API invoker authorization via enhancing existing client credentials flow.

Impacts to CCF-A:

If the CCF-A finds that the expected service/service operation/service API cannot be provided by AEF in its domain and finds that the expected service/service operation matches the discovered service API from CCA-B, the CCF-A sends the authorization related request to the CCF-B.

If the CCF-A finds that the expected service/service operation/service API is previously published by CCF-B, the CCF-A sends the authorization related request to the CCF-B.

Impacts to CCF-B:

CCF-B is required to receive authorization request from the CCF-A.

CCF-B is required to send the token to the API invoker via the CCF-A.

## 6.20 Solution #20: Security method negotiation mechanism in CAPIF interconnection scenarios

### 6.20.1 Introduction

This solution addresses part of KI#2 (i.e., The API invoker should support retrieval of the security method needed for accessing service APIs when these AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios.).

The existing secure method negotiation procedure defined in clause 6.3.1.2 of TS 33.122 [4] is enhanced to support the CAPIF inter-connection scenarios.

### 6.20.2 Solution details

Pre-conditions:

1. The API invoker is onboarded with the CCF-A.

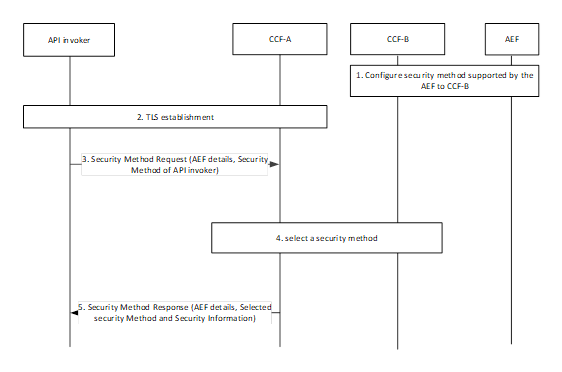


Figure 6.20.2 Selection of security method to be used in CAPIF interconnection scenarios

1. The AEF sends the supported security mechanisms to the CCF-B. The security mechanism may include TLS-PSK, TLS-PKI, TLS with OAuth token, OAuth client credential flow, authorization code flow, PKCE flow, etc.

2. Mutual authentication based on client and server certificates is established using TLS between the API invoker and the CCF-A.

3. The API invoker may send CAPIF-2/2e security capability information to the CAPIF core function in the Security Method Request message, indicating the list of security methods that the API invoker supports over CAPIF-2/2e reference point for each AEF. The security methods may include TLS-PSK, TLS-PKI, TLS with OAuth token, oauth client credential flow, authorization code flow, PKCE flow, etc.

4. If the CCF-A finds the target AEF is discovered by CCF-B, CCF-A sends common security methods that are supported by both CCF-A and the API invoker to CCF-B.

The CCF-B selects a security method to be used over CAPIF-2/2e reference point for each requested AEF, taking into account the information from the CCF-A and AEF capabilities.

The CCF-B sends Security Method Response message to the CCF-A, indicating the selected security method for each AEF.

5. The CCF-A sends Security Method Response message to the API invoker, indicating the selected security method for each AEF.

### 6.20.3 Evaluation

This solution addresses part of KI#2 (i.e., The API invoker should support retrieval of the security method needed for accessing service APIs when these AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios.).

The benefit of this solution is that it has no impact on API invoker.

Impacts to AEF:

AEF is required to provide the supported security mechanisms to the CCF-B.

Impacts to CCF-A

CCF-A is required to check if the target AEF is discovered by CCF-B.

If the CCF-A finds the target AEF is discovered by CCF-B, CCF-A is required to send the common security methods that are supported by both CCF-A and the API invoker to CCF-B.

Impacts to CCF-B

The CCF-B is required to select a security method to be used over CAPIF-2/2e reference point for each requested AEF, taking into account the information from the CCF-A and AEF capabilities.

The CCF-B is required to send the selected method to the CCF-A.

## 6.21 Solution #21: Solution for CAPIF interconnection security

### 6.21.1 Introduction

The solution address KI#2.

### 6.21.2 Solution details

The high-level functional architecture for CAPIF interconnection with multiple CAPIF provider domains provided in Clause 5.2.1 of the present document and TS 23.222 Clause 6.2.2 ‘Functional model description to support CAPIF interconnection’, are considered to describe further the security aspects as described below.

In the interconnection scenario, the security of CAPIF-6 reference points can be based on TLS to provide integrity protection, replay protection and confidentiality protection (i.e., same as described for CAPIF-3/4/5 reference points in TS 33.122 [4] Clause 6.6).

In the interconnection scenario, the security of CAPIF-6e reference points can either be based on reusing security procedures for CAPIF-3e/4e/5e refrence points described in TS 33.122 [4] Clause 6.10 (or) may be based on TLS to provide integrity protection, replay protection and confidentiality protection (e.g., like TS 33.501 Clause 12.3 on Protection of the NEF-AF interface).

The mutual authentication between API invoker and AEF (when AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios) over CAPIF-2 interface can reuse security procedure (i.e., authentication and authorization) described in TS 33.122 [4] Clause 6.4.

The mutual authentication between API invoker and AEF (when AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios) over CAPIF-2e interface can reuse security procedure (i.e., authentication and authorization) described in TS 33.122 [4] Clause 6.5.

The mutual authentication between CCFs over the CAPIF-6/6e reference point can reuse security procedures described in TS 33.122[4] Clause 6.6 and Clause 6.10 respectively.

The API invoker can retrieve the needed security method for accessing service APIs when these AEF service APIs are published via CAPIF-6/6e reference point in CAPIF interconnection scenarios by re-using the security method negotiation described in TS 33.122 [4] Clause 6.3.1.2.

The API invoker is onboarded to the CCF-A (called Onboarded CCF) based on TS 33.122 [4] Clause 6.1, where in interconnection case in step 5, AEF Authentication and authorization information additionally includes the 3rd party/designated CCF-B information (ID/address) per service API in case if any of the service API(s) belongs to CCF-B (called designated serving CCF to which the AEF belongs).

**Security Method negotiation**:

The security method to be used between API Invoker and the AEFs of CCF-B for CAPIF-2/2e security need to be selected by CCF-B in the case CAPIF interconnection as shown in Figure 6.21.2-1.



Figure 6.21.2-1: Selection of security method to be used in CAPIF-2/2e reference point

Precondition: In case of CAPIF interconnection scenario, the designated CCF (say CCF-A Onboarded CCF) in a CAPIF domain stores/maintains Service API/AEF information along with respective CCF-B information (IDs/address) for all the Service APIs/AEFs offered by the Other designated/3rd party CCF (say CCF-B designated serving CCF), where CCF-A and CCF-B are connected using CAPIF-6/6e interface.

1. Mutual authentication based on client and server certificates shall be established using TLS between the API invoker and the CCF-A same as described in TS 33.122 [4] Clause 6.3.1.2 step 1.

2. The API invoker may send CAPIF-2/2e security capability information to the CAPIF core function in the Security Method Request message, indicating the list of security methods that the API invoker supports over CAPIF-2/2e reference point for each AEF along with the 3rd party/designed CCF i.e., CCF-B information associated with each AEF (in case the AEF belongs to 3rd party CAPIF provider or belongs to a different CCF (say CCF B) which is other than the onboarded CCF (say CCF A).

3. The CCF-A shall select a security method to be used over CAPIF-2/2e reference point for each requested AEF which belongs to the CCF-A, taking into account the information from the API invoker in step 2, access scenarios and AEF capabilities as described in TS 33.122 [4] Clause 6.3.1.2 step 3.

4. The CCF-A based on the AEF details determines that part of the requested Service APIs/AEFs in step 2 belongs to different CCF (i.e., CCF-B) based on locally stored information and policy. If the CCF-A has information on CCF-B’s AEF’s Security method capabilities, then CCF-A can perform the security method selection (instead of step 5-7) for the CAPIF-2/2e reference point security.

5. The CCF-A sends an Interconnect Security Method Request to CCF-B which includes API Invoker ID, CCF-A Information (ID/address), API invoker’s subscribed CCF-B’s service API, access scenarios, AEF details (along with CCF-B information associated with each AEF), and Security Method.

6. The CCF-B checks the access policy on if the CCF-A is allowed to request the listed CCF-B associated AEF’s related security method request and if it is allowed, the CCF-B performs security method selection. Based on the API invoker's subscribed service APIs and/or AEF details, access scenarios (whether the API invoker access the AEF prior to service API invocation or upon the service API invocation) and AEF capabilities, the CCF-B choose the security method. Further the CCF-B stores the selected security method per service API/AEF for the API invoker ID along the CCF-A Information.

7. The CCF-B sends the Interconnection Security Method Response with the chosen security methods along with the information required for authentication of the API invoker at the AEF of CCF-B to the CCF-A. The information may include the validity time of the CAPIF-2e credentials and Security data sharing requirement indication per AEF.

8. The CCF-A sends Security Method Response message to the API invoker, indicating the selected security method for each AEF along with the designed CCF information associated with each AEF, Security data sharing requirement indication per AEF, any security information related to the security method. The API invoker use this method in the subsequent communication establishment with the API exposing function over CAPIF-2/2e reference point. Further the CCF-A also stores the Security data sharing requirement indication per AEF for the API Invoker to facilitate the security information transfer/forward from CCF-A to CCF-B for the CAPIF-2/CAPIF-2e security establishment between the API invoker and the AEF in the CCF-B domain. API Invoker also stores the AEF details along with CCF information, Selected security Method and Security Information, and Security data sharing requirement indication per AEF.

Authentication and Authorization

A. Method 1 – Using TLS-PSK



Figure 6.21.2-2: CAPIF-2e interface authentication and protection using TLS-PSK

1. CAPIF-1e authentication and secure session is established between API Invoker and it’s Onboarded CCF-A. The CCF-A shall provide the validity timer value for the key AEFPSK.

2. After successful establishment of TLS on CAPIF-1e, the API invoker and the CAPIF core function shall derive the key AEFPSK. The API invoker and the CAPIF core function-A starts the validity timer for the key AEFPSK.

3. The API invoker checks if the service APIs/AEF belong to the onboarded CCF (i.e., CCF-A) or belong to any 3rd party CCF (i.e., CCF-B). API Invoker includes CCF-A information (i.e., Onboarded CCF ID/address information) in Authentication Initiation request based on the Service APIs/AEF being provided by CCF-B (as indicated during API invoker Onboarding) and Security data sharing requirement indication per AEF received from the CCF-A (during Security Method Negotiation) to enable CAPIF 2/2e authentication and authorization between the API Invoker and the AEF of CCF-B.

4. The API Invoker shall send Authentication Initiation Request to the AEF, including the CAPIF core function assigned API invoker ID and Onboarded CCF Information.

5. The AEF sends the Request Security Information message which includes the API Invoker ID, CCF-A Information, Service API information to request for security information from the CCF-B to perform authentication and secure interface establishment with the API invoker.

6. The CCF-B if finds that it doesn’t have any security information related to the API Invoker, CCF-B contacts the CCF-A based on the CCF-A information (i.e., ID/address). The CCF-B sends the request Security information message to CCF-A, which includes the API Invoker ID, Service API information, and AEF details to request for security information from the CCF-B.

7-8. The CCF-A fetches the security information based on the received Service API information, AEF details and related to the API invoker ID and further provides the security information related to the chosen security method (TLS-PSK: AEFPSK) to the CCF-B in response message over CAPIF-6/6e interface. Further the CCF-B sends the security information related to the chosen security method (TLS-PSK: AEFPSK) in response message to the AEF over CAPIF-3 reference point. The CCF shall provide the remaining validity timer value for the key AEFPSK.

9. After fetching the relevant security information (AEFPSK) for the authentication, the AEF shall send Authentication Initiation Response message to API invoker to initiate the TLS session establishment. The AEF starts the validity timer based on the value received from the CCF-A or CCF-B in step 8.

10. The API Invoker and the AEF shall perform mutual authentication using the key AEFPSK and establish TLS session over the CAPIF-2e.

B. Method 2 – Using TLS-PKI

The step description of the authentication for interconnection case using TLS-PKI is same but only the security information exchanged is different, i.e., Root certificate of CA to validate API invoker certificate is sent to the AEF of CCF-B (in step 7 and 8) respectively. Further in step 10, the API Invoker and the AEF shall perform mutual authentication using certificates and establish TLS session over the CAPIF-2e. Certificate based authentication shall follow the profiles given in TS 33.310 [6].

B. Method 3 – TLS with OAuth token



Figure 6.21.2-3: CAPIF-2e interface authentication and protection using TLS with OAuth token

1. CAPIF-1e authentication and secure session establishment is performed.

2. After successful establishment of TLS session over CAPIF-1e, the API invoker sends an Access Token Request message to the CCF-A as per the OAuth 2.0specification which includes the API invoker ID, service API information, AEF details.

3. The CCF-A verify the Access Token Request message per OAuth 2.0 specification.

4. If the CCF-A successfully verifies the Access Token Request message, the CCF-A if finds that the service APIs and AEFs listed in step 2 belongs to a different 3rd party CCF-B, then the CCF-A sends an OAuth Access Token Request message to the CCF-B, where the OAuth Access Token Request message includes the API invoker ID, service API information, AEF details.

5-6. The CCF-B checks if the CCF-A is allowed to request the access token and/or CAPIF-2/2e authentication and authorization information related to the listed service APIs/AEF based on the local access policy stored for the CAPIF interconnection. If allowed, the CCF-B generate an access token specific to the API invoker, Issuer as CCF-B information and allowed service APIs related to the AEF and return it in an Access Token Response message. Meanwhile the CCF-B stores the API Invoker ID, Source CCF-A Info, AEF Information, Service API Information, and chosen security method (TLS with OAuth token).

NOTE 1: The API invoker may include the CCF-A assigned API invoker ID and the Onboard Secret in the OAuth access token request message for the CCF-A to validate the access token request.

7. The CCF-A sends the received OAuth access token to the API Invoker in the Access Token Response message.

8. The API Invoker checks if the service APIs/AEF belong to the onboarded CCF (i.e., CCF-A) or belong to any 3rd party CCF (i.e., CCF-B). API Invoker includes CCF-A information (i.e., Onboarded CCF ID/address information) in Authentication Initiation request based on the Service APIs/AEF being provided by CCF-B (as indicated during API invoker Onboarding) and Security data sharing requirement indication per AEF received from the CCF-A (during Security Method Negotiation) to enable CAPIF 2/2e authentication and authorization between the API Invoker and the AEF of CCF-B.

The API Invoker shall send Authentication Initiation Request to the AEF, including the CAPIF core function assigned API invoker ID and Onboarded CCF Information i.e., CCF-A ID/address.

The AEF sends the Request Security Information message which includes the API Invoker ID, CCF-A Information, Service API information, AEF details (IDs) to request for security information from the CCF-B to perform authentication and secure interface establishment with the API invoker, if the AEF does not have a valid security information.

The CCF-B if finds that it doesn’t have any security information related to the API Invoker, CCF-B contacts the CCF-A based on the CCF-A information. The CCF-B sends the request Security information message to CCF-A, which includes the API Invoker ID, Service API information, and AEF details to request for security information from the CCF-B.

The CCF-A fetches the security information based on the received Service API information, AEF details (IDs)) and related to the API invoker ID and further provides the security information related to the chosen security method (TLS with OAuth token, API invoker’s root CA certificate) in response message over CAPIF-6/6e interface. Further the CCF-B sends the security information related to the chosen security method (TLS with OAuth token, API invoker’s root CA certificate) in response message to the AEF over CAPIF-3 reference point.

After receiving the relevant security information for the authentication, AEF shall send Authentication Initiation Response message to API invoker to initiate the TLS session establishment procedure.

9. With successful authentication to the AEF on CAPIF-2e, the API invoker shall initiate invocation of a 3GPP northbound API with the AEF. The access token received from the CAPIF core shall be sent along with the northbound API invocation request as per OAuth 2.0.

10. The API exposing function shall validate the access token . The AEF verifies the integrity of the access token by verifying the CAPIF core function-B’s signature based on the CCF-B information. If validation of the access token is successful, the AEF shall verify the API invoker's Northbound API invocation request against the authorization claims in access token, ensuring that the API Invoker has access permission for the requested service API.

11. After successful verification of the access token and authorization claims of the API invoker, the requested northbound API shall be invoked, and the appropriate response shall be returned to the API invoker.

### 6.21.3 Evaluation

The solution has the following impacts:

The API invoker is provided with AEF details along with the respective CCF information during a successful onboarding in interconnection case.

The API invoker during a successful security method negotiation receives security information sharing required indication from the CCF-A (i.e., Onboarded CCF) if security information for CAPIF-2/2e authentication requires needs security information transfer CCF-A and CCF-B (i.e., designated serving CCF to which the AEF belongs).

The API Invoker sends onboarded CCF-A information (ID/address) in authentication initiation request to the AEF based on the CCF information received for the respective AEF details and based on the security information sharing indication, to let the AEF fetch the necessary security information from the onboarded CCF-A via the AEF’s CCF-B.

In case of TLS with OAuth token being selected, the CCF A provides API invoker ID, service API information/AEF details to CCF-B to let the CCF-B to generate finer granular access tokens.

## 6.22 Solution #22: CAPIF interconnection

### 6.22.1 Introduction

This solution addresses the requirements identified in key issue #2 (CAPIF interconnection security).

It is proposed to re-use the same security mechanisms specified for CAPIF-3/4/5 and CAPIF-3e/4e/5e reference points to address the security aspects of CAPIF-6 and CAPIF-6e reference points, respectively.

For the authentication and authorization of the API invoker by the AEF where the CCF serving the API invoker and serving the AEF are different, it is proposed to include interactions between two CCFs to allow the AEF to learn security and access control policy from the CCF serving the API invoker via the CCF serving the AEF and to allow the CCF serving the API invoker learn information about the AEF via the CCF serving the AEF to be used in the authorization of the API invoker.

It is proposed to re-use clause 6.3.1.2 of TS 33.122 [4] for the key issue requirement #2.

### 6.22.2 Solution details

For CAPIF-6 and CAPIF-6e reference points, same security mechanisms specified in clauses 6.6 and 6.10 of TS 33.122 [4] for CAPIF-3/4/5 and CAPIF-3e/4e/5e reference points are used, respectively.

For authentication and authorization of the API invoker in the CAPIF-2e reference point, the following figure and steps are the high-level presentation of the solution.

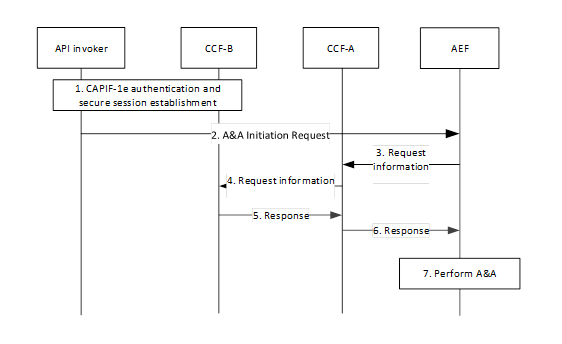


Figure 6.22.2-1: High level solution for authentication and authorization of the API invoker in the CAPIF-2e reference point.

CCF-B and CCF-A are respectively the CCF serving the API invoker and the CCF serving the AEF. It is assumed that the API invoker has been onboarded to CCF-B, the AEF is registered to CCF-A, and the API invoker has discovered service APIs provided by the AEF.

Step 1. The API invoker and CCF-B creates a secure session after successful authentication as specified in TS 33.122 [4]. If token-based authorization is used, then the CCF-B (or CCF-A) issues an access token with the help of CCF-A (or CCF-B). In the case that the token issuer is CCF-B, CCF-B communicates with the CCF-A to fetch authorization information and then issues the token. Otherwise, CCF-B sends information about the API invoker, the authorization decision is done by CCF-A and then CCF-A issues the token.

NOTE:  Consideration of all the authorization mechanisms supported in CAPIF is needed because the interconnection scenario is valid also for the legacy CAPIF (i.e., non-RNAA CAPIF).

Step 2. The API invoker interacts with the AEF for service API consumption. Before serving the request, the AEF starts to execute authentication and authorization of the API invoker.

Step 3-6. The AEF learns the security information to be used in the authentication of the API invoker from CCF-B with the help of CCF-A. The security information can be a pre-share key or a root certificate depending on the authentication method. The AEF also learns the access control policy from the CCF-A and CCF-B if token-based authorization is not used. In the case of token-based authorization, the AEF can learn a root certificate to be used the verification of the access token. If the token issuer is CCF-B, the AEF obtains the certificate or root certificate of CCF-B from CCF-A to be able to verify the token.

NOTE: Identification of the CCF-B by the CCF-A is not in the scope of this solution.

Step 7. The AEF performs the authentication and authorization of the API invoker.

### 6.22.3 Evaluation

The solution addresses the requirements of KI#2. It proposes to re-use the existing security mechanisms for CAPIF-6/6e reference point. It requires interaction between CCF-A and CCF-B for authorization information exchange.

There is no impact on the existing CAPIF mechanism for security method negotiation.

How the existing security method negotiation can be applied for the interconnection case steps is not described.

How the security information is fetched by the AEF’s CCF from the API invoker onboarded CCF for the case of CAPIF-2/2e security is not described.

## 6.23 Solution #23: Security protection mechanism for CAPIF-6 and CAPIF-6e reference points

### 6.23.1 Introduction

This solution proposes to use TLS to protect CAPIF-6 and CAPIF-6e reference points.

### 6.23.2 Solution details

Similar to CAPIF-3/4/5 reference points, for CAPIF-6 reference point,

- TLS shall be used to provide integrity protection, replay protection and confidentiality protection. The support of TLS is mandatory. Security profiles for TLS implementation and usage shall follow the provisions given in TS 33.310 [6], Annex E.

- Certificate based mutual authentication shall be performed between the CAPIF entities using TLS. Certificate based authentication shall follow the profiles given in TS 33.310 [6], subclauses 6.1.3a and 6.1.4a. The structure of the PKI used for the certificate is out of scope of the present document.

Similar to CAPIF-3e/4e/5e reference points, for CAPIF-6e reference point,

- TS 33.210 [7] shall be applied to secure messages on the reference points specified otherwise; and

- TS 33.310 [6] may be applied regarding the use of certificates with the security mechanisms of TS 33.210 [7] unless otherwise specified in the present document.

### 6.23.3 Evaluation

This solution proposes to use TLS to protect CAPIF-6 (similar to CAPIF-3/4/5 reference points) and CAPIF-6e (similar to CAPIF-3e/4e/5e reference points) reference points. No new security protection mechanism is needed to be defined.

## 6.24 Solution #24: Security procedure for CAPIF interconnection

### 6.24.1 Introduction

This solution addresses the security requirement of key issue#2.

### 6.24.2 Solution details

#### 6.24.2.1 Security method negotiation in CAPIF interconnection

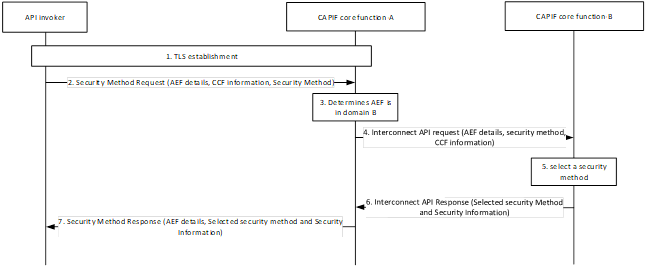


Figure 6.24.2.1-1: Selection of security method to be used in CAPIF-2/2e reference point

Detailed procedure is as follows:

Step 1: Mutual authentication based on client and server certificates shall be established using TLS between the API invoker and the CAPIF core function.

Step 2: The API invoker re-uses the service API invoker discovery procedure to get the list of service API information for which the API invoker has the required authorization. If the service APIs required are provided by the different domain than the onboarded CCF then the API invoker sends CAPIF-2e security capability information to the CAPIF core function in the Security Method Request message, indicating the API invoker ID, list of security methods that the API invoker supports over CAPIF-2/2e reference point, the service APIs it wants to access with the AEF identity information and the CCF information. The CCF information is provided by the onboarded CCF during onboarding.

Step 3: The CCF-A determines that the service APIs requested by the API invoker is provided by the AEF in CCF-B (another trust domain-B) based on the AEF details and CCF information.

Step 4: The CCF-A sends a security method request as interconnection API request to CCF-B with the AEF details i.e.API invoker ID, list of security methods that the API invoker supports over CAPIF-2/2e reference point and the service APIs it wants to access with the AEF identity information.

Step 5: The CCF-B selects a security method to be used over CAPIF-2e reference point for each requested AEF, taking into account the information from the API invoker and CCF-A and the AEF capabilities.

Step 6: The CCF-B provides a security method response, which is an interconnection API response to the CCF-A including AEF details, the selected security method and security information.

Step 7: The CCF-A sends the security method response message to the API invoker, including the AEF details, the selected security method and any security information related to the security method for each AEF in the domain B as per the request.

The API invoker uses the selected security method towards the AEF in the subsequent communication establishment with the API exposing function over CAPIF-2e reference point.

#### 6.24.2.2 Authentication and authorization

##### 6.24.2.2.1 Authentication and authorization with security method TLS-PSK or PKI

The following describes the authentication and authorization procedure during CAPIF Interconnect when security method 1 or 2 is selected.



Figure 6.24.2.2.1-1: Authentication and authorization procedure during CAPIF Interconnect – Method 1 and 2

Detailed procedure is as follows:

Step 1: CAPIF-1e authentication and secure session is established. The CAPIF core function-A provides the validity timer value for the key AEFPSK.

Step 2: After successful establishment of TLS on CAPIF-1e, the API invoker and the CAPIF core function-A derives the key AEFPSK.

The Key AEFPSK is bound to an AEF. The API invoker and the CAPIF core function starts the validity timer for the key AEFPSK.

Step 3: The API invoker re-uses the service API invoker discovery procedure to get the list of service API information for which the API invoker has the required authorization. If the service APIs required are provided by the different domain than the onboarded CCF then the API Invoker sends an Authentication Initiation Request to the AEF, including the CAPIF core function assigned API invoker ID and CCF identity information (source CCF) associated with the API invoker. The CCF information is provided by the onboarded CCF during onboarding.

Step 4: The AEF checks if it has a valid key material. Otherwise, the AEF requests for security information from the CAPIF Core Function-B to perform authentication and secure interface establishment with the API invoker, if the AEF does not have a valid key.

As the API invoker is registered to different trust domain (domain A), the AEF includes the API invoker Identity, CCF identity information, service APIs (shareable service APIs), AEF identity information in the request message to CCF-B.

Step 5: (Optional) The CAPIF Core Function-B, as the API invoker is from different trust domain does not have the required security materials. The CCF-B checks for the stored information on the shareable APIs based on the CCF identity information and received API invoker Identity.

Step 6: The CCF-B requests for security information from the CAPIF Core Function-A to perform authentication and secure interface establishment between the API invoker and API Exposure Function (AEF). The AEF includes the security method selected by it in the request message to the CCF-B.

Step 7: The CCF-A authorizes the CCF-B and AEF requesting the security material based on the received CCF information and AEF details (as it has service level agreement).

Step 8: Based on the received API invoker Identity and AEF selected security method the CCF-A retrieves the AEFPSK or the root certificate of the CA to validate the API invoker.

Step 9: The CCF-A provides the security information related to the chosen security method (TLS-PSK: AEFPSK or the root certificate of the CA to validate the API invoker) to the AEF over CAPIF-3 reference point.

Step 10-11: After fetching the relevant security information for the authentication, the AEF sends Authentication Initiation Response message to API invoker to initiate the TLS session establishment.

The API Invoker and the AEF performs mutual authentication using the key AEFPSK and establish TLS session over the CAPIF-2e.

After successful establishment of TLS on CAPIF-2e reference point, the API exposing function authorizes the API invoker's service API invocation request based on authorization information obtained from CAPIF core function.

##### 6.24.2.2.2 Authentication and authorization with security method TLS with OAuth token

The following describes the authentication and authorization procedure during CAPIF Interconnect when security method 3 is selected.

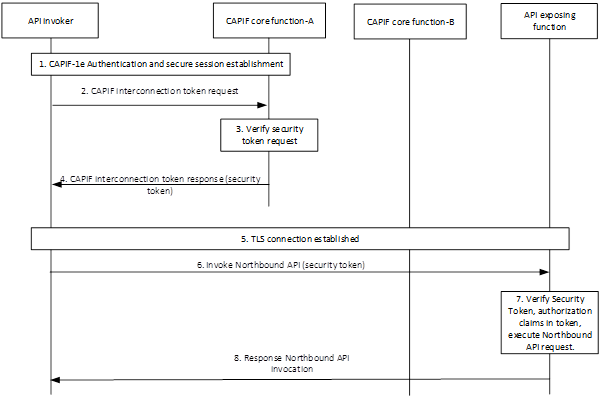


Figure 6.24.2.2-1: Authentication and authorization procedure during CAPIF Interconnect – Method 3

Detailed procedure is as follows:

Step 1: CAPIF-1e authentication and secure session establishment is performed.

Step 2: After successful establishment of TLS session over CAPIF-1e the API invoker sends a CAPIF interconnect token request message to the CAPIF core function-A.

Step 3: The CAPIF core function verify the Access Token Request message per OAuth 2.

Step 4: If the CAPIF core function successfully verifies the request message, the CAPIF core function generates a security token specific to the API invoker and return it in an Access Token Response message.

Step 5: On CAPIF-2e, the API invoker authenticates to the AEF by establishing a TLS session with the API exposing function based on the authentication and authorization method (i.e. Server (AEF) side certificate authentication or certificate-based mutual authentication) as indicated by CCF-A/CCF-B as described in clause 6.24..2.1. The following procedure shall be performed prior to establishment of TLS session.

The API invoker sends Authentication Initiation Request to the AEF, including API invoker ID.

The AEF requests for security information from the CCF-B to perform authentication and secure interface establishment with the API invoker. The CCF-B provides the security information related to the chosen security method (TLS with OAuth token) to the AEF over CAPIF-3 reference point (provided CCF-B is provisioned by CCF-A the required security materials).

The CCF-B returns API invoker's root CA certificate for the AEF to validate the API invoker's certificate. After fetching the relevant security information for the authentication, the AEF sends Authentication Initiation Response message to API invoker to initiate the TLS session establishment procedure.

Step 6: With successful authentication to the AEF on CAPIF-2e, the API invoker initiates invocation of a 3GPP northbound API with the AEF. The security token received from the CAPIF core is sent along with the northbound API invocation request as per OAuth 2.0.

Step 7: As the request is from API invoker in different trust domain, the AEF requests the CCF-B to verify the security token. The CCF-B is in possession of the required security material to verify the security token which was assigned/provided to the API invoker by the CCF-A. Otherwise, AEF retrieves the required security material from the CCF-B to verify the security token. The same principle as in IETF RFC 6749 [10] on OAuth 2.0 Authorization Framework applied for inter-domain service authorization. In order for the API invoker to obtain the domain B authorization access token, procedure as specified in clause 6.5.2.3 in TS 33.122 [4] is used where CCF-A provides API invoker a security token that identifies the API invoker to the AEF in domain B. This security token is specific to the AEF providing services in domain B and signed by the CCF-A in domain A as per IETF RFC 7515[12]. The domain A certificate(s) used to validate the security token at domain B are provisioned into the CCF-B using an out of band mechanism beyond the scope of this document. Upon validation of the security token, the API invoker is provided with authorization to the service(s) by AEF in domain B.

If validation of the security token is successful, the AEF verifies the API invoker's Northbound API invocation request against the authorization claims in the security token, ensuring that the API Invoker has access permission for the requested service API.

Step 8: After successful verification of the security token and authorization claims of the API invoker, the requested northbound API is invoked.

### 6.24.3 Evaluation

This solution partially addresses the security requirements for key issue#2.

Impacts to entities:

* CCFs in both trust domains (for inter domain communication)
* AEF (to contact CCF of the API Invoker)
* API Invoker (to send additional information in request message)

## 6.25 Solution #25: Backend based solution for UE-deployed API invoker accessing resources not owned by that UE

### 6.25.1 Introduction

This solution addresses the requirements identified in key Issue #3 (Authorizing API invoker on one UE accessing resources related to another UE).

In the real word application deployments, it is very common practice to have a backend server in addition to the application clients running/installed on end point devices such as mobile phones. For example, when the use case of gaming application is considered, even if the game application client instances are running on different devices and communicating with each other directly, there is always a backend server serving these clients. Thus, utilization of backend server can bring some benefits in terms of handling access permission and user identifiers. Taking these benefits into account, this solution proposes that when an application client instance running on a UE wants to access resources related to another UE which hosts another application client instance, the requester client can send the request to the backend server and then the backend server can consume the related APIs and returns the result back to the requester client. From 3GPP point of view, only permission check whether the backend server as AF is allowed to access the resources of the UE is needed. The remaining permission check such as which users’ resource can be accessed by which users can be handled in the application layer out of 3GPP scope.

### 6.25.2 Solution details

Figure 6.24.2-1 illustrates the high-level architecture.

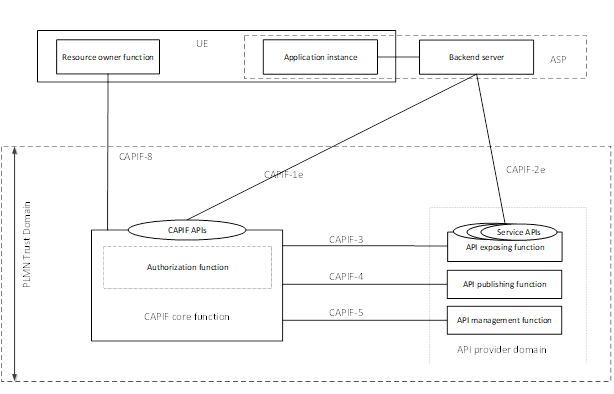


Figure 6.25.2-1: High level architecture

High-level steps of the solution are presented in Figure 6.25.2-2 and explained below.

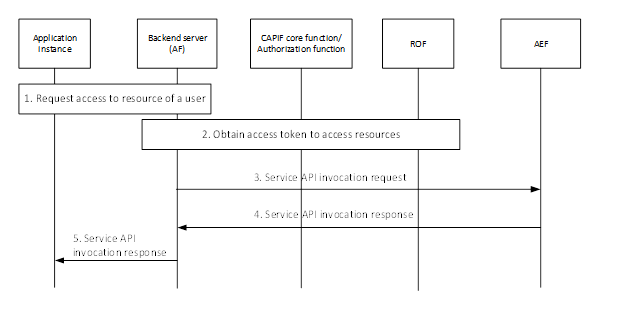


Figure 6.25.2-2: High-level procedure flow of the solution

1. The application instance running on UE 1 sends a request to the backend server to access resources related to user 2. The communication between the application instance and backend server and its security are out of scope of the solution.

2. The backend server identifies the UE ID of UE 2 related to the request. How to identify the UE ID is out of scope of the solution. Also, the backend server handles the resource owner authorization in the application layer (e.g., checks if user 1 can access resources related to user 2, gets permission from user 1). Application layer resource owner authorization handling is out of scope of the solution. The backend server acting the role of AF obtains an access token to access resources related to UE 2 by executing authentication and authorization the steps specified in TS 33.122 [4] if there is no valid access token. In the access token request, the AF can also send information about the UE 1 and the CCF can use that information in the authorization decision.

3. The backend server acting the role of AF invokes the API of the AEF with the access token.

4. After successful authentication of the backend server (i.e., AF) and verification of the access token, the AEF processes the API invocation and returns the result.

5. The backend server can return the result.

### 6.25.3 Evaluation

.

This solution addresses KI#3 by proposing invocation of the APIs by the back-end server assuming the role of AF, instead of direct API invocation by the application instance running on the UE. With that proposal, the solution have no impact on CAPIF except the impact on the API invoker side.

## 6.26 Solution #26: Nested API invocation

### 6.26.1 Introduction

This solution addresses the requirements identified in key issue #4 (nested API invocation).

According to the procedure in clause 8.32 of TS 23.222 [2], the API invoker gets authorization information (i.e., access token) from the CCF to invoke the API service of AEF 1 and then AEF 1 gets authorization information (i.e., access token) from the CCF to invoke the API service of AEF 2 which is in the same API provider domain as AEF 1. To avoid further interaction with the API invoker, it is stated that AEF 1 can exchange the access token of API invoker with the access token of AEF 1 to access the API exposed by AEF 2.

The solution proposes to utilize the token exchange procedure specified in RFC 8693 [5]. AEF 1 assumes the role of actor defined in the RFC. The access token of the API invokers to be used towards AEF 1 is used as the subject token. AEF 1 invokes the token end point of CCF by sending the subject token to receive a new access token to be used towards AEF 2. The solution also considers two different types of tokens (CAPIF legacy token and RNAA token) while adjusting the token exchange framework of the RFC to RNAA enhanced CAPIF.

### 6.26.2 Solution details

The solution is presented on the procedure specified in clause 8.32.3 of TS 23.222 [2]. Figure 6.26.2-1 illustrates the procedure.

Pre-conditions:

1. The resource owner function can communicate with the API invoker.

2. AEF 1 and AEF 2 are in the same trust domain.

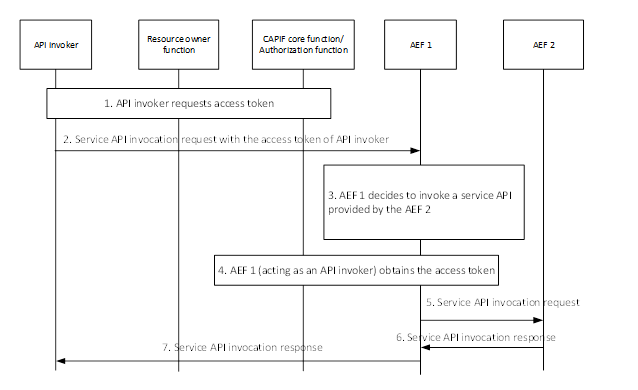


Figure 6.26.2-1: Procedure for obtaining authorization information in a nested API invocation

1. The API invoker requests access token to invoke the service API exposed by AEF 1. Resource owner function can be involved depending on whether RNAA is executed.

2. The API invoker sends a service API invocation request to AEF 1 with the access token received in step 1.

3. Based on the service API invocation request, AEF 1 decides to invoke another service API exposed by AEF 2.

4. AEF 1 invokes the token end point of CCF for token exchange. AEF 1 sends the received access token as the subject token to the CCF. AEF 1 can also optionally send an actor token issued by itself to the CCF. After token verification and access policy control, the CCF issues a new access token to be used by the AEF 1. The newly issued token includes the API invoker ID in the client id claim and optionally the AEF 1 ID in the act claim. If the subject token includes the resOwnerId claim, the CCF decides not to execute RNAA flow and includes that resOwnerId claim in the newly issued token. If the resOwnerId claim is not present in the subject token, then the CCF or AEF 1 can decide to execute RNAA flow. After the RNAA flow execution, the CCF includes the resOwnerId claim in the newly issued token. If the resOwnerId claim is not present in the subject token and the RNAA flow is not executed, then the CCF does not include the resOwnerId claim in the newly issued token. The CCF can also include the value in the scope claim of the subject token in the newly issued token. The CCF returns the newly issued access token to AEF 1.

5. AEF 1 sends a service API invocation request to AEF 2 with the newly issued access token received from the CCF.

6. AEF 2 performs the authentication and authorization. AEF 2 can also check whether the received access token includes the resOwnerId claim. If the received access token includes the resOwnerId claim, then AEF 2 can prefer to not execute any step to get permission from the resource owner.

7. The API invoker receives the service API invocation response resulting from the service API invocation.

### 6.26.3 Evaluation

This solution addresses the key issue #4, by proposing usage of IETF RFC 8693 [5] and providing a profiling by considering CAPIF RNAA aspects.

The solution has impact on the CCF.

The solution has the following impact on the AEF: the token received from the previous AEF needs to be sent to the CCF. There can also be additional impact on the AEF if the AEF wants to adjust its behaviour based on whether the token is an RNAA token.

There is no API invoker impacts i.e., to UE or AF when they are the actual API invokers.

Since the token newly issued by the CCF includes the API invoker in the client id claim, the AEF 2 ensures that the original request is coming from the API invoker.

## 6.27 Solution #27: Authorization for nested API invocation

### 6.27.1 Introduction

This solution addresses the security requirements of key issue#4. For nested API invocation, if the negotiated authentication method with AEF-2 is access token based, then it is proposed to re-use the OAuth 2.0 protocol with the extension that enables the API invoker(s) (AEF-1) to request and obtain security tokens from authorization server and use it as delegated access token when the AEF-1 decides to invoke the service API provided by the AEF-2. This solution proposes the API invoker – 2 (AEF – 1) requesting the CCF a delegated security token to invoke service APIs provided by the AEF – 2. Further, if the negotiated authentication method is TLS-PKI or certificate based then security procedure as specified in TS 33.122 [4] for generating or retrieving security materials.

### 6.27.2 Solution details

The nested API invocation scenario is a scenario where an API invocation towards a first API exposing function triggers that API exposing function to request an API invocation towards a second API exposing function.



Figure 6.27.2-1: Authorization for nested API invocation

1. CAPIF-1e authentication and secure session establishment is performed.
2. After successful establishment of TLS session over CAPIF-1e the API invoker requests authorization information to invoke the service API exposed by API exposing function 1.
3. The CAPIF core function verify the Access Token Request message as per OAuth 2.0.
4. If the CAPIF core function successfully verifies the request message, the CAPIF core function generates an access token and security token specific to the API invoker in an Access Token Response message.
5. The API invoker sends a service API invocation request to API exposing function 1 with the authorization information received in step 4.
6. Based on the service API invocation request, API exposing function 1 verifies the access token and decides to invoke another service API exposed by API exposing function 2.
7. API exposing function 1, acting as an API invoker, obtains from the CCF the authorization information to access the service API exposed by API exposing function 2. The API exposing function 1 sends token exchange request message to CCF, to get the authorization information to invoke the service API in API exposing function 2. The request message includes information as shown in Table 6.10.2.2-1.

Table 6.27.2.2-1: Token exchange request message

|  |  |  |
| --- | --- | --- |
| Information element | Status | Description |
| Authorization information | M | The authorization information with resource owner consent obtained from API invoker in the service API request message. |
| Security information | M | Security information related to API exposing function 1 to validate the request from API exposing function 1. |
| Resource Owner (s) Information | M | Identifiers or other information related to the resource owners for which the authorization information with resource owner consent is needed. |
| Service API access | M | Information related to the service API, service API request parameters and the API exposing function 2, for which the delegated authorization is requested. |

The CCF validates the request from API exposing function 1. CCF validates whether the requesting API exposing function 1 is allowed for delegated authorization to access service API related to the resource owners on API exposing function 2. Also, the CCF validates the Authorization information in the request message that is provided by the API invoker to the API exposing function 1. After successful validation, the CCF responds to API exposing function 1 with token exchange response message that includes the delegated authorization information to allow API exposing function 1 to invoke the service API on API exposing function 2. The response message includes information as shown in Table 6.10.2.2-2.

Table 6.27.2.2-2: Token exchange response message

|  |  |  |
| --- | --- | --- |
| Information element | Status | Description |
| Delegated authorization information | M | The delegated authorization information with resource owner consent. |
| > Resource owner (s) information | M | Identifiers or other information related to the resource owners for which the authorization information is applicable |
| > Authorization information about primary subject | M | The authorization information with resource owner consent provided by API invoker in the request message. |
| > Delegated subject | M | Information related to entities for which the delegated authorization is applicable. In this case, the information related to API exposing function 1. |
| > Expiry time | M | Time for which the delegated authorization is valid. |
| > Allowed permissions | M | Information related to allowed service API access and the permissions or permitted service operations or permitted API resources on the service APIs. |
| > Allowed API Exposing Functions | M | The API exposing function (s) where the allowed permissions are applicable. In this case, the information related to API exposing function 2. |

NOTE: Table 6.27.2.2-1 and Table 6.27.2.2-2 needs SA6 co-ordination.

1. API exposing function 1, sends a service API invocation request to API exposing function 2 with the authorization information i.e., security token received in step 7.
2. The API exposing function 1 receives the service API invocation response resulting from the service API invocation once API exposing function 2 has checked whether the API invoker is authorized to invoke that service API based on the authorization information.

10. The API invoker receives the service API invocation response resulting from the service API invocation.

#### 6.27.2.3 Access token claims

The standard claims would include client\_id of the API invoker – 2 (AEF – 1) acting as the actor. Further, the CAPIF OAuth 2.0 security token conveys the following actor claims for nested API invocation as specified in IETF RFC 8693 [5] additional to the token claims as specified in Annex C.2, TS 33.122 [4].

Table 6.27.2.3-1: Nested actor (act) claims

|  |  |
| --- | --- |
| Parameter | Description |
| client\_id | OPTIONAL. The identifier of the API Invoker-2 (AEF-1) making the API request. |
| iss | OPTIONAL. The issuer of the security token (in this case it is the CCF in the same or different trust domain) |

### 6.27.3 Evaluation

This solution addresses security requirements of key issue#4. If API invoker 2 and AEF-2 selects access token based method for authorization the below entities are impacted. Otherwise, it follows existing security procedure with no additional impacts implying that there will be no optimization case in nested API.

Impacts to entities:

- CCFs to provide delegated security token

- AEF-1 as an API invoker to contact CCF for delegated security token

## 6.28 Solution #28: Authenticating multiple API invokers of the same RO

### 6.28.1 Introduction

This solution addresses KI#5 and is an optimization when the ROF is engaging multiple API invokers at the same UE. The ROF has a valid certificate received from the CCF which can be used for interaction between API invoker and AEF after API invoker/ROF interaction within the UE.

### 6.28.2 Solution Details

Any API Invoker of the UE that needs to get authenticated with AEF asks ROF to sign its API Invoker ID (using ROF private key). ROF provides the signed API Invoker ID with expiry time and certificate information.

When the API invoker starts the authentication process with AEF, a server-based TLS connection between AEF and API Invoker is established first. To authenticate the API invoker towards AEF, the API Invoker provides its plain API Invoker ID together with the ROF signed API Invoker ID which also includes the ROF certificate or a URI where the AEF can find the necessary details for validation.

The AEF validates the ROF certificate and validates if the plain API Invoker ID and the verified signed API Invoker ID. match. If not expired and on successful verification and match, AEF and API Invoker are authenticated.

NOTE: The solution provides a way for the ROF to give access only to its own resources to API Invokers under its control. The certificate chain of the ROF (which was issued by CCF) is enough for AEF to verify the legitimate of ROF. API Invoker Authorization to ROF resources is automatically proven by the token.

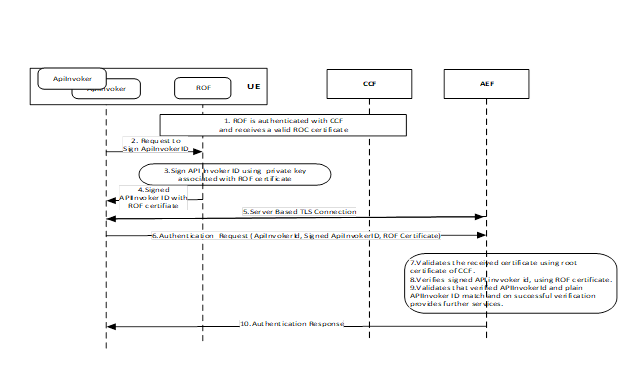


Figure 6.28.2-1: Authentication using the ROF certificate

Alternatively, if the ROF has received a valid OpenID [11] token during authentication between ROF and CCF, then the ROF can also provide to the API invoker the OpenID [11] Token instead of the ROF certificate. The flow is illustrated below.

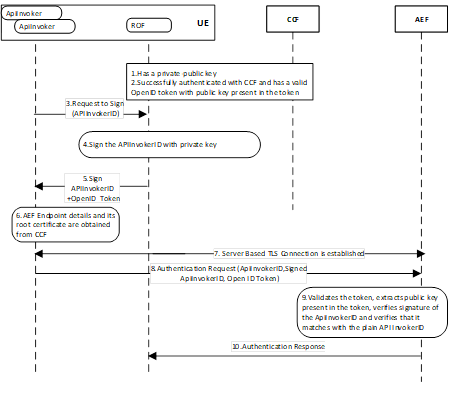


Figure 6.28.2-2: Authentication using OpenID token

NOTE: Not all API invokers should be be allowed to consume the resource owner data with same priority (e.g., duration/scope).

ROF authentication to API invoker is left to implementation.

ROF authentication to CCF is left to implementation.

### 6.28.3 Evaluation

The solution addresses KI#5 by providing a way for the ROF to give access only to its own resources to API Invokers under its control. The certificate chain of the ROF (which was issued by CCF) is enough for AEF to verify the legitimate of ROF. API Invoker authorization to ROF resources is automatically proven by the token.In this solution, ROF is used to authenticate the API invoker for the CAPIF system.

The solution influences:

The behaviour of the API Invoker and ROF to generate the new token

The AEF to verify the new token type before allowing access.

If all API invokers are going to be authenticated/authorized at once just because they are going to invoke services to access data of the same resource owner, it needs to be clarified how individual authorization/data exposure of RO data works.

## 6.29 Solution #29: Enhancing authorization through finer granularity access token.

### 6.29.1 Introduction

This solution is addressing KI#1.3 to enhance finer granularity authorization by reusing already existing mechanisms available in CAPIF ecosystems and is based on TR 23.700-22 [3] Solution 9 as selected by conclusion. I.e. to support RNAA, the CCF checks whether the API invoker is permitted to access the requested service API based on the API invoker's subscription information and resource owner consent using more granular information about API invoker in the context of RNAA.

In short: A Resource Owner, registered to a CAPIF instance, wants to provide its consent to an API Invoker to access its own resources. In this context, the CCF and AEF should be able to correctly identify the owner of the resource and authorize the access request based on their knowledge.

### 6.29.2 Solution details

#### 6.29.2.1 Summary

The solution proposes a way to enhance current OAuth2 based authorization mechanisms, both at CCF and AEF, to allow a finer granularity access control.

NOTE 1: A finer granularity can include: Resource Owner ID, operations (e.g. retrieve, create, etc), features (e.g. feature 1, feature 2, etc) and resources (e.g. resource 1, resource 2, etc).

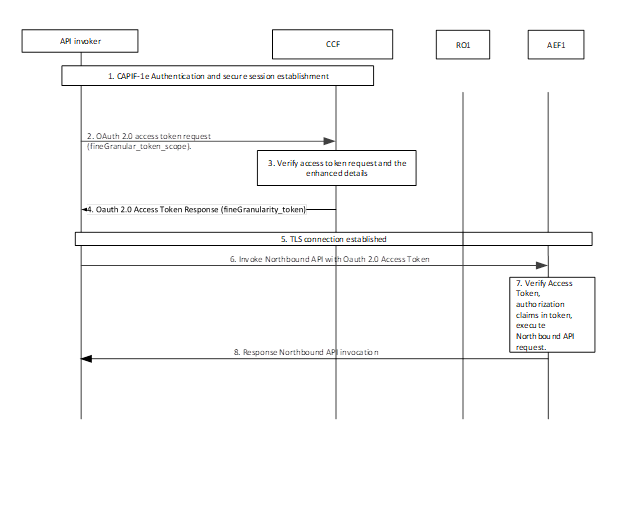
After authentication between the CCF and the API Invoker, the latter will include the required additional information to CCF during the Access token Request. The API Invoker will include in the scope parameter more authorization details that allow to distinguish both the resource owner and the resources that API Invoker intends to access. This will allow the CCF to verify if the resource owner allowed this specific API Invoker to access the requested resources.

When the verification is completed, the CCF will include the authorization details, with the new granularity, into the access token returned to the API Invoker.

The previously provided access token will allow the AEF to correctly authorize, or deny, the request by the mechanism already available to AEF. AEF must be able to verify the details for finer granularity access control.

NOTE 2: The inclusion of resource owner identifier is proposed to be a scope parameter while in Rel-18 RO id is part of claim. Granularity is not specified in Rel-18. The solution proposes that aA Rel-19 RNAA token is including the RO id and granularity details in scope and leaving empty the Rel-18 resOwenrId. This allows to distinguish between Rel-19 and Rel-18. It is done by purpose to distinguish between RO id in claims in Rel-18 and to avoid backward compatibility problems.

#### 6.29.2.2 Information flow



1. CAPIF-1e authentication and secure session establishment is performed as specified in subclause 6.3.1 of 33.122 [4].

2. After successful establishment of TLS session over CAPIF-1e, the API invoker shall send an Access Token Request message to the CAPIF core function with the enhanced scope.

NOTE 1: the enhanced scoped will include all the required additional details necessary for granular authorization, such as Resource owner ID, operation features and/or resources.

3. The CAPIF core function shall verify the Access Token Request message per OAuth 2.0 [4] specification, by verifying the required scope at the finer granularity required by the API Invoker.

4. The CAPIF core function shall generate an access token with the detailed scope containing finer granularity and return it in an Access Token Response message.

5. The API invoker authenticates to the AEF by establishing a TLS session with the API exposing function based on the choosed authentication method

6. With successful authentication to the AEF, the API invoker shall initiate invocation of a 3GPP northbound API with the AEF. The access token received from the CAPIF core shall be sent along with the northbound API invocation request as per OAuth 2.0 [4].

7. The API exposing function shall validate the access token. If validation of the access token is successful, the AEF shall verify the API invoker's Northbound API invocation request against the authorization claims in access token, ensuring that the API Invoker has access permission for the requested service API.

NOTE 2: the verification procedure should be enacted to verify the additional fields inserted in the access token from CCF.

8. After successful verification of the access token and authorization claims of the API invoker, the requested northbound API shall be invoked, and the appropriate response shall be returned to the API invoker.

### 6.29.3 Evaluation

The solution addresses the requirements of KI#1.3 by allowing to insert the required finer granularity as part of the token Request and the access token itself.

The solution affects:

- The API invoker, who will need to modify the access token request.

- The CCF which will need to understand the new fields and verify the permission of the API invoker

- The AEF which will need to verify the new enhanced scope.

## 6.30 Solution #30: Authentication of the origin API invoker in nested API invocation

### 6.30.1 Introduction

The solution addresses the authentication part of KI#4 on nested API invocation.

Before an API invoker can request the access token to invoke a service API exposed by AEF 1 it needs to get authenticated. However, if AEF1 decides to invoke another service API exposed by AEF2 that relates to the origin API Invoker, the token issued by CCF and finally received by AEF 2 does not authenticate the API invoker against AEF2.

The solution proposes to use the CCA concept. The origin API invoker includes CCA into its service request to AEF1, such that AEF1, if it needs to invoke another service from AEF2, can request an access token on behalf of the origin API invoker by presenting CCA to CCF.

### 6.30.2 Solution Details

1) API invoker and CCF have mutual authenticated.

2) API invoker discovers at CCF the AEF1.

3) CCF indicates in its service discovery response that nested API invocation is supported at AEF1. The indication allows the API invoker to include CCA for using the service later on. Eg. use of CCA required for nested API support.

4) The API invoker requests an Access Token for AEF1.

5) CCF returns the access token.

6) API invoker mutually authenticates with the discovered AEF 1 and hence, establishes a secure connection.

7) If nested API support was indicated during discovery, the API invoker generates a client credential assertion token (CCA) including the API Invoker Id, the CCF address and other information such as the expiry time of the CCA.

8) The Service request including the CCA is then sent to the AEF 1.

9) AEF1 receives the Service request including CCA. If AEF1 needs to invoke another service, it acts as an API invoker to AEF2.

10) AEF1 has established mTLS with CCF and sends its own AccessTokenRequest to CCF to access the service of AEF2 on behalf of the origin API invoker. This includes CCA, optionally it includes also the previously received token.

11) CCF will authenticate the origin APIInvoker via CCA and authorizes APIInvoker and AEF1 by provisioning a new access token to be used by AEF1 to access the nested service from AEF2 on behalf of the origin API invoker.

12) CCF provides the new access token to AEF1.

13) AEF1 creates the Service request to AEF2 along with the CCA received from the origin API invoker and the access token received in step 12.

14) AEF2 authenticate the origin API invoker via CCA and validates that the API invoker Id in CCA is matched with API invoker Id in the access token.



### 6.30.3 Evaluation

The solution assumes that CCF knows in advance that the AEF1 will invoke AEF2 API.

The solution requires a client credential authentication token to be introduced to allow for authentication when the nested API invocation scenario needs to authenticate the API Invoker.

The following entities are impacted:

API Invoker: The solution requires that the API invoker includes a CCA token to allow the final AEF to authenticate the API invoker in a nested API invocation scenario.

CCF/AEF: The solution requires CCF to process an access token request twice and AEF1 to request an access token and to update the service request (CCF authenticates and authorizes the API invoker of the original request. Then CCF needs to authenticate the access token request also from AEF1 to allow AEF1 to request the service with the updated token for service invocation from AEF2).

## 6.31 Solution#31: Authorization mechanism for nested API invocation

### 6.31.1 Introduction

This solution addresses KI#4.

In this solution, if the token sent by the API invoker contains the resource owner ID, to avoid redundant resource owner authorization, the token exchange procedure is used. Then, the AEF-1 requests AEF-2’s API on behalf of the API invoker. The token’s Client ID claim is set as API invoker ID.

If the token sent by the API invoker does not contain the resource owner ID, the token exchange procedure is not used. The AEF-1 directly requests the token to access AEF-2’s API.

### 6.31.2 Solution details

Pre-condtion:The CCF knows that some APIs of target AEF (e.g., AEF-1) need to trigger the target AEF to contact another AEF (e.g., AEF-2) to create the API-invoker-related response.



Figure 6.31.2-1: Authorization mechanism for nested API invocation

1. AEF-2 configures the authorization information to the CCF/authorization function.

The authorization information may contain the function identity (e.g., AEF-1 identity) and the service API information. The authorization information indicates the functions (e.g., AEF-1) that is authorized to access the AEF-2’s service API.

1. The API invoker requests the token#1 to invoke the service API exposed by API exposing function 1.

The CCF authorizes the API invoker optionally based on the authorization information from the resource owner.

If the API invoker is authorized and CCF/authorization function finds that the target AEF (e.g., AEF-1) needs to trigger another AEF (e.g., AEF-2) to create the API-invoker-related response, the CCF/authorization function adds the target AEF identity (e.g., AEF-1 identity) to the may\_act claim of the token. Then the token#1’s claims are listed as follows.

may\_act claim contains the AEF identity. One example of a may\_act claim is given as follows.

"may\_act":

{

sub":"AEF-1 identity"

}

May\_act is used to states that only the AEF listed in the may\_act claim can trigger the token exchange procedure related to this token.

2. The API invoker sends a service API invocation request to API exposing function 1 with the token received in step 1.

3. Based on the service API invocation request, API exposing function 1 decides to invoke another service API exposed by API exposing function 2 (i.e., AEF-2).

4a. If the token #1 contains the resource owner ID, AEF-1 sends a token exchange request to the CCF. The request includes token #1, scope , the audience, the grant-type, and AEF-1’s actor token. The grant-type is set as the token-exchange. The audience is set as the AEF-2 identity. The scope is set as the service API exposed by the AEF-2.

4b. If the token#1 does not contain the resource owner ID, AEF-1 sends a token request to the CCF. The request includes the AEF-2 identity and AEF-2’s service API.

5a. For token exchange requests, if the AEF-1 is authorized to access AEF-2’s service API based on AEF-2’s authorization information and AEF-1 identity in actor token is identical to the one in the token#1’s may\_act claim, the CCF generates the token#2 for AEF-1. Otherwise, CCF sends a failure message to the AEF-1. The failure message indicates that the token exchange fails. The scope of the token #2 contains the AEF-2’s service API. The actor claim of the token#2 is set as the AEF-1 identity. The token#2’s scope claim is set as AEF-2’s service API.

5b. For direct token request, the CCF authorizes AEF-1 based on authorization information provided by AEF-2. If AEF-1 is authorized, the CCF generates token #2. The token#2’s client id claim is set as the AEF-1 identity while the scope claim is set as the requested AEF-2’s service API.

6. The AEF-1 and AEF-2 do mutual authentication. Then AEF-2 obtains the AEF1 identity.

7. API exposing function 1, acting as an API invoker sends a service API invocation request to API exposing function 2 with the token#2.

8. If token#2 contains the actor claim, the AEF-2 checks if the AEF-1 identity obtained in step 6 is identical to the one in act(actor) claim of access token #2. If they are not identical, AEF-2 sends a failure message to the AEF-1. The failure message indicates that AEF-1 cannot delegate the service request.

If token#2 does not contain the actor claim, the AEF-2 uses an existing mechanism to check token#2.

9. The API exposing function 1 receives the service API invocation response resulting from the service API invocation once API exposing function 2 has checked whether the API invoker is authorized to invoke that service API based on the authorization information.

10. The API invoker receives the service API invocation response resulting from the service API invocation.

### 6.31.3 Evaluation

This solution addresses the nested API invocation authorization by enabling AEF-1 to request the token from the CCF.

This solution has impacts on AEF-1

If token#1 contains resource owner ID, the AEF-1 should support the token exchange procedure. The AEF-1 should be able to send the token#1, scope , the audience , the grant-type, and AEF-1’s actor token to the CCF. The grant-type is set as the token-exchange. The audience is set as the AEF-2 identity. The scope is set as the service API exposed by the AEF-2.

If token#1 does not contain the resource owner ID, the AEF-1 should support request token from CCF via non-token-exchange-based procedure.

This solution has impacts on CCF.

The CCF should support the token exchange procedure. If AEF-1 is not authorized to do the token exchange, the CCF should send the token exchange failure message to AEF-1.

The CCF should be able to authorize AEF-1 based on AEF-2’s authorization information and AEF-1’s actor token (optional).

The CCF should set AEF-1 identity as the may\_act claim of token#1 when nested API invocation is needed.

This solution has impacts on AEF-2.

If token#2 contains the actor claim, the AEF-2 checks if the authenticated AEF-1 identity is identical to the one in act(actor) claim of access token #2. If they are not identical, AEF-2 sends a failure message to the AEF-1. The failure message indicates that AEF-1 cannot request the service request on behalf of the API invoker.

## 6.32 Solution #32: Validation of correct GPSI in API invoker information

### 6.32.1 Introduction

This solution addresses KI#6. It is based on KI#5 of TR 23.700-22 [3].

GPSI provided by the APIinvoker in an onboarding or modification request needs to be confirmed to be associated to the UE on which the APIinvoker is running. The solution enables the CCF to validate apiInvokerInformation details by UE interaction.

### 6.32.2 Solution Details

The API invoker requests UE to create a MAC (Message Authentication Code) for the apiInvokerInformation or the GPSI as a proof that the GPSI belongs to this UE. Such MAC must be generated out of information known to both the UE and the 5GC. A MAC can be for example derived from K\_AUSF or AKMA using the UE subscriber credentials.

GPSI is identifying the RO.

During Onboarding/Update/Modify request to the CCF, the API invoker sends the MAC provided by the UE in addition to the apiInvokerInformation, i.e. application and device details (GPSI, etc).

How the API invoker residing on the UE gets knowledge of the GPSI is not detailed in this solution.

Upon receiving the Onboarding/Update/Modify request, the CCF requests the 5GC to also generate the MAC on the apiInvokerInformation that the CCF needs to pass over for this. CCF receives a MAC result and compares both MACs. CCF only processes the request, if successfully matching.

### 6.32.3 Evaluation

The solution enables the CCF to validate apiInvokerInformation details by UE interaction.

Further evaluation (e.g., impacts) is not provided.

## 6.33 Solution #33: Onboarding of API Invoker residing in UE

### 6.33.1 Introduction

The solution addresses KI#6.

### 6.33.2 Solution details

The API Invoker residing in the UE can perform onboarding using the procedure described in TS 33.122 Clause 6.1 as the baseline principles with the following adaptations as applicable for the case of API invoker being part of the UE as shown in Figure 6.33.2-1.



Figure 6.33.2-1 Security procedure for onboarding of API invoker residing in the UE

The steps shown in Figure 6.33.2-1 is described below to clarify the adaptations specific to the onboarding of API invoker (residing UE).

1. Step 1 same as in TS 33.122 Clause 6.1 step 1.

NOTE 1: The API provider domain functions are described in TS 23.222 Clause 8.28.

2. Step 2 same as in TS 33.122 Clause 6.1 step 2.

3. With the secure session established, the API Invoker sends an Onboard API Invoker Request message to the CCF, which includes Existing CCF access token, existing information elements and along with GPSI, authentication info (i.e., hash/MAC of GPSI, A-ID and other information if any needed computed using an existing security context e.g., AUSF key or AKMA key related to UE. Where the details of the specific inputs and key to be used are not in the scope of the solution and Application Identifier(s) (A-ID(s)).

4. The CCF validates the enrolment credential (OAuth access token) as in the existing TS 33.122 and validates the authentication information based on UE’s security context available in the core network related to the GPSI, where the validation of authentication info is up to further normative details. If the validation is successful, the CCF generates the API invoker profile as specified in TS 33.122 step 4 and additionally it includes GPSI and A-ID(s) if the API invoker corresponds to a UE i.e., a UE service. Further the CCF derives Onboard\_Secret as in TS 33.122 and also it is bound to the respective GPSI and the A-ID(s).

7. The CCF respond with an Onboard API invoker response message, same as in TS 33.122 Clause 6.1 step 5.

The offboarding of the API Invoker when applicable works as described in TS 33.122 Clause 6.8.

### 6.33.3 Evaluation

The solution impacts are listed as follows.

API invoker/UE:

* Initiates Onboard API Invoker request towards CCF, with UE ID (GPSI), authentication information and application identifier(s) to let CCF to validate if the API invoker is residing as part of the UE and using information only related to the specific UE while onboarding.

CCF:

* The CCF validates the authentication information received from the UE based on a security context related to GPSI as available in the core network.
* The CCF on a successful verification of GPSI generates and provides API invoker profile and Onboard secret to the API Invoker residing as part of the UE, which binds to the GPSI and the application identifier(s).

## 6.34 Solution #34: UE-deployed API invoker accessing resources not owned by that UE

### 6.34.1 Introduction

The solution is for key issue #3 (Authorizing API invoker on one UE accessing resources related to another UE) and proposes a security mechanism for the cases that the application running on a UE directly or via a backend server invokes NB APIs to access network-hosted resources of another UE.

This solution additionally addresses key issue #1 (Security of resource owner authorization management and CAPIF-8 reference point) by providing a mechanism how to obtain authorization form the Resource Owner and identifying required information to be included in the authorization data.

### 6.34.2 Solution details

#### 6.34.2.1 Obtaining resource owner authorization and authorization revocation information

According to Rel-18 TS 33.122 [4], the resource owner can be the user of the UE or the subscription owner.

The resource owner authorization information is stored in the Resource Owner Management Function (ROMF) of the CCF / Authorization Function.

NOTE: ROMF can be a separate entity.

The information includes Resource Owner identifier, Application identifier, Purpose, Scope and optionally a list of UEs (e.g., UE2, UE3, etc.). The resource owner authorization information can also include policy and conditions on when and how the authorization information can be used in the authorization decision and/or under which conditions the access is allowed (e.g., days, time slots etc.). The resource owner authorization information can additionally include an indication to indicate whether the ROF can be used for obtaining/provision of resource owner authorization and authorization revocation information. Additionally, the resource owner authorization information can include an indication and information to indicate that the resource owner can be accessible to request resource owner authorization information by the CCF / Authorization Function / ROMF in which conditions. Also, the information can include resource owner accessibility information such as ROF instance ID, a URI or an SMS endpoint that can be used to reach to the Resource Owner. An example record can look like the following:

- Resource Owner identifier,

- Application identifier, and per Application Identifier

- Purpose,

- Scope,

- A list of UEs who can access the resource of the Resource Owner,

- Indication for the Resource Owner accessibility,

- Indication for the ROF usability for obtaining authorization and authorization revocation information,

- Policy and condition on when and how the authorization information can be used in the authorization decision and/or under which conditions the access is allowed.

- Conditions for the Resource Owner accessibility (e.g., reachability of the resource owner),

- Resource Owner accessibility information, for example ROF instance ID, a URI, SMS endpoint etc.

Obtaining resource owner authorization and authorization revocation information can be done by using out of band mechanism(s) and can be configured in the CCF / Authorization Server / ROMF. The ROF via CAPIF-8 can also be used to obtain resource owner authorization and authorization revocation information.

To be able to use ROF to obtain / provide resource owner authorization and authorization revocation information, there needs to be a related record in the CCF / Authorization Server / ROMF and the indication for the ROF usability is set to the value which indicates that the ROF can be used to obtain / provide resource owner authorization and authorization revocation information. The authorization information sent by the ROF can include Resource Owner identifier, Application identifier, Purpose, Scope, the UE identifier in the App, a list of UE identifiers allowed to access the API Exposing Function(s), AEF identifiers, policy and condition for authorization information, indication for the Resource Owner accessibility to be notified to provide RO authorization, conditions for the Resource Owner accessibility, and Resource Owner Accessibility information. The procedure of providing authorization and authorization revocation information can be triggered by the Resource Owner.

The authorization information can also be requested by the CCF / Authorization Server / ROMF. In that case, the CCF / Authorization Server / ROMF first checks 1) the indication for the ROF usability, 2) the indication for the Resource Owner accessibility and 3) conditions for the Resource Owner accessibility and if the checks are successful then sends the request to the ROF of the Resource Owner by using the Resource Owner Identifier and/or the Resource Owner accessibility information to obtain the resource owner authorization information. In the request the CCF /Authorization Server / ROMF can send the Application identifier, purpose, scope, AEF ID, the requesting API invoker / UE identifier, and the response can include whether the Resource Owner allows the access and policy and condition for authorization information.

#### 6.34.2.2 UE-deployed API invoker accessing resources not owned by that UE

Figure 6.34.2.2-1 presents the procedure for enabling a UE-hosted API invoker accessing network-hosted resources owned by other UEs.

Pre-condition:

1. CCF / Authorization Function / ROMF has connectivity and can interact with ROF via CAPIF-8.

1. The API invoker in UE2 knows the identifier for UE1.
2. The API invoker knows the Application identifier requiring the access to RO resources. While onboarding of the API invoker, the Application identifier is also stored in the API invoker profile in the CCF.



Figure 6.34.2.2-1: UE-deployed API invoker accessing other UEs’ resources

0. The ROF for UE1 stores Resource Owner authorization information for UE1 in the Resource Owner Management Function (ROMF) of the CCF or the authorization information is configured by an out of band mechanism. The ROMF keeps the authorization information as stated in clause 6.34.1.2. The ROF can also send Application identifier, the indication for the Resource Owner accessibility, conditions for the Resource Owner accessibility, policy and condition for the authorization information, and Resource Owner accessibility information to the CCF / Authorization Server / ROMF. The authentication and authorization of the ROF and the human user behind the ROF occurs within the PLMN domain and is deployment/implementation specific.

1. The CCF authenticates the API invoker in UE2 by using the credentials for the API invoker issued during onboarding of the API invoker and by using the API access authorization information for the API invoker. If the request is coming via the backend server of the API invoker service provider, the CCF validates and authorizes the back-end server which has already been onboarded to the CCF.

2. The API invoker in UE2 sends an Obtain Service API Authorization (e.g., access token) request to the CCF for obtaining permission (e.g., access token) to access the service API. The request includes UE1 identifier, whose resources are trying to be accessed, and the information about the Purpose and Scope to be performed on the targeted resources (service API). The request also includes the Application identifier that is requesting the access and UE2 identifier.

NOTE 1: The UE1 identifier can be the Resource Owner ID. The UE2 identifier can be a UE ID token issued by a UE ID server in the network during step1.

3. UE2 is authenticated by the CCF / Authorization Function. For authentication, UE2 UE ID token can be used. If the API invoker profile includes the hosting UE identifier information, then the CCF can compare the UE identifier in the API invoker profile with the UE2 identifier. The CCF / Authorization Function retrieves Resource Owner authorization information for UE1 from the ROMF and validates whether UE2 is authorized to access UE1 resources for the specified purpose, scope, and Application identifier, by also considering the policy and access condition on when and how the authorization information can be used in the authorization decision and how and when the access is allowed.

4. If Resource Owner authorization needs to be captured because the CCF / Authorization Function / ROMF does not contain Resource Owner authorization for UE1 or the retrieved Resource Owner authorization does not include authorization information for UE2, the CCF / Authorization Server / ROMF can trigger the capture of the UE1 authorization information by taking the indication for the Resource Owner accessibility and conditions for the Resource Owner accessibility into account. The CCF / Authorization Server / ROMF can use Resource Owner ID and/or Resource Owner accessibility information to reach to Resource Owner or the ROF of UE1.

NOTE 2: ROF of UE1 is accessed by the CCF / Authorization Function / ROMF to request permission from the Resource Owner. Since the response from the Resource Owner can take time, the CCF / Authorization Function / ROMF can inform the API invoker to try again by executing step 1 and 2. The information that a resource owner authorization information obtain request has been sent to the ROF of UE1 can be stored in the CCF / Authorization Function so that there will be no repeated resource owner authorization information obtain requests sent to the ROF of UE1.

5. If the CCF / Authorization Function / ROMF granted to UE2 the Resource Owner authorization for accessing resources of UE1, the authorization information (i.e., RNAA access token) to access the service API is sent to the API invoker in the Obtain service API authorization response (e.g. access granted for API invoker in UE2 to the resources of UE1).

6. The service API invocation takes place including the authorization information (i.e., RNAA access token) received in the previous step and also optionally including the UE2 identifier.

### 6.34.3 Solution evaluation

This solution is in line with corresponding key issue conclusion in TR 23.700-22 [3] and addresses the key issue requirements of key issues #1, and #3of the present document.

This solution has impact on the CCF regarding authorization information management and authorization of the API invoker.

There is no impact on the AEF.

## 6.35 Solution #35: Onboarding of UE-hosted API invoker

### 6.35.1 Introduction

This solution also addresses key issue #6 (Onboarding security issues) by proposing a mechanism for onboarding of API invokers residing on a UE.

### 6.35.2 Solution details

Clause 6.1 of TS 33.122 [4] applies with the following changes for the onboarding of the API invoker residing on a UE.

- An access token as the onboarding credential can be provided to the API invoker by the API invoker service provider (API invoker backend server). The token also includes the Application identifier.

- In step 3, the API invoker can send the certificate of the API invoker service provider to the CCF.

- In step 4, the CCF verifies the token and checks whether the API invoker service provider provides the application identified by the Application identifier in the access token. If the check is successful, the CCF also stores the Application identifier of the API invoker in the API invoker profile in the CCF.

- In step 3, the UE hosting the API invoker can be authenticated by the CCF. For example, UE ID token issued by the network can be used.

- In step 4, if the UE hosting the API invoker is authenticated by the CCF then the CCF can store the UE identifier (e.g. GPSI) in the API invoker profile in the CCF.

Figure 6.35.2-1 presents the changes to the existing onboarding procedure. The changes are show in *italic* for newly added parts and ~~strikethrough~~ for the removed parts.



Figure 6.35.2-1: Potential changes to the existing onboarding procedure.

### 6.35.3 Solution evaluation

This solution addresses the requirement on how UE hosted APIs can be onboarded. It proposes the following changes to the existing onboarding procedure.

- The onboarding credential access token can include the Application ID.

- The onboarding request can include the token issuer’s certificate.

- The CCF needs to store the Application ID in the API invoker profile.

- The CCF can authenticate the UE hosting the API invoker and can store the UE ID in the API invoker profile.

Impacts on the API invoker:

- Since obtaining of the enrolment information is out of scope, there is no impact on the API invoker whether it obtains the enrolment information from API provider domain or its own backend server.

- There is impact on the API invoker for sending the token issuer’s certificate to the CCF.

- There can be impact on the API invoker for authentication of the UE towards the CCF.

Impacts on the CCF:

- There is impact on the CCF to store the Application ID in the API invoker profile and use this information in the authorization decision.

- There is impact on the CCF to authenticate the UE hosting the API invoker.

There is no impact on the AEF.

There can be business relation between the CCF and API provider domain as in the existing system, so the certificate or necessary info of API provider domain to perform token validation can be same as existing system. So, why an access token is not sufficient and why additionally token issuer’s certificate and API invoker Public key need to be sent in Onboard API invoker Request is not addressed.

## 6.36 Solution #36: Reusing existing mechanism to enable cross-UE authorization

### 6.36.1 Introduction

In this contribution, the existing mechanisms defined in TS 33.122 are reused to address the KI#3.

### 6.36.2 Solution details

The RNAA procedures defined in clause 6.5.3.2 and 6.5.3.3 of TS 33.122 [4] are reused to authorize API invoker on one UE to access resources related to another UE with the following modifications. CCF should not check if the UE is accessing its own resources when the API invoker is on UE.

### 6.36.3 Evaluation

The RNAA procedures defined in clause 6.5.3.2 and 6.5.3.3 of TS 33.122 [4] are reused to authorize API invoker on one UE to access resources related to another UE.

How this step ‘CCF should not check if the UE is accessing its own resources when the API invoker is on UE.’ can let CAPIF to do authorization of the API invoker on one UE to access resources related to another UE is not addressed.

## 6.37 Solution #37: Enabling mTLS between ROF and CCF using AKMA

### 6.37.1 Introduction

This solution is addressing KI#1.2 requirement of mutual authentication between ROF and CCF.

This solution proposes that ROF can get a certificate using AKMA, which allows to perform mTLS between ROF and CCF in any subsequent communication.

### 6.37.2 Solution details

As a part of first-time connection of ROF with CCF, ROF realize that it does not have a ROF certificate (or CCF insists on further authentication to allow for mutual authentication in future and triggers ROF for authentication).

The solution utilizes AKMA specified in TS 33.535 [9]. ROF initiates Application Session Request to CCF with A-KID (see TS 33.535 [9], figure 6.1-1, for A-KID derivation at UE). ROF authenticates the CCF via CCF certificate.

The CCF can then authenticate the ROF based on A-KID and provides a ROF certificate via application session response. The ROF can store the certificate for further use, e.g if CCF closes the connection and has marked that the certificate has been provided so that it can avoid sending the ROF certificate again and again.

Hence, ROF can now create mTLS connections with CCF using the certificate received in the first-time connection.

The related information flow and description is provided as follows.



1) The UE and 5GC shall generate the AKMA Anchor Key (K-AKMA) and the A-KID from the K-AUSF and AanF stores the A-KID, k-AKMA and SUPI of the UE. See TS 33.535 [9], Figure 6.1-1: Deriving KAKMA after primary authentication for more details.

2) ROF knows the root CA certificate.

3) ROF establishes a TLS connection based on CCF certificate (CCF certificate is known to ROF either from UE or from the network or through another channel).

4) ROF generates private – public key pair.

5) On the secure channel (unlike in AKMA where Application Session Establishment goes over non secure channel), ROF sends Application Session Establishment request to CCF along with the A-KID, callbackUri and public key.

6) CCF acting as AF in AKMA terminology follows the procedure, mentioned in TS 33.535 [9], Figure 6.2-1 on KAF generation from KAKMA, to get the SUPI/GPSI of the UE associated with the A-KID. This way CCF indirectly authenticates the ROF running on the UE as only the right ROF knows the A-KID. If AKMA procedures are followed, along with SUPI/GPSI, k-AF is also received at CCF, but it is not used by CCF. If k-AF shouldn’t be sent to CCF, AKMA procedure can be enhanced to receive only GPSI/SUPI and not the k-AF.

7) CCF generates ROF certificate using the public key received from ROF and sends the same to ROF as part of Application Session Response.

8) ROF further uses the ROF certificate to establish a mTLS with CCF.

9) ROF establishes mTLS with CCF

10) CCF revokes the ROF certificate.

11) CCF informs the ROF via the CallbackUri received in step 4 from the ROF.

12) ROF may retry steps 3 to 7 to get the new certificate.

NOTE: The CAPIF specific enhancements over AKMA if any required, would be left to normative work.

### 6.37.3 Evaluation

In scenarios where ROF and CCF need to mutually authenticate, this solution provides ROF with a certificate.

The solution is based on the assumption that ROF is running on the UE

The impact of the solution is not analyzed.

## 6.38 Solution #38: Renewal of onboarding

### 6.38.1 Introduction

This solution is addressing KI#6 security aspects of onboarding.

The API Invoker onboarding at CCF is usually valid for long time. Expiry time is only optionally provided. The Onboard\_Secret remains valid as long as the API invoker is registered. Long-time secrets are vulnerable.

The only way of renewal from API invoker side is to initiate an offboarding process and do a new onboarding. The procedure is triggered by the API invoker over CAPIF-1 or CAPIF-1e. However, this offboarding makes the API invoker no longer a recognized user of the CCF.

Furthermore, this creates ambiguity, e.g. if the API invoker continues to use the service during the renewal process, it is not feasible as per the current standards.

### 6.38.2 Solution details

This solution introduces the option to update onboarding information to allow for renewal of the Onboard registration or Onboard\_Secret or updating of APIInvokerEnrolmentDetails (APIList,apiInvokerInformation) at CCF without the need to offboard first.

Further, a CAPIF Event could be introduced such that it can be subscribed by API Invoker to get notification about onboarding expiry in advance.



Step 1: API Invoker is preconfigured with credentials required for onboarding with CAPIF Core function by the service provider.

Step 2: TLS connection is established between API Invoker and CAPIF Core function.

Step 3: API Invoker sends the Onboard API Invoker request with o-auth access token to CAPIF Core function.

Step 4: CAPIF Core function verifies the o-auth access token and generates API Invoker profile for the API Invoker.

Step 5: CAPIF Core function sends the Onboard API Invoker Response with API Invoker ID, API Invoker Certificate and Onboard\_Secret, which will be used by API Invoker during the authentication with API Exposing Function (CAPIF-2e interface authentication and protection using Access Tokens).

Step 6: API Invoker can renew its Onboarding using OnboardingUpdateRequest by providing APIInvokerID,OnboardingInformation (received from CCF as part of Onboarding) and the parameters API Invoker wanted to renew (APIList,ApiInvokerInformation etc).

Also, API Invoker can request CCF to generate a new Onboard\_Secret as part of OnboardingUpdateRequest.

If ProposedExpirationTime is present in the OnboardingUpdateRequest and CCF agrees, CCF sends the negotiated ExpirationTime in the OnboardingUpdateResponse. Otherwise CCF may provide its own ExpirationTime in the response.

GenerateOnBoardSecret is an optional parameter which can take True/False.

ProposedExpirationTime is an optional parameter to propose expiration time of the onboarding.

APIList, APIInvokerInformation are optional parameters defined in CAPIF specs and if present CCF updates the information received from APIInvoker.

Step 7: If GenerateOnBoardSecret is received as True, CAPIF Core Function generates new Onboard\_Secret.

If GenerateOnBoardSecret is not received or received as false, existing Onboard\_Secret will be reused.

If ProposedExpirationTime is received, CAPIF Core Function generates the new expiration time for onboarding.

Other parameters are updated in CCF.

Step 8: CAPIF Core Function sends the OnboardingUpdateResponse with the OnboardingStatus OnboaringInformation, APIList etc received in the request, as well as a new Expiration Time to API Invoker.

TS 33.122 [4] Annex B.1 diagram is proposed to be updated as follows:

Renewal process:



### 6.38.3 Evaluation

The solution allows the API invoker to receive the updated APIInvokerEnrolmentDetails from CCF but does not lose the API invoker context information already registered at the CCF.The need of onboarding update request following a successful onboarding needs further considerations.

Whether this solution is in the scope of the key issue would need further evaluation.

## 6.39 Solution #39: ROF certificate generation

### 6.39.1 Introduction

This solution is addressing KI#1.

The security procedures between the ROF and the authorization function/CCF supporting the Resource owner-aware Northbound API Access (RNAA) were left open in Release 18. To allow for mutual authentication, ROF and CCF need to be issued with a certificate.

### 6.39.2 Solution details

ROF gets a TLS certificate from CCF and using the same with CCF for mTLS connection further.

1) Primary authentication happens between UE and AUSF in 5GC and k-AUSF is derived at both UE and AUSF.

2) ROF gets the CCF certificate from the network/local APIInvoker.

3) ROF establishes a server based TLS connection with CCF.

4) ROF generates a random string, generates a hash using the k-AUSF.

5) ROF sends a Authentication request to CCF with GPSI/SUPI, random string and the generated hash.

6) CCF sends GenerateHash request to AUSF by passing GPSI/SUPI and the random string.

7) AUSF identifies the k-AUSF associated with SUPI/GPSI and generates the hash on the random string using k-AUSF.

8) AUSF sends the generated hash to CCF.

9.C CF validates the hash received from AUSF with that of received from ROF.

10) On successful validation generates a ROF certificate with resourceOwnerId in the certificate.



### 6.39.2 Evaluation

The solution allows for mutual authentication. Whether the solution works depends on the ROF definition.

Impact: a new key is needed. The impact needs further to be analyzed.

Evaluation would need to be provided before further consideration.

## 6.Y Solution #Y: <Title>

### 6.Y.1 Introduction

### 6.Y.2 Solution details

### 6.Y.3 Evaluation

# 7 Conclusions

### 7.1.1 Conclusions for KI#1.1 CAPIF-8 reference point

Normative work is recommended to protect the CAPIF-8 reference point based on the following principles:

The authentication of the CCF is based on TLS using the CCF’s certificate.

The ROF authentication is to be decided in the normative work.

The TLS secure channel is used to provide messages exchanged between the ROF and the CCF with integrity protection, confidentiality protection and. replay protection.

### 7.1.2 Conclusions for KI#1.2 Resource owner authorization management

The following statements are agreed as a basis for normative work.:

#### 7.1.2.1 Authentication and authorization of the end points and security of transferred authorization data

Authorization information/authorization revocation information is transferred between the ROF and the CCF utilizing secure CAPIF-8 reference point between ROF and CCF.

The resource owner is authenticated before being allowed to manage the permissions (including revocation) to allow the API invoker to access the resource owner’s resource via the northbound API.

How to authenticate and authorize the resource owner is left to implementation.

NOTE: Authentication between the ROF and the CCF is addressed in KI#1.1.

#### 7.1.2.2 Resource owner authorization data

Authorization information includes Resource Owner Identifier. Other information to be obtained and stored for the resource owner authorization is to be determined in the normative work.

#### 7.1.2.3 Revocation

The existing mechanisms for the revocation in TS 33.122 [4] are reused. What revocation information is sent by the ROF for the CCF to identify the revoked RNAA-related token is to be determined in the normative work.

### 7.1.3 Conclusions for KI#1.3 Finer granular authorization

Normative work is recommended to support API service operation level and API resource level granularity in RNAA scenarios.

NOTE 1: The finer level authorization will be aligned with that specified in TS 23.222 [3].

NOTE 2: Whether finer level authorization can be used in non-RNAA scenarios will be discussed in normative work and aligned with TS 23.222 [3].

NOTE 3: Whether purpose of data access is to be supported will be discussed in normative phase.

NOTE 4: Whether API feature level is included is to be discussed in normative phase.

## 7.2 Conclusion for KI #2: CAPIF interconnection security

### 7.2.0 General

It is assumed that the API invoker onboards to CCF-A, which is referred as onboarded CCF.

It is assumed that the API invoker is onboarded to CCF-A and the target AEF is registered to a different CCF-B.

### 7.2.1 Conclusion for CAPIF 6/6e security

It is concluded that for CAPIF-6 and CAPIF-6e reference points, same security mechanisms specified in clauses 6.6 and 6.10 of TS 33.122 [4] for CAPIF-3/4/5 and CAPIF-3e/4e/5e reference points will be used, respectively.

### 7.2.2 Conclusion for security method negotiation

For security method negotiation procedure (as per requirement 2), clause 6.3.1.2 in TS 33.122 [4] will be used as baseline with the necessary enhancements (if any).

The details of how security method selection is done for the CAPIF 2/2e reference point based on the capabilities of the API Invoker and the AEF capabilities (that belongs to CCF-B) are up to normative work.

### 7.2.3 Conclusion for API invoker authentication and authorization mechanism

Authorization between API invoker (on-boarded to CCF-A) and the AEF (registered to CCF-B) is specified in TS 23.222 [2].

For mutual authentication between API invoker (on-boarded to CCF-A) and the AEF (registered to CCF-B), the procedures as defined in clause 6.5.2 of TS 33.122 [4] can be re-used with the following enhancement:

- When using TLS-PSK or PKI:

- On receiving the request from the AEF, CCF-B requests the security information (AEFPSK/root CA) from CCF-A (over CAPIF-6/6e reference point).

- The AEF learns the authorization information from CCF-B. Whether CCF-B can obtain access control policy from CCF-A to be decided during normative work

- When using TLS with OAuth token:

- On receiving the request from the AEF, CCF-B requests the security information (e.g., root CA) from CCF-A (over CAPIF-6/6e reference point).

- CCF-A can send the access token request to CCF-B.

- CCF-B can provide an access token to the API invoker via CCF-A as specified in clause 6.5.2.3 in TS 33.122 [4].

- The AEF verifies the access token as described in 6.5.2.3 in TS 33.122 [4].

- Whether CCF-A can issue an access token and whether CCF-B needs to verify CCF-A is authorized for the authorization decision are to be determined during normative work.

- For the case of TLS-PSK or PKI, and TLS with OAuth token methods listed above, the specific details of how the CCF-B requests the security information of the API Invoker from the right CCF-A, i.e. where API invoker is onboarded, are up to normative work.

- For the case of TLS with OAuth token methods, how CCF-A determines to request token from CCF-B is up to normative work.

- Further details of the procedure are to be determined during normative work.

## 7.3 Conclusion for KI #3: Authorizing API invoker on one UE accessing resources related to another UE

The API invoker residing on the UE needs to be authorized to access resources related to another UE.

The RNAA procedures defined in clause 6.5.3.2 and 6.5.3.3 of TS 33.122 [4] will be reused as baseline.

Whether additional flows and details are needed is to be discussed in the normative work.

## 7.4 Conclusion for KI #4: Nested API Invocation

Potential enhancements can have impacts to AEF and CCF.

For authorization of nested API invocation token exchange framework as specified in IETF RFC 8693 [5] can be used.

Details of the procedure will be specified in normative work.

## 7.5 Conclusion for KI #5: Authenticating multiple API invokers of the same Resource Owner

When the ROF is engaging multiple API invokers at the same UE and the ROF has a valid certificate, the authentication process could be optimized to allow for mutual authentication for interaction of several API invokers with AEF as described by solution #28 having impacts on CAPIF security architecture.

However, no conclusion on normative work is reached.

## 7.6 Conclusion for KI #6: Onboarding security issues

The API invoker onboarding/offboarding procedure defined in TS 33.122 [4] can be reused as baseline in RNAA scenarios.

Using the existing procedure whether and how CCF is able to onboard the actual API invoker residing in the UE and if any related enhancement is needed will be discussed in the normative work.

Annex <X>:  
Change history

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| **Change history** | | | | | | | |
| **Date** | **Meeting** | **TDoc** | **CR** | **Rev** | **Cat** | **Subject/Comment** | **New version** |
| 2024-08 | SA3#117 | S3-243637 |  |  |  | Skeleton | 0.0.0 |
| 2024-08 | SA3#117 | S3-243718 |  |  |  | Incorporate pCRs that add S3‑243636, S3‑243638, S3‑243640, S3‑243700, S3‑243716, S3‑243136 | 0.1.0 |
| 2024-10 | SA3#118 | S3-244305 |  |  |  | Incorporate pCRs that add S3-244298, S3-244341, S3-244068, S3-244342, S3-244343, S3-244344, S3-244345, S3-244346, S3-244347, S3-244348, S3-244349, S3-244350, S3-244351, S3-244352, S3-244353, S3-244354, S3-244355, S3-244356, S3-244357, S3-244070, S3-244358, S3-244359, S3-244360, S3-244361, S3-244362, S3-244363, S3-244364, S3-244159, S3-244461, S3-244462, S3-244463, S3-244466, S3-244168 | 0.2.0 |
| 2024-11 | SA3#119 | S3-245205 |  |  |  | Incorporate pCRs that add S3‑244773, S3‑245220, S3‑245221, S3‑245222, S3‑245223, S3‑245224, S3‑245225, S3‑245226, S3‑245227, S3‑245228, S3‑245229, S3‑245230, S3‑245233, S3‑245234, S3‑245236, S3‑245237, S3‑245238, S3‑245239, S3‑245240, S3‑245241, S3‑245242, S3‑245243, S3‑245244, S3‑245245, S3‑245246, S3‑245247, S3‑245113, S3‑245249, S3‑245250, S3‑245251, S3‑245252, S3‑245253, S3‑245254, S3‑245255, S3‑244940, S3‑244941, S3‑245248, S3‑245231, S3‑245235, S3-245371 | 0.3.0 |
| 2025-01 | SA3#119AdHoc-e | S3-250235 |  |  |  | Incorporate pCRs that add S3-250182, S3 -250196, S3-250189,  S3-250197, S3-250020, S3-250212, S3-250216, S3-250187, S3-250040, S3-250188, S3-250199, S3-250200, S3-250201, S3-250161, S3-250162, S3-250163, S3-250191, S3-250198 | 0.4.0 |
| 2025-01 | SA3#119 AdHoc-e | S3-250236 |  |  |  | Editorial corrections in clauses 7.1.2.1, 6.2.2.1, 6.4.3  Clause 7.2.3 updated to align with S3-250189 | 0.5.0 |
| 2025-02 | SA3#120 | S3-250976 |  |  |  | Incorporate pCRs that add S3‑250998, S3‑250999, S3‑251000, S3‑251001, S3‑251002, S3‑251023, S3‑251024, S3‑251025, S3‑250582, S3‑251026, S3‑251028, S3‑251027, S3‑251003, S3‑251004, S3‑251005, S3‑251006, S3‑250896, S3‑251020, S3‑250903, S3‑250906, S3‑250907, S3‑250908, S3‑250909, S3‑250910, S3‑250911, S3‑251021, S3‑250915, S3‑251015,  S3‑250918, S3‑251018, S3‑250922, S3‑251019, S3‑251022, S3‑251014, S3‑251016, S3‑251017, S3‑251029 | 0.6.0 |