|  |  |
| --- | --- |
| 3GPP TR 33.700-22 V0.2.0 (2024-08) | |
| Technical Report | |
| 3rd Generation Partnership Project;  Technical Specification Group Services and System Aspects;  Study on security aspects of CAPIF Phase3  (FS\_CAPIF\_Ph3-sec)  (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.  © 2022, 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 8

1 Scope 10

2 References 10

3 Definitions of terms, symbols and abbreviations 11

3.1 Terms 11

3.2 Symbols 11

3.3 Abbreviations 11

4 High-level architectures 11

4.1 High-level architecture for RNAA 12

4.2 High-level architecture for CAPIF interconnection 12

5 Key issues 14

5.1 Key Issue #1: Security of resource owner authorization management and CAPIF-8 reference point 14

5.1.0 Introduction 14

5.1.1 Key Issue #1.1: CAPIF-8 reference point 14

5.1.1.1 Key issue details 14

5.1.1.2 Security threats 14

5.1.1.3 Potential Security Requirement 14

5.1.2 Key Issue #1.2: Resource owner authorization management 15

5.1.2.1 Key issue details 15

5.1.2.2 Security threats 15

5.1.2.3 Potential Security Requirement 15

5.1.3 Key Issue #1.3: Finer granular authorization 15

5.1.3.1 Key issue details 15

5.1.3.2 Security threats 15

5.1.3.3 Potential Security Requirement 16

5.2 Key issue #2: CAPIF interconnection security 16

5.2.1 Key issue details 16

5.2.2 Security threats 17

5.2.3 Potential security requirements 17

5.3 Key Issue #3: Authorizing API invoker on one UE accessing resources related to another UE 18

5.3.1 Key issue details 18

5.3.2 Security threats 18

5.3.3 Potential security requirements 18

5.4 Key issue #4: Nested API invocation 19

5.4.1 Key issue details 19

5.4.2 Security threats 19

5.4.3 Potential security requirements 19

5.5 Key Issue KI#5: Authenticating multiple API invokers of the same Resource Owner 19

5.5.1 Key Issue details 19

5.5.2 Security threats 19

5.5.3 Security requirements 19

6 Proposed solutions 20

6.0 Mapping of solutions to key issues 20

6.1 Solution #1: Security protection mechanism for CAPIF-8 reference point 22

6.1.1 Introduction 22

6.1.2 Solution details 22

6.1.3 Evaluation 22

6.2 Solution #2: CAPIF-8 reference point security 22

6.2.1 Introduction 22

6.2.2 Solution details 22

6.2.2.1 Mutual authentication 22

6.2.2.2 Protection of messages between ROF – AzF 22

6.2.3 Evaluation 22

6.3 Solution #3: Security procedures for CAPIF-8 reference points 23

6.3.1 Introduction 23

6.3.2 Solution details 23

6.3.3 Evaluation 23

6.4 Solution #4: resource owner authorized revocation 23

6.4.1 Introduction 23

6.4.2 Solution details 23

6.4.3 Evaluation 24

6.5 Solution #5: Security of resource owner authorization management and CAPIF-8 24

6.5.1 Introduction 24

6.5.2 Solution details 24

6.5.3 Evaluation 26

6.6 Solution #6: Security procedures for resource owner authorization management 26

6.6.1 Introduction 26

6.6.2 Solution details 26

6.6.2.1 Security procedure for obtaining resource owner authorization 26

6.6.2.2 Security procedure for authorizing the API invoker in RNAA 27

6.6.2.3 Security procedure for revoking resource owner authorization 28

6.6.3 Evaluation 28

6.7 Solution #7: RO permission/authorization management 28

6.7.1 Introduction 28

6.7.2 Solution details 29

6.7.3 Evaluation 29

6.8 Solution #8: Resource owner triggered revocation procedure 29

6.8.1 Introduction 29

6.8.2 Solution details 30

6.8.3 Evaluation 31

6.9 Solution #9: Resource owner authentication and authorization mechanism 31

6.9.1 Introduction 31

6.9.2 Solution details 31

6.9.3 Evaluation 32

6.10 Solution #10: resource-level and/or api-level authorization and revocation 32

6.10.1 Introduction 32

6.10.2 Solution details 32

6.10.2.1 Service operation/resource level authorization 32

6.10.2.2 Service operation/resource level revocation 32

6.10.3 Evaluation 32

6.11 Solution #11: Client initiated backchannel authorization (CIBA) based solution 33

6.11.1 Introduction 33

6.11.2 Solution details 33

6.11.3 Evaluation 33

6.12 Solution #12: Security method retrieval in CAPIF interconnect 34

6.12.1 Introduction 34

6.12.2 Solution details 34

6.12.2.1 Summary 34

6.12.2.2 Information flow 35

6.12.3 Evaluation 35

6.13 Solution #13: Requesting security information from another CCF in order to authenticate using TLS-PSK in CAPIF interconnect 36

6.13.1 Introduction 36

6.13.2 Solution details 36

6.13.2.1 Summary 36

6.13.2.2 Information flow 36

6.13.3 Evaluation 37

6.14 Solution #14: Authentication aspect in CAPIF interconnect when API invoker has not included CCF information 37

6.14.1 Introduction 37

6.14.2 Solution details 37

6.14.2.1 Summary 37

6.14.2.2 Information flow 38

6.14.3 Evaluation 38

6.15 Solution #15: Authorization token request handling in CAPIF interconnect 39

6.15.1 Introduction 39

6.15.2 Solution details 39

6.15.2.1 Summary 39

6.15.2.2 Information flow 40

6.15.3 Evaluation 41

6.16 Solution #16: Mapping an API invoker authorization request to the correct CCF in CAPIF interconnect 42

6.16.1 Introduction 42

6.16.2 Solution details 42

6.16.2.1 Summary 42

6.16.2.2 Information flow 43

6.16.3 Evaluation 43

6.17 Solution #17: Security procedures for CAPIF interconnection 43

6.17.1 Introduction 43

6.17.2 Solution details 43

6.17.2.1 Security procedure for API invoker authentication and authorization in CAPIF interconnection 43

6.17.2.2 Security procedure for API invoker offboarding in CAPIF interconnection 45

6.17.3 Evaluation 45

6.18 Solution #18: API invoker authentication mechanism in CAPIF interconnection scenarios 45

6.18.1 Introduction 45

6.18.2 Solution details 46

6.18.2.1 TLS-PSK based authentication mechanism for CCF interconnection scenarios 46

6.18.2.2 TLS-PKI based authentication mechanism for CCF interconnection scenarios 47

6.18.3 Evaluation 47

6.19 Solution #19: API invoker authorization mechanism in CAPIF interconnection scenarios 47

6.19.1 Introduction 47

6.19.2 Solution details 48

6.19.3 Evaluation 49

6.20 Solution #20: Security method negotiation mechanism in CAPIF interconnection scenarios 49

6.20.1 Introduction 49

6.20.2 Solution details 49

6.20.3 Evaluation 50

6.21 Solution #21: Solution for CAPIF interconnection security 50

6.21.1 Introduction 50

6.21.2 Solution details 51

6.21.3 Evaluation 51

6.22 Solution #22: CAPIF interconnection 51

6.22.1 Introduction 51

6.22.2 Solution details 51

6.22.3 Evaluation 52

6.23 Solution #23: Security protection mechanism for CAPIF-6 and CAPIF-6e reference points 52

6.23.1 Introduction 52

6.23.2 Solution details 53

6.23.3 Evaluation 53

6.24 Solution #24: Security procedure for CAPIF interconnection 53

6.24.1 Introduction 53

6.24.2 Solution details 53

6.24.2.1 Security method negotiation in CAPIF interconnection 53

6.24.2.2 Authentication and authorization 54

6.24.2.2.1 Authentication and authorization with security method TLS-PSK or PKI 54

6.24.2.2.2 Authentication and authorization with security method TLS with OAuth token 56

6.24.3 Evaluation 57

6.25 Solution #25: Backend based solution for UE-deployed API invoker accessing resources not owned by that UE 57

6.25.1 Introduction 57

6.25.2 Solution details 57

6.25.3 Evaluation 59

6.26 Solution #26: Nested API invocation 59

6.26.1 Introduction 59

6.26.2 Solution details 59

6.26.3 Evaluation 60

6.27 Solution #27: Authorization for nested API invocation 61

6.27.1 Introduction 61

6.27.2 Solution details 61

6.27.2.3 Access token claims 63

6.27.3 Evaluation 63

6.28 Solution #28: Authenticating multiple API invokers of the same RO 63

6.28.1 Introduction 63

6.28.2 Solution Details 63

6.28.3 Evaluation 65

6.29 Solution #29: Enhancing authorization through finer granularity access token. 65

6.29.1 Introduction 65

6.29.2 Solution details 65

6B.29.2.1 Summary 65

6.29.2.2 Information flow 66

B.X.3 Evaluation 67

6.Y Solution #Y: <Title> 67

6.Y.1 Introduction 67

6.Y.2 Solution details 67

6.Y.3 Evaluation 67

7 Conclusions 67

Annex <X>: Change history 68

# 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

Editor's 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.

# 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 4.2-1 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 3GPP 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 3GPP 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.

Editor’s note: What privacy threats exist is FFS if CAPIF-8 reference point has been protected by confidentially, integrity and anti-replay mechanisms.

#### 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 3GPP TS 33.122 [4].

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

As from clause 6.5.3. in TS 33.122 [4], (for Release 18) three authorization flows are introduced to be used as potential procedures in RNAA enhanced CAPIF. These are client credentials, authorization code and authorization code with PKCE. Client credentials flow generally becomes applicable when the authorization is preconfigured in the authorization server by an out of band mechanism. The authorization code and authorization code with PKCE require online user interaction. The case where there is no preconfigured authorization information and the authorization from the user can only be obtained by an out of band mechanism is not addressed.

Editor’s Note: For the case where there is no preconfigured authorization information and the authorization from the resource owner can only be obtained by an out of band mechanism, any new authorization flow needs to be studied or not is FFS.

#### 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.

### 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 A (CCF-A), where it is registered. It does not communicate with the interconnected CCF in CAPIF provider domain (CCF-B), 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 23.700-22 KI #6 [3].

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.

### 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.

# 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 |
| 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 #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 |  |  |  |  |

## 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

TBD.

## 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 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.

### 6.2.2 Solution details

#### 6.2.2.1 Mutual authentication

For authentication between a ROF and an AzF, mutual authentication based on client and server certificates is performed using TLS. The certificate profiles follow the TS 33.310 [6], clause 6.1.3a. The identities in the end entity certificates are used for authentication and policy checks.

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

Editor’s Note: the need for Authentication between a ROF and an AzF in this solution is ffs.

Editor’s Note: whether client certificate can be used here is ffs.

#### 6.2.2.2 Protection of messages between ROF – AzF

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 clause 6.2 of TS 33.210 [7].

### 6.2.3 Evaluation

TBD

## 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 3GPP TS 33.220 [8], or Authentication and Key Management for Applications (AKMA) specified in 3GPP 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 ROC 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 ROC 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 ROC and CCF using TLS. Certificate based authentication shall follow the profiles given in 3GPP 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.

Editor’s Note: The authentication method to be used in TLS is FFS.

Editor’s Note: Whether GBA, AKMA and TLS need to be simultaneously supported is FFS.

Editor’s Note: How the solution works for the ROF that can be a web browser is FFS.

### 6.3.3 Evaluation

TBD.

## 6.4 Solution #4: resource owner authorized revocation

### 6.4.1 Introduction

This solution addresses revocation part of the "Key Issue #1.2: Resource owner authorization management ".

In TS 33.122 [4], it is stated

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

In this solution, this 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

The procedure for revoking API invoker authorization initiated by the resource owner through the resource owner function (ROF) is given below:

Editor’s Note: how authorization data is provided to CCF/AEF is ffs

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).

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., for the QoS service API, the AEF may inform PCF/SMF to modify the QoS level of corresponding PDU sessions after revocation.

Editor’s Note: the need for resource update is ffs.

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 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

TBD

Editor’s Note: whether short-lived token can be used is ffs.

## 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). Alternatively, CAPIF-8 can be secured by reusing AKMA procedure described in TS 33.535 Clause 6. Figure 6.Y.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.

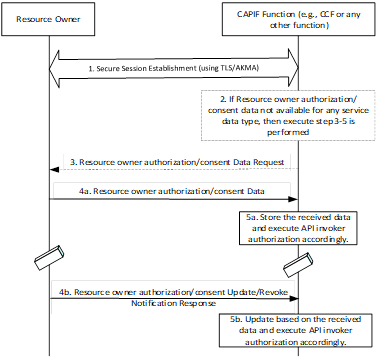


Figure 6.5.2-1 Resource authorization management procedure

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 of the API Invoker), 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.

Editor's Note: The authentication mechanisms to be used in TLS and the AKMA based protocol to be used for communication security are FFS.

### 6.5.3 Evaluation

TBD

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

### 6.6.1 Introduction

This solution addresses the 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

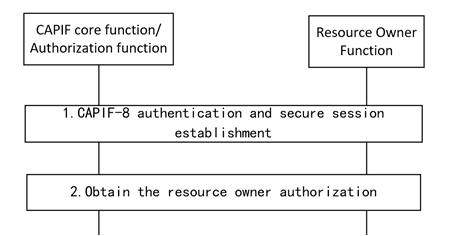


Figure 6.6.2.1-1: Procedure for obtaining resource owner authorization

1. The CCF authenticates the ROF and establishes a secure session via CAPIF-8 reference point.

2. The ROF provides the resource owner authorization to the CCF. The CCF stores the resource owner authorization and security information related to authenticate the ROF.

Editor’s Note: The security information related to authenticate the ROF is FFS.

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

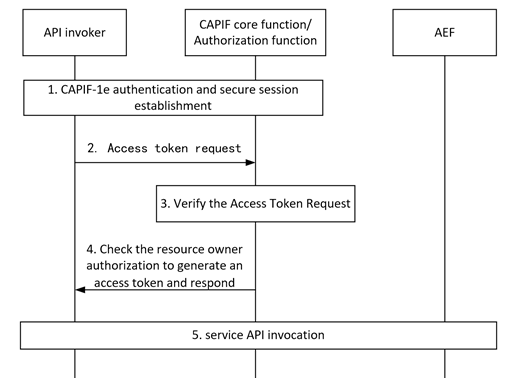


Figure 6.6.2.2-1: Procedure for authorizing the API invoker in RNAA

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 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.

5. Authentication between the API invoker and the AEF and service API invocation in RNAA are performed following step 5-8 of subclause 6.5.2.3 in TS 33.122 [4].

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

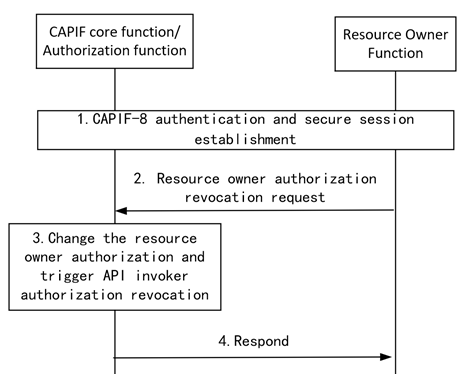


Figure 6.6.2.3-1: Procedure for revoking resource owner authorization

1. The CCF authenticates the ROF and establishes a secure session via CAPIF-8 reference point.

2. The ROF sends a resource owner authorization revocation request to the CCF to provide resource owner revocation information. The information indicates which resource owner authorization is requested for revocation.

3. After receiving the resource owner revocation information, the CCF changes the resource owner authorization and determines whether to trigger a revocation procedure specified in subclause 8.23.4 of 3GPP TS 23.222 [i.z] to revoke the API invoker authorization.

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

### 6.6.3 Evaluation

TBD.

## 6.7 Solution #7: RO permission/authorization management

Editor's Note: The term "permission" may need to be updated to align with discussion outcome.

### 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 resource owner permission from the ROF over CAPIF-8 interface.

### 6.7.2 Solution details

RO authorization management needs a set of service operations to manage the permissions. The following sketches them.

- 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.

Editor's Note: How CCF sends messages to the ROF in the UE via AMF NAS operation is ffs.

- 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.

### 6.7.3 Evaluation

TBD

## 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

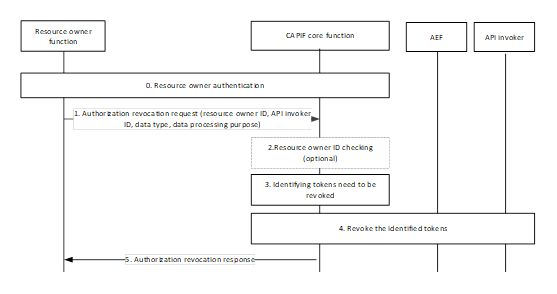


Figure 6.8.2 Resource owner triggered revocation procedure

1. The resource owner function obtains the address information of CCF from the network or API invoker. 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).

Editor’s Note: ROF authentication is FFS.

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.

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].

Editor’s Note: The token revocation is FFS.

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

### 6.8.3 Evaluation

TBD.

## 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

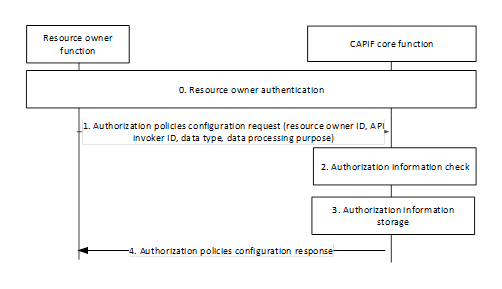


Figure 6.9.2 Resource-owner-related authorization

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. 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)

Editor’s Note: The ROF authentication is FFS.

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.

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

TBD.

## 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 authoriztaion 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.

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

### 6.10.3 Evaluation

TBD

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

### 6.11.1 Introduction

This solution addresses the requirements identified in key issue #1.2 (Resource owner authorization management).

Editor’s Note: Clarification on which requirements is addressed by this solution is FFS.

In some cases, there can be no preconfigured authorization information and the permission from the resource owner is requested by an out of band mechanism. Client credentials and authorization code flow (including PKCE) 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.

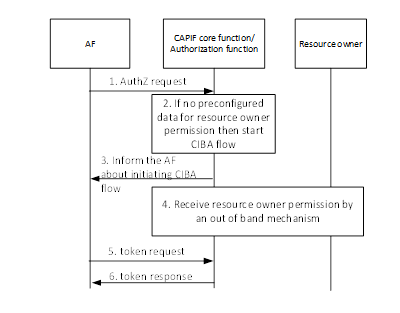


Figure 6.11.2-1: High-level procedure flow of the solution

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

2. The CCF 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 decides to use CIBA flow.

3. The CCF informs the AF about initiating the CIBA flow, so that the AF executes steps 5 and 6 regularly until the CCF obtains permission from the resource owner.

4. The CCF obtains permission from the resource owner by using an out of band mechanism.

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

Editor’s Note: Details and impact of the execution of steps 5-6 multiple times is FFS.

### 6.11.3 Evaluation

TBD.

## 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

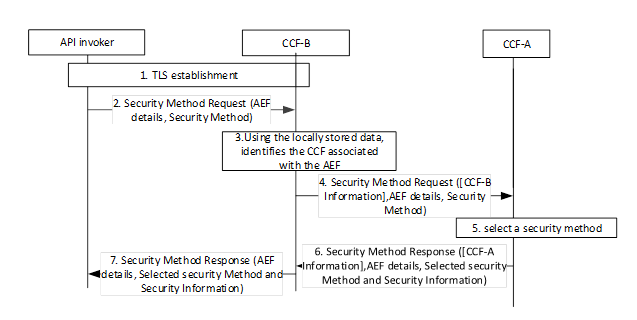


Figure 6.12.2.2-1: Information flow to retrieve security method in interconnect

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

TBD

## 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

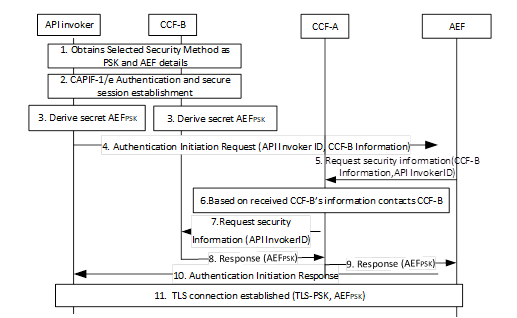


Figure 6.13.2.2-1: Information flow to allow authenticating API invoker to AEF in a different security domain

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.

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.

Editor's Note: The need of the APIInvoker to provide the CCF address is ffs.

### 6.13.3 Evaluation

TBD

## 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

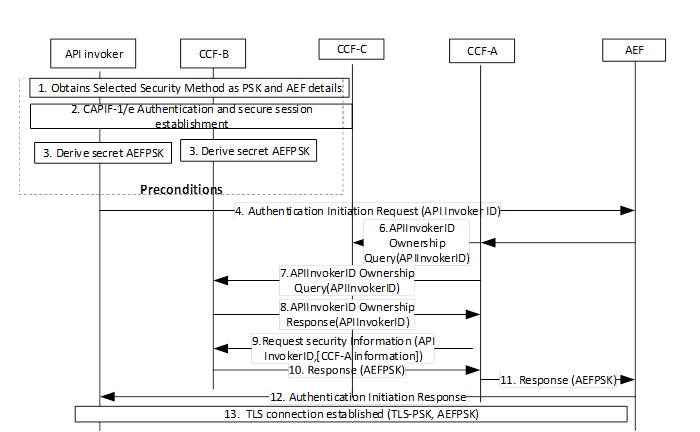


Figure 6.14.2.2-1: Information flow to allow establishment of TLS-PSK in interconnect

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

TBD

## 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

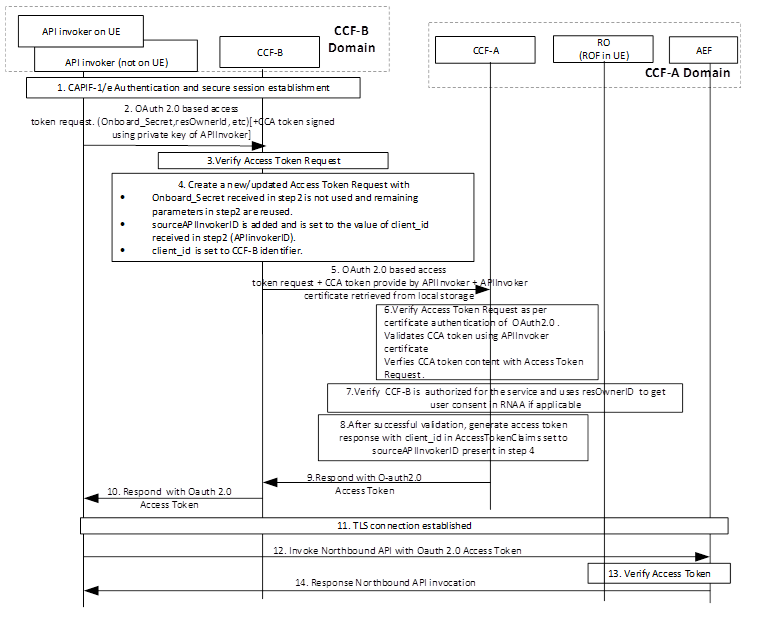


Figure 6.15.2.2-1: Information flow to retrieve security method in interconnect

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).

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.

Editor's Note: The onboarding part of API invoker being part of UE is ffs.

### 6.15.3 Evaluation

TBD

## 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.

### 6.16.2 Solution details

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.

#### 6.16.2.1 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.

Editor's Note: Alignment of step 1-5 with existing authorization code flow is ffs.

Editor's Note: Step 5/6 are interrupting the RO.

Editor's Note: The clients involved in steps 14/15 related to capif interconnect in API invoker access token request are ffs.

#### 6.16.2.2 Information flow

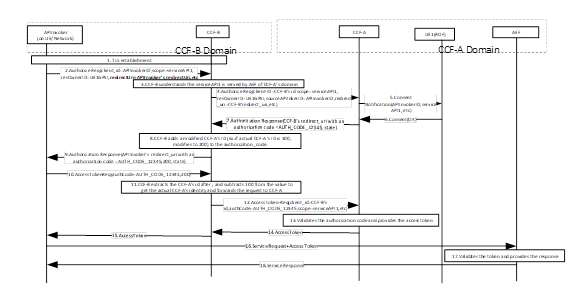


Figure 6.16.2.2-1: Identifying in an API access token request with authorization code the correct CCF in CAPIF interconnect

### 6.16.3 Evaluation

TBD

## 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”.

The solution 6.17.2.1 enhances the Authentication and Authorization specified in subclause 6.5.2 of 3GPP 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. It is proposed to reach an agreement that, whatever the security is (method 1, method 2 or method 3), the CCF A authorizes the service API invocation and the CCF B provides the security information in CAPIF interconnection scenario.

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 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 via procedure defined in step 1 and 2 of clause 8.25.3.3 of 3GPP TS 23.222 [i.y].

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.

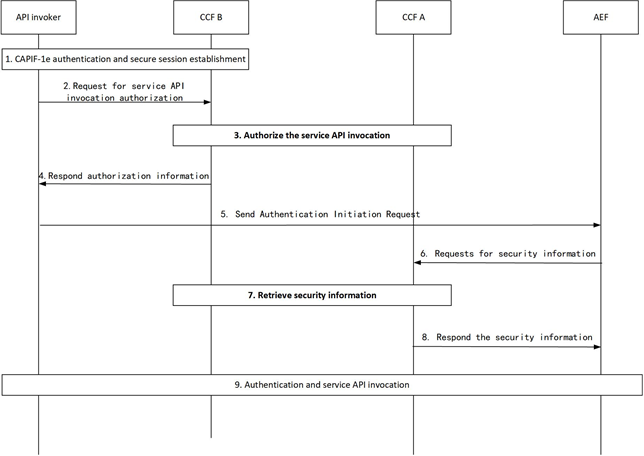


Figure 6.17.2.1-1: Procedure for API invoker authentication and authorization in CAPIF interconnection

1. CAPIF-1e authentication and secure session is established as specified in subclause 6.3.1 of 3GPP TS 33.122 [4].

2. The API invoker requests to the CCF B for obtaining authorization to access the service API published by CCF A via CAPIF-6/6e.

3. The CCF B requests the CCF A to authorize the service API invocation of the API invoker.

4. If the CCF A permits the service API invocation, the CCF B responds the authorization information to the API invoker.

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

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

7. The CCF A retrieves security information based on API invoker 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.

9. Authentication and secure interface establishment between the AEF and the API invoker is performed with the security information. And the AEF authorizes the API invoker's service API invocation request based on authorization information obtained from CCF A as specified in subclause 8.16 of TS 23.222 [i.y].

#### 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 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

TBD.

## 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 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

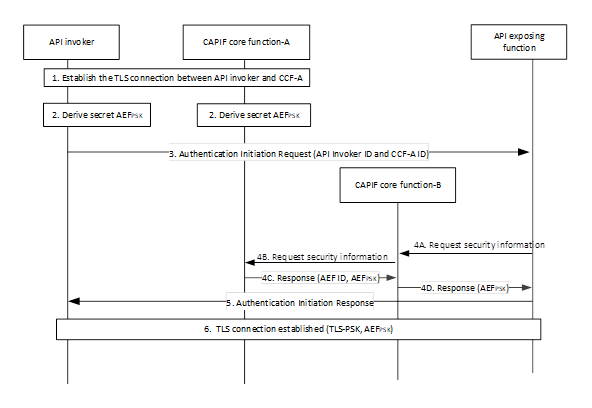


Figure 6.18.2.1 TLS-PSK based authentication mechanism for CCF interconnection scenarios

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

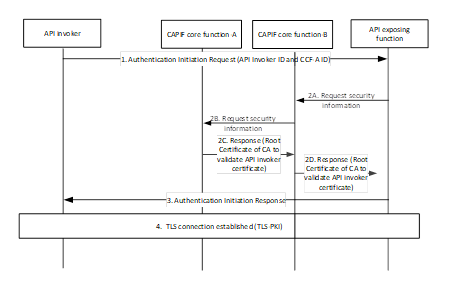


Figure 6.18.2.2: TLS-PKI based authentication mechanism for CCF interconnection scenarios

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

TBD.

## 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

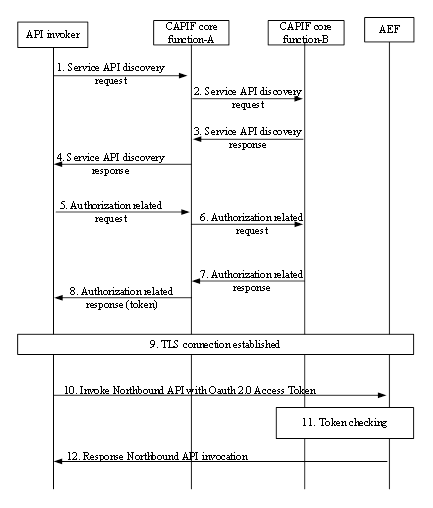


Figure 6.19.2 API invoker authorization mechanism in CAPIF interconnection scenarios

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.

Editor’s Note: The alignment with SA6 is FFS.

### 6.19.3 Evaluation

TBD.

## 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 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.

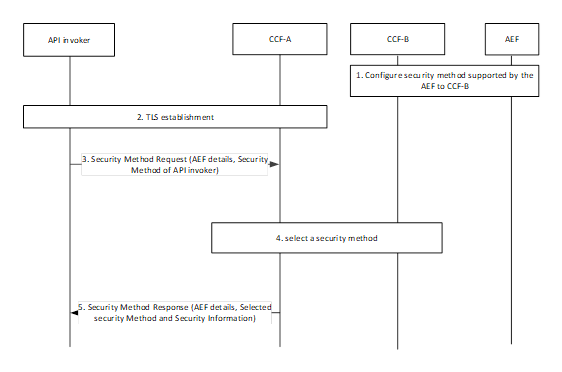


Figure 6.20.2 Selection of security method to be used in CAPIF interconnection scenarios

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

TBD.

## 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 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 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 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 Clause 6.5.

The mutual authentication between CCFs over the CAPIF-6/6e reference point can reuse security procedures described in TS 33.122 Clause 6.6 and TS 33.122 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 Clause 6.3.1.2.

Editor’s notes: How to forward the security information and/or token request is FFS.

### 6.21.3 Evaluation

TBD

## 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.

### 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.

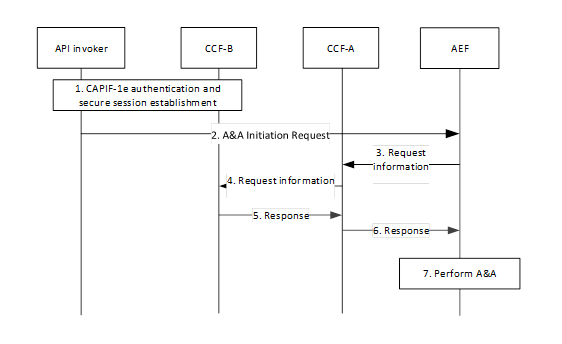


Figure 6.22.2-1: High level solution for authentication and authorization of the API invoker in the CAPIF-2e reference point.

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).

Editor’s Note: The details of "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)" and the claims verification is FFS.

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.

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

### 6.22.3 Evaluation

TBD.

## 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 3GPP 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,

- 3GPP TS 33.210 [7] shall be applied to secure messages on the reference points specified otherwise; and

- 3GPP TS 33.310 [6] may be applied regarding the use of certificates with the security mechanisms of 3GPP TS 33.210 [7] unless otherwise specified in the present document.

### 6.23.3 Evaluation

TBD

## 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

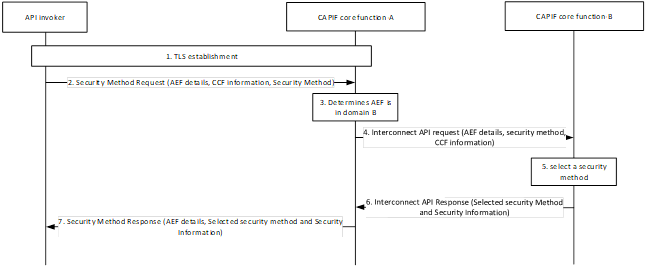


Figure 6.24.2.1-1: Selection of security method to be used in CAPIF-2/2e reference point

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 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.

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 1: 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 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.

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: 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.

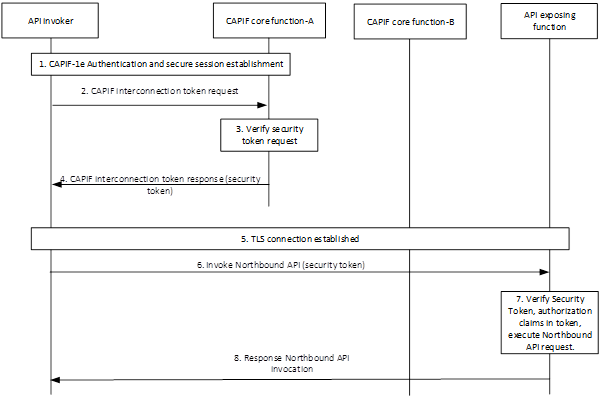


Figure 6.24.2.2-1: Authentication and authorization procedure during CAPIF Interconnect – Method 3

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.Y.2.1.

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.

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

- TBD

## 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.

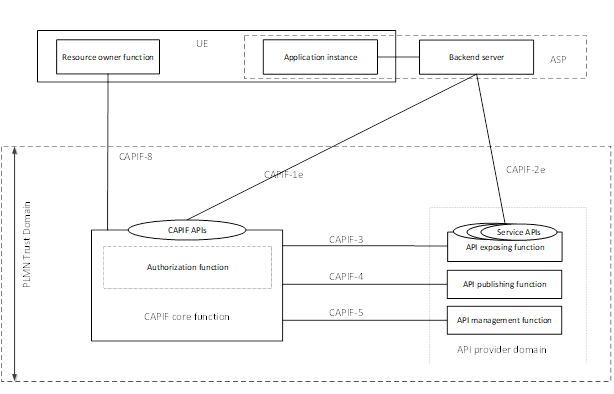


Figure 6.25.2-1: High level architecture

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

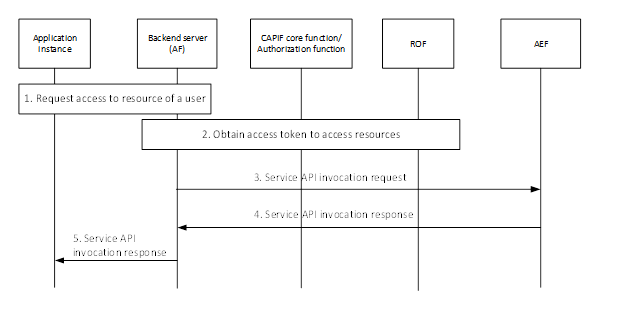


Figure 6.25.2-2: High-level procedure flow of the solution

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.

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

TBD.

## 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 [XX]. 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.

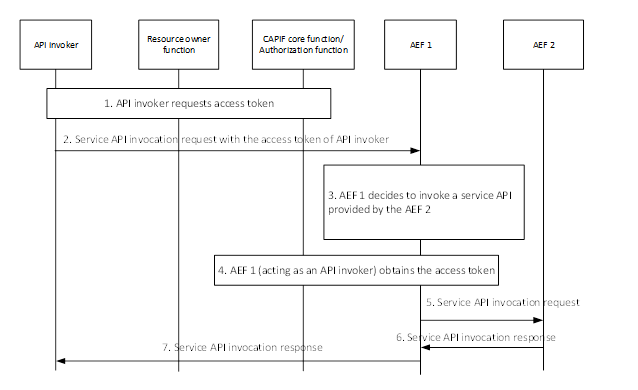


Figure 6.26.2-1: Procedure for obtaining authorization information in a nested API invocation

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. 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.

Editor’s Note: Whether the actor token is need and if so, how it is issued is FFS.

### 6.26.3 Evaluation

TBD.

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

### 6.27.1 Introduction

This solution addresses the security requirements of key issue#4. 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.

### 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 additionally conveys the following actor claims for nested API invocation as specified in IETF RFC 8693 [5].

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.

## Editor’s Note: Additional evaluation is FFS.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 signd 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.

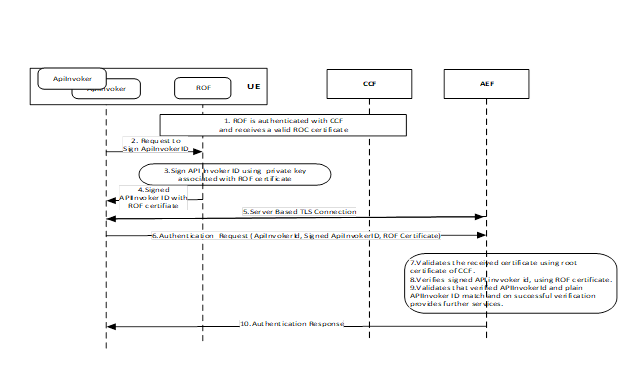


Figure 6.28.2-1: Authentication using the ROF certificate

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

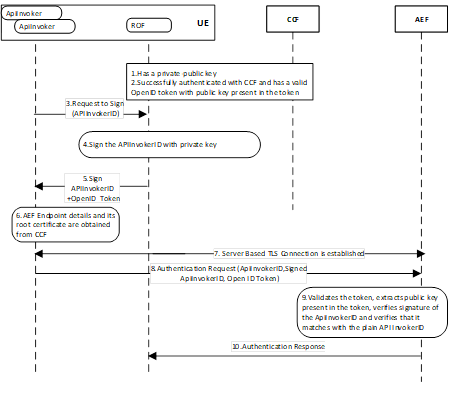


Figure 6.28.2-2: Authentication using OpenID token

Editor's Note: Whether the ROF can be used to authenticate the API invoker is ffs.

Editor’s Note: How the solution works if ROF is web browser is ffs.

### 6.28.3 Evaluation

TBD

## 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 23.700-22 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

#### 6B.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.

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.

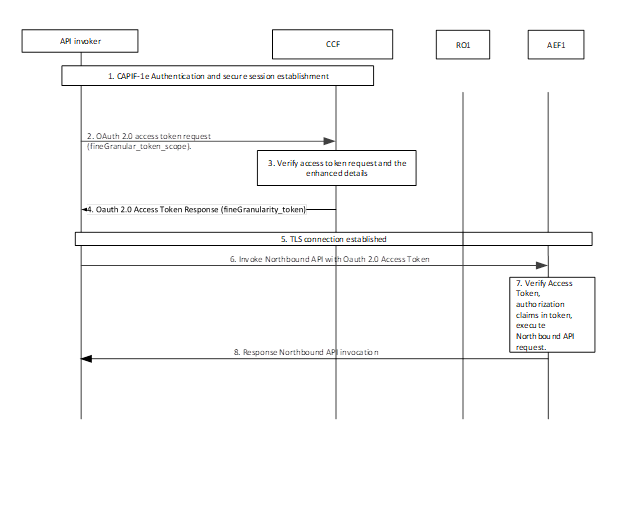
When the verification is completed, the CCF will include the authorization details, with the new granularity, into the access token returned to the API Invoker.

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.

Editor's note: The inclusion of resource owner identifier is proposed to be a scope parameter which is creating a misalignment with Rel-18, where it is captured in a claim. FFS how to handle.

Editor's note: Granularity details to be added. The authorization information granularity and its details are FFS.

#### 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.

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, such as Resource owner, operation 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.

### B.X.3 Evaluation

TBD

## 6.Y Solution #Y: <Title>

### 6.Y.1 Introduction

### 6.Y.2 Solution details

### 6.Y.3 Evaluation

# 7 Conclusions

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 | 0.2.0 |