**3GPP TSG-SA3 Meeting #109AdHoc-e *draft\_S3-230338-r1***

**Electronic meeting, 16 - 20 January 2023**

**Source: Samsung**

**Title: New Solution on Resource owner Authorization in API Invocation using OAuth 2.0 Authorization Code Grant**

**Document for: Approval**

**Agenda Item: 5.11**

# 1 Decision/action requested

***The contribution proposes a new solution for key issue #2 of TR 33.884.***

# 2 References

NA

# 3 Rationale

This contribution proposes a new solution for a resource owner to provide/revoke the resource owner’s authorization to API Invoker.

# 4 Detailed proposal

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of 1st Change \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

# 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 22.261: "Service requirements for the 5G system".

[3] 3GPP TR 23.700-95: “Study on application enablement aspects for subscriber-aware northbound API access”.

[4] IETF RFC 6749: “The OAuth 2.0 Authorization Framework”.

[5] 3GPP TS 33.122: “Security aspects of Common API Framework (CAPIF) for 3GPP northbound APIs”.

[6] openID.net: " OpenID Connect Core 1.0 incorporating errata set 1". Available at: <https://openid.net/specs/openid-connect-core-1_0.html>

[7] IETF RFC 7009: “OAuth 2.0 Token Revocation”.

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

[xx] IETF RFC 7542: "The Network Access Identifier".

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of 1st Change \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* Start of 2nd Change \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

## 6.X Solution #X: Providing and Revoking Resource Owner Authorization using OAuth 2.0 Authorization Code Grant

### 6.X.1 Introduction

This solution addresses Key Issue #2 "Checking authorization before allowing access".

This solution proposes to use OAuth 2.0 Authorization Code Grant as specified in clause 4.1 of RFC 6749 [4], in which the resource owner can be authenticated by the resource owner's 3GPP credentials.

UE, API Invoker, and Authorization Function in this solution performs the role of User-Agent, Client, and Authorization server in RFC 6749 [4], respectively.

### 6.X.2 Solution details

### 6.X.2.1 Architecture



Figure 6.X.2.1-1 architecture for CAPIF with SNA enhancement

This solution uses an architecture proposed in solution #2 of TR 23.700-95 [3]. As defined in TR 23.700-95 [3], the Resource owner client is an application client used by end-user or subscriber of the API provider domain's service provider.

### 6.X.2.2 Procedure

Pre-requisite:

1. During the primary authentication, AUSF receives SNAAPPY Indication from UDM, which indicates that the AUSF and the UE need to generate the following pre-requisite 2.
2. After the primary authentication, UE and AUSF generate SNAAPPY Key Identifier (S-KID) and KSNAAPPY from KAUSF as detailed in 6.X.2.3 and 6.X.2.4 of this document, respectively. After the S-KID and KSNAAPPY are generated, AUSF sends the KSNAAPPY, S-KID, and SUPI to Authorization Function. The Authorization Function stores this information sent by the AUSF. If there were KSNAAPPY and S-KID corresponding to the same SUPI, they are overridden by the new KSNAAPPY and S-KID.



Figure 6.X.2.2-1 Procedure for resource owner authorization based API invocation

1. After API Invoker performs onboarding procedure to CAPIF core function as specified in clause 6.1 of TS 33.122 [5], the API Invoker mutually authenticates with the CAPIF core function as specified in clause 6.3 of TS 33.122 [5].
2. API Invoker performs mutual authentication with API exposing function and gets an authorization to invoke a service API using one of three methods specified in clause 6.5 of TS 33.122 [5].
3. For a service API which needs a resource owner's authorization, the API Invoker shall get an authorization from the resource owner in addition to the authorization that was obtained in step 2.
4. The API Invoker obtains Authorization Code via OAuth 2.0 Authorization Code Grant as specified in RFC 6749 [4]. While the API Invoker redirects the UE to the Authorization Function, the UE sends SNAAPPY indicator, which indicates that the UE supports the resource owner authentication with KSNAAPPY, to the Authorization Function. If the Authorization Function decides to authenticate the resource owner using KSNAAPPY, the Authorization Function sends a challenge to the UE. UE responds with the S-KID and a hash signature which is generated using the S-KID, the challenge, and KSNAAPPY. The Authorization Function verifies the hash signature using KSNAAPPY which the Authorization Function can find based on the S-KID. If the verification is successful and then the resource owner authorizes the API Invoker to invoke the service API, the Authorization Function sends Authorization Code to the API Invoker.

NOTE: Although the Authorization Function is illustrated as a separate entity from CAPIF core function in Figure 6.X.2.2-1, it may be deployed within the CAPIF core function according to the decision in SA3.

NOTE: Other authentication method between the resource owner and the Authorization Function can be performed if the UE does not send SNAAPPY indicator.

Editor's Note: Whether authenticating the resource owner using 3GPP credential is sufficient is FFS.

1. The Authorization Function generates an OAuth 2.0 token, TokenSNAAPPY. The TokenSNAAPPY conveys the S-KID and the generated time of the TokenSNAAPPY, in addition to the token claims speicified in Annex C.2.2 of TS 33.122 [5].
2. The API Invoker requests the TokenSNAAPPY from the Authorization Function by presenting the Authorization Code sent by the Authorization Function in step 4.
3. The Authorization Function sends the TokenSNAAPPY to the API Invoker.
4. The API Invoker performs the service API invocation by presenting the TokenSNAAPPY.
5. If the API Invoker requested a service API that needs resource owner's authorization, API exposing function shall check whether the API Invoker presented a TokenSNAAPPY. If the API Invoker performed the service API invocation without TokenSNAAPPY in step 8, the API exposing function shall reject the request. If the verification of the TokenSNAAPPY is successful, API exposing function identifies the UE using the S-KID which is included in the TokenSNAAPPY, by communicating with the Authorization Function that stored the S-KID and SUPI.
6. API Invoker receives the service API invocation response from the API exposing function.
7. If the resource owner does not want for the API Invoker to invoke the service API, the resource owner can revoke the TokenSNAAPPY at anytime even before the validity time of the TokenSNAAPPY.
8. The UE requests the API Invoker to revoke the TokenSNAAPPY for the service API.
9. The API Invoker requests the Authorization Function to revoke the TokenSNAAPPY for the service API as specified in RFC 7009 [7].
10. If the Authorization Function receives a revocation request for TokenSNAAPPY,
11. The Authorization Function responds to the revocation request. The response includes revocation time and the TokenSNAAPPY with a hash signature which the Authorization Function generates using the TokenSNAAPPY, revocation time, and KSNAAPPY; and
12. The Authorization Function notifies the API exposing function of the revocation of the TokenSNAAPPY, with the revocation time. After the API exposing function receives the revocation notification of the TokenSNAAPPY, the API exposing function shall reject the API invocation from the API Invoker if the API Invoker invokes the service API with TokenSNAAPPY of which the generated time is prior to the revocation time.
13. The UE verifies the hash signature using KSNAAPPY. The UE may inform the resource owner of the revocation result based on the verification.

### 6.X.2.3 S-KID

S-KID is in NAI format as specified in clause 2.2 of IETF RFC 7542 [xx], i.e. username@realm. The username part includes SNAAPPY Temporary UE Identifier (S-TID), and the realm part includes Home Network Identifier or Authorization Function Address.

When deriving S-TID from KAUSF, the following parameters shall be used to form the input S to the KDF:

* FC = 0xXX;
* P0 = "S-TID";
* L0 = length of "S-TID";
* P1 = SUPI;
* L1 = length of SUPI.

The input key KEY shall be KAUSF.

NOTE: FC value to be determined during normative phase.

### 6.X.2.4 KSNAAPPY derivation function

When deriving KSNAAPPY from KAUSF, the following parameters shall be used to form the input S to the KDF:

* FC = 0xYY;
* P0 = “Authorization”;
* L0 = length of “Authorization”;

The input key KEY shall be the KAUSF.

NOTE: FC value to be determined during normative phase.

### 6.X.3 Evaluation

TBD

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\* End of 2nd Change \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*