**3GPP TSG-SA3 Meeting #110Ad-Hoc-e *draft\_S3-231910-r2***

**Electronic meeting, Online, 17 - 21 April 2023** (revision of S3-yyxxxx)

**Source: Huawei, HiSilicon**

**Title: Update on solution #1**

**Document for: Approval**

**Agenda Item: 5.11**

# 1 Decision/action requested

***Approve the proposed update on solution#1 in TR 33.884 [1].***

# 2 References

[1] 3GPP TR 33.884 “Study on security of application enablement aspects for subscriber-aware northbound API access”

# 3 Rationale

The solution proposes to address the following ENs in solution #1.

For EN 1 “Details of content and verification of token is ffs”, it is proposed to add more details in step 5 in clause 6.1.2.2 and clause 6.1.2.4.

For EN 2 “Mapping to SA6 defined use case is ffs”, since the text is already depicted in clause 6.1.2.3, it is proposed to add a NOTE for reference.

For EN 3 “Further evaluation is FFS”, since there is no more evaluation needed, it is proposed to delete it.

# 4 Detailed proposal

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

## 6.1 Solution #1: Resource Owner Authorization in API Invocation using OAuth Token

### 6.1.1 Introduction

This solution addresses the requirement in KI#2.

This solution proposes to reuse OAuth 2.0 with authorization code grant model to obtain resource owner’s authorization in case that an API invocation of network exposure is to process the resource owner’s data.

NOTE1: how the API invoker is authenticated to the authorization server is out of scope of this solution.

NOTE2: how the Resource owner is authenticated to the authorization server is out of scope of this solution.

### 6.1.2 Solution details

### 6.1.2.1 Architecture



Figure 6.1.2.1-1 architecture for CAPIF with SNA enhancement

The architecture is derived from solution 2 in TR 23.700-95 [3]. Definition of the authorization function, CAPIF-8, CAPIF-10, and CAPIF-10e is the same. Triggerer has the same definition of resource owner client(s).

It is proposed that the authorization function is collocated with the CCF.

However, the difference is that CAPIF-9 is not needed in this solution because there is no communication with AEF. CAPIF-8 is application layer, which is out of 3GPP

### 6.1.2.2 Procedure



Figure 6.1.2.2-1 Procedure of Obtaining Resource owner Authorization

As shown in the Figure 6.1.2.2-1, the details of obtaining resource owner authorization in API invocation is summarized as following:

1. API invoker obtains authentication and authorization method (e.g. method 1: TLS-PSK, or method 2: PKI, or method 3: TLS with OAuth token) as specified in clause 6.1 in TS 33.122 [5].

NOTE 3: CCF needs to pre-configure with TLS with OAuth token method.

NOTE 4: Onboarding procedure is reused.

2. API invoker discovers service API as specified in clause 6.3.1.3 in TS 33.122 [5].

3. Resource owner triggers the API invocation. If resource owner authorization is needed for the invoked API and the method 3: TLS with OAuth token is selected, then the API invoker obtains tokenCAPIF via OAuth 2.0 with authorization code grant model as depicted in clause 6.1.2.3.

4. The API invoker invokes northbound API to the AEF as depicted in clause 6.5.2.1 or 6.5.2.2 in TS 33.122 [5]. The API Invocation message includes tokenCAPIF.

The API invoker is pre-configured with a certificate and use TLS to authenticate with AEF.

5. The AEF verifies the tokenCAPIF in the message, and the profile of this token is depicted in clause 6.1.2.4. The AEF verifies the integrity of tokenCAPIF and shall check whether IDs in the scope are align with the IDs in the API invocation message. If the verification is successful, it means the resource owner has authorized the API Invoker to access the API for the resource owned by the resource owner.

### 6.1.2.3 OAuth 2.0 role mapping

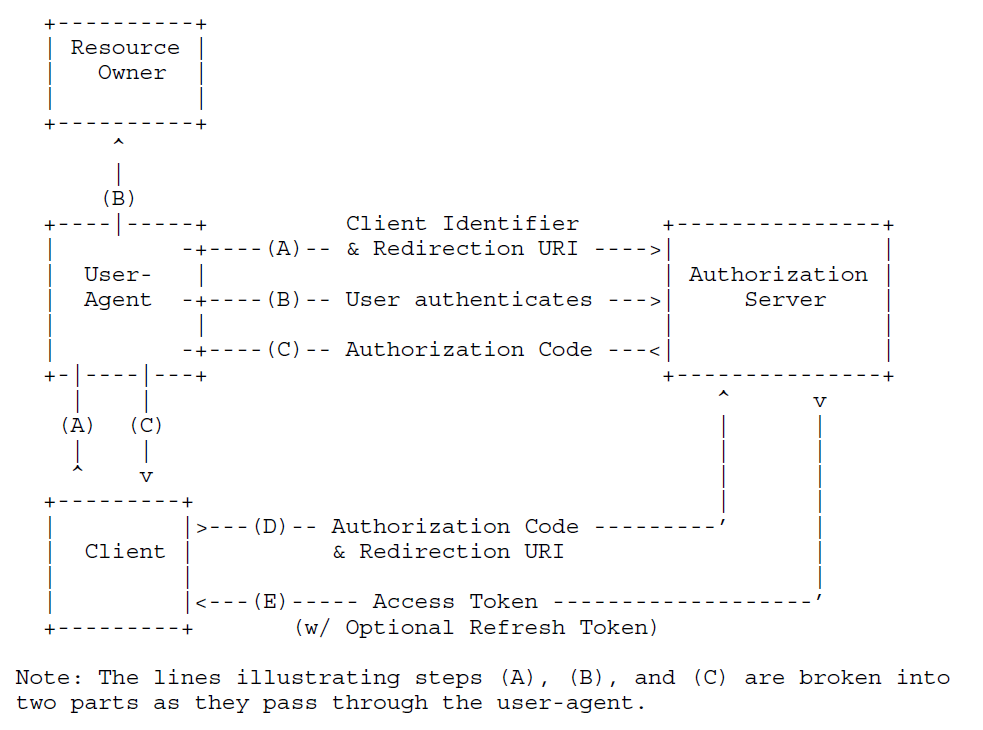


Figure 6.1.2.3-1 Procedure of Obtaining Resource owner Authorization (from RFC 6749 [4])

OAuth 2.0 with authorization code grant model is depicted in clasue 4.1 in RFC 6749 [4]. In this solution, the API invoker endorses the role of client, the triggerer endorses the role of user-agent. The Authorization Function authenticates the resource owner in step B, which depends on existing mechanism and is out of this solution. The API invoker (i.e. client) retrieves tokenCAPIF from the Authorization function in CCF. The scope of the tokenCAPIF includes API invoker ID, service API ID and resource owner ID (i.e. GPSI).

Authorization can be revoked according to mechanism defined in IETF RFC 7009 [7].

The resource owner ID is equal to the UE ID in the API invocation message, e.g. GPSI.

It takes use case 1 defined in Annex A.1 in TR 23.700-95 [3] as an example. An end user (i.e. **resource owner**) is playing a time-sensitive game using a game client application (i.e. **triggerer**) on the end user’s UE communicating with a game server (i.e. **API invoker**), and wants to have a high-quality and low-latency communication for better service experience.

In step A, the end user requests the game client application, and the game client application requests game server to try to invoke the QoS API via **application layer**. The game server discoveries QoS API, and initiates OAuth procedure by contacting the game client application, and the game client application sends the requests to authorization server via **CAPIF-8**.

In step B, the Authorization Server requests the game client application to do user authentication and authorization via **CAIPF-8**, the end user may type in his MNO username and password and click “consent” for extra charging for QoS enhancement.

NOTE: It is assumed that authorization server has linkage between resource owner and GPSI.

Editor’s Note: How Authorization Function maps username to ID of the UE that the user is using when the user has multiple subscriptions is FFS.

In step C, after successful authentication and authorization, the authorization server provides authorization code to the game client application via **CAPIF-8**, and the game client application sends the authorization code to the game server via **application layer**.

In step D, the game server sends authorization code to authorization server via **CAPIF-10/CAPIF-10e**.

In step E, the game server gets tokenCAPIF from authorization server. The game server uses the tokenCAPIF to invoke QoS API via **CAPIF-2/CAPIF-2e** to modify the end user’s QoS.

Editor’s Note: More clarification of Integration of out scope and in scope messages is FFS.

6.1.2.4 TokenCAPIF Profile

The tokenCAPIF is protected by the JSON signature profile as specified in IETF RFC 7515 [8].

The claim in the tokenCAPIF includes the parameters defined in C.2.2 in TS 33.122 [5]. In addition, the “resource owner ID” and the resource ID related to the list services are also added in scope. The resource owner ID is equal to the UE ID in the API invocation message, e.g. GPSI.

### 6.1.3 Evaluation

The solution works when the TLS with OAuth token is selected.

This solution assumes that the resource owner is human user.

This solution addresses the requirements Authz-1, Authz-2, Authz-3 and Authz-4, but does not address the following requirements in KI#2:

This solution doesn’t touch authentication of the resource owner or API invoker by the authorization server, it may be addressed by other solution.

This solution doesn’t touch authentication between API invoker and CCF and authentication between API invoker and AEF, it may be addressed by other solution.

This solution doesn’t touch revocation of authorization, it may be addressed by other solution.

This solution doesn’t touch privacy of the UE's external identity against the third-party, it may be addressed by other solution. This solution doesn’t address a static token claim issue after token revocation.

NOTE: Clause 6.1.2.3 has provided mapping to the use case defined in TR 23.700-95 [3].

Editor’s Note: Further evaluation is FFS

Editor’s Note: Evaluation on sharing user credential to application client is FFS.

Editor’s Note: Evaluation on user authentication is FFS.

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