Workgroup: SIP Core
Internet-Draft:
draft-ietf-sipcore-sip-token-authnz-13
Updates: 3261 (if approved)
Published: 15 April 2020
Intended Status: Standards Track
Expires: 17 October 2020
Authors: R. Shekh-Yusef   C. Holmberg   V. Pascual
Avaya   Ericsson   webrtchacks

# Third-Party Token-based Authentication and Authorization for Session Initiation Protocol (SIP)

## Abstract

This document defines the "Bearer" authentication scheme for the Session Initiation Protocol (SIP), and a mechanism by which user authentication and SIP registration authorization is delegated to a third party, using the OAuth 2.0 framework and OpenID Connect Core 1.0. This document updates RFC 3261 to provide guidance on how a SIP User Agent Client (UAC) responds to a SIP 401/407 response that contains multiple WWW-Authenticate/Proxy-Authenticate header fields.

## Status of This Memo



## Copyright Notice

**Table of Contents**

## 1. Introduction

The Session Initiation Protocol (SIP) [RFC3261] uses the same framework as HTTP [RFC7230] to authenticate users: a simple challenge-response authentication mechanism that allows a SIP server to challenge a SIP client request and allows a SIP client to provide authentication information in response to that challenge.

OAuth 2.0 [RFC6749] defines a token-based authorization framework to allow an OAuth client to access resources on behalf of its user.

The OpenID Connect 1.0 specification [OPENID] defines a simple identity layer on top of the OAuth 2.0 protocol, which enables clients to verify the identity of the user based on the authentication performed by a dedicated authorization server, as well as to obtain basic profile information about the user.

This document defines the "Bearer" authentication scheme for the Session Initiation Protocol (SIP), and a mechanism by which user authentication and SIP registration authorization is delegated to a third party, using the OAuth 2.0 framework and OpenID Connect Core 1.0. This kind of user authentication enables single sign-on, which allows the user to authenticate once and gain access to both SIP and non-SIP services.

This document also updates [RFC3261], by defining the User Agent Client (UAC) procedures when a UAC receives a 401/407 response with multiple WWW-Authenticate/Proxy-Authenticate header fields, providing challenges using different authentication schemes for the same realm.

### 1.1. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

### 1.2. SIP User Agent Types

The OAuth 2.0 authorization framework [RFC6749] defines two types of clients, confidential and public, that apply to the SIP UACs.

* Confidential User Agent: a SIP UAC that is capable of maintaining the confidentiality of the user credentials and any tokens obtained using these user credentials.

* Public User Agent: a SIP UAC that is incapable of maintaining the confidentiality of the user credentials and any obtained tokens.

The mechanism defined in this document MUST only be used with Confidential User Agents, as the UAC is expected to obtain and maintain tokens to be able to access the SIP network.

### 1.3. Token Types and Formats

The tokens used in third-party authorization depend on the type of authorization server (AS).

An OAuth authorization server provides the following tokens to a successfully authorized UAC:

* Access token: the UAC will use this token to gain access to services by providing the token to a SIP server.

* Refresh token: the UAC will present this token to the AS to refresh a stale access token.

An OpenID Connect server returns an additional token:

* ID Token: this token contains the SIP URI and other user-specific details that will be consumed by the UAC.

Tokens can be represented in two different formats:

* Structured Token: a token that consists of a structured object that contains the claims associated with the token, e.g. JWT as defined in [RFC7519].

* Reference Token: a token that consists of a random string that is used to obtain the details of the token and its associated claims, as defined in [RFC6749].

Access Tokens could be represented in one of the above two formats. Refresh Tokens usualy are represented in a reference format, as this token is consumed only the AS that issued the token. ID Token is defined as a structured token in the form of a JWT.

### 1.4. Example Flows

#### 1.4.1. Registration

Figure 1 below shows an example of a SIP registration, where the registrar informs the UAC about the authorization server from which the UAC can obtain an access token.

```
  UAC                         Registrar                           AS
---------------------------------------------------------------------
   |                               |                               |
   | [1] REGISTER                  |                               |
   |------------------------------>|                               |
   |                               |                               |
   | [2] 401 Unauthorized          |                               |
   |     WWW-Authenticate: Bearer "authz_server"="<authz_server>"  |
   |<------------------------------|                               |
   |                               |                               |
   | [3] The UAC interacts with the AS and obtains tokens, using   |
   |     some out-of-scope mechanism.                              |
   |<=============================================================>|
   |                               |                               |
   | [4] REGISTER                  |                               |
   |     Authorization: Bearer <access_token>                      |
   |------------------------------>|                               |
   |                               | [5] HTTP POST /introspect     |
   |                               |     {access_token}            |
   |                               |------------------------------>|
   |                               |                               |
   |                               | [6] 200 OK {metadata}         |
   |                               |<------------------------------|
   |                               |                               |
   | [7] 200 OK                    |                               |
   |<------------------------------|                               |
   |                               |                               |
```

Figure 1: Example Registration Flow

In step [1], the UAC starts the registration process by sending a SIP REGISTER request to the registrar without any credentials.

In step [2], the registrar challenges the UA, by sending a SIP 401 (Unauthorized) response to the REGISTER request. In the response, the registrar includes information about the AS to contact in order to obtain a token.

In step [3], the UAC interacts with the AS via an out-of-scope mechanism, potentially using the OAuth Native App mechanism defined in [RFC8252]. The AS authenticates the user and provides the UAC with the tokens needed to access the SIP service.

In step [4], the UAC retries the registration process by sending a new REGISTER request that includes the access token that the UAC obtained in the step above.

The registrar validates the access token. If the access token is a reference token, the registrar MAY perform an introspection [RFC7662], as in steps [5] and [6], in order to obtain more information about the access token and its scope, per [RFC7662]. Otherwise, after the registrar validates the token to make sure it was signed by a trusted entity, it inspects its claims and acts upon it.

In step [7], once the registrar has successfully verified and accepted the access token, it sends a 200 (OK) response to the REGISTER request.

### 1.4.2. Registration with Preconfigured AS

Figure 2 shows an example of a SIP registration where the UAC has been preconfigured with information about the AS from which to obtain the access token.

```
  UAC                         Registrar                          AS
---------------------------------------------------------------------
  |                               |                               |
  | [1] The UAC interacts with the AS and obtains tokens, using   |
  |     some out of scope mechanism.                              |
  |<=============================================================>|
  |                               |                               |
  | [2] REGISTER                  |                               |
  |     Authorization: Bearer <access_token>                      |
  |------------------------------>|                               |
  |                               | [3] HTTP POST /introspect     |
  |                               |     {access_token}            |
  |                               |------------------------------>|
  |                               |                               |
  |                               | [4] 200 OK {metadata}         |
  |                               |<------------------------------|
  |                               |                               |
  | [5] 200 OK                    |                               |
  |<------------------------------|                               |
  |                               |                               |
```

Figure 2: Example Registration Flow - Authorization Server Information Preconfigured

In step [1], the UAC interacts with the AS using an out-of-scope mechanism, potentially using the OAuth Native App mechanism defined in [RFC8252]. The AS authenticates the user and provides the UAC with the tokens needed to access the SIP service.

In step [2], the UAC initiates the registration process by sending a new REGISTER request that includes the access token that the UAC obtained in the step above.

The registrar validates the access token. If the access token is a reference token, the registrar MAY perform an introspection, as in steps [3] and [4], in order to obtain more information about the access token and its scope, per [RFC7662]. Otherwise, after the registrar validates the token to make sure it was signed by a trusted entity, it inspects its claims and acts upon it.

In step [5], once the registrar has successfully verified and accepted the access token, it sends a 200 (OK) response to the REGISTER request.

## 2. SIP Procedures

Section 22 of [RFC3261] defines the SIP procedures for the Digest authentication mechanism. The same procedures apply to the Bearer authentication mechanism, with the changes described in this section.

## 2.1. UAC Behavior

### 2.1.1. Obtaining Tokens and Responding to Challenges

When a UAC sends a request without credentials (or with invalid credentials), it could receive either a 401 (Unauthorized) response with a WWW-Authenticate header field or a 407 (Proxy Authentication Required) response with a Proxy-Authenticate header field. If the WWW-Authenticate or Proxy-Authenticate header field indicates "Bearer" scheme authentication and contains an address to an authorization server, the UAC contacts the authorization server in order to obtain tokens, and includes the requested scopes, based on a local configuration (Figure 1).

The detailed OAuth2 procedure to authenticate the user and obtain these tokens is out of scope of this document. The address of the authorization server might already be known to the UAC via configuration. In which case, the UAC can contact the authorization server for tokens before it sends a SIP request (Figure 2). Procedures for native applications are defined in [RFC8252]. When using the mechanism defined in [RFC8252] the user of the UAC will be directed to interact with the authorization server using a web browser, allowing the authorization server to prompt the user for multi-factor authentication, to redirect the user to third-party identity providers, and to enable the use of single sign-on sessions.

The tokens returned to the UAC depend on the type of authorization server (AS): an OAuth AS provides an access token and refresh token [RFC6749]. The UAC provides the access token to the SIP servers to authorize UAC's access to the service. The UAC uses the refresh token only with the AS to get a new access token and refresh token before the expiry of the current access token (see [RFC6749], section 1.5 Refresh Token for details). An OpenID Connect server returns an additional ID-Token containing the SIP URI and other user-specific details that will be consumed by the UAC.

If the UAC receives a 401/407 response with multiple WWW-Authenticate/Proxy-Authenticate header fields, providing challenges using different authentication schemes for the same realm, the UAC selects one or more of the provided schemes (based on local policy) and provides credentials for those schemes.

NOTE: The address of the Authorization Server might be known to the UAC e.g., using means of configuration, in which case the UAC can contact the Authorization Server in order to obtain the access token before it sends SIP request without credentials.

#### 2.1.2. Protecting the Access Token

[RFC6749] mandates that access tokens are protected with TLS when in transit. However, TLS only guarantees hop-to-hop protection when used to protect SIP signaling. Therefore the access token MUST be protected in a way so that only authorized SIP servers will have access to it. Endpoints that support this specification MUST support encrypted JSON Web Tokens (JWT) [RFC7519] for encoding and protecting access tokens when they are included in SIP requests, unless some other mechanism is used to guarantee that only authorized SIP endpoints have access to the access token.

#### 2.1.3. REGISTER Request

The procedures in this section apply when the UAC has received a challenge that contains a "Bearer" scheme, and the UAC has obtained a token as specified in Section 2.1.1.

The UAC sends a REGISTER request with an Authorization header field containing the response to the challenge, including the Bearer scheme carrying a valid access token in the request, as specified in [RFC6750].

Note that, if there were multiple challenges with different schemes, then the UAC may be able to successfully retry the request using non-Bearer credentials.

Based on local policy, the UAC MAY include an access token that has been used for another binding associated with the same Address Of Record (AOR) in the request.

If the access token included in a REGISTER request is not accepted, and the UAC receives a 401 response or a 407 response, the UAC follows the procedures in Section 2.1.1.

#### 2.1.4. Non-REGISTER Request

The procedures in this section apply when the UAC has received a challenge that contains a "Bearer" scheme, and the UAC has obtained a token as specified in Section 2.1.1.

When the UAC sends a request, it MUST include an Authorization header field with a Bearer scheme, carrying a valid access token in the request, as specified in [RFC6750]. Based on local policy, the UAC MAY include an access token that has been used for another dialog, or for another stand-alone request, if the target of the new request is the same.

If the access token included in a request is not accepted, and the UAC receives a 401 response or a 407 response, the UAC follows the procedures in Section 2.1.1.

### 2.2. UAS and Registrar Behavior

When a UAS or Registrar receives a request that fails to contain authorization credentials acceptable to it, it SHOULD challenge the request by sending a 401 (Unauthorized) response. To indicate that it is willing to accept an access token as a credential, the UAS/Registrar MUST include a Proxy-Authentication header field in the response that indicates "Bearer" scheme and includes an address of an authorization server from which the originator can obtain an access token.

When a UAS/Registrar receives a SIP request that contains an Authorization header field with an access token, the UAS/Registrar MUST validate the access token, using the procedures associated with the type of access token (Structured or Reference) used, e.g. [RFC7519]. If the token provided is an expired access token, then the UAS MUST reply with 401 Unauthorized, as defined in section 3 of [RFC6750]. If the validation is successful, the UAS/Registrar can continue to process the request using normal SIP procedures. If the validation fails, the UAS/Registrar MUST reject the request.

### 2.3. Proxy Behavior

When a proxy receives a request that fails to contain authorization credentials acceptable to it, it SHOULD challenge the request by sending a 407 (Proxy Authentication Required) response. To indicate that it is willing to accept an access token as a credential, the proxy MUST include a Proxy-Authentication header field in the response that indicates "Bearer" scheme and includes an address to an authorization server from which the originator can obtain an access token.

When a proxy wishes to authenticate a received request, it MUST search the request for Proxy-Authorization header fields with 'realm' parameters that match its realm. It then MUST successfully validate the credentials from at least one Proxy-Authorization header field for its realm. When the scheme is "Bearer", the proxy

MUST validate the access token, using the procedures associated with the type of access token (Structured or Reference) used, e.g., [RFC7519].

## 3. Access Token Claims

The type of services to which an access token grants access can be determined using different methods. The methods used and the access provided by the token is based on local policy agreed between the AS and the registrar.

If an access token is encoded as a JWT, it might contain a list of claims [RFC7519], some registered and some application-specific. The REGISTRAR can grant access to services based on such claims, some other mechanism, or a combination of claims and some other mechanism. If an access token is a reference token, the REGISTRAR will grant access based on some other mechanism. Examples of such other mechanisms are introspection [RFC7662], user profile lookups, etc.

## 4. WWW-Authenticate Response Header Field

This section uses ABNF [RFC5234] to describe the syntax of the WWW-Authenticate header field when used with the "Bearer" scheme to challenge the UAC for credentials, by extending the 'challenge' parameter defined by [RFC3261].

```
challenge  =/  ("Bearer" LWS bearer-cln *(COMMA bearer-cln))
bearer-cln = realm / scope / authz-server / error /
             auth-param
authz-server = "authz_server" EQUAL authz-server-value
authz-server-value = https-URI
realm = <defined in RFC3261>
auth-param = <defined in RFC3261>
scope = <defined in RFC6749>
error = <defined in RFC6749>
https-URI = <defined in RFC7230>
```

Figure 3: Bearer Scheme Syntax

The authz-server parameter contains the HTTPS URI, as defined in [RFC7230], of the authorization server. The UAC can discover metadata about the AS using a mechanism like the one defined in [RFC8414].

The realm and auth-param parameters are defined in [RFC3261].

Per [RFC3261], the realm string alone defines the protection domain. [RFC3261] states that the realm string must be globally unique and recommends that the realm string contain a hostname or domain name. It also states that the realm string should be a human-readable identifier that can be rendered to the user.

The scope and error parameters are defined in [RFC6749].

The scope parameter could be used by the registrar/proxy to indicate to the UAC the minimum scope that must be associated with the access token to be able to get service. As defined in [RFC6749], the value of the scope parameter is expressed as a list of space-delimited, case-sensitive strings. The strings are defined by the authorization server. The values of the scope parameter are out of scope of this document. The UAC will use the scope provided by the registrar to contact the AS and obtain a proper token with the requested scope.

The error parameter could be used by the registrar/proxy to indicate to the UAC the reason for the error, with possible values of "invalid_token" or "invalid_scope".

## 5. Security Considerations

The security considerations for OAuth are defined in [RFC6749]. The security considerations for bearer tokens are defined in [RFC6750]. The security considerations for JSON Web Tokens (JWT) are defined in [RFC7519]. These security considerations also apply to SIP usage of access token as defined in this document.

[RFC6749] mandates that access tokens are protected with TLS. However, TLS only guarantees hop-to-hop protection when used to protect SIP signaling. Therefore the access token MUST be protected in a way so that only authorized SIP endpoints will have access to it. Endpoints that support this specification MUST support encrypted JSON Web Tokens (JWT) [RFC7519] for encoding and protecting access tokens when included in SIP requests, unless some other mechanism is used to guarantee that only authorized SIP endpoints have access to the access token.

## 6. IANA Considerations

## 7. Acknowledgments

The authors would like to specially thank Paul Kyzivat for his multiple detailed reviews and suggested text that significantly improved the quality of the document.

The authors would also like to thank the following for their review and feedback on this document:

Olle Johansson, Roman Shpount, Dale Worley, and Jorgen Axell.

The authors would also like to thank the following for their review and feedback of the original document that was replaced with this document:

Andrew Allen, Martin Dolly, Keith Drage, Paul Kyzivat, Jon Peterson, Michael Procter, Roy Radhika, Matt Ryan, Ivo Sedlacek, Roman Shpount, Robert Sparks, Asveren Tolga, Dale Worley, and Yehoshua Gev.

The authors would also like to specially thank Jean Mahoney for her multiple reviews, editorial help, and the coversion of the XML source file from v2 to v3.

## 8. Normative References

**[OPENID]** Sakimura, N., Bradley, J., Jones, M., de Medeiros, B., and C. Mortimore, "OpenID Connect Core 1.0", February 2014.

**[RFC2119]** Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

**[RFC3261]** Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, DOI 10.17487/RFC3261, June 2002, <https://www.rfc-editor.org/info/rfc3261>.

**[RFC5234]**
Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/info/rfc5234>.

**[RFC6749]** Hardt, D., Ed., "The OAuth 2.0 Authorization Framework", RFC 6749, DOI 10.17487/RFC6749, October 2012, <https://www.rfc-editor.org/info/rfc6749>.

**[RFC6750]** Jones, M. and D. Hardt, "The OAuth 2.0 Authorization Framework: Bearer Token Usage", RFC 6750, DOI 10.17487/RFC6750, October 2012, <https://www.rfc-editor.org/info/rfc6750>.

**[RFC7230]** Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/info/rfc7230>.

**[RFC7519]** Jones, M., Bradley, J., and N. Sakimura, "JSON Web Token (JWT)", RFC 7519, DOI 10.17487/RFC7519, May 2015, <https://www.rfc-editor.org/info/rfc7519>.

**[RFC7662]** Richer, J., Ed., "OAuth 2.0 Token Introspection", RFC 7662, DOI 10.17487/RFC7662, October 2015, <https://www.rfc-editor.org/info/rfc7662>.

**[RFC8174]** Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

**[RFC8252]** Denniss, W. and J. Bradley, "OAuth 2.0 for Native Apps", BCP 212, RFC 8252, DOI 10.17487/RFC8252, October 2017, <https://www.rfc-editor.org/info/rfc8252>.

**[RFC8414]** Jones, M., Sakimura, N., and J. Bradley, "OAuth 2.0 Authorization Server Metadata", RFC 8414, DOI 10.17487/RFC8414, June 2018, <https://www.rfc-editor.org/info/rfc8414>.

## Authors' Addresses

Rifaat Shekh-Yusef
Avaya
425 Legget Drive
Ottawa Ontario
Canada

Phone: +1-613-595-9106

Email: rifaat.ietf@gmail.com

Christer Holmberg
Ericsson
Hirsalantie 11
FI- Jorvas 02420
Finland

Email: christer.holmberg@ericsson.com

Victor Pascual
webrtchacks
Spain

Email: victor.pascual.avila@gmail.com