Workgroup: SIP Core
Internet-Draft:
draft-ietf-sipcore-sip-token-authnz-17
Updates: 3261 (if approved)
Published: 5 May 2020
Intended Status: Standards Track
Expires: 6 November 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

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at https://datatracker.ietf.org/drafts/current/.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 6 November 2020.

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 User Agent Server (UAS), proxy or registrar to challenge a SIP User Agent Client (UAC) request and allows the UAC 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 OAuth/OpenID clients to verify the identity of the user based on the authentication performed by a dedicated authorization server (AS), referred to as OpenID Provider (OP), 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 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", "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. Applicability

This document covers cases where grants that allow the UAC to obtain an access token from the AS are used. Cases where the UAC is not able to obtain an access token (e.g., in the case of an authorization code grant) are not covered.

1.3. Token Types and Formats

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

An OAuth AS 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 OP returns an additional token:

- ID Token: this token contains a SIP URI associated with the user 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., JSON Web Token (JWT) as defined in [RFC7519].

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

Access Tokens are represented in one of the above two formats. Refresh Tokens usually 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 AS from which the UAC can obtain an access token.

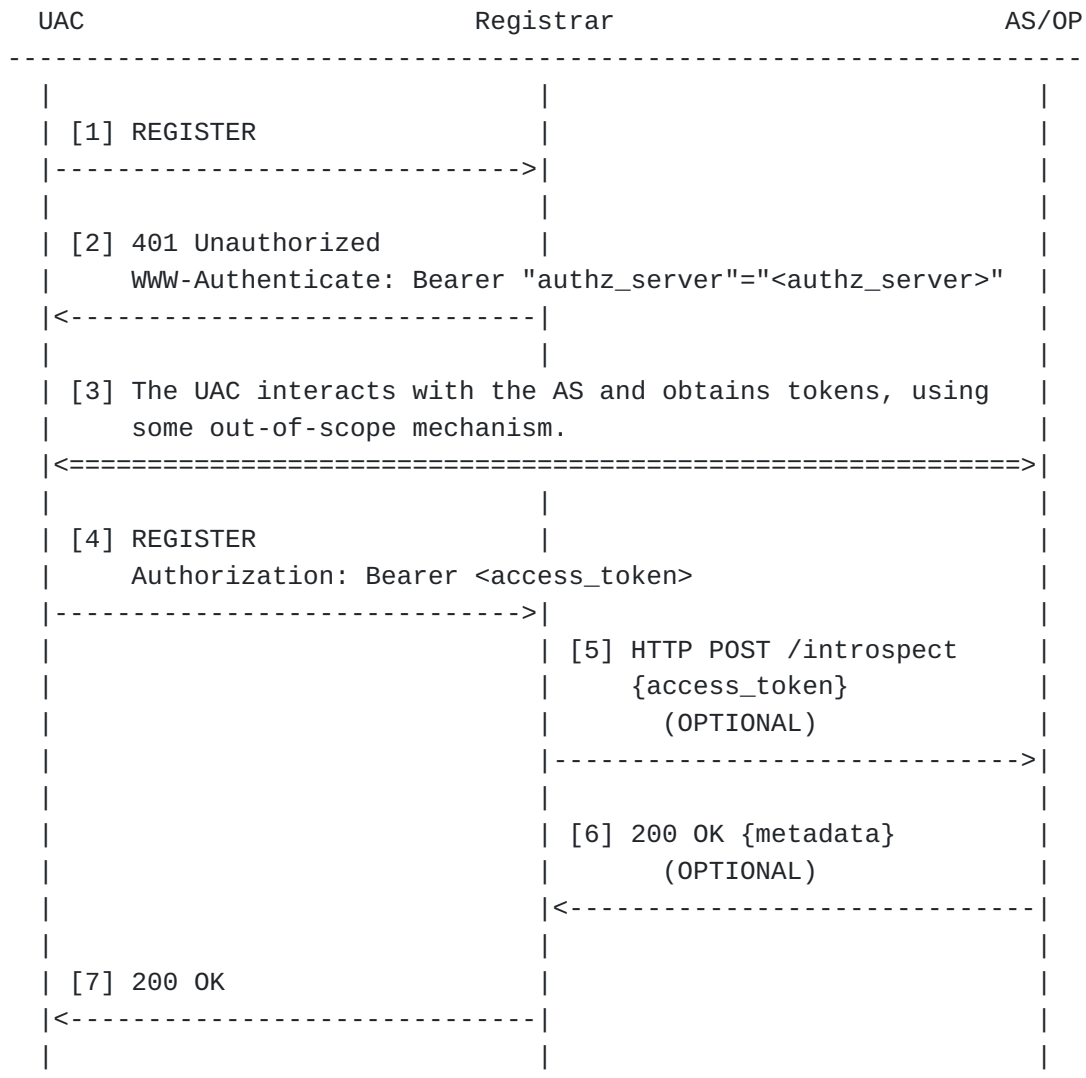

Figure 1: Example Registration Flow

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, 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.

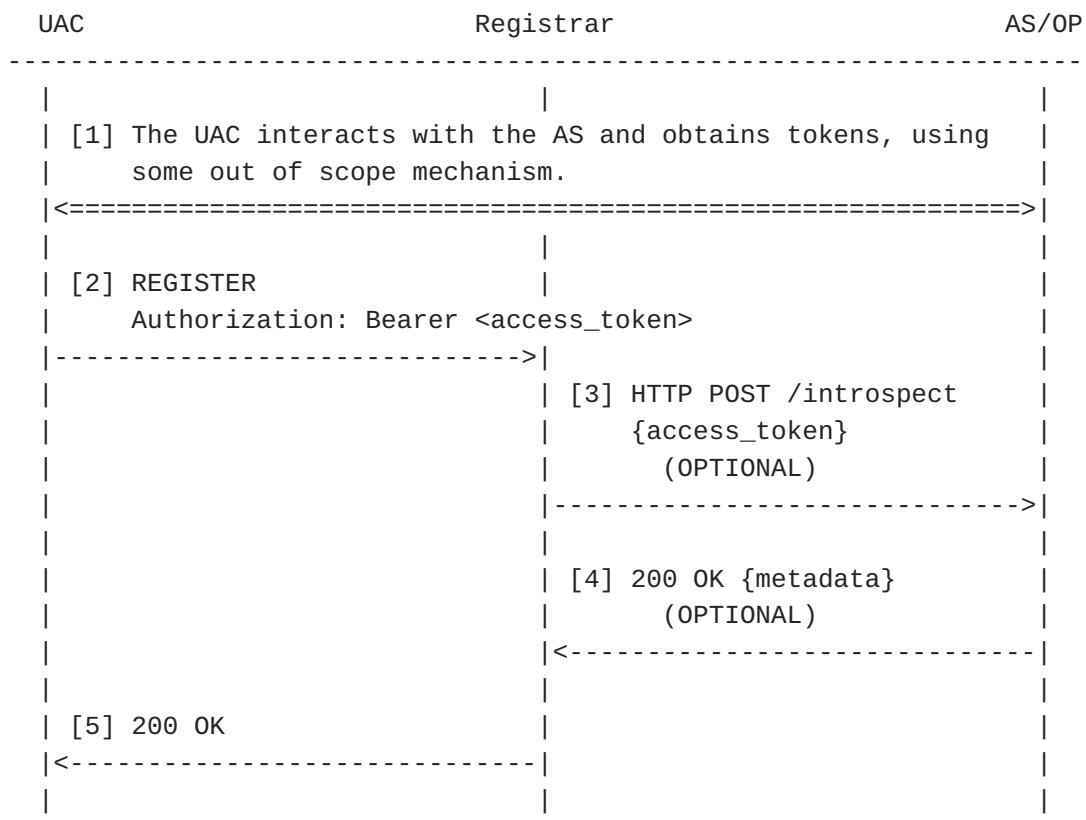

Figure 2: Example Registration Flow - AS Information Preconfigured

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 [RFC7662], as in steps [4] and [5], in order to obtain more information about the access token and its scope, per [RFC7662]. Otherwise, after the registrar validates the token, 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 AS, the UAC contacts the AS in order to obtain tokens, and includes the requested scopes, based on a local configuration (Figure 1). The UAC MUST check the AS URL received in the 401/407 response against a list of trusted ASs configured on the UAC, in order to prevent several classes of possible vulnerabilities when a client blindly attempts to use any provided AS.

The detailed OAuth2 procedure to authenticate the user and obtain these tokens is out of scope of this document. The address of the AS might already be known to the UAC via configuration. In such cases, the UAC can contact the AS 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 AS using a web browser, allowing the AS 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 AS: an OAuth AS provides an access token and optionally a refresh token [RFC6749]. The refresh token is only used between the UAC and the AS. If the AS provides a refresh token to the UAC, the UAC uses it to request a new access token from the AS before the currently used access token expires ([RFC6749], Section 1.5). If the AS does not provide a refresh token, the UAC needs to re-authenticate the user, in order to get a new access token, before the currently used access token expires. An OP returns an additional ID Token that contains claims about the authentication of the user by an authorization server. The ID Token can potentially include other optional claims about the user, e.g. the SIP URI, that will be consumed by the UAC and later used to register with the registrar.

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 provides credentials for one of the schemes that it supports, based on local policy.

NOTE: At the time of writing this document, detailed procedures for the cases where a UAC receives multiple different authentication schemes had not been defined. A future specification might define such procedures.

NOTE: The address of the AS might be known to the UAC e.g., using means of configuration, in which case the UAC can contact the AS 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, SIP makes use of intermediary SIP proxies, and 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. SIP endpoints that support this document MUST use 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. TLS can still be used for protecting traffic between SIP endpoints and the AS, as defined in [RFC6749].

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.

Typically, a UAC will obtain a new access token for each new binding, However, based on local policy, a 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 obtained from the AS indicated in the challenge, 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. User Agent Server (UAS) and Registrar Behavior

When a UAS or registrar receives a request that fails to contain authorization credentials acceptable to it, the UAS/registrar SHOULD challenge the request by sending a 401 (Unauthorized) response. If the UAS/registrar chooses to challenge the request, and is willing to accept an access token as a credential, it MUST include a WWW-Authenticate header field in the response that indicates "Bearer" scheme and includes an AS address, encoded as an https URI [RFC7230], from which the UAC can obtain an access token.

When a UAS or 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/registrar MUST reply with a 401 (Unauthorized) response, 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 reply with 401 (Unauthorized) response.

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. If the proxy chooses to challenge the request, and is willing to accept an access token as a credential, it MUST include a Proxy-Authenticate header field in the response that indicates "Bearer" scheme and includes an AS address, encoded as an https URI [RFC7230], from which the UAC 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 are based on local policy agreed between the AS and the registrar.

If an access token is encoded as a JWT, it will contain a list of claims [RFC7519], including both registered and application-specific claims. 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] and user profile lookups.

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-param / authz-server-param / error-param /
             auth-param
realm = <defined in RFC3261>
scope-param = "scope" EQUAL DQUOTE scope DQUTE
scope = <defined in RFC6749>
authz-server-param = "authz_server" EQUAL DQUOTE authz-server DQUOTE
authz-server = https-URI
https-URI = <defined in RFC7230>
error-param = "error" EQUAL DQUOTE error DQUOTE
error = <defined in RFC6749>
auth-param = <defined in RFC3261>
```

Figure 3: Bearer Scheme Syntax

The authz_server parameter contains the HTTPS URI, as defined in [RFC7230], of the AS. 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 can 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 AS. 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 when in transit. However, SIP makes have use of intermediary SIP proxies, and 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. SIP endpoints that support this document MUST use 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. TLS can still be used for protecting traffic between SIP endpoints and the AS, as defined in [RFC6749].

Single Sign-On (SSO) enables the user to use one set of credentials to authenticate once and gain access to multiple SIP and non-SIP services using access token(s). If the SSO login is compromised, that single point of compromise has a much broader effect than is the case without SSO. Further, an attacker can often use a compromised account to set up Single Sign-On for other services that the victim has not established an account with, and sometimes can even switch a dedicated account into Single-Sign-On mode, creating a still broader attack.

Because of that, it is critical to make sure that extra security measures be taken to safeguard credentials used for Single Sign-On. Examples of such measures include long passphrase instead of a password, enabling multi-factor factor authentication, and the use of the native platform browser when possible, as defined in [RFC8252].

Although this is out of scope for this document, it is important to carefully consider the claims provided in the tokens used to access these services to make sure of the privacy of the user accessing these services. As mentioned above, this document calls for encrypting JWT representing the access token.

It is important that both parties participating in SSO provide mechanisms for users to sever the SSO relationship, so that it is possible without undue difficulty to mitigate a compromise that has already happened.

The operator of a Single-Sign-On authentication system has access to private information about sites and services that their users log into, and even, to some extent, about their usage patterns. It's important to call these out in privacy disclosures and policies, and to make sure that users can be aware of the tradeoffs between convenience and privacy when they choose to use SSO.

When a registrar chooses to challenge a REGISTER request, if the registrar can provide access to different levels of services, it is RECOMMENDED that the registrar includes a scope in the response in order to indicate the minimum scope needed to register and access basic services. The access token might include an extended scope that gives the user access to more advanced features beyond basic services. In SIP, the AS administrator will typically decide what level of access is provided for a given user.

The UAC MUST check the AS URL received in the 401/407 response against a list of trusted ASs configured on the UAC, in order to prevent several classes of possible vulnerabilities when a client blindly attempts to use any provided AS.

6. IANA Considerations

6.1. New Proxy-Authenticate header field parameters

This section defines new SIP header field parameters in the "Header Field Parameters and Parameter Values" subregistry of the "Session Initiation Protocol (SIP) Parameters" registry: https://www.iana.org/assignments/sip-parameters

```
Header Field:  Proxy-Authenticate

Parameter Name:  authz_server
Predefined Values:  No
Reference:  RFC XXXX

Parameter Name:  error
Predefined Values:  No
Reference:  RFC XXXX

Parameter Name:  scope
Predefined Values:  No
Reference:  RFC XXXX
```

Figure 4

6.2. New WWW-Authenticate header field parameters

This section defines new SIP header field parameters in the "Header Field Parameters and Parameter Values" subregistry of the "Session Initiation Protocol (SIP) Parameters" registry: https://www.iana.org/assignments/sip-parameters

Header Field: WWW-Authenticate

Parameter Name: authz_server
Predefined Values: No
Reference: RFC XXXX

Parameter Name: error
Predefined Values: No
Reference: RFC XXXX

Parameter Name: scope
Predefined Values: No
Reference: RFC XXXX

Figure 5

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.

Roman Danyliw, Benjamin Kaduk, Erik Kline, Barry Leiba, Eric Vyncke and Magnus Westerlund provided feedback and suggestions for improvements as part of the IESG evaluation of the document. Special thanks to Benjamin Kaduk for his detailed and comprehensive reviews and comments.

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>.

9. Informative References

[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