

SIPCORE
Internet-Draft
Updates: 3263 (if approved)
Intended status: Standards Track
Expires: March 4, 2017

O. Johansson
Edvina AB
G. Salgueiro
Cisco Systems
V. Gurbani
Bell Labs, Nokia Networks
D. Worley, Ed.
Ariadne
August 31, 2016

Locating Session Initiation Protocol (SIP) Servers in a Dual-Stack IP
Network
draft-ietf-sipcore-dns-dual-stack-08

Abstract

RFC 3263 defines how a Session Initiation Protocol (SIP) implementation, given a SIP Uniform Resource Identifier (URI), should locate the next-hop SIP server using Domain Name System (DNS) procedures. As SIP networks increasingly transition from IPv4-only to dual-stack, a quality user experience must be ensured for dual-stack SIP implementations. This document updates the DNS procedures described in RFC 3263 for dual-stack SIP implementations in preparation for forthcoming specifications for applying Happy Eyeballs principles to SIP.

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 <http://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 March 4, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

Table of Contents

1. Introduction	3
2. Terminology	3
3. DNS Procedures in a Dual-Stack Network	4
3.1. Dual-Stack SIP UA DNS Record Lookup Procedure	4
3.2. Indicating Address Family Preference in DNS SRV Records	5
4. Clarification of Interaction with RFC 6724	6
5. Security Considerations	8
6. IANA Considerations	8
7. Acknowledgments	8
8. Revision History	8
8.1. Changes from draft-ietf-sipcore-dns-dual-stack-07 to draft-ietf-sipcore-dns-dual-stack-08	8
8.2. Changes from draft-ietf-sipcore-dns-dual-stack-06 to draft-ietf-sipcore-dns-dual-stack-07	9
8.3. Changes from draft-ietf-sipcore-dns-dual-stack-05 to draft-ietf-sipcore-dns-dual-stack-06	9
8.4. Changes from draft-ietf-sipcore-dns-dual-stack-04 to draft-ietf-sipcore-dns-dual-stack-05	9
8.5. Changes from draft-ietf-sipcore-dns-dual-stack-03 to draft-ietf-sipcore-dns-dual-stack-04	9
8.6. Changes from draft-ietf-sipcore-dns-dual-stack-02 to draft-ietf-sipcore-dns-dual-stack-03	10
9. References	10
9.1. Normative References	10
9.2. Informative References	11
Authors' Addresses	11

1. Introduction

The Session Initiation Protocol (SIP, [RFC3261]) and the additional documents that extended it provide support for both IPv4 and IPv6. However, this support does not fully extend to the highly hybridized environments that are characteristic of the transitional migratory phase from IPv4 to IPv6 networks. During this phase, many server and client implementations run on dual-stack hosts. In such environments, a dual-stack host will likely suffer greater connection delay, and by extension an inferior user experience, than an IPv4-only host. The need to remedy this diminished performance of dual-stack hosts led to the development of the Happy Eyeballs [RFC6555] algorithm, which has since been implemented in many protocols and applications.

This document updates the DNS lookup procedures of RFC 3263 [RFC3263] in preparation for the specification of the application of Happy Eyeballs to SIP. Happy Eyeballs will provide enhanced performance, and consequently user experience, in highly hybridized dual-stack SIP networks. The procedures described herein are such that a dual-stack client should look up both A and AAAA records in DNS and then select the best way to set up a network flow. The details of how the latter is done is considered out of scope for this document. See the Happy Eyeballs algorithm and implementation and design considerations in RFC 6555 [RFC6555] for more information about issues with setting up dual-stack network flows.

Section 4 of this document clarifies the interaction of [RFC3263] with [RFC6157] and [RFC6724].

2. 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 RFC 2119 [RFC2119].

RFC 3261 [RFC3261] defines additional terms used in this document that are specific to the SIP domain such as "proxy", "registrar", "redirect server", "user agent server" or "UAS", "user agent client" or "UAC", "back-to-back user agent" or "B2BUA", "dialog", "transaction", and "server transaction".

This document uses the term "SIP server" that is defined to include the following SIP entities: user agent server, registrar, redirect server, a SIP proxy in the role of user agent server, and a B2BUA in the role of a user agent server.

While this document focuses on the dual-stack situation described in RFC 6555 and other documents, concerning the migration from an IPv4-only network to a network supporting both IPv4 and IPv6, the techniques described can be used in other situations. Possible situations include when a device has multiple interfaces with distinct addressing characteristics and when additional IP address families are created in the future. This document uses the general term "dual-stack" to include all situations where the client has access to multiple communication methods that have distinct addressing characteristics.

The term "address records" means the DNS records which translate a domain name into addresses within the address family(ies) that the entity supports (as A records provide IPv4 addresses and AAAA records provide IPv6 addresses), regardless of whether the address family was defined before or after this document was approved.

3. DNS Procedures in a Dual-Stack Network

This specification introduces two normative DNS lookup procedures. These are designed to improve the performance of dual-stack clients in IPv4/IPv6 networks.

3.1. Dual-Stack SIP UA DNS Record Lookup Procedure

Once the transport protocol has been determined, the procedure for discovering an IP address if the TARGET is not a numeric IP address but the port is explicitly stated in the URI, is detailed in Section 4.2 of RFC 3263 [RFC3263]. The piece relevant to this discussion is:

If the TARGET was not a numeric IP address, but a port is present in the URI, the client performs an A or AAAA record lookup of the domain name. The result will be a list of IP addresses, each of which can be contacted at the specific port from the URI and transport protocol determined previously.

Section 4.2 of RFC 3263 [RFC3263] also goes on to describe the procedure for discovering an IP address if the TARGET is not a numeric IP address, and no port is present in the URI. The piece relevant to to this discussion is:

If no SRV records were found, the client performs an A or AAAA record lookup of the domain name. The result will be a list of IP addresses, each of which can be contacted using the transport protocol determined previously, at the default port for that transport. Processing then proceeds as described above for an explicit port once the A or AAAA records have been looked up.

Happy Eyeballs [RFC6555] documents that looking up the "A or AAAA record" is not an effective practice for dual-stack clients and that it can add significant connection delay and greatly degrade user experience. Therefore, this document makes the following normative addendum to the DNS lookup procedures of Section 4.2 of RFC 3263 [RFC3263] for IPv4/IPv6 hybrid SIP networks and recommends it as a best practice for such dual-stack networks:

The dual-stack client SHOULD look up address records for all address families that it supports for the domain name and add the resulting addresses to the list of IP addresses to be contacted. A client MUST be prepared for the existence of DNS resource records containing addresses in families that it does not support; if such records may be returned by the client's DNS queries, such records MUST be ignored as unusable and the supported addresses used as specified herein.

3.2. Indicating Address Family Preference in DNS SRV Records

The Happy Eyeballs algorithm [RFC6555] is particularly effective for dual-stack HTTP client applications that have significant performance differences between their IPv4 and IPv6 network paths. This is because the client can initiate two TCP connections to the server, one using IPv4 and one using IPv6, and then use the connection which completes first. This works properly because the client can test each route by initiating a TCP connection, but simply opening a TCP connection to an HTTP server does not change the server's state; the client will send the HTTP request on only one connection.

Unfortunately, in common SIP situations, it is not possible to "race" simultaneous request attempts using two address families. If the SIP requests are transmitted as single UDP packets, sending two copies of the request to two different addresses risks having two copies of the request propagating through the SIP network at the same time. The difference between SIP and HTTP is that in SIP the sender cannot test a route in a non-state-changing way.

(If two copies of the same request arrive at the destination client, the client MUST reject the second of them with a 482 "Merged Request" response.[RFC3261] But this rule is not sufficient to prevent user-visible differences in behavior. A proxy that is upstream of the second request to arrive at the client may (almost immediately!) serially fork the second request to further destinations (e.g., the voicemail service for the destination client).)

In this common scenario it is often necessary for a dual-stack client to indicate a preference for either IPv4 or IPv6. A service may use DNS SRV records to indicate such a preference for an address family.

This way, a server with a high-latency and/or low-capacity IPv4 tunnel may indicate a preference for being contacted using IPv6. A server that wishes to do this can use the lowest SRV priority to publish hostnames that only resolve in IPv6 and the next priority with host names that resolve in both address families.

Note that hostnames that have addresses in only one address family are discouraged by [RFC6555]. Such special-purpose hostnames SHOULD be used only as described in this section, as targets of SRV records for an aggregate host name, where the aggregate host name ultimately resolves to addresses in all families supported by the client.

4. Clarification of Interaction with RFC 6724

Section 5 of [RFC6157] specifies that the addresses from the address records for a single target DNS name for a server's DNS name must be contacted in the order specified by the source and destination address selection algorithms defined in [RFC6724]. The set of addresses provided to a single invocation of the destination address selection algorithm MUST be the address records for the target DNS name in a single SRV record (or, if there are no SRV records, the DNS name in the URI or derived via NAPTR) -- the destination address selection algorithm MUST NOT reorder addresses derived from different SRV records. Typically, destination address selection is done by using the (relatively new) `getaddrinfo()` function to translate the target DNS name into a list of IPv4 and/or IPv6 addresses in the order in which they are to be contacted, as that function implements [RFC6724].

Thus, if SRV lookup on the server's DNS name is successful, the major ordering of the complete list of destination addresses is determined by the priority and weight fields of the SRV records (as specified in [RFC2782]) and the (minor) ordering among the destinations derived from the "target" field of a single SRV record is determined by [RFC6724].

For example, consider a server with DNS name `example.com`, with TCP transport specified. The relevant SRV records for `example.com` are:

```
_sip._tcp.example.com. 300 IN SRV 10 1 5060 sip-1.example.com.  
_sip._tcp.example.com. 300 IN SRV 20 1 5060 sip-2.example.com.
```

The processing of [RFC2782] results in this ordered list of target domain names:

```
sip-1.example.com  
sip-2.example.com
```

The address records for sip-1.example.com, as ordered by [RFC6724], are

```
sip-1.example.com. 300 IN AAAA 2001:0db8:58:c02::face
sip-1.example.com. 300 IN AAAA 2001:0db8:c:a06::2:cafe
sip-1.example.com. 300 IN AAAA 2001:0db8:44:204::d1ce
sip-1.example.com. 300 IN A 192.0.2.45
sip-1.example.com. 300 IN A 203.0.113.109
sip-1.example.com. 300 IN A 198.51.100.24
```

and the address records for sip-2.example.com, as ordered by [RFC6724], are:

```
sip-2.example.com. 300 IN AAAA 2001:0db8:58:c02::dead
sip-2.example.com. 300 IN AAAA 2001:0db8:c:a06::2:beef
sip-2.example.com. 300 IN AAAA 2001:0db8:44:204::c0de
sip-2.example.com. 300 IN A 192.0.2.75
sip-2.example.com. 300 IN A 203.0.113.38
sip-2.example.com. 300 IN A 198.51.100.140
```

Thus, the complete list of destination addresses has this ordering:

```
2001:0db8:58:c02::face
2001:0db8:c:a06::2:cafe
2001:0db8:44:204::d1ce
192.0.2.45
203.0.113.109
198.51.100.24
2001:0db8:58:c02::dead
2001:0db8:c:a06::2:beef
2001:0db8:44:204::c0de
192.0.2.75
203.0.113.38
198.51.100.140
```

In particular, the destination addresses derived from sip-1.example.com and those derived from sip-2.example.com are not interleaved; [RFC6724] does not operate on the complete list. This would be true even if the two SRV records had the same priority and were (randomly) ordered based on their weights, as the address records of two target DNS names are never interleaved.

5. Security Considerations

This document introduces two new normative procedures to the existing DNS procedures used to locate SIP servers. A client may contact additional target addresses for a URI beyond those prescribed in [RFC3263], and/or it may contact target addresses in a different order than prescribed in [RFC3263]. Neither of these changes introduce any new security considerations because it has always been assumed that a client desiring to send to a URI may contact any of its targets that are listed in DNS.

The specific security vulnerabilities, attacks and threat models of the various protocols discussed in this document (SIP, DNS, SRV records, Happy Eyeballs requirements and algorithm, etc.) are well documented in their respective specifications.

6. IANA Considerations

This document does not require any actions by IANA.

7. Acknowledgments

The authors would like to acknowledge the support and contribution of the SIP Forum IPv6 Working Group. This document is based on a lot of tests and discussions at SIPit events, organized by the SIP Forum.

This document has benefited from the expertise and review feedback of many participants of the IETF DISPATCH and SIPCORE WG mailing lists as well as those on the SIP Forum IPv6 Task Group mailing list. The authors wish to specifically call out the efforts and express their gratitude for the detailed and thoughtful comments and corrections of Dan Wing, Brett Tate, Rifaat Shekh-Yusef, Carl Klatsky, Mary Barnes, Keith Drage, Cullen Jennings, Simon Perreault, Paul Kyzivat, Adam Roach, Richard Barnes, Ben Campbell, Stefan Winter, Spencer Dawkins, and Suresh Krishnan. Adam Roach devised the example in Section 4.

8. Revision History

[Note to RFC Editor: Please remove this entire section upon publication as an RFC.]

8.1. Changes from draft-ietf-sipcore-dns-dual-stack-07 to draft-ietf-sipcore-dns-dual-stack-08

Remove the reference to RFC 3484, since that RFC has been superseded, and the reference was only the statement that 3484 had been superseded by RFC 6724.

Added explanation why "racing" simultaneous copies of a SIP requests causes incorrect behavior. Acknowledged Spencer Dawkins for this.

In Section 4, made explicit the ordered list of target domain names that result from processing the SRV records. Acknowledged Suresh Krishnan for suggesting this.

Updated the Terminology section to remove the definitions of "IPv4-only", etc. (which weren't being used) and add a definition of "dual-stack" that includes all multiple-stack situations.

8.2. Changes from draft-ietf-sipcore-dns-dual-stack-06 to draft-ietf-sipcore-dns-dual-stack-07

Update per Ben Campbell's AD evaluation.

Update Vijay Gurbani's affiliation.

Update per Stefan Winter's OPS-DIR review.

8.3. Changes from draft-ietf-sipcore-dns-dual-stack-05 to draft-ietf-sipcore-dns-dual-stack-06

Acknowledged Adam Roach for providing the example in Section 4.

Correct references to [RFC6157] vs. references to [RFC6724].

8.4. Changes from draft-ietf-sipcore-dns-dual-stack-04 to draft-ietf-sipcore-dns-dual-stack-05

Simplified the acknowledgments.

Improve wording and punctuation.

Rewrote Section 4 based on critiques on the Sipcore list. Included an example by Adam Roach.

Replaced "RR's" with "records" per suggestion by Jean Mahoney.

8.5. Changes from draft-ietf-sipcore-dns-dual-stack-03 to draft-ietf-sipcore-dns-dual-stack-04

Changed the "updates" specification to add RFC 3263 and remove RFC 6157.

Added Simon Perreault to the acknowledgments.

Minor wording changes.

8.6. Changes from draft-ietf-sipcore-dns-dual-stack-02 to draft-ietf-sipcore-dns-dual-stack-03

Described the relationship to RFC 3263 as "update", since the existing wording in 3263 is not what we want. Arguably, the new wording is what was intended in 3263, but the existing wording either does not say that or says it in a way that is easily misunderstood.

Described the relationship to RFC 6157 as "clarification", since the described interaction between 3263 and 6157 appears to be the only reasonable interpretation.

Revised wording, punctuation, and capitalization in various places.

Clarified that this draft does not document Happy Eyeballs for SIP, but is preparatory for it.

Attempted to use "update" for text that is definitively a change to the preexisting text and "clarify" for text that is a more clear statement of the (presumed) intention of the preexisting text.

Removed normative words from section 1, the introduction.

Copied definition of "address records" from RFC 2782 (SRV records) to allow the specifications to expand automatically to include any new address families.

Relocated the text requiring a client to ignore addresses that it discovers in address families it does not support from section 4.2 (which describes why the situation arises) to section 4.1 (which describes how clients look up RRs).

Clarified the interaction with RFC 6157 (source and destination address selection in IPv6) to specify what must have been intended: The major sort of the destinations is the ordering determined by priority/weight in the SRV records; the addresses derived from a single SRV record's target are minorly sorted based on RFC 6157.

Removed editor's name from the acknowledgments list.

9. References

9.1. Normative References

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

- [RFC2782] Gulbrandsen, A., Vixie, P., and L. Esibov, "A DNS RR for specifying the location of services (DNS SRV)", RFC 2782, DOI 10.17487/RFC2782, February 2000, <<http://www.rfc-editor.org/info/rfc2782>>.
- [RFC3263] Rosenberg, J. and H. Schulzrinne, "Session Initiation Protocol (SIP): Locating SIP Servers", RFC 3263, DOI 10.17487/RFC3263, June 2002, <<http://www.rfc-editor.org/info/rfc3263>>.
- [RFC6157] Camarillo, G., El Malki, K., and V. Gurbani, "IPv6 Transition in the Session Initiation Protocol (SIP)", RFC 6157, DOI 10.17487/RFC6157, April 2011, <<http://www.rfc-editor.org/info/rfc6157>>.
- [RFC6724] Thaler, D., Ed., Draves, R., Matsumoto, A., and T. Chown, "Default Address Selection for Internet Protocol Version 6 (IPv6)", RFC 6724, DOI 10.17487/RFC6724, September 2012, <<http://www.rfc-editor.org/info/rfc6724>>.

9.2. Informative References

- [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, <<http://www.rfc-editor.org/info/rfc3261>>.
- [RFC6555] Wing, D. and A. Yourtchenko, "Happy Eyeballs: Success with Dual-Stack Hosts", RFC 6555, DOI 10.17487/RFC6555, April 2012, <<http://www.rfc-editor.org/info/rfc6555>>.

Authors' Addresses

Olle E. Johansson
Edvina AB
Runbovaegen 10
Sollentuna SE-192 48
SE

Email: oej@edvina.net

Gonzalo Salgueiro
Cisco Systems
7200-12 Kit Creek Road
Research Triangle Park, NC 27709
US

Email: gsalguei@cisco.com

Vijay Gurbani
Bell Labs, Nokia Networks
1960 Lucent Lane
Rm 9C-533
Naperville, IL 60563
US

Email: vkg@bell-labs.com

Dale R. Worley (editor)
Ariadne Internet Services
738 Main St.
Waltham, MA 02451
US

Email: worley@ariadne.com

SIPCore
Internet-Draft
Updates: 3261 (if approved)
Intended status: Standards Track
Expires: March 9, 2017

R. Shekh-Yusef, Ed.
Avaya
V. Pascual
Oracle
C. Holmberg
Ericsson
September 5, 2016

The Session Initiation Protocol (SIP) OAuth
draft-yusef-sipcore-sip-oauth-04

Abstract

This document defines an authorization framework for SIP that is based on the OAuth 2.0 framework, and adds a simple identity layer on top of that, based on the OpenID Connect Core 1.0, to enable Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User.

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 <http://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 March 9, 2017.

Copyright Notice

Copyright (c) 2016 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<http://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect

to this document. Code Components extracted from this document must include Simplified BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Simplified BSD License.

This document may contain material from IETF Documents or IETF Contributions published or made publicly available before November 10, 2008. The person(s) controlling the copyright in some of this material may not have granted the IETF Trust the right to allow modifications of such material outside the IETF Standards Process. Without obtaining an adequate license from the person(s) controlling the copyright in such materials, this document may not be modified outside the IETF Standards Process, and derivative works of it may not be created outside the IETF Standards Process, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1.	Introduction	3
1.1.	Terminology	4
1.2.	Definitions	4
1.3.	Use Cases	4
1.3.1.	Enterprise SSO	4
1.3.2.	3GPP	5
1.3.3.	Confidential SIP Hardphone	5
1.3.4.	Public SIP Hardphone	5
1.3.5.	SIP SSO	6
1.4.	Roles	7
1.5.	ID Token	7
1.6.	Authentication Types	8
2.	Benefits	8
2.1.	Single Sign-On	8
2.2.	Service Authorization	8
2.3.	Third-Party Authentication	9
3.	Authorization Code Grant type	9
3.1.	Operations Overview	9
3.2.	Authentication	12
3.3.	Registration	13
3.4.	Subsequent Requests	14
3.5.	Token Refresh	14
3.6.	Services	15
4.	Implicit Grant Type	16
4.1.	OAuth Implicit Grant	16
4.1.1.	Overview	16
4.1.2.	Authentication	17
4.1.3.	Registration	18
4.1.4.	Subsequent Requests	19

4.1.5. Services	19
4.2. OpenID Implicit Grant	20
5. Resource Owner Password Credentials Grant type	21
5.1. Operations Overview	21
5.2. Registration and Acquiring Tokens	22
5.3. Discarding Credentials	23
5.4. Token Refresh	23
5.5. Authenticated Requests	23
5.6. Examples	24
6. Outbound	25
6.1. Authorization Code Grant type	25
6.2. Resource Owner Password Credentials Grant type	25
7. Security Considerations	25
8. IANA Considerations	25
9. Acknowledgments	25
10. Normative References	25
Authors' Addresses	26

1. Introduction

The SIP protocol [RFC3261] uses the framework used by the HTTP protocol for authenticating users, which is a simple challenge-response authentication mechanism that allows a server to challenge a client request and allows a client to provide authentication information in response to that challenge.

The SIP protocol does not have an authorization framework to allow the system to control access to various services provided by the system.

OAuth 2.0 [RFC6749] defines a token based authorization framework to allow clients to access resources on behalf of their user. It also defines four types of authorization grants, which the client uses to request the access token.

The OpenID Connect 1.0 [OPENID] specifications defines a simple identity layer on top of the OAuth 2.0 protocol, which enables Clients to verify the identity of the End-User based on the authentication performed by an Authorization Server, as well as to obtain basic profile information about the End-User.

This document defines an authorization framework for SIP that is based on the OAuth 2.0 framework, and adds the identity layer on top of that, based on the OpenID Connect Core 1.0 specification

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 [RFC2119].

1.2. Definitions

Types of SIP services:

- * Basic SIP Services: make/receive call, transfer, call forward, etc.
- * Advanced SIP Services: services provided by SIP application servers, e.g. Voice Mail, Conference Services, Presence, IM, ...

Single Sign-On (SSO)

SSO is a property that allows the user to be authenticated once and as a result have access to multiple services in the system.

Authentication

The process of verifying the identity of a user trying to get access to some network services.

Authorization

The process of controlling an authenticated user access to network services and the level of service provided to the user.

1.3. Use Cases

1.3.1. Enterprise SSO

An enterprise is interested in providing its users with an SSO capability to the various corporate services. The enterprise has an authorization server for controlling the user access to their network and would like to extend that existing authorization server to control the user access to the various services provided by their SIP network.

The user is expected to provide his corporate credentials to login to the corporate network and get different types of services, regardless of the protocol used to provide the service, and without the need to create different accounts for these different types of services.

1.3.2. 3GPP

The 3GPP network has a requirement to allow a user using a WebRTC IMS Client (WIC) to authenticate to a WebRTC Authorization Function (WAF) and in response be given an access token that allows the user to register and get service from the 3GPP SIP network.

The WIC downloads an IMS webpage from the WebRTC Web Server Function (WWSF) using HTTP. The WIC then requests an access token from the WAF using HTTP, which the WIC then uses to register to the SIP network through the P-CSCF enhanced for WebRTC (eP-CSCF) element.

1.3.3. Confidential SIP Hardphone

A SIP hardphone with rich UI, that has the capability to maintain the confidentiality of user's credentials, is used to authenticate to an authorization server, get a token, and use that token to register and get service from the SIP network.

When the phone interacts with the authorization server and gets challenged to provide credentials, the phone will prompt the user to enter his credentials which will be used to authenticate to the authorization server.

1.3.4. Public SIP Hardphone

A SIP hardphone with limited UI capabilities, that is incapable of maintaining confidentiality of user's credentials, is used to register with the SIP network by providing an access code obtained from an authorization server.

When the phone interacts with the SIP network without providing any credentials, the phone gets challenged to provide proper credentials.

The user will then use an out of band method, e.g browser, to authenticate to the authorization server and get a short-lived numeric access code.

The user will then use the phone's keypad to provide the numeric access code to the SIP phone. The phone will then use the access code to register and get service from the SIP network. The SIP Proxy will exchange the access code with access token from the authorization server.

1.3.5. SIP SSO

An enterprise is interested in providing its users with an SSO capability to the various corporate SIP services.

The enterprise wants to control the services provided to their SIP users and the level of service provided to the user by their SIP application servers without the need to create different accounts for these services.

The enterprise wants to utilize an existing authentication mechanism provided by SIP, but would like to be able to control who gets access to what service and when.

The user is expected to use his SIP credentials to login to the SIP network and get access to the basic services, and to get access to the services provided by the various SIP application servers without being challenged to provide credentials for each type of service.

1.4. Roles

resource owner

An entity capable of granting access to a protected resource. When the resource owner is a person, it is referred to as an end-user.

In a typical SIP network, it is the management element in the system that acts as a resource owner.

resource server

The server hosting the protected resources or services, capable of accepting and responding to protected resource and services requests using access tokens.

OAuth 2.0 client

An application making protected resource requests on behalf of the resource owner and with its authorization. The term "client" does not imply any particular implementation characteristics (e.g., whether the application executes on a server, a desktop, or other devices).

SIP client

An application making requests to access SIP services on behalf of the end-user.

authorization server

The server issuing tokens to the OAuth 2.0 client or SIP Client after successfully authenticating the resource owner and obtaining authorization.

proof-of-possession (pop)

A hash used by one party to prove to another party that it is in possession of some shared credentials, without sending the credentials on the wire.

1.5. ID Token

ID token, as defined in the OpenID document, is a security token that contains claims about the authentication of an end-user by an authorization server.

1.6. Authentication Types

There are two types of user authentications in SIP:

- o Proxy-to-User: which allows a server that is providing a service to authenticate the identity of a user before providing the service.
- o User-to-User: which allows a user receiving a request to authenticate the identity of the remote user before processing the request.

The mechanism defined in this document addresses the proxy-to-user authentication only. For user-to-user authentication refer to the mechanism defined in [STIR].

2. Benefits

This section describes the benefit of this authorization framework:

2.1. Single Sign-On

With the existing mechanism, the proxy and application servers might need to challenge many of the requests sent by a client, which adds traffic that could be avoided with this authorization mechanism.

Single Sign-On is a property that allows the user to be authenticated once and as a result have access to multiple services in the system.

This authorization mechanism would enable Single Sign-On, as the user will be authenticated once and as a result given a token and a refresh token to allow the user access to various services based on the token scope.

2.2. Service Authorization

This authorization mechanism allows the system to centrally control the services provided to the user, e.g conference services, voice mail, etc. The mechanism also allow control over the level of services provided to the user; for example, if the user is given access to conference services, the system controls whether the user gets access to video conference services or only audio conference services.

2.3. Third-Party Authentication

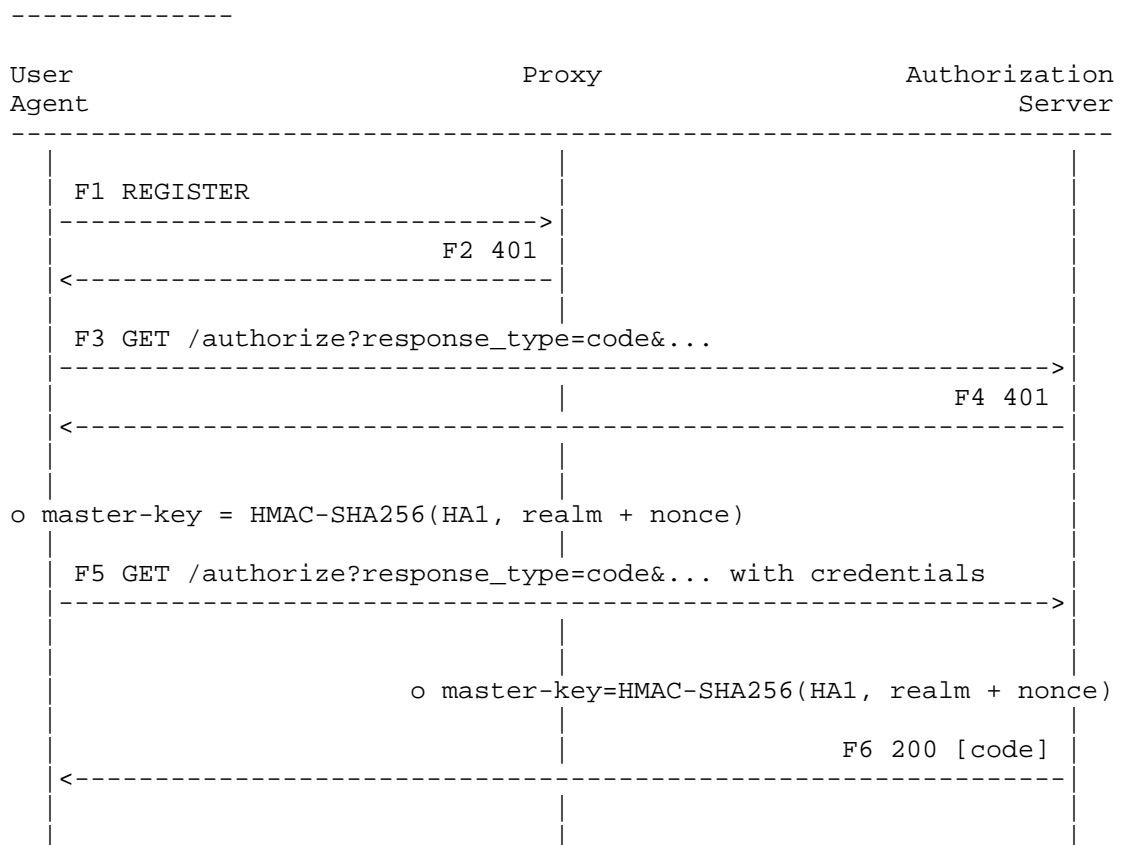
This authorization mechanism allows the user to be authenticated and obtain tokens using some Third-Party Authorization mechanism and still get services from the system.

3. Authorization Code Grant type

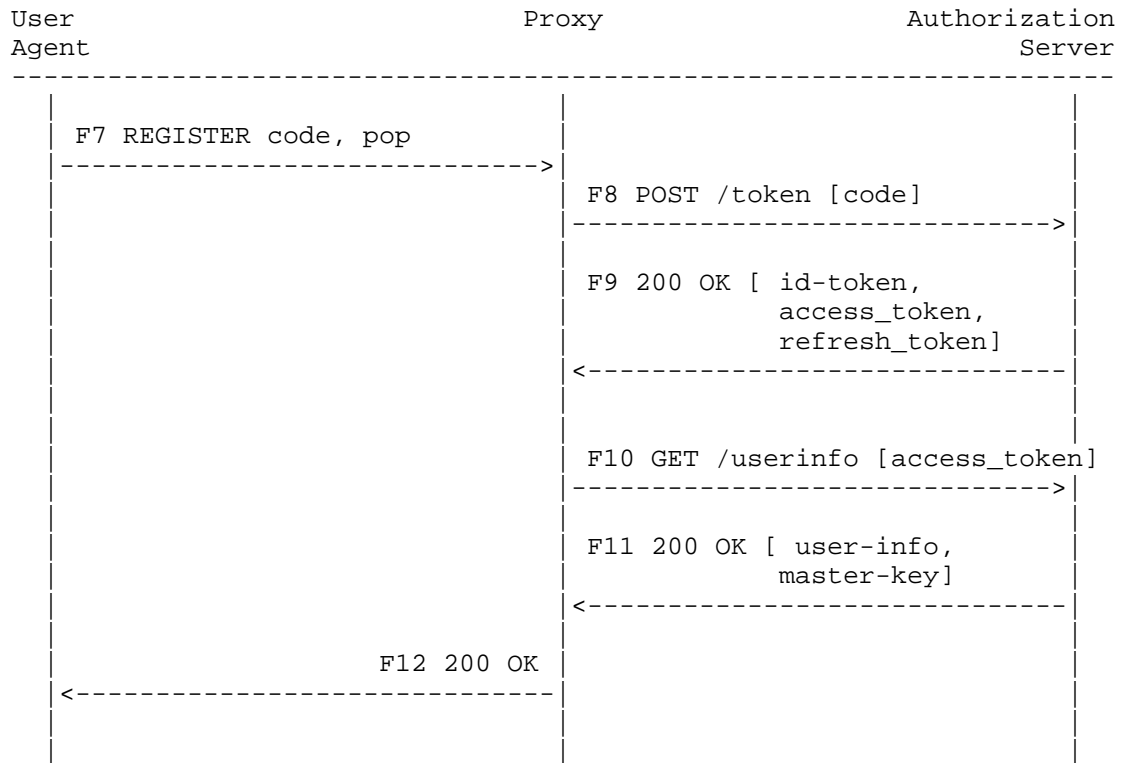
3.1. Operations Overview

The following figure provides a high level view of flow of messages for the Authorization Code Grant type:

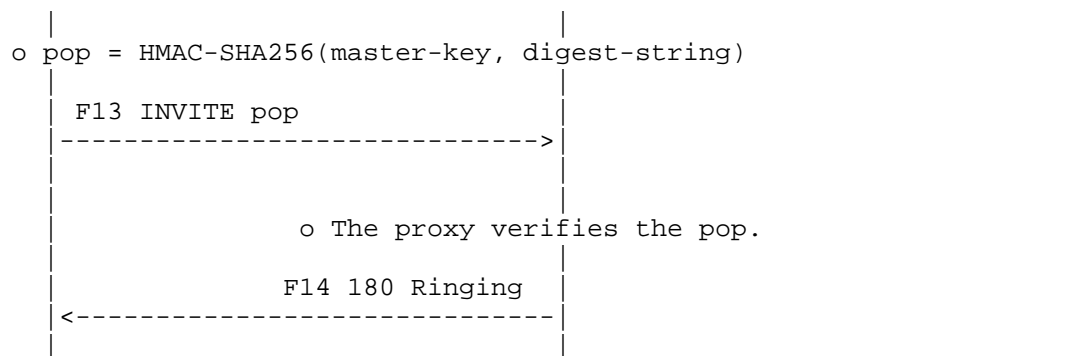
Authentication



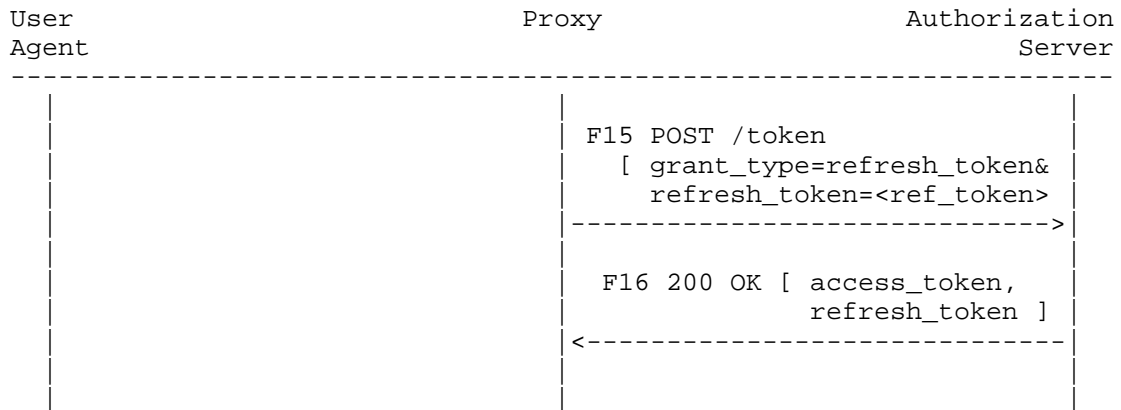
Registration



Subsequent Requests



Token Refresh



The UA initially sends a REGISTER request (F1) without providing any credentials.

The proxy challenges the UA by responding with 401 (F2) that includes the address of the Authorization Server.

[[OPEN ISSUE]] How should the UA be redirected to the Authorization Server: 1. New SIP parameter? 2. Extend the Bearer scheme? 3. Define a new Scheme?

The UA will then contact the Authorization Server without providing any credentials in the first request (F3). The Authorization Server challenges the request using the Digest scheme (F4), and the client retries the request (F5) and provides the user's credentials.

The Authorization Server verifies the request from the client; if the verification is successful, the Authorization Server responds with 200 OK (F6) and includes a code in the body part.

The UA then retries the request (F7) and include the code in the body of the request. The proxy then contacts the Authorization Server and exchanges the code for tokens (F8 and F9), and gets the user information (F10 and F11). The proxy then sends 200 OK to the UA to complete the registration process.

3.2. Authentication

The UA initiates the process by sending a REGISTER request (F1) to the proxy. The proxy will redirect the UA to the Authorization Server by responding with 401 (F2) that includes the address of the Authorization Server in the form of an HTTP URI.

The UA constructs the initial request (F3) to the Authorization Server without providing any user credentials, but with the following URI parameters in the query component:

response_type (REQUIRED)

Value MUST be set to "code".

user_id (REQUIRED)

The user's identification with the Authorization Server.

scope (OPTIONAL)

The scope of the access request

state (RECOMMENDED)

The value of this parameter is a nonce created by the client to prevent replay attack. The nonce is a uniquely generated value for each request. This parameter might not be included with the initial request that does not include credentials (F3).

The Authorization Server uses the user identification specified in the user_id parameter to verify that the user has an account in the system, and then challenges the request by responding with 401 (F4) with Digest scheme.

The UA will generate a master-key that is based on an HMAC-Hash algorithm, e.g. HMAC-SHA256, that takes an input the user's HA1 and the concatenation of realm and nonce received in the challenge from the server.

The UA will then send a new authorization request (F5), but this time include the credentials requested by the server. The UA will use the same parameters values used in the initial authorization request with the exception of the state parameter which will get a new nonce value.

When the server receives the request with the credentials (F5), the server will verify the digest provided by the UA; if that is

successful, the server will respond with 200 OK (F6) and include a code in the body of the response with the following parameters:

grant_type (REQUIRED)

Value MUST be set to "authorization_code".

code (REQUIRED)

The authorization code received from the authorization server.

The server then generates a master-key that is based on an HMAC-Hash algorithm, e.g. HMAC-SHA256, that takes an input the user's HA1, and the concatenation of realm and nonce sent in the challenge (F4) to the client.

3.3. Registration

The UA will send a new REGISTER request (F7) and include the code in the body of the request with the following parameters:

grant_type (REQUIRED)

Value MUST be set to "authorization_code".

code (REQUIRED)

The authorization code received from the authorization server.

The proxy sends a POST request (F8) to the Authorization Server and include the following parameters in the body:

grant_type (REQUIRED)

Value MUST be set to "authorization_code".

code (REQUIRED)

The authorization code received from the authorization server.

If the request is valid and authorized, the authorization server responds with a 200 OK (F9) with id_token, access token, and refresh_token in the body.

The UA sends a GET request (F10) to the Authorization Server to fetch the user information, and includes the access token in the body of the request. In response the Authorization Server will respond with

200 OK and include the user information and the master-key associated with the user in the body part.

The proxy then responds with 200 OK (F12) to the UA to complete the registration process.

3.4. Subsequent Requests

When the UA wants to send any request to the proxy, it MUST include the Authorization header and use the Bearer scheme to carry the proof-of-possession of the master-key.

The pop is calculated using the master-key as follows:

```
pop = HMAC-SHA256(master-key, digest-string)
```

The following is an example of an Authorization header with Bearer scheme:

```
Authorization: Bearer pop=<pop>
```

See rfc4474, section 9, for the SIP headers to hash to create digest-string.

[[OPEN ISSUE]] The Bearer scheme is used to deliver tokens without providing any proof of possession. We probably need to use different scheme later on.

3.5. Token Refresh

The proxy makes a refresh request to the Authorization Server by sending a refresh POST request (F13) that includes a body with the grant_type and the refresh_token.

For example:

```
grant_type=refresh_token&refresh_token=<refresh_token>
```

If the proxy fails to refresh the token, then it MUST challenge the next request from the UA, and as a result the UA MUST go through the authorization process again to obtain new tokens.

3.6. Services

When the UA tries to access a service on behalf of a user, e.g. Voice Mail Service, the proxy forwards the request to the server providing the service and MUST include an Authorization header with the Bearer scheme that carries the token needed to get service, as follows:

Authorization: Bearer token=<token>

4. Implicit Grant Type

The implicit grant type is used by the SIP UA to directly obtain access tokens from the Authorization Server to be able to register and get service from the SIP network.

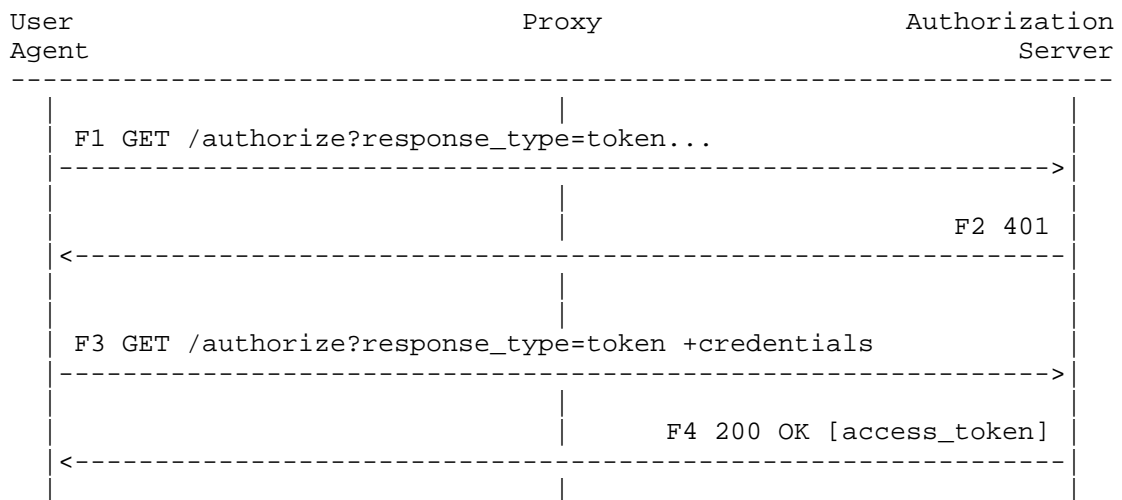
This grant type does not support the issuance of refresh tokens, which means that the SIP UA must re-authenticate again to the Authorization Server to get a new token before the current token expires.

4.1. OAuth Implicit Grant

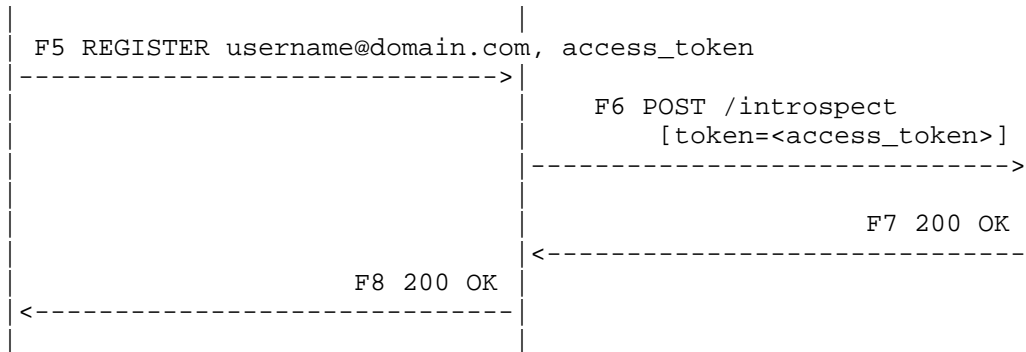
4.1.1. Overview

The following figure provides a high level view of flow of messages for the OAuth Implicit Grant type:

Authentication



Registration



4.1.2. Authentication

The UA starts the process by sending an HTTP GET request to the Authorization Server without providing any credentials in the first request (F1).

The UA constructs the initial request (F1) to the Authorization Server with the following URI parameters in the query component:

response_type (REQUIRED)

Value MUST be set to "token".

user_id (REQUIRED)

The user's identification with the Authorization Server.

scope (OPTIONAL)

The scope of the access request.

The Authorization Server challenges the request using the Digest scheme (F2). The client retries the request (F3) and provides the user's credentials. In response the Authorization Server responds with 200 OK (F4) with the Access Token in the body.

4.1.3. Registration

The UA starts the registration process with the SIP proxy by sending a REGISTER request (F5) with the access token it obtained in the previous steps (F1-F4).

The UA adds the following parameters to the body of the REGISTER request:

access_token (REQUIRED)

The access token issued by the authorization server.

token_type (REQUIRED)

The type of the token issued by the authorization server. Value is case insensitive.

expires (RECOMMENDED)

The lifetime in seconds of the access token.

scope (OPTIONAL)

The scope of the access request.

If introspection is used [RFC7662], then the proxy validates the access token by sending an HTTP POST request (F6), with the parameters sent as "application/x-www-form-urlencoded" data, to the Authorization Server and include the following parameters:

token (REQUIRED)

The string value of the token.

token_type_hint (OPTIONAL)

A hint about the type of the token submitted for introspection.

Authorization Server then validates the request and responds with 200 OK (F7), with a JSON object in the body with the following parameters:

active (REQUIRED)

Boolean indicator of whether or not the presented token is currently active.

scope (OPTIONAL)

The scope of the access request.

Other parameters

TBD.

4.1.4. Subsequent Requests

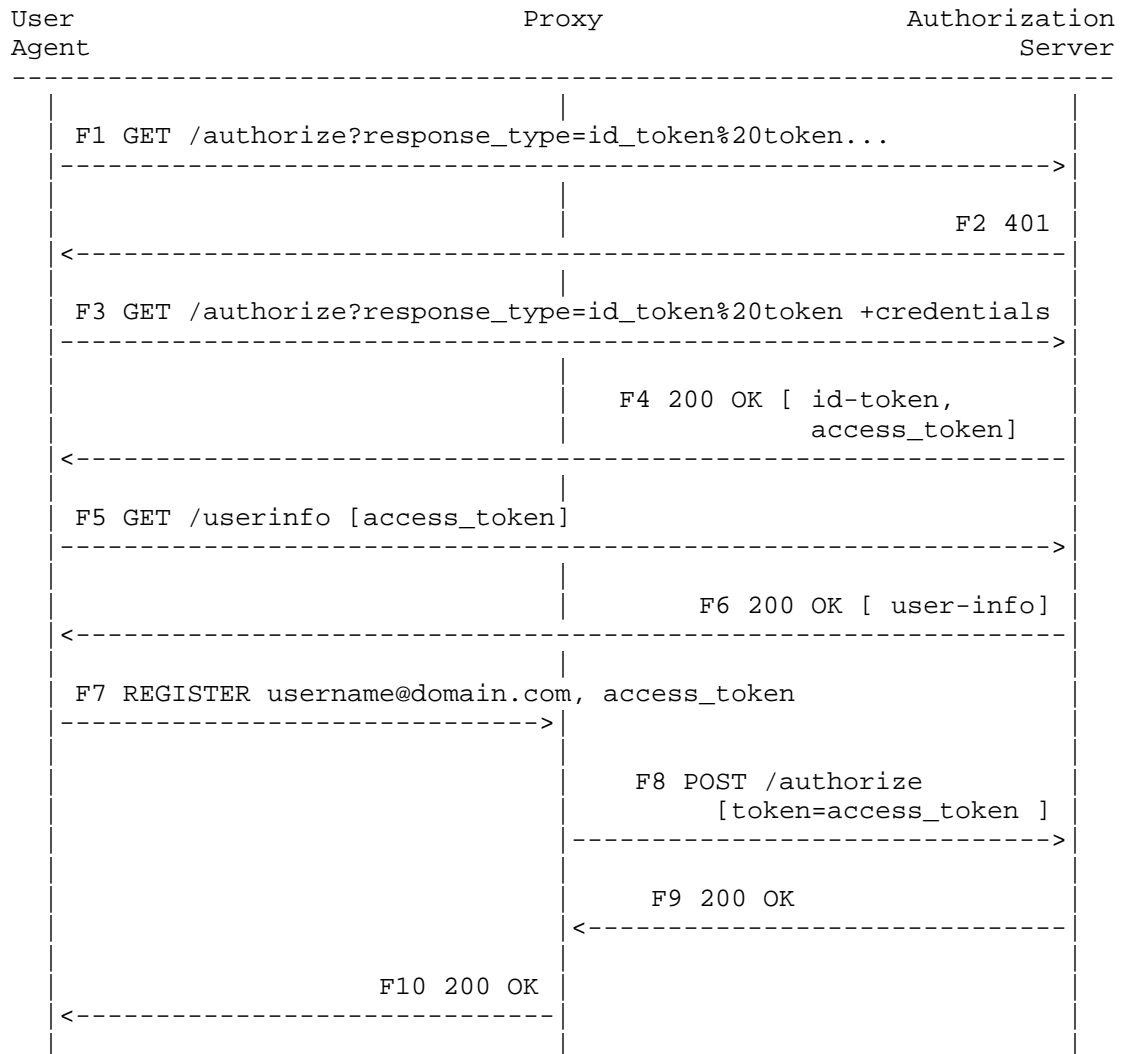
All subsequent requests from the UA MUST include a valid access token. The UA MUST obtain a new access token before the access token expiry period to continue to get service from the system.

4.1.5. Services

When the proxy forwards a request from a UA to an application server, it makes sure to keep the access token and scope in the message to allow the application server to provide the proper service to the user.

4.2. OpenID Implicit Grant

The following figure provides a high level view of flow of messages for the OpenID Implicit Grant type:



5. Resource Owner Password Credentials Grant type

5.1. Operations Overview

The following figure provides a high level view of flow of messages for the Resource Owner Password Credentials Grant type:



During registration the UA initially sends a REGISTER request (F1) without providing any credentials.

The proxy then challenges the UA by responding with 401 (F2) that includes the Digest scheme in the www-authenticate header.

The UA will generate a master-key that is based on an HMAC-Hash algorithm, e.g. HMAC-SHA256, that takes an input the user's HA1 and the concatenation of realm and nonce received in the challenge from the server. The UA will continue to use the existing operation of handling the Digest challenge and then sends a new REGISTER request (F3) with the credentials to the server.

When the server receives the request with the credentials (F3), the server will verify the digest provided by the UA; if that is successful, the server will accept the registration (F4) and include the details of the token in the response.

The server then generates a master-key that is based on an HMAC-Hash algorithm, e.g. HMAC-SHA256, that takes an input the user's HA1, and the concatenation of realm and nonce sent in the challenge to the client.

At the end of the above process the UA would have registered with the proxy and both the UA and the proxy would have created the same master-key without sending the master-key on the wire.

Later when the UA wants to send a request to the proxy it MUST always include the token and SHOULD include the pop as defined in section 4.6.

5.2. Registration and Acquiring Tokens

The UA MUST request the access token during the registration process with the proxy, by including a body with the grant_type as "password". Initially, the UA sends a REGISTER request without providing any credentials.

The proxy MUST then challenge the UA by responding with 401 with the Digest scheme in the WWW-Authenticate header.

When the UA gets challenged by the proxy to provide its credentials, the UA MUST include its credentials in the new REGISTER request in the authorization header as it is done with the existing mechanism, and MUST include a body with the grant_type as "password".

In addition, the UA MUST generate a master-key as follows:

master-key = HMAC-SHA256(HA1, realm + nonce)

Where

- o HA1 - this is the user's H(A1) as defined in [DIGEST].

- o realm - this is the realm that is returned by the server in the response to the initial request from the UA.
- o nonce - this is the nonce that is returned by the server in the response to the initial request from the UA.

When the server receives the request with the credentials, the server will verify the digest provided by the UA; if that is successful, the server will accept the registration and include the details of the token in the response.

[[OPEN ISSUE]] How should the tokens be transported to the UA? in the body of the 200 OK? or a SIP header?

The server then generates a master-key following the same procedure followed by the client.

As a result of this procedure both the UA and the server would have created the same master-key without sending the master-key on the wire.

5.3. Discarding Credentials

After successfully receiving the access and refresh tokens from the proxy, the UA SHOULD discard the user credentials.

5.4. Token Refresh

The UA makes a refresh request to the token by sending a refresh REGISTER request that includes the authorization header and a body with the grant_type, the refresh_token, and the proof-of-possession of the master-key.

For example:

```
grant_type=refresh_token&refresh_token=<refresh_token>&pop=<pop>
```

5.5. Authenticated Requests

When the UA wants to send any request to the proxy, it MUST include the Authorization header and use the Bearer scheme to carry the access token, and the proof-of-possession of the master-key.

For example:

```
Authorization: Bearer token=<token>, pop=<pop>
```

See rfc4474, section 9, for the SIP headers to hash to create the value for the proof.

[[OPEN ISSUE]] The Bearer scheme is used to deliver tokens without providing any proof of possession. We probably need to use different scheme later on.

5.6. Examples

```
REGISTER sip:registrar.biloxi.com SIP/2.0
Via: SIP/2.0/TCP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
Max-Forwards: 70
To: Bob <sip:bob@biloxi.com>
From: Bob <sip:bob@biloxi.com>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:bob@192.0.2.4>
Expires: 7200
Content-Length: 19
```

```
grant_type=password&pop=<pop>
```

```
SIP/2.0 200 OK
Via: SIP/2.0/TCP bobspc.biloxi.com:5060;branch=z9hG4bKnashds7
    ;received=192.0.2.4
To: Bob <sip:bob@biloxi.com>;tag=2493k59kd
From: Bob <sip:bob@biloxi.com>;tag=456248
Call-ID: 843817637684230@998sdasdh09
CSeq: 1826 REGISTER
Contact: <sip:bob@192.0.2.4>
Expires: 7200
Content-Length: 0
```

```
{
  "access_token": "2YotnFZFEjrlzCsicMWpAA",
  "token_type": "example",
  "expires_in": 3600,
  "refresh_token": "tGzv3JOkF0XG5Qx2TlKWIA",
  "example_parameter": "example_value"
}
```

6. Outbound

RFC5626 defines a mechanism that allows a UA to simultaneously connect and establish registration with multiple outbound proxies to get service.

This section describes that impact of outbound on this authorization mechanism.

6.1. Authorization Code Grant type

During initial registration with the primary proxy, the UA is able to get an authorization code that it will use to register with the primary proxy. Assuming the authorization server is shared between the various outbound proxies, the UA will be able to use the same authorization code to register with the secondary proxies and as a result each one of the secondary proxies will get the master-key associated with the user to be used for the calculation of the proof-of-possession.

6.2. Resource Owner Password Credentials Grant type

During registration the proxy challenges the UA, and both the proxy and the UA create a master-key based on HA1, realm, and nonce. Since the nonce is not shared between the various proxies, it is not possible for the outbound proxies to use the same master-key; as a result, the UA is expected to maintain a master-key and token per outbound proxy.

7. Security Considerations

<Security considerations text>

8. IANA Considerations

<IANA considerations text>

9. Acknowledgments

<Acknowledgments text>

10. 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, March 1997.

- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, H., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC6749] Hardt, D., "The OAuth 2.0 Authorization Framework", RFC 6749, October 2012.
- [RFC7662] Richer, J., "OAuth 2.0 Token Introspection", RFC 7662, October 2015.

Authors' Addresses

Rifaat Shekh-Yusef (editor)
Avaya
250 Sidney Street
Belleville, Ontario
Canada

Phone: +1-613-967-5267
EMail: rifaat.ietf@gmail.com

Victor Pascual
Oracle
Spain

EMail: victor.pascual.avila@oracle.com

Christer Holmberg
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

EMail: christer.holmberg@ericsson.com