IPv6 Operations Working Group (v6ops)
Internet-Draft
Intended status: Experimental
Expires: April 30, 2015

O. Nakamura
Keio Univ./WIDE Project
H. Hazeyama
NAIST / WIDE Project
Y. Ueno
Keio Univ./WIDE Project
A. Kato
Keio Univ. / WIDE Project

October 27, 2014

A Special Purpose TLD to resolve IPv4 Address Literal on DNS64/NAT64 environments
draft-osamu-v6ops-ipv4-literal-in-url-02

Abstract

In an IPv6-only environment with DNS64/NAT64 based translation service, there is no way to get access a URL whose domain name part includes an IPv4 address literal. This memo proposes a special purpose TLD so that the IPv4 address literal is accessible from such a DNS64/NAT64 environments.

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 April 30, 2015.

1. Introduction and Overview

When a host in an IPv6 only environment (an IPv6-only host) has to access an IPv4-only destination, a translator-based approach is a powerful tool. The translator-based approach is usually composed of a DNS64 server [RFC6147] and a stateful NAT64 translator [RFC6146]. The DNS64 server responds with a AAAA record of an IPv4 embedded IPv6 address with a certain IPv6 prefix assigned to the NAT64 translator, for example, the well known NAT64 prefix (64:ff9b::) or a global IPv6 prefix. The IPv6-only host sends an IPv6 packet, which is translated by the NAT64 box to an IPv4 packet. In this memo, an IPv4 embedded IPv6 address with a NAT64 prefix is described as ``Pref64::/n address''. The translation of responded IPv4 packet back into an IPv6 packet is also performed in the NAT64 translator.

The NAT64 with DNS64 approach works well for most destinations. But it does not work well when the DNS response packet resulted NXDOMAIN or SERVFAIL to the AAAA query, partly described in [RFC4074]. Resolutions of this case are out of scope of this memo.

It is legitimate to embed an IPv4 address literal in an URL such as follows:

http://192.0.2.10/index.html

In the environment described above, the destination is not accessible from an IPv6-only host. This problem has already been reported in [RFC6586] and others.

The reason why the destination specified by above notation cannot be

accessible is that no DNS lookup is performed, and no DNS64 service is able to tell a Pref64::/n address to the host. To perform DNS64/NAT64 translation against such an IPv4 address literal notation, some mechanism will be required.

This memo proposes a special-purpose TLD and defines behaviors of resolvers and of the authoritative servers to treat the special-purpose TLD. This memo also considers implementation strategy of .TLD and side effects of .TLD usages to the current communications on the Internet. The special-purpose TLD is denoted as .TLD which will be replaced with an actual TLD allocated by IANA.

The concept of .TLD is simple: All IPv4 address literal notations are rewritten to ``<ipv4-address-literal>.TLD'' on a host. As ``<ipv4-address-literal>.TLD'' is seemed to be a regular FQDN, ``<ipv4-address-literal>.TLD'' lets DNS64 servers resolve IPv4 address literal as a regular FQDN and translate the A record of ``<ipv4-address-literal>.TLD'' to a corresponding Pref64::/n address on each leaf network. For example, 192.0.2.10.TLD in DNS64/NAT64 environment would be translated to a Pref64::c000:020a. In an IPv4 environment, 192.0.2.10.TLD would be resolved just as an A record about 192.0.2.10.

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].

2. Scope of this memo

This memo focuses only on smooth migration to an IPv6-only environment with the DNS64/NAT64 solution. Therefore, this memo focuses on only ``IPv4 address literal'' problem mentioned in [RFC6586].

The ``IPv6 address literal'' is out of scope of this memo, because an URL including IPv6 address literal can be accessible in IPv6-only networks and in dual stack networks. The solutions to keep IPv4-only hosts or IPv4-only applications in IPv6 only environment are out of scope on this memo.

3. A special-purpose TLD for IPv4 Address Literal

When the part of IPv4 address literal is written to form a pseudo FQDN and the pseudo FQDN is resolved as an IPv4 address, a DNS64 server can return a AAAA record with the specified IPv4 address that is mapped to an appropriate NAT64 prefix.

Once a AAAA record is obtained, the IPv6-only host can send IPv6 packets to the destination. IPv6 packets will be translated back via NAT64 translator in exactly the same as a regular IPv4-only destination.

3.1. .TLD Authoritative DNS server behavior

The authoritative DNS server of .TLD SHOULD be operated only for a special purpose.

1. If a DNS query asks ``<ipv4-address-literal>.TLD '', .TLD authoritative server MUST return ``<ipv4-address-literal>'' as the A record of ``<ipv4-address-literal>.TLD ''.
2. Otherwise, .TLD authoritative server MUST return NXDOMAIN.

3.2. DNS64 behaviors

When a DNS64 receives a query of <ipv4-address-literal>.TLD, it SHOULD issue a DNS query to one of the .TLD authoritative servers. The response from .TLD authoritative server will be either an A record of the issued <ipv4-address-literal> or NXDOMAIN. If the response contains an A record, the DNS64 MUST translate the IPv4 address in the A record to the AAAA record by Pref64::/n address according to [RFC6147].

Taking into account of scalability, the DNS64 WOULD cache the AAAA record of <ipv4-address-literal>.TLD in a certain interval. As one of possible ways to get more scalability, the DNS64 CLOUD have the function of .TLD authoritative server.

3.3. Client behaviors

3.3.1. Case 1: manual type-writing

When a client (human) wants to access an IPv4 only server by IPv4 address literal in a DNS64/NAT64 network, he / she manually attaches .TLD to the IPv4 address of the IPv4 only server. When the network has DNS64/NAT64 function, the AAAA record, that is Pref64::/n address of the issued <ipv4-address-literal> , will be return.

The client COULD attach .TLD to the IPv4 address of the IPv4 only server in an IPv4 only network or a dual stack network. When the network situation is IPv4 only or dual stack, the A record of the issued <ipv4-address-literal>.TLD will be returned.

If the client uses FQDN or IPv6 address literal, he / she MUST NOT attach .TLD.

3.3.2. Case 2: device or application

A client (device or application), that has a name resolution function, SHOULD attach .TLD when the input value of getaddrinfo is an IPv4 address literal. For example, <ipv4-address-literal> SHOULD be rewritten to <ipv4-address-literal>.TLD. If the input value of getaddrinfo is not IPv4 address literal, the client MUST NOT attach .TLD.

Of course, the client CAN take self-synthesizing of mapped address mentioned in [RFC7050], or MAY combine .TLD method and [RFC7050] self-synthesizing method.

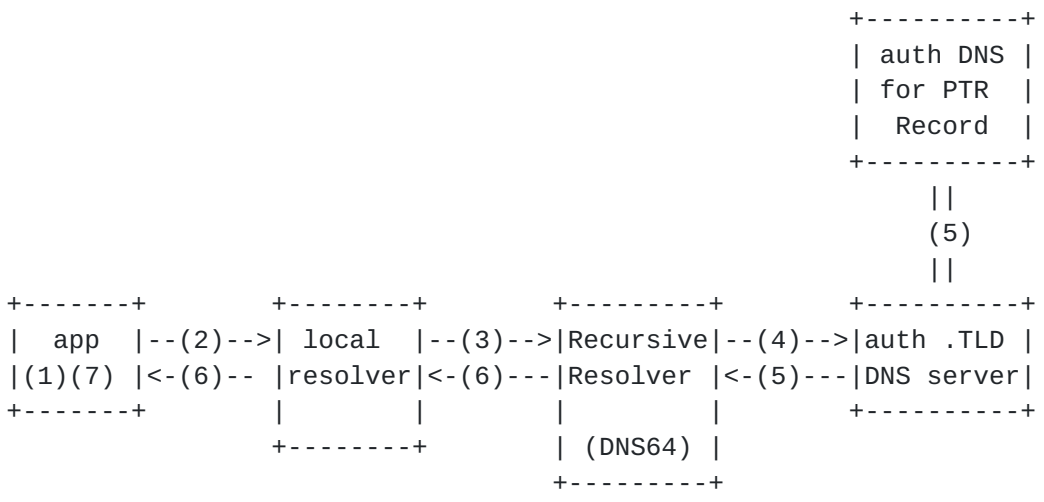
Some access authentication may not allow any external accesses until access authentication procedure is finished, and may use an IPv4 address literal on the redirected authentication web page. Taking into account such corner case, client WOULD check the reachability to the external network initially.

NOTE: migrating from IPv4 to IPv6, access authentication SHOULD avoid to use IPv4 address literal and SHOULD use FQDN for dual stack client or IPv6 only client.

3.4. DNS query flow

Figure 1 shows a DNS query flow on the .TLD.

1. An application on a client creates <ipv4-address-literal>.TLD.
2. The application inputs the query of AAAA or ANY about <ipv4-address-literal>.TLD. to its local resolver.
3. The local resolver forwards the query to a recursive resolver that would be a DNS64 server in DNS64/NAT64 environment.
4. The recursive resolver sends a recursive query of <ipv4-address-literal>.TLD.
5. .TLD authoritative server creates the A record of the issued <ipv4-address-literal>.TLD, and MAY check PTR record of the issued <ipv4-address-literal>. Then, .TLD authoritative server returns the DNS response to the recursive resolver.
6. When the recursive resolver has DNS64 function, it creates the AAAA record according to [RFC6147] and replies the AAAA record to the local resolver on the client. If the recursive resolver does not have DNS64 function, the recursive resolver returns the A record responded from .TLD authoritative server.
7. The application on the client gets the appropriate IP address (IPv4 address or Pref64::/n address), then creates an appropriate socket.



DNS Query Flow on .TLD

Figure 1

This solution would not require the modification of common shared libraries on any Operating Systems. The DNS implementations, SHOULD support .TLD. As the query flow mentioned above, .TLD authoritative server SHOULD be placed. The modification of NAT64 or DHCP are not required in this method.

3.5. Use cases

3.5.1. Use case 1: manual type-writing

For example, consider living on an IPv6-only network with DNS64/ NAT64, and receiving a message like ``please download a file foo.doc from a ftp server 192.0.2.10''. Usually, you may estimate the NAT64 prefix and calculate Pref64::/n address through [RFC7050] or [RFC7051]. Under the proposed mechanism on this memo, you can just type as follow;

```
% ftp 192.0.2.10.TLD
```

The packet would be transferred along with [RFC6384].

3.5.2. Use case 2: browser plug-in

An IPv4 address literal is often used in URL for the lazy DNS operation, a temporary HTTP server or a hidden (private) server. Taking into account user convenience, a browser plug-in can be developed that it converts the <ipv4-address-literal> on the hostname

part of an URL to <ipv4-address-literal>.TLD. It may be suggested to turn this function on when the host is on IPv6-only network, however, it may not be easy to detect the situation of the network (IPv4 only, dual stack or DNS64/NAT64 environment). A sample of Google Chrome plug-in is attached in Appendix B

3.6. Recommendation

For usability in manual type-writing, the .TLD SHOULD be as short as possible, and SHOULD express the special purpose in the name space. ``.v4'' is recommended as a candidate of .TLD, because of the simplicity and the expression of IPv4.

4. Considerations

4.1. Attached the special-purpose TLD to a regular FQDN

Conceptually, the special-purpose TLD would be attached to only IPv4 address literals, however, the special-purpose TLD may be attached to a regular FQDN notation like ``foo.bar.com.TLD''. Such misuses SHOULD be avoided.

4.2. An embedded IP address literal in the content part of URL

In some case, <ipv4-address-literal> may be embedded into the content part of a URL, however, it may be difficult for users or browser plug-ins to recognize unambiguously that a string like <ipv4-address-literal> surely means some IPv4 address. From the point of view of IPv6 migration, embedded IP address literal in the content part of an URL MUST be avoided.

4.3. Prevention the leak of the special-purpose TLD

When .TLD is actually employed in the operation, .TLD may leak to the public DNS infrastructure including root DNS servers as seen in ``.local''. Therefore, once consensus is obtained, the relevant TLD SHOULD be delegated to a set of DNS servers.

Two possible DNS operation methods can be considered. One is to delegate the TLD to AS112 servers [as112-servers]. When one of the AS112 servers received a query with .TLD, it returns with NXDOMAIN.

The other possible DNS operation is to deploy a set of special purpose DNS servers which accept queries with .TLD and synthesize an A record corresponding to the IPv4 address in the QNAME when it is a legitimate IPv4 address. Otherwise, NXDOMAIN MUST be returned.

4.4. Possibility to break connections with Apache VirtualHost concept

Changing the URL (swapping the DNS name or adding in a Pref64) frequently breaks the connections since the application is aware of the name it expects, and connecting correctly to the correct IP address is not sufficient, the name must also be the same in many cases.

For example, many websites use the Apache VirtualHost concept. When a web site that changes contents along with accessed IP address family like http://www.kame.net/ or http://dual.tlund.se/ , and if some client accesses such web site by <ipv4-address-literal>.TLD instead of FQDN, the VirtualHost may not work as intended.

Therefore, such web site, that uses the Apache VirtualHost concept, SHOULD NOT use <ipv4-address-literal> in URL and SHOULD use appropriate FQDN.

4.5. Inaffinity with HTTP/HTTPS Cookie

This solution may not work with HTTP/HTTPS cookie. We should also consider the HTTP security considerations for the cases where someone puts one of the names into a URL. For example, consider http://192.0.2.10.TLD/ to an origin that sets a cookie on the domain "*.10.TLD".

There are likely already plenty of ways to do the same thing out there, so this may not be a major issue.

4.6. TLD alternatives

In Section 3.6, we propose .v4 as the TLD, and comparisons with other candidates are discussed as follows.

4.6.1. .v4.arpa

``v4.arpa'' may be a candidate of .TLD that does not require new TLD, however, it may be confued with [RFC7050] ``ipv4only.arpa'', and the length (8 characters) of ``.v4.arpa'' is bit longer than the length (3 characters) of ``.v4'' for type-writing usages.

4.6.2. .host

``.host'' has already been assigned as one of the new gTLDs, and not considered a candidate here unless the authority of .host offers 256 (or 356 -- see discussion in Section 4.6.3) delegations to this purpose.

4.6.3. TLD less delegation

When it is feasible to "delegate" 256 TLDs (from ".0" through ".255") or 366 TLDs (".00", ".000", and others are added) for this particular purpose, it is possible to implement the functionality described in this memo without assigning a particular .TLD. It contributes 256 (or 356) extra TLDs in the Root zone.

It is known that DNS queries with such TLDs have been observed, and this delegation may interfere with undocumented usage of such TLDs.

If such 256 (or 366) delegations is suitable, bogus such queries to the root servers will be redirected to the DNS server described in Section 5.

4.7. Usages of IPv6 address literal

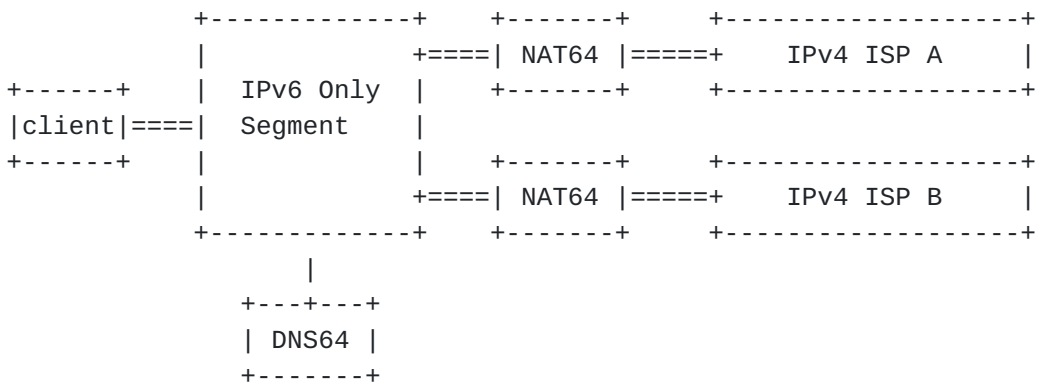
The special-purpose TLD may be applied to IPv6 address cases in same ways, however, such notation is not required in dual stack / IPv6-only environment, generally.

4.8. RFC7050 ipv4only.arpa

[RFC7050] defines a method to estimate a NAT64 prefix by querying Well-Known IPv4-only Name ``ipv4only.arpa''. [RFC7050] does not cover several situations. .TLD method is aimed to solve such situations as follows:

4.8.1. Multiple NAT64 prefixes for load balancing

One of situations is multihoming, illustrated in Figure 2. In this situation, the NAT64 prefix estimated by [RFC7050] method may be different from the one that the operator intends.

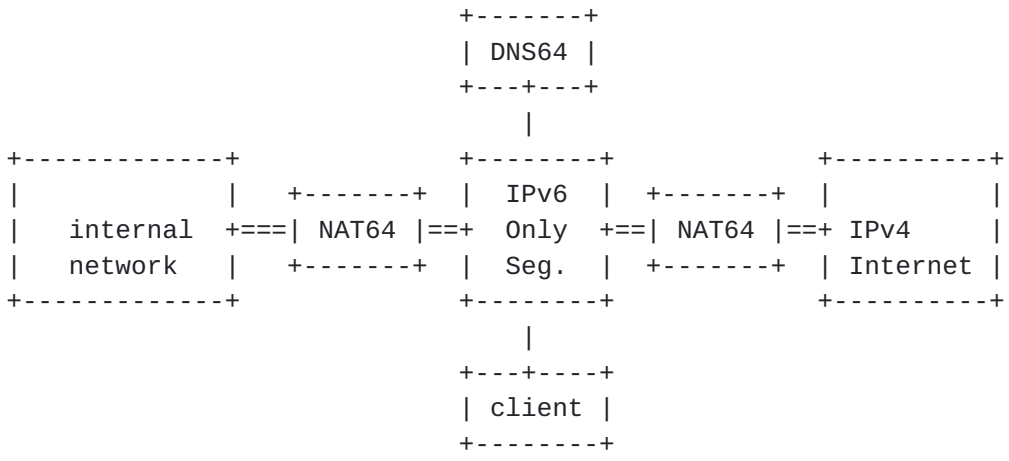


Situation A : multiple NAT64 prefixes for optimizing routes on multihoming

Figure 2

4.8.2. Multiple NAT64 prefixes for external / internal IPv4 only networks

Another situation is where multiple NAT64 prefixes are operated for accessing the external IPv4 Internet and an internal private IPv4 only network from an internal IPv6 only network. Figure 3 draws this situation. In this situation, the NAT64 prefix estimated by [RFC7050] method could not be reached to the internal IPv4 only network.



Situation B : multiple NAT64 prefixes for internal / external

Figure 3

4.8.3. Difficulty of conversion from octet expression to hex expression by human type-writing

As the initial motivation of this memo, IPv4 address literal is often used for a personal / private server that is not registered in DNS record because of lazy operation, temporal usage, or the intention to hide from DNS query scans. ``ipv4only.arpa'' solution can be available to synthesize the Pref64::/n address for the private server, however, the owner of the private server has to convert the octet expression of the IPv4 address on his/her private server to the hex expression by manual. Usually, conversion from octet expression to hex expression by manual is difficult or tiresome operation.

5. Implementation Strategy

It is suggested to implement the .TLD rewriting as in the following order:

1. Define .TLD
   Once the community agrees to accept the rewriting scheme described in this memo, it must fix the .TLD to be used. The .TLD WOULD require the update of [RFC6761].
2. .TLD delegation
   DNS queries with .TLD can leak to the DNS of the global Internet, it is highly suggested to delegate .TLD to a set of authoritative DNS servers as discussed in Section 4.3.
3. DNS64 modification
   DNS64 implementation is suggested to modify to respond corresponding AAAA record to a query with .TLD. This process can be done in parallel to the step 2 above.
4. Start using .TLD rewriting
   After, at least the step 2 is completed, the TLD rewriting may be used in manually described in Section 3.5.1 or automatically by browser plugins described in Section 3.5.2. While further discussions and observation is required, the use of an URL in IPv4 literal embedded might be discouraged. Instead, the use of .TLD notation as a legitimate URL might be encouraged even in the server side.

6. Security Considerations

The recommendation contains security considerations related to DNS. The special purpose DNS servers of this memo only treats the IPv4 address literal with .TLD. Therefore, the special DNS MAY use self-signed / authorized key for DNS responses.

When a client is to access an URL with IPv4 literal address embedded,

it triggers a DNS query, and the query may be sent over the Internet to the nearest authoritative .TLD DNS server. It may break the confidentiality against the DNS service.

TBD

7. IANA Considerations

This memo calls for ``.v4'' as the special-purpose TLD to the IANA registry.

8. Acknowledgments

Authors thank to WIDE Project members for their active discussion, implementations, and evaluations. Especially, we thank to Atsushi ONOE for the revision of this solution, Hirochika ASAI for the contribution of the prototype implementation of the special purpose authoritative DNS, and Hirotaka NAKAJIMA for the contribution of the Google chrome plug-in. We also thank to Yoshiaki KITAGUCHI, Yu-ya KAWAKAMI and others who evaluated our proof of concept special purpose DNS (.v4.wide.ad.jp) and the Google Chrome plugin-in at JANOG34 DNS64/NAT64 experiment networks. Teeme Savolainen, Cameron Byrne, Dan Wing, Erik Nygren gave us various considerations on the actual operation of .TLD.

9. References

9.1. Normative References

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

[RFC4074] Morishita, Y. and T. Jinmei, "Common Misbehavior Against DNS Queries for IPv6 Addresses", RFC 4074, May 2005.

[RFC6146] Bagnulo, M., Matthews, P., and I. van Beijnum, "Stateful NAT64: Network Address and Protocol Translation from IPv6 Clients to IPv4 Servers", RFC 6146, April 2011.

[RFC6147] Bagnulo, M., Sullivan, A., Matthews, P., and I. van Beijnum, "DNS64: DNS Extensions for Network Address Translation from IPv6 Clients to IPv4 Servers", RFC 6147, April 2011.

[RFC6384] van Beijnum, I., "An FTP Application Layer Gateway (ALG)

for IPv6-to-IPv4 Translation", RFC 6384, October 2011.

[RFC6586] Arkko, J. and A. Keranen, "Experiences from an IPv6-Only Network", RFC 6586, April 2012.

[RFC6761] Cheshire, S. and M. Krochmal, "Special-Use Domain Names", RFC 6761, February 2013.

[RFC7050] Savolainen, T., Korhonen, J., and D. Wing, "Discovery of the IPv6 Prefix Used for IPv6 Address Synthesis", RFC 7050, November 2013.

[RFC7051] Korhonen, J. and T. Savolainen, "Analysis of Solution Proposals for Hosts to Learn NAT64 Prefix", RFC 7051, November 2013.

9.2. Informative References

[as112-servers]
AS112 Project, "AS112 Project", October 2009, <https://www.as112.net/>.

Appendix A. A Test Server of the special TLD

We run a prototype implementation of the special-purpose DNS server in the WIDE backbone (AS 2500). We use ``.v4.wide.ad.jp'' as .TLD.

Appendix B. Sample extension for Google Chrome

We developed a sample plug-in code for Google Chrome ``IPv4 Address Literal Appender'' that automatically converts <ipv4-address-literal> in URL to <ipv4-address-literal>.TLD. The .TLD can be customized in the option. The ``IPv4 Address Literal Appender'' is freely available in Google Chrome Web Store, and also in github https://github.com/nunnun/nat64-v4-literal-extension.

```
var wr = chrome.webRequest;

var v4Suffix = ".TLD";
var ipAddrRegex = /^(\d|[01]?\d\d|2[0-4]\d|25[0-5])\.(\d|[01]?\d\d|
2[0-4]\d|25[0-5])\.(\d|[01]?\d\d|2[0-4]\d|25[0-5])\.(\d|[01]?\d\d|2
[0-4]\d|25[0-5])$/;

function onBeforeRequest(details) {
  var tmpuri = new URI(details.url);
  var tmphost = tmpuri.host();
  var finalUri = '';
  tmphost.replace(ipAddrRegex,function(str,p1,p2,p3,p4,offset,s){
  finalUri=tmpuri.host(p1+"."+p2+"."+p3+"."+p4+v4Suffix).toString();
  });
 if('' != finalUri) {
  console.log(finalUri);
  return {redirectUrl: finalUri};
 }
};

wr.onBeforeRequest.addListener(onBeforeRequest,{urls: ["https://*/*",
"http://*/*", "ftp://*/*"]}, ["blocking"]);
```

Authors' Addresses

Osamu Nakamura
Keio Univ./WIDE Project
5322 Endo
Fujisawa, Kanagawa 252-0882
JP

Phone: +81 466 49 1100
Email: osamu@wide.ad.jp

Hiroaki Hazeyama
NAIST / WIDE Project
8916-5 Takayama
Ikoma, Nara 630-0192
JP

Phone: +81 743 72 5111
Email: hiroa-ha@is.naist.jp

Yukito Ueno
Keio Univ./WIDE Project
5322 Endo
Fujisawa, Kanagawa 252-0882
JP

Phone: +81 466 49 1100
Email: eden@sfc.wide.ad.jp

Akira Kato
Keio Univ. / WIDE Project
Graduate School of Media Design, 4-1-1 Hiyoshi
Kohoku, Yokohama 223-8526
JP

Phone: +81 45 564 2490
Email: kato@wide.ad.jp