

P2PSIP
Internet-Draft
Intended status: Standards Track
Expires: August 20, 2013

M. Petit-Huguenin
Impedance Mismatch
February 16, 2013

Configuration of Access Control Policy in REsource LOcation And
Discovery (RELOAD) Base Protocol
draft-ietf-p2psip-access-control-00

Abstract

This document describes an extension to the REsource LOcation And Discovery (RELOAD) base protocol to distribute the code of new Access Control Policies without having to upgrade the RELOAD implementations in an overlay.

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 August 20, 2013.

Copyright Notice

Copyright (c) 2013 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 not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	2
2. Terminology	3
3. Processing	4
4. Security Considerations	6
5. IANA Considerations	7
6. References	7
6.1. Normative References	7
6.2. Informative References	8
Appendix A. Examples	8
A.1. Standard Access Control Policies	8
A.1.1. USER-MATCH	8
A.1.2. NODE-MATCH	9
A.1.3. USER-NODE-MATCH	9
A.1.4. NODE-MULTIPLE	9
A.2. Service Discovery Access Control Policy NODE-ID-MATCH . .	10
A.3. VIPR Access Control Policy	11
A.4. ShaRe Access Control Policy USER-CHAIN-ACL	12
Appendix B. Release notes	12
B.1. Modifications between ietf-p2psip-reload-access-control and petithuguenin-p2psip-access-control-05	12
B.2. Running Code Considerations	13
B.3. TODO List	13
Author's Address	13

1. Introduction

The RELOAD base protocol specifies an Access Control Policy as "defin[ing] whether a request from a given node to operate on a given value should succeed or fail." The paragraph continues saying that "[i]t is anticipated that only a small number of generic access control policies are required", but there is indications that this assumption will not hold. On all the RELOAD Usages defined in other documents than the RELOAD base protocol, roughly 50% defines a new Access Control Policy.

The problem with a new Access Control Policy is that, because it is executed when a Store request is processed, it needs to be implemented by all the peers and so requires an upgrade of the software. This is something that is probably not possible in large overlays or on overlays using different implementations. For this reason, this document proposes an extension to the RELOAD configuration document that permits to transport the code of a new Access Control Policy to each peer.

This extension defines a new element `<code>` that can be optionally added to a `<configuration>` element in the configuration document. The `<code>` element contains ECMAScript [ECMA-262] code that will be called for each `StoredData` object that use this access control policy. The code receives four parameters, corresponding to the Resource-ID, Signature, Kind and `StoredDataValue` of the value to store. The code returns true or false to signal to the implementation if the request should succeed or fail.

For example the USER-MATCH Access Control Policy defined in the base protocol could be identically defined by inserting the following code in an `<code>` element:

```
return resource.equalsHash(signer.user_name.bytes());
```

The `<kind>` parameters are also passed to the code, so the NODE-MULTIPLE Access Control Policy could be implemented like this:

```
for (var i = 0; i < kind.max_node_multiple; i++) {  
    if (resource.equalsHash(signer.node_id, i.width(4))) {  
        return true;  
    }  
}  
return false;
```

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 [RFC2119] with the caveat that "SHOULD", "SHOULD NOT", "RECOMMENDED", and "NOT RECOMMENDED" are appropriate when valid exceptions to a general requirement are known to exist or appear to exist, and it is infeasible or impractical to enumerate all of them. However, they should not be interpreted as permitting implementors to fail to implement the general requirement when such failure would result in interoperability failure.

3. Processing

A peer receiving a configuration document containing one or more `<code>` elements, either by retrieving it from the configuration server or in a ConfigUpdateReq message, MUST reject this configuration if it is not signed or if the signature verification fails.

The Compact Relax NG Grammar for this element is:

```
namespace acp = "urn:ietf:params:xml:ns:p2p:access-control"

parameter &= element acp:code {
  attribute name { xsd:string },
  xsd:base64Binary
}?

```

All peers in an overlay MUST implement this specification. One way to do this is to add a `<mandatory-extension>` element containing the "urn:ietf:params:xml:ns:p2p:access-control" namespace in the configuration document.

The mandatory "name" attribute identifies the access control policy and can be used in the "name" attribute of a `<kind>` element as if it was defined by IANA.

If the `<code>` element is present in the namespace allocated to this specification, and the Access Control Policy is not natively implemented, then the code inside the element MUST be called for each DataValue found in a received StoreReq for a Kind that is defined with this access control policy. The content of the `<code>` element MUST be decoded using the base64 [RFC4648] encoding, uncompressed using gzip [RFC1952] then converted to characters using UTF-8. `<code>` elements that are not encoded using UTF-8, compressed with gzip or finally converted to the base64 format MUST be ignored.

For each call to the code, the following ECMAScript objects, properties and functions MUST be available:

configuration.instance_name: The name of the overlay, as a String object.

configuration.topology_plugin: The overlay algorithm, as a String object.

configuration.node_id_length: The length of a NodeId in bytes, as a Number object.

`configuration.kinds`: An array of kinds (with the same definition than the kind object), indexed by id and eventually by name.

`configuration.evaluate(String, String, String)`: A function that evaluates the first parameter as an XPath expression against the configuration element, and returns the result as a String object. The second parameter contains a namespace prefix and the third parameter contains a namespace.

`kind.id`: The id of the Kind associated with the entry, as a Number object.

`kind.name`: If the Kind associated with the entry is registered by IANA, contains the name as a String object. If not, this property is undefined.

`kind.data_model`: The name of the Data Model associated with the entry, as a String object.

`kind.access_control`: The name of the Access Control Policy associated with the entry, as a String object.

`kind.max_count`: The value of the max-count element in the configuration file, as a Number object.

`kind.max_size`: The value of the max-size element in the configuration file as a Number object.

`kind.max_node_multiple`: If the Access Control is MULTIPLE-NODE, contains the value of the max-node-multiple element in the configuration file, as a Number object. If not, this property is undefined.

`kind.evaluate(String, String, String)`: A function that evaluates the first parameter as an XPath expression against the kind element, and returns the result as a String object. The second parameter must contain a namespace prefix and the third parameter must contain a namespace.

`resource`: An opaque object representing the Resource-ID, as an array of bytes.

`resource.entries`: An array of arrays of entry objects, with the first array level indexed by Kind-Id and kind names, and the second level indexed by index, key or nothing, depending on the data model of the kind. This permits to retrieve all the values of all Kinds stored at the same Resource-ID than the entry currently processed.

`resource.equalsHash(Object...)`: A function that returns true if hashing the concatenation of the arguments according to the mapping function of the overlay algorithm is equal to the Resource-ID. Each argument is an array of bytes.

`entry.index`: If the Data Model is ARRAY, contains the index of the entry, as a Number object. If not, this property is undefined.

`entry.key`: If the Data Model is DICTIONARY, contains the key of the entry, as an array of bytes. If not, this property is undefined.

`entry.storage_time`: The date and time used to store the entry, as a Date object.

`entry.lifetime`: The validity for the entry in seconds, as a Number object.

`entry.exists`: Indicates if the entry value exists, as Boolean object.

`entry.value`: This property contains an opaque object that represents the whole data, as an array of bytes.

`entry.signer.user_name`: The rfc822Name stored in the certificate that was used to sign the request, as a String object.

`entry.signer.node_id`: The Node-ID stored in the certificate that was used to sign the request, as an array of bytes.

The properties SHOULD NOT be modifiable or deletable and if they are, modifying or deleting them MUST NOT modify or delete the equivalent internal values (in other words, the code cannot be used to modify the elements that will be stored).

The value returned by the code is evaluated to true or false, according to the ECMAScript rules. If the return value of one of the call to the code is evaluated to false, then the StoreReq fails, the state MUST be rolled back and an Error_Forbidden MUST be returned.

4. Security Considerations

Because the configuration document containing the ECMAScript code is under the responsibility of the same entity that will sign it, using a scripting language does not introduce any additional risk if the RELOAD implementers follow the rules in this document (no side effect when modifying the parameters, only base classes of ECMAScript implemented, etc...). It is even possible to deal with less than perfect implementations as long as they do not accept a configuration

file that is not signed correctly. One way for the signer to enforce this would be to deliberately send in a ConfigUpdate an incorrectly signed version of the configuration file and blacklist all the nodes that accepted it in a newly issued configuration file.

By permitting multiple overlay implementations to interoperate inside one overlay, RELOAD helps build overlays that are not only resistant to hardware or communication failures, but also to programmer errors. Distributing the access control policy code inside the configuration document reintroduces this single point of failure. To mitigate this problem, new access control policies should be implemented natively as soon as possible, but if all implementations uses the ECMAScript code as a blueprint for the native code, an hidden bug can be unwillingly duplicated. This is why developers should implement new access control policies from the normative text instead of looking at the code itself. To help developers do the right thing the code in the configuration document is obfuscated by compressing and encoding it as a base64 character string.

5. IANA Considerations

This section requests IANA to register the following URN in the "XML Namespaces" class of the "IETF XML Registry" in accordance with [RFC3688].

URI: urn:ietf:params:xml:ns:p2p:access-control

Registrant Contact: The IESG

XML: This specification.

6. References

6.1. Normative References

[RFC1952] Deutsch, P., "GZIP file format specification version 4.3", RFC 1952, May 1996.

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

[RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.

[I-D.ietf-p2psip-base]

Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-24 (work in progress), January 2013.

[ECMA-262]

Ecma, , "ECMAScript Language Specification 3rd Edition", December 2009.

6.2. Informative References

[I-D.ietf-p2psip-service-discovery]

Maenpaa, J. and G. Camarillo, "Service Discovery Usage for REsource LOcation And Discovery (RELOAD)", draft-ietf-p2psip-service-discovery-06 (work in progress), April 2013.

[I-D.petithuguenin-vipr-reload-usage]

Petit-Huguenin, M., Rosenberg, J., and C. Jennings, "A Usage of Resource Location and Discovery (RELOAD) for Public Switched Telephone Network (PSTN) Verification", draft-petithuguenin-vipr-reload-usage-04 (work in progress), March 2012.

[I-D.ietf-p2psip-share]

Knauf, A., Schmidt, T., Hege, G., and M. Waehlich, "A Usage for Shared Resources in RELOAD (ShaRe)", draft-ietf-p2psip-share-00 (work in progress), April 2013.

Appendix A. Examples

A.1. Standard Access Control Policies

This section shows the ECMAScript code that could be used to implement the standard Access Control Policies defined in [I-D.ietf-p2psip-base].

A.1.1. USER-MATCH

```
String.prototype['bytes'] = function () {
    var bytes = [];
    for (var i = 0; i < this.length; i++) {
        bytes[i] = this.charCodeAt(i);
    }
    return bytes;
};

return resource.equalsHash(entry.signer.user_name.bytes());
```

A.1.1.2. NODE-MATCH

```
return resource.equalsHash(entry.signer.node_id);
```

A.1.1.3. USER-NODE-MATCH

```
String.prototype['bytes'] = function () {
    var bytes = [];
    for (var i = 0; i < this.length; i++) {
        bytes[i] = this.charCodeAt(i);
    }
    return bytes;
};

var equals = function (a, b) {
    if (a.length !== b.length) return false;
    for (var i = 0; i < a.length; i++) {
        if (a[i] !== b[i]) return false;
    }
    return true;
};

return resource.equalsHash(entry.signer.user_name.bytes()
    && equals(entry.key, entry.signer.node_id));
```

A.1.1.4. NODE-MULTIPLE

```
Number.prototype['width'] = function (w) {
    var bytes = [];
    for (var i = 0; i < w; i++) {
        bytes[i] = (this >>> ((w - i - 1) * 8)) & 255;
    }
    return bytes;
};
```

```
for (var i = 0; i < kind.max_node_multiple; i++) {
    if (resource.equalsHash(entry.signer.node_id, i.width(4))) {
        return true;
    }
}
return false;
```

A.2. Service Discovery Access Control Policy NODE-ID-MATCH

[I-D.ietf-p2psip-service-discovery] defines a new Access Control Policy (NODE-ID-MATCH) that need to access the content of the entry to be written. If implemented as specified by this document, the ECMAScript code would look something like this:

```
/* Insert here the code from
   http://jsfromhell.com/classes/bignumber
*/

var toBigNumber = function (node_id) {
    var bignum = new BigNumber(0);
    for (var i = 0; i < node_id.length; i++) {
        bignum = bignum.multiply(256).add(node_id[i]);
    }
    return bignum;
};

var checkIntervals = function (node_id, level, node, factor) {
    var size = new BigNumber(2).pow(128);
    var node = toBigNumber(node_id);
    for (var f = 0; f < factor; f++) {
        var temp = size.multiply(new BigNumber(f)
            .pow(new BigNumber(level).negate()));
        var min = temp.multiply(node.add(new BigNumber(f)
            .divide(factor)));
        var max = temp.multiply(node.add(new BigNumber(f + 1)
            .divide(factor)));
        if (node.compare(min) === -1 || node.compare(max) == 1
            || node.compare(max) == 0) return false;
    }
    return true;
};

var equals = function (a, b) {
    if (a.length !== b.length) return false;
    for (var i = 0; i < a.length; i++) {
        if (a[i] !== b[i]) return false;
    }
}
```

```
    return true;
};

var level = function (value) {
    var length = value[16] * 256 + value[17];
    return value[18 + length] * 256 + value[18 + length + 1];
};

var node = function (value) {
    var length = value[16] * 256 + value[17];
    return value[18 + length + 2] * 256
        + value[18 + length + 3];
};

var namespace = function (value) {
    var length = value[16] * 256 + value[17];
    return String.fromCharCode.apply(null,
        value.slice(18, length + 18));
};

var branching_factor =
    kind.evaluate('/branching-factor',
        'redir', 'urn:ietf:params:xml:ns:p2p:redir');
return equals(entry.key, entry.signer.node_id)
    && (!entry.exists || checkIntervals(entry.key,
        level(entry.value), node(entry.value),
        branching_factor))
    && (!entry.exists
        || resource.equalsHash(namespace(entry.value),
            level(entry.value), node(entry.value)));
```

Note that the code for the `BigNumber` object was removed from this example, as the licensing terms are unclear. The code is available at [1].

A.3. VIPR Access Control Policy

[I-D.petithuguenin-vipr-reload-usage] defines a new Access Control Policy. If implemented as specified by this document, the ECMAScript code would look something like this:

```
var equals = function (a, b) {
    if (a.length !== b.length) return false;
    for (var i = 0; i < a.length; i++) {
        if (a[i] !== b[i]) return false;
    }
    return true;
};
```

```
};  
var length = configuration.node_id_length;  
return equals(entry.key.slice(0, length),  
    entry.value.slice(4, length + 4))  
    && equals(entry.key.slice(0, length), entry.signer.node_id);
```

A.4. ShaRe Access Control Policy USER-CHAIN-ACL

[I-D.ietf-p2psip-share] defines a new Access Control Policy, USER-CHAIN-ACL. If implemented as specified by this document, the ECMAScript code would look something like this:

```
var pattern = kind.evaluate('/share:pattern',  
    'share', 'urn:ietf:params:xml:ns:p2p:config-share');  
var username = entry.signer.user_name.match(/^([^\@]+)\@(.+)\$/);  
var new_pattern = new RegExp(  
    pattern.replace('$USER', username[1])  
    .replace('$DOMAIN', username[2]));  
var length = entry.value[0] * 256 + entry.value[1];  
var resource_name = String.fromCharCode.apply(null,  
    entry.value.slice(2, length + 2));  
return new_pattern.test(resource_name);\n");
```

[[Note: the code is incomplete]]

Appendix B. Release notes

This section must be removed before publication as an RFC.

B.1. Modifications between ietf-p2psip-reload-access-control and petithuguenin-p2psip-access-control-05

- o Removed inconsistency in the terminology section.
- o Updated the IANA section and added reference to RFC 3688.
- o Removed "This is probably not legal..." in the security section.
- o Renamed "access-control-code" to simply "code" as it has to be prefixed by the namespace anyway, so there is no risk of conflict.

B.2. Running Code Considerations

- o Reference Implementation and Access Control Policy script tester (<http://debian.implementers.org/testing/source/reload.tar.gz>). Marc Petit-Huguenin. Implements version -03.

B.3. TODO List

- o Finish the code for ShaRe.
- o Update the reference implementation.

Author's Address

Marc Petit-Huguenin
Impedance Mismatch

Email: petithug@acm.org

P2PSIP Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 28, 2013

A. Knauf
T C. Schmidt, Ed.
HAW Hamburg
G. Hege
daviko GmbH
M. Waehlich
link-lab & FU Berlin
February 24, 2013

A Usage for Shared Resources in RELOAD (ShaRe)
draft-ietf-p2psip-share-01

Abstract

This document defines a RELOAD Usage for managing shared write access to RELOAD Resources. Shared Resources in RELOAD (ShaRe) form a basic primitive for enabling various coordination and notification schemes among distributed peers. Access in ShaRe is controlled by a hierarchical trust delegation scheme maintained within an access list. A new USER-CHAIN-ACL access policy allows authorized peers to write a Shared Resource without owning its corresponding certificate. This specification also adds mechanisms to store Resources with a variable name which is useful whenever peer-independent rendezvous processes are required.

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 August 28, 2013.

Copyright Notice

Copyright (c) 2013 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	4
3. Shared Resources in RELOAD	5
3.1. Mechanisms for Isolating Stored Data	6
4. Access Control List Definition	7
4.1. Overview	7
4.2. Data Structure	8
5. Extension for Variable Resource Names	10
5.1. Overview	10
5.2. Data Structure	10
5.3. Overlay Configuration Document Extension	11
6. Access Control to Shared Resources	13
6.1. Granting Write Access	13
6.2. Revoking Write Access	14
6.3. Validating Write Access through an ACL	14
6.4. Operations of Storing Peers	15
6.5. Operations of Accessing Peers	15
6.6. USER-CHAIN-ACL Access Policy	15
7. ACCESS-CONTROL-LIST Kind Definition	17
8. Security Considerations	18
8.1. Resource Exhaustion	18
8.2. Malicious or Misbehaving Storing Peer	18
8.3. Privacy Issues	18
9. IANA Considerations	19
9.1. Access Control Policy	19
9.2. Data Kind-ID	19
10. Acknowledgments	20
11. References	21
11.1. Normative References	21
11.2. Informative References	21
Appendix A. Change Log	22
Authors' Addresses	24

1. Introduction

This document defines a RELOAD Usage for managing shared write access to RELOAD Resources and a mechanism to store Resources with a variable name. The Usage for Shared Resources in RELOAD (ShaRe) enables overlay users to share their exclusive write access to specific Resource/Kind pairs with others. Shared Resources form a basic primitive for enabling various coordination and notification schemes among distributed peers. Write permission is controlled by an Access Control List (ACL) Kind that maintains a chain of Authorized Peers for a particular Shared Resource. A newly defined USER-CHAIN-ACL access control policy enables shared write access in RELOAD.

The Usage for Shared Resources in RELOAD is designed for jointly coordinated group applications among distributed peers (e.g., third party registration, see [I-D.ietf-p2psip-sip] , or distributed conferencing, see[I-D.ietf-p2psip-disco]). Of particular interest are rendezvous processes, where a single identifier is linked to multiple, dynamic instances of a distributed cooperative service. Shared write access is based on a trust delegation mechanism. It transfers the authorization to write a specific Kind data by storing logical Access Control Lists. An ACL contains the ID of the Kind to be shared and contains trust delegations from one authorized to another (previously unauthorized) user.

Shared write access augments the RELOAD security model, which is based on the restriction that peers are only allowed to write resources at a small set of well defined locations (Resource-IDs) in the overlay. Using the standard access control rules in RELOAD, these locations are bound to the username or Node-ID in the peer's certificate. This document extends the base policies to enable a controlled write access for multiple users to a common Resource Id.

Additionally, this specification defines an optional mechanism to store Resources with a variable Resource Name. It enables the storage of Resources whose name complies to a specific pattern. Definition of the pattern is arbitrary, but must contain the username of the Resource creator.

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 [RFC2119].

This document uses the terminology and definitions from the RELOAD base [I-D.ietf-p2psip-base] and the peer-to-peer SIP concepts draft [I-D.ietf-p2psip-concepts]. Additionally, the following terms are used:

Shared Resource: The term Shared Resource in this document defines a RELOAD Resource with its associated Kinds, that can be written or overwritten by multiple RELOAD users following the specifications in this document.

Access Control List: The term Access Control List in this document defines a logical list of RELOAD users allowed to write a specific RELOAD Resource/Kind pair by following the specifications in this document. The list items are stored as Access Control List Kinds that map trust delegations from user A to user B, where A is allowed to write a Shared Resource and the Access Control List, while B is a user that obtains write access to specified Kinds from A.

Resource Owner: The term Resource Owner in this document defines a RELOAD peer that initially stored a Resource to be shared. The Resource Owner possesses the RELOAD certificate that grants write access to a specific Resource/Kind pair using the RELOAD certificate-based access control policies.

Authorized Peer: The term Authorized Peer in this document defines a RELOAD peer that was granted write access to a Shared Resource by permission of the Resource Owner or another Authorized Peer.

3. Shared Resources in RELOAD

A RELOAD user that owns a certificate for writing at a specific overlay location can maintain one or more RELOAD Kinds that are designated for a non-exclusive write access shared with other RELOAD users. The mechanism to share those Resource/Kind pairs with a group of users consists of two basic steps.

1. Storage of the Resource/Kind pairs to be shared.
2. Storage of an Access Control List (ACL) associated with those Kinds.

ACLs are created by the Resource Owner and contain ACL items, each delegating the permission of writing the shared Kind to a specific user called Authorized Peer. For each shared Kind data, its Resource owner stores a root item that initiates an Access Control List. Trust delegation to the Authorized Peer can include the right to further delegate the write permission, enabling a tree of trust delegations with the Resource Owner as trust anchor at its root.

The Resource/Kind pair to be shared can be any RELOAD Kind that complies to the following specifications:

Isolated Data Storage: To prevent concurrent writing from race conditions, each data item stored within a Shared Resource SHALL be exclusively maintained by the RELOAD user who created it. Hence, Usages that allow the storage of Shared Resources are REQUIRED to use either the array or dictionary data model and apply additional mechanisms for isolating data as described in Section 3.1.

Access Control Policy: To ensure write access to Shared Resource by Authorized Peers, each Usage MUST use the USER-CHAIN-ACL access policy as described in Section 6.6.

Resource Name Extension: To enable Shared Resources to be stored using a variable resource name, this document defines an optional ResourceNameExtension structure. It contains the Resource Name of the Kind data to be stored and allows any receiver of a shared data to validate whether the Resource Name hashes to the Resource-ID. The ResourceNameExtension is made optional by configuration. The ResourceNameExtension field is only present in the Kind data structure when configured in the corresponding kind-block of the overlay configuration document (for more details see Section 5.3). If the configuration allows variable resource names, a Kind using the USER-CHAIN-ACL policy MUST use the ResourceNameExtension as initial field within the Kind data

structure definition. Otherwise the Kind data structure does not contain the ResourceNameExtension structure.

3.1. Mechanisms for Isolating Stored Data

This section defines mechanisms to avoid race conditions while concurrently writing an array or dictionary of a Shared Resource.

If a dictionary is used in the Shared Resource, the dictionary key MUST be the Node-ID of the certificate that will be used to sign the stored data. Thus data access is bound to the unique ID-holder.

If the data model of the Shared Resource is an array, each Authorized Peer SHALL obtain its exclusive range of the array indices. The following algorithm will generate an array indexing scheme that avoids collisions.

1. Obtain the Node-ID of the certificate that will be used to sign the stored data
2. Take the least significant 24 bits of that Node-ID
3. Concatenate an 8 bit long short individual index value to those 24 bit of the Node-ID

The resulting 32 bits long integer MUST be used as the index for storing an array entry in a Shared Resource. The 8 bit individual index can be incremented individually for further array entries and allows for 256 distinct entries per Peer.

The mechanism to create the array index is related to the pseudo-random algorithm to generate an SSRC identifier in RTP, see Section 8.1 in [RFC3550] for calculating the probability of a collision.

4. Access Control List Definition

4.1. Overview

An Access Control List (ACL) is a (self-managed) shared resource that contains a list of `AccessControlItem` structures as defined in Section 4.2. Each entry delegates write access for a specific Kind data to a single RELOAD user. An ACL enables the RELOAD user who is authorized to write a specific Resource-ID to delegate his exclusive write access to a specific Kind to further users of the same RELOAD overlay. Each Access Control List data structure therefore carries the information about who obtains write access, the Kind-ID of the Resource to be shared, and whether delegation includes write access to the ACL itself. The latter condition grants the right to delegate write access further for the Authorized Peer. Access Control Lists are stored at the same overlay location as the Shared Resource and use the RELOAD array data model. They are initially created by the Resource Owner.

Figure 1 shows an example of an Access Control List. We omit the `res_name_ext` field to simplify illustration. The array entry at index `0x123abc001` displays the initial creation of an ACL for a Shared Resource of Kind-ID 1234 at the same Resource-ID. It represents the root item of the trust delegation tree for this shared RELOAD Kind. The root entry MUST contain the username of the Resource owner in the `"to_user"` field and can only be written by the owner of the public key certificate associated with this Resource-ID. The `allow_delegation` (ad) flag for a root ACL item is set to 1 by default. The array index is generated by using the mechanism for isolating stored data as described in Section 3.1. Hence, the most significant 24 bits of the array index (`0x123abc`) are the least significant 24 bits of the Node-ID of the Resource Owner.

The array item at index `0x123abc002` represents the first trust delegation to an Authorized Peer that is thus permitted to write to the Shared Resource of Kind-ID 1234. Additionally, the Authorized peer Alice is also granted (limited) write access to the ACL as indicated by the `allow_delegation` flag (ad) set to 1. This configuration authorizes Alice to store further trust delegations to the Shared Resource, i.e., add items to the ACL. On the contrary, index `0x456def001` illustrates trust delegation for Kind-ID 1234, in which the Authorized Peer Bob is not allowed to grant access to further peers (ad = 0). Each Authorized Peer signs its ACL items by using its own signer identity along with its own private key. This allows other peers to validate the origin of an ACL item and makes ownership transparent.

To manage Shared Resource access of multiple Kinds at a single

location, the Resource Owner can create new ACL entries that refer to another Kind-ID as shown in array entry index 0x123abc003. Note that overwriting existing items in an Access Control List that reference a different Kind-ID revokes all trust delegations in the corresponding subtree (see Section 6.2). Authorized Peers are only enabled to overwrite existing ACL item they own. The Resource Owner is allowed to overwrite any existing ACL item, but should be aware of its consequences.

Access Control List			
#Index	Array Entries		signed by
123abc001	to_user:Owner Kind:1234 ad:1	Owner	
123abc002	to_user:Alice Kind:1234 ad:1	Owner	
123abc003	to_user:Owner Kind:4321 ad:1	Owner	
123abc004	to_user:Carol Kind:4321 ad:0	Owner	
...	
456def001	to_user:Bob Kind:1234 ad:0	Alice	
...	

Figure 1: Simplified example of an Access Control including entries for two different Kind-IDs and varying delegation (ad) configurations

Implementations of ShaRe should be aware that the trust delegation in an Access Control List need not be loop free. Self-contained circular trust delegation from A to B and B to A are syntactically possible, even though not very meaningful.

4.2. Data Structure

The Kind data structure for the Access Control List is defined as follows:

```
struct {
    /* res_name_ext is optional, see documentation */
    ResourceNameExtension res_name_ext;
    opaque                to_user<0..2^16-1>;
    KindId                kind;
```

```
        Boolean          allow_delegation;  
    } AccessControlItem;
```

The AccessControlItem structure is composed of:

res_name_ext: This optional field contains the Resource Name of a ResourceNameExtension (see Section 5.2) to be used by a Shared Resource with variable resource name. This name serves the storing peer for validating, whether a variable resources name matches one of the predefined naming pattern from the configuration document for this Kind. The presence of this field is bound to a variable resource name element in the corresponding kind-block of the configuration document whose "enable" attribute is set to true (see Section 5.3). Otherwise, if the "enable" attribute is false, the res_name_ext field SHALL NOT be present in the Kind data structure.

to_user: This field contains the username of the RELOAD peer that obtains write permission to the Shared Resource.

kind: This field contains the Kind-ID of the Shared Resource.

allow_delegation: If true, this Boolean flag indicates that the Authorized Peer in the 'to_user' field is allowed to add additional entries to the ACL for the specified Kind-ID.

5. Extension for Variable Resource Names

5.1. Overview

In certain use cases such as conferencing (c.f., [I-D.ietf-p2psip-disco]) it is desirable to increase the flexibility of a peer in using Resource Names beyond those defined by the username or Node-ID fields in its certificate. For this purpose, this document presents the concept for variable Resources Names that enables providers of RELOAD instances to define relaxed naming schemes for overlay Resources.

Each RELOAD node uses a certificate to identify itself using its user name (or Node-ID) while storing data under a specific Resource-ID. The specifications in this document scheme adhere to this paradigm, but enable a RELOAD peer to store values of Resource Names that are derived from the username in its certificate. This is done by using a Resource Name with a variable substring that still matches the user name in the certificate using a pattern defined in the overlay configuration document. Thus despite being variable, an allowable Resource Name remains tied to the Owner's certificate. A sample pattern might be formed as follows.

Example Pattern:

```
.*-conf-\'$USER@$DOMAIN
```

When defining the pattern, care must be taken to avoid conflicts arising from two usernames of which one is a substring of the other. In such cases, the holder of the shorter name could threaten to block the resources of the longer-named peer by choosing the variable part of a Resource Name to contain the entire longer username. This problem can easily be mitigated by delimiting the variable part of the pattern from the username part by some fixed string, that by convention is not part of a username (e.g., the "-conf-" in the above Example).

5.2. Data Structure

This section defines the optional ResourceNameExtension structure for every Kind that uses the USER-CHAIN-ACL access control policy.

```
enum { pattern (1),  
      (255)} ResourceNameType;  
  
struct {  
    ResourceNameType type;  
    uint16          length;  
    select(type) {  
        case pattern:  
            opaque resource_name<0..2^16-1>;  
  
        /* Types can be extended */  
    }  
} ResourceNameExtension
```

The content of the ResourceNameExtension consist of

length: This field contains the length of the remaining data structure. It is only used to allow for further extensions to this data structure.

The content of the rest of the data structure depends of the ResourceNameType. Currently, the only defined type is "pattern".

If the type is "pattern", then the following data structure contains an opaque <0..2^16-1> field containing the Resource Name of the Kind being stored. The type "pattern" further indicates that the Resource Name MUST match to one of the variable resource name pattern defined for this Kind in the configuration document.

The ResourceNameType enum and the ResourceNameExtension structure can be extended by further Usages to define other naming schemes.

5.3. Overlay Configuration Document Extension

This section extends the overlay configuration document by defining new elements for patterns relating resource names to user names. Configurations in this overlay document MUST adhere in syntax and semantic of names as defined by the context of use. For example, AOR syntax restrictions apply when using P2PSIP[I-D.ietf-p2psip-sip] , while a more general naming is feasible in plain RELOAD.

The <variable-resource-names> element serves as a container for one or multiple <pattern> sub-elements. It is an additional parameter within the kind block and has a boolean "enable" attribute that indicates, if true, that the overlay provider allows variable resource names for this Kind. The default value of the "enable" attribute is "false". In the absence of a <variable-resource-names> element for a Kind using the USER-CHAIN-ACL access policy (see

Section 6.6), implementors SHOULD assume this default value.

A <pattern> element MUST be present if the "enabled" attribute of its parent element is set to true. Each <pattern> element defines a pattern for constructing extended resource names for a single Kind. It is of type xsd:string and interpreted as a regular expression according to "POSIX Extended Regular Expression" (see the specifications in [IEEE-Posix]). In this regular expression, \$USER and \$DOMAIN are used as variables for the corresponding parts of the string in the certificate user name field (with \$USER preceding and \$DOMAIN succeeding the '@'). Both variables MUST be present in any given pattern definition. If no pattern is defined for a Kind or the "enabled" attribute is false, allowable Resource Names are restricted to the username of the signer for Shared Resource.

The Relax NG Grammar for the Variable Resource Names Extension reads:

```
<!-- VARIABLE RESOURCE URN SUB-NAMESPACE -->

namespace share = "urn:ietf:params:xml:ns:p2p:config-base:share"

<!-- VARIABLE RESOURCE NAMES ELEMENT -->

kind-parameter &= element share:variable-resource-names {
    attribute enable { xsd:boolean }
    <!-- PATTERN ELEMENT -->
    element pattern { xsd:string }*
}?
```

6. Access Control to Shared Resources

6.1. Granting Write Access

Write access to a Kind that is intended to be shared with other RELOAD users can solely be issued by the Resource Owner. A Resource Owner can share RELOAD Kinds by using the following procedure.

- o The Resource Owner stores an ACL root item at the Resource-ID of the Shared Resource. The root item contains the resource name extension field (see Section 5.2), the username of the Resource Owner and Kind-ID of the Shared Resource. The allow_delegation flag is set to 1. The index of array data structure MUST be generated as described in Section 3.1
- o Further ACL items for this Kind-ID stored by the Resource Owner will delegate write access to Authorized Peers. These ACL items contain the same resource name extension field, the username of the Authorized Peer and the Kind-Id of the Shared Resource. Optionally, the Resource Owner sets the "ad" to 1 (the default equals 0) to enable the Authorized Peer to further delegate write access. Each succeeding ACL item created by the Resource Owner can be stored in the numerical order of the array index starting with the index of the root item incremented by one.

An Authorized Peer with delegation allowance ("ad"=1) can extend the access to an existing Shared Resource as follows.

- o An Authorized Peer can store additional ACL items at the Resource-ID of the Shared Resource. These ACL items contain the resource name extension field, the username of the newly Authorized Peer, and the Kind-Id of the Shared Resource. Optionally, the "ad" flag is set to 1 for allowing the Authorized Peer to further delegate write access. The array index MUST be generated as described in Section 3.1. Each succeeding ACL item can be stored in the numerical order of the array index.

A store request by an Authorized Peer that attempts to overwrite any ACL item signed by another Peer is unauthorized and causes an Error_Forbidden response from the Storing Peer. Such access conflicts could be caused by an array index collision. However, the probability of a collision of two or more identical array indices will be negligibly low using the mechanism for isolating stored data (see Section 3.1)

6.2. Revoking Write Access

Write permissions are revoked by storing a non-existent value [I-D.ietf-p2psip-base] at the corresponding item of the Access Control List. Revoking a permission automatically invalidates all delegations performed by that user including all subsequent delegations. This allows to invalidate entire subtrees of the delegations tree with only a single operation. Overwriting the root item with a non-existent value of an Access List invalidates the entire delegations tree.

An existing ACL item MUST only be overwritten by the user who initially stored the corresponding entry, or by the Resource Owner that is allowed to overwrite all ACL items for revoking write access.

6.3. Validating Write Access through an ACL

Access Control Lists are used to transparently validate authorization of peers for writing a data value at a Shared Resource. Thereby it is assumed that the validating peer is in possession of the complete and most recent ACL for a specific Resource/Kind pair. The corresponding procedure consists of recursively traversing the trust delegation tree and proceeds as follows.

1. Obtain the username of the certificate used for signing the data stored at the Shared Resource.
2. Validate that an item of the corresponding ACL (i.e., for this Resource/Kind pair) contains a "to_user" field whose value equals the username obtained in step 1. If the Shared Resource under examination is an Access Control List Kind, further validate if the "ad" flag is set to 1.
3. Select the username of the certificate that was used to sign the ACL item obtained in step 2.
4. Validate that an item of the corresponding ACL contains a "to_user" field whose value equals the username obtained in step 3. Additionally validate that the "ad" flag is set to 1.
5. Repeat steps 3 and 4 until the "to_user" value is equal to the username of the signer of the previously selected ACL item. This final ACL item is expected to be the root item of this ACL which SHALL be further validated by verifying that the root item was signed by the owner of the ACL Resource.

The trust delegation chain is valid if and only if all verification steps succeed. In this case, the creator of the data value of the

Shared Resource is an Authorized Peer.

Note that the ACL validation procedure can be omitted whenever the creator of data at a Shared Resource is the Resource Owner itself. The latter can be verified by its public key certificate as defined in Section 6.6.

6.4. Operations of Storing Peers

Storing peers at which Shared Resource and ACL are physically stored, are responsible for controlling storage attempts to a Shared Resource and its corresponding Access Control List. To assert the USER-CHAIN-ACL access policy (see Section 6.6), a storing peer MUST perform the access validation procedure described in Section 6.3 on any incoming store request using the most recent Access Control List for every Kind that uses the USER-CHAIN-ACL policy. It SHALL further ensure that only the Resource Owner stores new ACL root items for Shared Resources.

6.5. Operations of Accessing Peers

Accessing peers, i.e., peers that fetch a Shared Resource, MAY validate that the originator of a Shared Resource was authorized to store data at this Resource-ID by processing the corresponding ACL. To enable an accessing peer to perform the access validation procedure described in Section 6.3, it first needs to obtain the most recent Access Control List in the following way.

1. Send a Stat request to the Resource-ID of the Shared Resource to obtain all array indexes of stored ACL Kinds.
2. Fetch all indexes of existing ACL items at this Resource-ID by using the array ranges retrieved in the Stat request answer.

Peers can cache previously fetched Access Control Lists up to the maximum lifetime of an individual item. Since stored values could have been modified or invalidated prior to their expiration, an accessing peer SHOULD use a Stat request to check for updates prior to using the data cache.

6.6. USER-CHAIN-ACL Access Policy

This document specifies an additional access control policy to the RELOAD base draft [I-D.ietf-p2psip-base]. The USER-CHAIN-ACL policy allows Authorized Peers to write a Shared Resource, even though they do not own the corresponding certificate. Additionally, the USER-CHAIN-ACL allows the storage of Kinds with a variable resource name that are following one of the specified naming pattern. Hence, on an

inbound store request on a Kind that uses the USER-CHAIN-ACL access policy, the following rules MUST be applied:

In the USER-CHAIN-ACL policy, a given value MUST be written or overwritten, if either one of USER-MATCH or USER-NODE-MATCH (mandatory if the data model is dictionary) access policies of the base document [I-D.ietf-p2psip-base] applies.

Otherwise, the value MUST be written if the certificate of the signer contains a username that matches to one of the variable resource name pattern (c.f. Section 5) specified in the configuration document and, additionally, the hashed Resource Name matches the Resource-ID. The Resource Name of the Kind to be stored MUST be taken from the mandatory ResourceNameExtension field in the corresponding Kind data structure.

Otherwise, the value MUST be written if the ACL validation procedure described in Section 6.3 has been successfully applied.

7. ACCESS-CONTROL-LIST Kind Definition

This section defines the ACCESS-CONTROL-LIST Kind previously described in this document.

Name: ACCESS-CONTROL-LIST

Kind IDs: The Resource Name for ACCESS-CONTROL-LIST Kind-ID is the Resource Name of the Kind that will be shared by using the ACCESS-CONTROL-LIST Kind.

Data Model: The data model for the ACCESS-CONTROL-LIST Kind-ID is array. The array indexes are formed by using the mechanism for isolated stored data as described in Section 3.1

Access Control: USER-CHAIN-ACL (see Section 6.6)

8. Security Considerations

In this section we discuss security issues that are relevant to the usage of shared resources in RELOAD.

8.1. Resource Exhaustion

Joining a RELOAD overlay inherently poses a certain resource load on a peer, because it has to store and forward data for other peers. In common RELOAD semantics, each Resource ID and thus position in the overlay may only be written by a limited set of peers - often even only a single peer, which limits this burden. In the case of Shared Resources, a single resource may be written by multiple peers, who may even write an arbitrary number of entries (e.g., delegations in the ACL). This leads to an enhanced use of resources at individual overlay nodes. The problem of resource exhaustion can easily be mitigated for Usages based on the ShaRe-Usage by imposing restrictions on size, i.e., <max-size> element for a certain Kind in the configuration document.

8.2. Malicious or Misbehaving Storing Peer

The RELOAD overlay is designed to operate despite the presence of a small set of misbehaving peers. This is not different for Shared Resources since a small set of malicious peers does not disrupt the functionality of the overlay in general, but may have implications for the peers needing to store or access information at the specific locations in the ID space controlled by a malicious peer. A storing peer could withhold stored data which results in a denial of service to the group using the specific resource. But it could not return forged data, since the validity of any stored data can be independently verified using the attached signatures.

8.3. Privacy Issues

All data stored in the Shared Resource is publicly readable, thus applications requiring privacy need to encrypt the data. The ACL needs to be stored unencrypted, thus the list members of a group using a Shared Resource will always be publicly visible.

9. IANA Considerations

9.1. Access Control Policy

IANA shall register the following entry in the "RELOAD Access Control Policies" Registry (cf., [I-D.ietf-p2psip-base]) to represent the USER-CHAIN-ACL Access Control Policy, as described in Section 6.6. [NOTE TO IANA/RFC-EDITOR: Please replace RFC-AAAA with the RFC number for this specification in the following list.]

Kind	RFC
USER-CHAIN-ACL	RFC-AAAA

9.2. Data Kind-ID

IANA shall register the following code point in the "RELOAD Data Kind-ID" Registry (cf., [I-D.ietf-p2psip-base]) to represent the ShaRe ACCESS-CONTROL-LIST kind, as described in Section 7. [NOTE TO IANA/RFC-EDITOR: Please replace RFC-AAAA with the RFC number for this specification in the following list.]

Kind	Kind-ID	RFC
ACCESS-CONTROL-LIST	17	RFC-AAAA

10. Acknowledgments

This work was stimulated by fruitful discussions in the P2PSIP working group and SAM research group. We would like to thank all active members for constructive thoughts and feedback. In particular, the authors would like to thank (in alphabetical order) Lothar Grimm, Cullen Jennings, Peter Musgrave, Joerg Ott, Marc Petit-Huguenin, Peter Pogrzeba, and Jan Seedorf. This work was partly funded by the German Federal Ministry of Education and Research, projects HAMcast, Mindstone, and SAFEST.

11. References

11.1. Normative References

- [I-D.ietf-p2psip-base]
Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-25 (work in progress), February 2013.
- [IEEE-Posix]
"IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities (Vol. 1)", IEEE Std 1003.2-1992, ISBN 1-55937-255-9, January 1993.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.

11.2. Informative References

- [I-D.ietf-p2psip-concepts]
Bryan, D., Willis, D., Shim, E., Matthews, P., and S. Dawkins, "Concepts and Terminology for Peer to Peer SIP", draft-ietf-p2psip-concepts-04 (work in progress), October 2011.
- [I-D.ietf-p2psip-disco]
Knauf, A., Schmidt, T., Hege, G., and M. Waehlich, "A RELOAD Usage for Distributed Conference Control (DisCo)", draft-ietf-p2psip-disco-00 (work in progress), October 2012.
- [I-D.ietf-p2psip-sip]
Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., Schulzrinne, H., and T. Schmidt, "A SIP Usage for RELOAD", draft-ietf-p2psip-sip-08 (work in progress), December 2012.
- [RFC3550] Schulzrinne, H., Casner, S., Frederick, R., and V. Jacobson, "RTP: A Transport Protocol for Real-Time Applications", STD 64, RFC 3550, July 2003.

Appendix A. Change Log

The following changes have been made from version draft-ietf-p2psio-share-00:

1. Clarified use of identities in ACLs
2. Specified use of Posix regular expressions in configuration document
3. Added IANA considerations
4. Editorial improvements
5. Updated References

The following changes have been made from version draft-knauf-p2psip-share-02:

1. Editorial improvements
2. Updated References

The following changes have been made from version draft-knauf-p2psip-share-01:

1. Simplified the ACL data structure in response to WG feedback
2. Added ResourceNameExtension data structure to simplify the use of variable resource names
3. Restructured document
4. Many editorial improvements

The following changes have been made from version draft-knauf-p2psip-share-00:

1. Integrated the USER-PATTERN-MATCH access policy into USER-CHAIN-ACL
2. Access Control List Kind uses USER-CHAIN-ACL exclusively
3. Resources to be shared use USER-CHAIN-ACL exclusively
4. More precise specification of mandatory User_name and Resource_name fields for Shared Resources

5. Added mechanism for isolating stored data to prevent race conditions while concurrent storing
6. XML Extension for variable resource names uses its own namespace
7. Many editorial improvements

Authors' Addresses

Alexander Knauf
HAW Hamburg
Berliner Tor 7
Hamburg D-20099
Germany

Phone: +4940428758067
Email: alexanderknauf@gmail.com
URI:

Thomas C. Schmidt
HAW Hamburg
Berliner Tor 7
Hamburg D-20099
Germany

Email: schmidt@informatik.haw-hamburg.de
URI: <http://inet.cpt.haw-hamburg.de/members/schmidt>

Gabriel Hege
daviko GmbH
Am Borsigturm 50
Berlin D-13507
Germany

Phone: +493043004344
Email: hege@daviko.com
URI:

Matthias Waehlich
link-lab & FU Berlin
Hoenower Str. 35
Berlin D-10318
Germany

Email: mw@link-lab.net
URI: <http://www.inf.fu-berlin.de/~waehl>

P2PSIP
Internet-Draft
Intended status: Standards Track
Expires: August 29, 2013

C. Jennings
Cisco
B. Lowekamp
Skype
E. Rescorla
RTFM, Inc.
S. Baset
H. Schulzrinne
Columbia University
T C. Schmidt, Ed.
HAW Hamburg
February 25, 2013

A SIP Usage for RELOAD
draft-ietf-p2psip-sip-09

Abstract

This document defines a SIP Usage for REsource LOcation And Discovery (RELOAD). The SIP Usage provides the functionality of a SIP proxy or registrar in a fully-distributed system and includes a lookup service for Address of Records (AORs) stored in the overlay. It also defines Globally Routable User Agent Uris (GRUUs) that allow the registrations to map an AOR to a specific node reachable through the overlay. After such initial contact of a peer, the AppAttach method is used to establish a direct connection between nodes through which SIP messages are exchanged.

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 August 29, 2013.

Copyright Notice

Copyright (c) 2013 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	4
2. Terminology	6
3. Registering AORs in the Overlay	6
3.1. Overview	6
3.2. Data Structure	7
3.3. Access Control	9
3.4. Overlay Configuration Document Extension	9
4. Looking up an AOR	11
4.1. Finding a Route to an AOR	11
4.2. Resolving an AOR	11
5. Forming a Direct Connection	11
6. Using GRUUs	12
7. SIP-REGISTRATION Kind Definition	13
8. Security Considerations	13
8.1. RELOAD-Specific Issues	13
8.2. SIP-Specific Issues	14
8.2.1. Fork Explosion	14
8.2.2. Malicious Retargeting	14
8.2.3. Misuse of AORs	14
8.2.4. Privacy Issues	14
9. IANA Considerations	14
9.1. Data Kind-ID	15
9.2. XML Name Space Registration	15
10. Acknowledgments	15
11. References	15
11.1. Normative References	15
11.2. Informative References	16
Appendix A. Third Party Registration	17
Appendix B. Change Log	17
B.1. Changes since draft-ietf-p2psip-sip-08	17
B.2. Changes since draft-ietf-p2psip-sip-07	17
B.3. Changes since draft-ietf-p2psip-sip-06	17
Authors' Addresses	18

1. Introduction

The REsource LOcation And Discovery (RELOAD) [I-D.ietf-p2psip-base] specifies a peer-to-peer (P2P) signaling protocol for the general use on the Internet. This document defines a SIP Usage of RELOAD that allows SIP [RFC3261] user agents (UAs) to establish peer-to-peer SIP (or SIPS) sessions without the requirement for permanent proxy or registration servers, e.g., a fully distributed telephony service. In such a network, the RELOAD overlay itself performs the registration and rendezvous functions ordinarily associated with such servers.

The SIP Usage involves two basic functions.

Registration: SIP UAs can use the RELOAD data storage functionality to store a mapping from their address-of-record (AOR) to their Node-ID in the overlay, and to retrieve the Node-ID of other UAs.

Rendezvous: Once a SIP UA has identified the Node-ID for an AOR it wishes to call, it can use the RELOAD message routing system to set up a direct connection for exchanging SIP messages.

Mappings are stored in the SipRegistration Resource Record defined in this document. All operations required to perform a SIP registration or rendezvous are standard RELOAD protocol methods.

For example, Bob registers his AOR, "bob@dht.example.com", for his Node-ID "1234". When Alice wants to call Bob, she queries the overlay for "bob@dht.example.com" and receives Node-ID 1234 in return. She then uses the overlay routing to establish a direct connection with Bob and can directly transmit a standard SIP INVITE. In detail, this works along the following steps.

1. Bob, operating Node-ID 1234, stores a mapping from his AOR to his Node-ID in the overlay by applying a Store request for "bob@dht.example.com -> 1234".
2. Alice, operating Node-ID 5678, decides to call Bob. She retrieves Node-ID "1234" by performing a Fetch request on "bob@dht.example.com".
3. Alice uses the overlay to route an AppAttach message to Bob's peer (ID 1234). Bob responds with his own AppAttach and they set up a direct connection, as shown in Figure 1. Note that mutual ICE checks are invoked automatically from AppAttach message exchange.

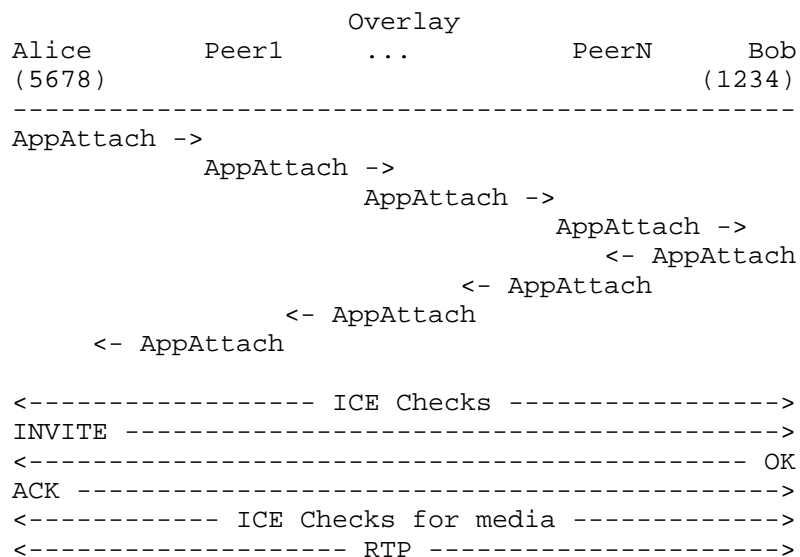


Figure 1: Connection setup in P2P SIP using the RELOAD overlay

It is important to note that here the only role of RELOAD is to set up the direct SIP connection between Alice and Bob. As soon as the ICE checks complete and the connection is established, ordinary SIP or SIPS is used. In particular, the establishment of the media channel for a phone call happens via the usual SIP mechanisms, and RELOAD is not involved. Media never traverses the overlay. After the successful exchange of SIP messages, call peers run ICE connectivity checks for media.

In addition to mappings from AORs to Node-IDs, the SIP Usage also allows mappings from AORs to other AORs. This enables an indirection useful for call forwarding. For instance, if Bob wants his phone calls temporarily forwarded to Charlie, he can store the mapping "bob@dht.example.com -> charlie@dht.example.com". When Alice wants to call Bob, she retrieves this mapping and can then fetch Charlie's AOR to retrieve his Node-ID. These mechanisms are described in Section 3.

Alternatively, Globally Routable User Agent URIs (GRUUs) can be used for directly accessing peers. They are handled via a separate mechanism, as described in Section 6.

The SIP Usage for RELOAD addresses a fully distributed deployment of session-based services among overlay peers. Two opposite scenarios of deploying P2P SIP services are in the focus of this document: A

highly regulated environment of a "single provider" that admits parties using AORs with domains from controlled namespace(s), only, and an open, multi-party infrastructure that liberally allows a registration and rendezvous for various or any domain namespace. It is noteworthy in this context that - in contrast to regular SIP - domain names play no role in routing to a proxy server. Once connectivity to an overlay is given, any name registration can be technically processed.

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].

We use the terminology and definitions from Concepts and Terminology for Peer to Peer SIP [I-D.ietf-p2psip-concepts] and the RELOAD Base Protocol [I-D.ietf-p2psip-base] extensively in this document.

In addition, term definitions from SIP [RFC3261] apply to this memo. The term AOR is the SIP "Address of Record" used to identify a user in SIP. For example, `alice@example.com` could be the AOR for Alice. For the purposes of this specification, an AOR is considered not to include the scheme (e.g `sip:`) as the AOR needs to match the `rfc822Name` in the X509v3 certificates. It is worth noting that SIP and SIPS are distinguished in P2PSIP by the Application-ID.

3. Registering AORs in the Overlay

3.1. Overview

In ordinary SIP, a UA registers its AOR and location with a registrar. In RELOAD, this registrar function is provided by the overlay as a whole. To register its location, a RELOAD peer stores a `SipRegistration` Resource Record under its own AOR using the `SIP-REGISTRATION` Kind, which is formally defined in Section 7. A RELOAD overlay MAY restrict the storage of AORs. Namespaces (i.e., the right hand side of the AOR) that are supported for registration and lookup can be configured for each RELOAD deployment as described in Section 3.4.

As a simple example, consider Alice with AOR `"alice@dht.example.org"` at Node-ID `"1234"`. She might store the mapping `"alice@dht.example.org -> 1234"` telling anyone who wants to call her to contact node `"1234"`.

RELOAD peers MAY store two kinds of SIP mappings,

- o from an AOR to a destination list (a single Node-ID is just a trivial destination list), or
- o from an AOR to another AOR.

The meaning of the first kind of mapping is "in order to contact me, form a connection with this peer." The meaning of the second kind of mapping is "in order to contact me, dereference this AOR". The latter allows for forwarding. For instance, if Alice wants her calls to be forwarded to her secretary, Sam, she might insert the following mapping "alice@dht.example.org -> sam@dht.example.org".

3.2. Data Structure

This section defines the SipRegistration Resource Record as follows:

```
enum { sip_registration_uri(1), sip_registration_route(2),
      (255) } SipRegistrationType;

select (SipRegistration.type) {
  case sip_registration_uri:
    opaque          uri<0..2^16-1>;

  case sip_registration_route:
    opaque          contact_prefs<0..2^16-1>;
    Destination     destination_list<0..2^16-1>;

  /* This type can be extended */
} SipRegistrationData;

struct {
  SipRegistrationType  type;
  uint16              length;
  SipRegistrationData data;
} SipRegistration;
```

The contents of the SipRegistration Resource Record are:

type
the type of the registration

length
the length of the rest of the PDU

data
the registration data

- o If the registration is of type "sip_registration_uri", then the contents are an opaque string containing the URI.
- o If the registration is of type "sip_registration_route", then the contents are an opaque string containing the callee's contact preferences and a destination list for the peer.

The encoding of contact_prefs - the callee's contact preferences - follows the media feature set syntax of [RFC2533] (see also [RFC2738]). As an example, a voicemail server that is a UA that supports audio and video media types and is not mobile would carry the following feature set description in its contact_prefs attribute:

```
(& (sip.audio=TRUE)
  (sip.video=TRUE)
  (sip.actor=msg-taker)
  (sip.automata=TRUE)
  (sip.mobility=fixed)
  (| (sip.methods=INVITE) (sip.methods=BYE) (sip.methods=OPTIONS)
    (sip.methods=ACK) (sip.methods=CANCEL)))
```

A callee MAY indicate that it prefers contact via a particular SIP scheme - SIP or SIPS - by using one of the following contact_prefs attribute:

```
(sip.schemes=SIP)
(sip.schemes=SIPS)
```

RELOAD explicitly supports multiple registrations for a single AOR. The registrations are stored in a Dictionary with Node-IDs as the dictionary keys. Consider, for instance, the case where Alice has two peers:

- o her desk phone (1234)
- o her cell phone (5678)

Alice might store the following in the overlay at resource "alice@dht.example.com".

- o A SipRegistration of type "sip_registration_route" with dictionary key "1234" and value "1234".
- o A SipRegistration of type "sip_registration_route" with dictionary key "5678" and value "5678".

Note that this structure explicitly allows one Node-ID to forward to another Node-ID. For instance, Alice could set calls to her desk phone to ring at her cell phone by storing a SipRegistration of type "sip_registration_route" with dictionary key "1234" and value "5678".

3.3. Access Control

In order to prevent hijacking or other misuse, registrations are subject to access control rules. Two kinds of restrictions apply:

- o A Store is permitted only for AORs with domain names that fall into the namespaces supported by the RELOAD overlay instance.
- o Storing requests are performed according to the USER-NODE-MATCH access control policy of RELOAD.

Before issuing a Store request to the overlay, any peer SHOULD verify that the AOR of the request is a valid Resource Name with respect to its domain name and the namespaces defined in the overlay configuration document (see Section 3.4).

Before a Store is permitted, the storing peer MUST check that:

- o The AOR of the request is a valid Resource Name with respect to the namespaces defined in the overlay configuration document.
- o The certificate contains a username that is a SIP AOR which hashes to the Resource-ID it is being stored at.
- o The certificate contains a Node-ID that is the same as the dictionary key it is being stored at.

Note that these rules permit Alice to forward calls to Bob without his permission. However, they do not permit Alice to forward Bob's calls to her. See Section 8.2.2 for additional descriptions.

3.4. Overlay Configuration Document Extension

The use of a SIP-enabled overlay MAY be restricted to users with AORs from specific domains. When deploying an overlay service, providers can decide about these use case scenarios by defining a set of namespaces for admissible domain names. This section extends the overlay configuration document by defining new elements for patterns that describe a corresponding domain name syntax.

A RELOAD overlay can be configured to accept store requests for any

AOR, or to apply domain name restrictions. For the latter, an enumeration of admissible domain names including wildcarded name patterns of the following form MAY be configured.

Example of Domain Patterns:

```
dht\.example\.com
.*\.my\.name
```

In this example, any AOR will be accepted that is either of the form <user>@dht.example.com, or ends with the domain "my.name". When restrictions apply and in the absence of domain patterns, the default behavior is to accept only AORs that exactly match the domain name of the overlay. Otherwise, i.e., when restrictions are not configured (attribute enable not set), the default behavior is to accept any AOR. In the absence of a <domain-restrictions> element, implementors SHOULD assume this default value. Encoding of the domain name complies to the restricted ASCII character set without character escaping as defined in Section 19.1 of [RFC3261].

The <domain-restrictions> element serves as a container for zero to multiple <pattern> sub-elements. A <pattern> element MAY be present if the "enable" attribute of its parent element is set to true. Each <pattern> element defines a pattern for constructing admissible resource names. It is of type xsd:string and interpreted as a regular expression according to "POSIX Extended Regular Expression" (see the specifications in [IEEE-Posix]).

The Relax NG Grammar for the AOR Domain Restriction reads:

```
<!-- AOR DOMAIN RESTRICTION URN SUB-NAMESPACE -->

namespace sip = "urn:ietf:params:xml:ns:p2p:config-base:sip"

<!-- AOR DOMAIN RESTRICTION ELEMENT -->

Kind-parameter &= element sip:domain-restriction {

    attribute enable { xsd:boolean }

    <!-- PATTERN ELEMENT -->

    element pattern { xsd:string }*

}?


```

4. Looking up an AOR

4.1. Finding a Route to an AOR

A RELOAD user, member of an overlay, who wishes to call another user with given AOR SHALL proceed in the following way.

AOR is GRUU? If the AOR is a GRUU for this overlay, the callee can be contacted directly as described in Section 6.

AOR domain is hosted in overlay? If the domain part of the AOR matches a domain pattern configured in the overlay, the user can continue to resolve the AOR in this overlay. The user MAY choose to query the DNS service records to search for additional support of this domain name.

AOR domain not supported by overlay? If the domain part of the AOR is not supported in the current overlay, the user SHOULD query the DNS (or other discovery services at hand) to search for an alternative overlay that services the AOR under request. Alternatively, standard SIP procedures for contacting the callee SHOULD be used.

AOR inaccessible? If all of the above contact attempts fail, the call fails.

The procedures described above likewise apply when nodes are simultaneously connected to several overlays.

4.2. Resolving an AOR

A RELOAD user that has discovered a route to an AOR in the current overlay SHALL execute the following steps.

1. Perform a Fetch for Kind SIP-REGISTRATION at the Resource-ID corresponding to the AOR. This Fetch SHOULD NOT indicate any dictionary keys, so that it will fetch all the stored values.
2. If any of the results of the Fetch are non-GRUU AORs, then repeat step 1 for that AOR.
3. Once only GRUUs and destination lists remain, the peer removes duplicate destination lists and GRUUs from the list and initiates SIP or SIPS connections to the appropriate peers as described in the following sections. If there are also external AORs, the peer follows the appropriate procedure for contacting them as well.

5. Forming a Direct Connection

Once the peer has translated the AOR into a set of destination lists, it then uses the overlay to route AppAttach messages to each of those

peers. The "application" field MUST be either 5060 to indicate SIP or 5061 for using SIPS. If certificate-based authentication is in use, the responding peer MUST present a certificate with a Node-ID matching the terminal entry in the route list. Note that it is possible that the peers already have a RELOAD connection mutually established. This MUST NOT be used for SIP messages unless it is a SIP connection. A previously established SIP connection MAY be used for a new call.

Once the AppAttach succeeds, the peer sends plain or (D)TLS encrypted SIP messages over the connection as in normal SIP. A caller MAY choose to contact the callee using SIP or secure SIPS, but SHOULD follow a preference indicated by the callee in its `contact_prefs` attribute (see Section 3.2). A callee MAY choose to listen on both SIP and SIPS ports and accept calls from either SIP scheme, or select a single one. However, a callee that decides to accept SIPS calls, only, SHOULD indicate its choice by setting the corresponding attribute in its `contact_prefs`.

6. Using GRUUs

Globally Routable User Agent Uris (GRUUs) [RFC5627] have been designed to allow direct routing without the indirection of a SIP proxy function. The concept is transferred to RELOAD overlays as follows. GRUUs in RELOAD are constructed by embedding a base64-encoded destination list in the `gr` URI parameter of the GRUU. The base64 encoding is done with the alphabet specified in table 1 of [RFC4648] with the exception that `~` is used in place of `=`.

Example of a RELOAD GRUU:

```
alice@example.com;gr=MDEyMzQ1Njc4OTAxMjM0NTY3ODk~
```

GRUUs do not require to store data in the Overlay Instance. Rather when a peer needs to route a message to a GRUU in the same P2P overlay, it simply uses the destination list and connects to that peer. Because a GRUU contains a destination list, it MAY have the same contents as a destination list stored elsewhere in the resource dictionary.

Anonymous GRUUs [RFC5767] are constructed analogously, but require either that the enrollment server issues a different Node-ID for each anonymous GRUU required, or that a destination list be used that includes a peer that compresses the destination list to stop the Node-ID from being revealed.

7. SIP-REGISTRATION Kind Definition

This section defines the SIP-REGISTRATION Kind.

Name SIP-REGISTRATION

Kind IDs The Resource Name for the SIP-REGISTRATION Kind-ID is the AOR of the user. The data stored is a SipRegistration, which can contain either another URI or a destination list to the peer which is acting for the user.

Data Model The data model for the SIP-REGISTRATION Kind-ID is dictionary. The dictionary key is the Node-ID of the storing peer. This allows each peer (presumably corresponding to a single device) to store a single route mapping.

Access Control USER-NODE-MATCH. Note that this matches the SIP AOR against the rfc822Name in the X509v3 certificate. The rfc822Name does not include the scheme so that the "sip:" prefix needs to be removed from the SIP AOR before matching.

Data stored under the SIP-REGISTRATION Kind is of type SipRegistration. This comes in two varieties:

`sip_registration_uri`
a URI which the user can be reached at.

`sip_registration_route`
a destination list which can be used to reach the user's peer.

8. Security Considerations

8.1. RELOAD-Specific Issues

This Usage for RELOAD does not define new protocol elements or operations. Hence no new threats arrive from message exchanges in RELOAD.

This document introduces an AOR domain restriction function that must be surveyed by the storing peer. A misconfigured or malicious peer could cause frequent rejects of illegitimate storing requests. However, domain name control relies on a lightweight pattern matching and can be processed prior to validating certificates. Hence no extra burden is introduced for RELOAD peers beyond loads already present in the base protocol.

8.2. SIP-Specific Issues

8.2.1. Fork Explosion

Because SIP includes a forking capability (the ability to retarget to multiple recipients), fork bombs are a potential DoS concern. However, in the SIP usage of RELOAD, fork bombs are a much lower concern than in a conventional SIP Proxy infrastructure, because the calling party is involved in each retargeting event. It can therefore directly measure the number of forks and throttle at some reasonable number.

8.2.2. Malicious Retargeting

Another potential DoS attack is for the owner of an attractive AOR to retarget all calls to some victim. This attack is common to SIP and difficult to ameliorate without requiring the target of a SIP registration to authorize all stores. The overhead of that requirement would be excessive and in addition there are good use cases for retargeting to a peer without its explicit cooperation.

8.2.3. Misuse of AORs

A RELOAD overlay and enrollment service that liberally accept registrations for AORs of domain names unrelated to the overlay instance and without further justification, eventually store presence state for misused AORs. An attacker could hijack names, register a bogus presence and attract calls dedicated to a victim that resides within or outside the Overlay Instance.

A hijacking of AORs can be mitigated by restricting the name spaces admissible in the Overlay Instance, or by additional verification actions of the enrollment service. To prevent an (exclusive) routing to a bogus registration, a caller can in addition query the DNS (or other discovery services at hand) to search for an alternative presence of the callee in another overlay or a normal SIP infrastructure.

8.2.4. Privacy Issues

All RELOAD SIP registration data is public. Methods of providing location and identity privacy are still being studied. Location privacy can be gained from using anonymous GRUUs.

9. IANA Considerations

9.1. Data Kind-ID

IANA shall register the following code point in the "RELOAD Data Kind-ID" Registry (cf., [I-D.ietf-p2psip-base]) to represent the SIP-REGISTRATION Kind, as described in Section 7. [NOTE TO IANA/ RFC-EDITOR: Please replace RFC-AAAA with the RFC number for this specification in the following list.]

Kind	Kind-ID	RFC
SIP-REGISTRATION	1	RFC-AAAA

9.2. XML Name Space Registration

This document registers the following URI for the config XML namespace in the IETF XML registry defined in [RFC3688]

URI: urn:ietf:params:xml:ns:p2p:config-base:sip

Registrant Contact: The IESG

XML: N/A, the requested URI is an XML namespace

10. Acknowledgments

This document was generated in parts from initial drafts and discussions in the early specification phase of the P2PSIP base protocol. Significant contributions (in alphabetical order) were from David A. Bryan, James Deverick, Marcin Matuszewski, Jonathan Rosenberg, and Marcia Zangrilli, which is gratefully acknowledged.

Additional thanks go to all those who helped with ideas, discussions, and reviews, in particular (in alphabetical order) Michael Chen, Marc Petit-Huguenin, Brian Rosen, and Matthias Waehlich.

11. References

11.1. Normative References

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

[I-D.ietf-p2psip-base]

Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-26 (work in

progress), February 2013.

- [RFC3261] Rosenberg, J., Schulzrinne, H., Camarillo, G., Johnston, A., Peterson, J., Sparks, R., Handley, M., and E. Schooler, "SIP: Session Initiation Protocol", RFC 3261, June 2002.
- [RFC2533] Klyne, G., "A Syntax for Describing Media Feature Sets", RFC 2533, March 1999.
- [RFC2738] Klyne, G., "Corrections to "A Syntax for Describing Media Feature Sets"", RFC 2738, December 1999.
- [RFC3688] Mealling, M., "The IETF XML Registry", BCP 81, RFC 3688, January 2004.
- [RFC5627] Rosenberg, J., "Obtaining and Using Globally Routable User Agent URIs (GRUUs) in the Session Initiation Protocol (SIP)", RFC 5627, October 2009.
- [RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, October 2006.
- [IEEE-Posix]
"IEEE Standard for Information Technology - Portable Operating System Interface (POSIX) - Part 2: Shell and Utilities (Vol. 1)", IEEE Std 1003.2-1992, ISBN 1-55937-255-9, January 1993.

11.2. Informative References

- [I-D.ietf-p2psip-concepts]
Bryan, D., Willis, D., Shim, E., Matthews, P., and S. Dawkins, "Concepts and Terminology for Peer to Peer SIP", draft-ietf-p2psip-concepts-04 (work in progress), October 2011.
- [RFC5767] Munakata, M., Schubert, S., and T. Ohba, "User-Agent-Driven Privacy Mechanism for SIP", RFC 5767, April 2010.
- [I-D.ietf-p2psip-share]
Knauf, A., Schmidt, T., Hege, G., and M. Waehlich, "A Usage for Shared Resources in RELOAD (ShaRe)", draft-ietf-p2psip-share-01 (work in progress), February 2013.

Appendix A. Third Party Registration

In traditional SIP, the mechanism of a third party registration (i.e., an assistant acting for a boss, changing users register a role-based AOR, ...) is defined in Section 10.2 of [RFC3261]. This is a REGISTER which uses the URI of the third-party in its From header and cannot be translated directly into a P2PSIP registration, because only the owner of the certificate can store a SIP-REGISTRATION in a RELOAD overlay.

A way to implement third party registration is by using the extended access control mechanism USER-CHAIN-ACL defined in [I-D.ietf-p2psip-share]. Creating a new Kind "SIP-3P-REGISTRATION" that is ruled by USER-CHAIN-ACL allows the owner of the certificate to delegate the right for registration to individual third parties. In this way, original SIP functionality can be regained without weakening the security control of RELOAD.

Appendix B. Change Log

B.1. Changes since draft-ietf-p2psip-sip-08

- o Added the handling of SIPS
- o Specified use of Posix regular expressions in configuration document
- o Added IANA registration for namespace
- o Editorial polishing
- o Updated and extended references

B.2. Changes since draft-ietf-p2psip-sip-07

- o Cleared open issues
- o Clarified use cases after WG discussion
- o Added configuration document extensions for configurable domain names
- o Specified format of contact_prefs
- o Clarified routing to AORs
- o Extended security section
- o Added Appendix on Third Party Registration
- o Added IANA code points
- o Editorial polishing
- o Updated and extended references

B.3. Changes since draft-ietf-p2psip-sip-06

- o Added Open Issue

Authors' Addresses

Cullen Jennings
Cisco
170 West Tasman Drive
MS: SJC-21/2
San Jose, CA 95134
USA

Phone: +1 408 421-9990
Email: fluffy@cisco.com

Bruce B. Lowekamp
Skype
Palo Alto, CA
USA

Email: bbl@lowekamp.net

Eric Rescorla
RTFM, Inc.
2064 Edgewood Drive
Palo Alto, CA 94303
USA

Phone: +1 650 678 2350
Email: ekr@rtfm.com

Salman A. Baset
Columbia University
1214 Amsterdam Avenue
New York, NY
USA

Email: salman@cs.columbia.edu

Henning Schulzrinne
Columbia University
1214 Amsterdam Avenue
New York, NY
USA

Email: hgs@cs.columbia.edu

Thomas C. Schmidt (editor)
HAW Hamburg
Berliner Tor 7
Hamburg 20099
Germany

Email: schmidt@informatik.haw-hamburg.de

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: July 18, 2013

M. Petit-Huguenin
Impedance Mismatch
January 14, 2013

Anycast, Multicast or Broadcast Bootstrap Nodes for REsource LOcation
And Discovery (RELOAD)
draft-petithuguenin-p2psip-reload-one-to-many-00

Abstract

This document describes an extension to REsource LOcation And Discovery (RELOAD) that permits to contact a bootstrap node using an Anycast, Multicast or Broadcast IP address.

Status of this Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79. This document may not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

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 July 18, 2013.

Copyright Notice

Copyright (c) 2013 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. Searching for Bootstrap Node	3
4. Acknowledgments	4
5. Security Considerations	4
6. IANA Considerations	4
7. Normative References	5
Appendix A. Example	5
Author's Address	6

1. Introduction

RELOAD [I-D.ietf-p2psip-base] explains that to join an overlay a node must, if the bootstrap node cache is empty, contact one of the bootstrap peers listed in a <bootstrap-node> element of the configuration document. The process described works only with unicast addresses, as TCP - and so TLS - is not supported with anything else than unicast addresses, and DTLS does not work at all with multicast or broadcast addresses, and will be unreliable with anycast addresses.

2. Terminology

The key words "MUST" and "MAY" in this document are to be interpreted as described in [RFC2119].

One-to-many Address: An Anycast, Multicast or Broadcast IP address.

3. Searching for Bootstrap Node

An overlay that runs one or more bootstrap node over a one-to-many address MUST add a <bootstrap-node> element for each of them in the configuration document, using the XML namespace designating them as such. The same configuration document MAY also contain unicast <bootstrap-node> elements to let nodes that do not implement this specification join the overlay.

The RELAX NG grammar for the one-to-many <bootstrap-node> element is as follow. Whitespace and case processing MUST follow the rules of [OASIS.relax_ng] and XML Schema Datatypes [W3C.REC-xmlschema-2-20041028].

```
namespace otm = "http://implementers.org/reload-one-to-many"
```

```
parameter &= element otm:bootstrap-node {  
    attribute address { xsd:string },  
    attribute port { xsd:int }?  
}*
```

The <bootstrap-node> element has an attribute called "address" that contains an IPv4 or IPv6 address and an optional attribute called "port" that represents the port and defaults to 6084. The IPv6 address MUST use the hexadecimal form using standard period and colon separators as specified in [RFC5952]. More than one "bootstrap-node" element MAY be present.

Instead of the bootstrap node, a STUN [RFC5389] server implementing the ALTERNATE-SERVER Mechanism (Section 11) MUST run on the one-to-many address. The STUN server only serves Binding Request and the ALTERNATE-SERVER attribute returned MUST contain the unicast address and the port of a bootstrap node. Credentials MUST NOT be required by the STUN server.

If no cached bootstrap nodes are available, a joining node that implements this specification MUST first try to join the bootstrap nodes designated as listening on a one-to-many address. If trying to contact all the one-to-many addresses fails, the joining node MUST try the eventual unicast bootstrap nodes listed in the configuration document. The joining node MUST send a STUN Binding Request to one of the one-to-many addresses, chosen randomly. Any response other than a 300 will put the one-to-many address in a blacklist for 3 minutes, and the joining node MUST try the next one-to-many address. The IP address and port returned in a 300 response MUST be used as if a unicast bootstrap address was retrieved from the configuration file, and MAY be cached the same way a unicast bootstrap node address is cached.

As a consequence of this design, a bootstrap node running on a unicast address do not have to listen for other protocols than the Overlay Link protocols defined in RELOAD. A client that need to know the Node-ID of the bootstrap node can send a Probe or Ping request over DTLS-UDP-SR, TLS-TCP-FH-NO-ICE or DTLS-UDP-SR-NO-ICE with a destination list containing a single wildcard Node-ID. After this, the standard process to join the overlay described in [I-D.ietf-p2psip-base] can be used.

4. Acknowledgments

Some of the text in this document was taken from version 23 of [I-D.ietf-p2psip-base], authored by Cullen Jennings, Bruce B. Lowekamp, Eric Rescorla, Salman A. Baset and Henning Schulzrinne.

5. Security Considerations

6. IANA Considerations

If this document is accepted as a standard track document this section will request a URN in the "XML Namespaces" class of the "IETF XML Registry" from IANA. Until this is done, implementations should use the following URN:

<http://implementers.org/reload-one-to-many>

7. Normative References

- [I-D.ietf-p2psip-base]
Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and
H. Schulzrinne, "REsource LOcation And Discovery (RELOAD)
Base Protocol", draft-ietf-p2psip-base-23 (work in
progress), November 2012.
- [OASIS.relax_ng]
Bray, T. and M. Murata, "RELAX NG Specification".
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate
Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC5389] Rosenberg, J., Mahy, R., Matthews, P., and D. Wing,
"Session Traversal Utilities for NAT (STUN)", RFC 5389,
October 2008.
- [RFC5952] Kawamura, S. and M. Kawashima, "A Recommendation for IPv6
Address Text Representation", RFC 5952, August 2010.
- [W3C.REC-xmlschema-2-20041028]
Malhotra, A. and P. Biron, "XML Schema Part 2: Datatypes
Second Edition", World Wide Web Consortium
Recommendation REC-xmlschema-2-20041028, October 2004,
<<http://www.w3.org/TR/2004/REC-xmlschema-2-20041028>>.

Appendix A. Example

The following example shows that an anycast address and a multicast address can be used to find a bootstrap node. A joining node that does not recognize the extension can still join the overlay by using the unicast addresses.

```
<?xml version="1.0" encoding="UTF-8" ?>
<overlay xmlns="urn:ietf:params:xml:ns:p2p:config-base"
  xmlns:otm="http://implementers.org/reload-one-to-many">
  <configuration instance-name="overlay.example.org" sequence="22"
    expiration="2002-10-10T07:00:00Z">
    <bootstrap-node address="192.0.0.1" port="6084" />
    <bootstrap-node address="192.0.2.2" port="6084" />
    <bootstrap-node address="2001:DB8::1" port="6084" />
    <otm:bootstrap-node address="192.0.0.1" />
    <otm:bootstrap-node address="233.252.0.1" port="6084" />
  </configuration>
  <signature> VGhpcyBpcyBub3QgcmlnaHQhCg== </signature>
</overlay>
```

Author's Address

Marc Petit-Huguenin
Impedance Mismatch

Email: petithug@acm.org

Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: August 21, 2013

M. Petit-Huguenin
Impedance Mismatch
February 17, 2013

SCTP Overlay link for REsource LOcation And Discovery (RELOAD)
draft-petithuguenin-p2psip-reload-sctp-00

Abstract

This document defines two new Overlay Link protocols to be used with REsource LOcation And Discovery (RELOAD), both using SCTP. The Overlay Link protocol DTLS-SCTP-NO-ICE uses DTLS on top of native SCTP for overlays that do not require the use of ICE. The Overlay Link protocol DTLS-SCTP-UDP uses DTLS on top of an UDP encapsulation of SCTP for overlays that require the use of ICE.

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 August 21, 2013.

Copyright Notice

Copyright (c) 2013 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 not be modified, and derivative works of it may not be created, except to format it for publication as an RFC or to translate it into languages other than English.

Table of Contents

1. Introduction	2
2. Terminology	2
3. DTLS-SCTP-NO-ICE Overlay Link Protocol	3
4. DTLS-SCTP-UDP Overlay Link Protocol	3
5. Security Considerations	3
6. IANA Considerations	3
7. Normative References	3
Author's Address	4

1. Introduction

RELOAD [I-D.ietf-p2psip-base] defines 3 Overlay Link protocols, DTLS-UDP-SR, TLS-TCP-FH-NO-ICE and DTLS-UDP-SR-NO-ICE but none of them prevents Head of Line (HoL) blocking. DTLS for SCTP [RFC6083], is a good candidate as Overlay Link protocol, but because most NATs cannot handle SCTP [RFC4960], it can only be used with overlays that are known to not have nodes behind a NAT. For overlays that may have nodes behind a NAT, a UDP encapsulation of SCTP [I-D.ietf-tsvwg-sctp-udp-encaps] is used in place of the native SCTP, so it can traverse NAT. Additionally using a UDP encapsulation does not require to define how using SCTP in ICE.

2. Terminology

The key words "MUST" and "MUST NOT" in this document are to be interpreted as described in [RFC2119].

3. DTLS-SCTP-NO-ICE Overlay Link Protocol

This overlay link protocol consists of DTLS over SCTP, as specified in [RFC6083]. The RELOAD messages MUST be sent on a stream other than 0 (which is reserved for the DTLS ChangeCipherSpec, Alert and Handshake protocols), with unlimited reliability but without the ordered delivery feature.

The Framing Header or the Simple Reliability protocols described in RELOAD MUST NOT be used.

4. DTLS-SCTP-UDP Overlay Link Protocol

This overlay link protocol consists of DTLS over an UDP encapsulation of SCTP, as specified in [RFC6083] and [I-D.ietf-tsvwg-sctp-udp-encaps]. The RELOAD messages MUST be sent on a stream other than 0 (which is reserved for the DTLS ChangeCipherSpec, Alert and Handshake protocols), with unlimited reliability but without the ordered delivery feature.

The Framing Header or the Simple Reliability protocols described in RELOAD MUST NOT be used.

5. Security Considerations

TBD.

6. IANA Considerations

If this document is accepted as a standard track document this section will request that IANA assign codes for the DTLS-SCTP-NO-ICE and DTLS-SCTP-UDP overlay protocols. Until this is done, implementations should use code 5 (reserved for experimentations as EXP-LINK). Using the same code is not an issue in this case as ICE and NO-ICE Overlays Protocols cannot be simultaneously used in the same overlay.

[[Do we need to also assign a PPID?]]

7. Normative References

[I-D.ietf-p2psip-base]

Jennings, C., Lowekamp, B., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", draft-ietf-p2psip-base-24 (work in progress), January 2013.

[I-D.ietf-tsvwg-sctp-udp-encaps]

Tuexen, M. and R. Stewart, "UDP Encapsulation of SCTP Packets", draft-ietf-tsvwg-sctp-udp-encaps-09 (work in progress), January 2013.

- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, March 1997.
- [RFC4960] Stewart, R., "Stream Control Transmission Protocol", RFC 4960, September 2007.
- [RFC6083] Tuexen, M., Seggelmann, R., and E. Rescorla, "Datagram Transport Layer Security (DTLS) for Stream Control Transmission Protocol (SCTP)", RFC 6083, January 2011.

Author's Address

Marc Petit-Huguenin
Impedance Mismatch

Email: petithug@acm.org