Network Working Group	M.P.H. Petit-Huguenin
Internet-Draft	Stonyfish, Inc.
Intended status: Standards Track	March 06, 2011
Expires: September 07, 2011	

Configuration of Access Control Policy in REsource LOcation And Discovery (RELOAD) Base Protocol
draft-petithuguenin-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

Copyright Notice

Table of Contents

1. Introduction

The RELOAD base protocol defines 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 they are executed when a Store request is processed, they need to be implemented by all the peers, and so require 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 set of new elements that can be optionally added to a <kind> element in the configuration document. The most important of this elements is the <access-control-code> element that contains JavaScript code that will be called for each StoredData object in a StoreReq processed by a peer. 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 redefined by inserting the following code in an <access-control-code> element:

```
return resource.equals(signature.user_name);
```

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

```
for (int i = 0; i < kind.params['max-node-multiple']; i++) {
  if (resource.equals(signature.node_id, i)) return true;
}
return false;
```

Some Access Control Policies requires access to the content of the value to be stored. To permit this a <data-stored> element can be added to describe the content of the value. This description uses the same syntax that is used in the RELOAD base protocol to describe the various messages and is automatically converted to a JavaScript object accessible from the Access Control Policy code. For example ReDiR *[I-D.ietf-p2psip-service-discovery]* requires such mechanism so if the structure described in section 4.1. is copied in the <data-stored> element then the following code can be used to define the NODE-ID-MATCH Access Control Policy:

```
return entry.key === signature.node_id
  && (!entry.exists
      || resource.equals(entry.value.data.namespace,
          entry.value.data.level, entry.value.data.node);
```

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].

3. Processing an extended Kind

A peer receiving a <kind>, either by retrieving it from the configuration server or in a ConfigUpdateReq message, MUST verify the signature in the kind-signature element before executing the code. If the <access-control-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 this Kind. For each call to the code, the following JavaScript objects, properties and functions MUST be available:

resource: An opaque object representing the Resource-ID.

resource.equals(Object...): Returns true if hashing the concatenation of the arguments according to the mapping function of the overlay algorithm is equal to the Resource-ID.

signature.user_name: The rfc822Name stored in the certificate that was used to sign the request.

signature.node_id: The Node-ID stored in the certificate that was used to sign the request.

kind.id: The id of the Kind associated with the entry.

kind.name: The name of the Kind associated with the entry.

kind.data_model: The name of the Data Model associated with the entry.

kind.access_control: The name of the Access Control Policy associated with the entry.

kind.params: An associative array containing the parameters of the Access Control Policy as specified in the configuration file.

- **max-count:** The value of the max-count element in the configuration file.
- **max-size:** The value of the max-size element in the configuration file.
- **max-node-multiple:** If the Access Control is MULTIPLE-NODE, contains the value of the max-node-multiple element in the configuration file. If not, this property is undefined.

entry.index: If the Data Model is ARRAY, contains the index of the entry. If not, this property is undefined.

entry.key: If the Data Model is DICTIONARY, contains the key of the entry. If not, this property is undefined.

entry.storage_time: A Date object containing the time for the storage.

entry.lifetime: A number that contain the validity for the data in seconds.

entry.exist: A boolean that indicates if the entry value exists.

entry.value: If a <description> extension element is present in the <kind> element, then the content of this property is automatically generated from the definition. If not, this property contains an opaque object that represent the whole data.

If addition to the "max-count", "max-size" and eventually "max-node-multiple" properties in the kind.params associative array, any extension element in any namespace found in the <kind> element MUST be added to this array, using the element name as key and the content as value.
The value returned by the code is evaluated to true or false, according to the JavaScript 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.
If the <data-stored> element is present in the namespace allocated to this specification, then its content MUST be parsed and converted in a way that will permit to parse the value in the DataValue structure and generate a JavaScript object with properties corresponding to each label. The "value" attribute MUST be filled with the label of the statement inside the element that must be used as the root of the parsing. Each statement in the content MUST be converted as follow:

Vectors: Variable-length vectors are converted to arrays.

Numbers: uint8, uint16, uint24, uint32, uint64, and uint128 are converted to a JavaScript number.

Enumerateds: TBD

Structures: Structures are converted to an object, which each fields being converted to a property of same name.

Variants: TBD

If the <data-stored> element is not present, the value in the DataValue structure will still be passed as an opaque object to the code.

4. Security Considerations

TBD

5. IANA Considerations

No IANA considerations.

6. Acknowledgements

This document was written with the xml2rfc tool described in [RFC2629].

7. References

7.1. 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", Internet-Draft draft-ietf-p2psip-base-12, November 2010.

7.2. Informative References

[RFC2629]	Rose, M.T., "Writing I-Ds and RFCs using XML", RFC 2629, June 1999.
[I-D.ietf-p2psip-service-discovery]	Maenpaa, J and G Camarillo, "Service Discovery Usage for REsource LOcation And Discovery (RELOAD)", Internet-Draft draft-ietf-p2psip-service-discovery-02, January 2011.
[I-D.knauf-p2psip-disco]	Knauf, A, Hege, G, Schmidt, T and M Waehlisch, "A RELOAD Usage for Distributed Conference Control (DisCo)", Internet-Draft draft-knauf-p2psip-disco-01, December 2010.

Appendix A. Examples

Appendix A.1. Standard Access Control Policies

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

Appendix A.1.1. USER-MATCH

```
return resource.equals(signature.user_name);
```

Appendix A.1.2. NODE-MATCH

```
return resource.equals(signature.node_id);
```

Appendix A.1.3. USER-NODE-MATCH

```
return resource.equals(signature.user_name)
   && entry.key === signature.node_id;
```

Appendix A.1.4. NODE-MULTIPLE

```
for (int i = 0; i < kind.params['max-node-multiple']; i++) {
   if (resource.equals(signature.node_id, i)) return true;
}
return false;
```

Appendix A.2. Service Discovery Usage

[I-D.ietf-p2psip-service-discovery] defines a specific 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 <kind> element would look something like this:

```
<kind name='REDIR'
  xmlns:acp='http://implementers.org/access-control-policy'>
    <data-model>DICTIONARY</data-model>
    <access-control>NODE-ID-MATCH</access-control>
    <max-count>100</max-count>
    <max-size>60</max-size>

    <acp:access-control-code>
       return entry.key === signature.node_id
          && true /* placeholder */
          && (!entry.exists
              || (resource.equals(entry.value.data.namespace,
                  entry.value.data.level, entry.value.data.node);
    </acp:access-control-code>

    <acp:data-stored value='RedirServiceProvider'>
       struct {
         NodeId                  serviceProvider;
         opaque                  namespace&lt;0..2^16-1&gt;;
         uint16                  level;
         uint16                  node;

         /* This type can be extended */

       } RedirServiceProviderData;

       struct {
         uint16                  length;
         RedirServiceProviderData data;
       } RedirServiceProvider;
    </acp:data-stored>
</kind>
```

Appendix B. Release notes

This section must be removed before publication as an RFC.

Appendix B.1. TODO List

*Need to present the complete list of certificates for the DisCo *[I-D.knauf-p2psip-disco]* Usage USER-CHAIN-MATCH.

*Add ABNF for the presentation language.

Author's Address

Marc Petit-Huguenin Petit-Huguenin Stonyfish, Inc. EMail: petithug@acm.org