Internet Engineering Task Force (IETF)
Request for Comments: 7650
Category: Standards Track
ISSN: 2070-1721

J. Jimenez
Ericsson
J. Lopez-Vega
University of Granada
J. Maenpaa
G. Camarillo
Ericsson

September 2015

A Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD)

Abstract

This document defines a Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD). The CoAP Usage provides the functionality to federate Wireless Sensor Networks (WSNs) in a peer-to-peer fashion. The CoAP Usage for RELOAD allows CoAP nodes to store resources in a RELOAD peer-to-peer overlay, provides a lookup service, and enables the use of RELOAD overlay as a cache for sensor data. This functionality is implemented in the RELOAD overlay itself, without the use of centralized servers. The RELOAD AppAttach method is used to establish a direct connection between nodes through which CoAP messages are exchanged.

Status of This Memo

This is an Internet Standards Track document.

This document is a product of the Internet Engineering Task Force (IETF). It represents the consensus of the IETF community. It has received public review and has been approved for publication by the Internet Engineering Steering Group (IESG). Further information on Internet Standards is available in Section 2 of RFC 5741.

Information about the current status of this document, any errata, and how to provide feedback on it may be obtained at http://www.rfc-editor.org/info/rfc7650.

Copyright Notice

Copyright (c) 2015 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 5
3. Architecture 5
4. Registering CoAP URIs 7
5. Lookup 8
6. Forming a Direct Connection and Reading Data 9
7. Caching Mechanisms 11
7.1. ProxyCache 11
7.2. SensorCache 13
8. CoAP Usage Kinds Definition 14
8.1. CoAP-REGISTRATION Kind 14
8.2. CoAP-CACHING Kind 15
9. Access Control Rules 15
10. Security Considerations 16
11. IANA Considerations 17
11.1. CoAP-REGISTRATION Kind-ID 17
11.2. CoAP-CACHING Kind-ID 17
11.3. Access Control Policies 17
12. References 18
12.1. Normative References 18
12.2. Informative References 18
Authors' Addresses 19

1. Introduction

The Constrained Application Protocol (CoAP) Usage for REsource LOcation And Discovery (RELOAD) allows CoAP nodes to store resources in a RELOAD peer-to-peer overlay, provides a lookup service, and enables the use of RELOAD overlay as a cache for sensor data. This functionality is implemented in the RELOAD overlay itself, without the use of centralized servers.

This usage is intended for interconnected devices over a wide-area geographical coverage, such as in cases where multiple Wireless Sensor Networks (WSNs) need to be federated over some wider-area network. These WSNs would interconnect by means of nodes that are equipped with long range modules (e.g., 2G, 3G, 4G) as well as short range ones (e.g., XBee, ZigBee, Bluetooth LE).

Constrained devices are likely to be heterogeneous when it comes to their radio layer; however, we expect them to use a common application-layer protocol -- CoAP, which is a specialized web transfer protocol [RFC7252]. It realizes the Representational State Transfer (REST) architecture for the most constrained nodes, such as sensors and actuators. CoAP can be used not only between nodes on the same constrained network but also between constrained nodes and nodes on the Internet. The latter is possible since CoAP can be translated to Hypertext Transfer Protocol (HTTP) for integration with the web. Application areas of CoAP include different forms of machine-to-machine (M2M) communication, such as home automation, construction, health care or transportation. Areas with heavy use of sensor and actuator devices that monitor and interact with the surrounding environment.

As specified in [RFC6940], RELOAD is fundamentally an overlay network. It provides a layered architecture with pluggable application layers that can use the underlaying forwarding, storage, and lookup functionalities. Figure 1 illustrates where the CoAP Usage is placed within the RELOAD architecture.

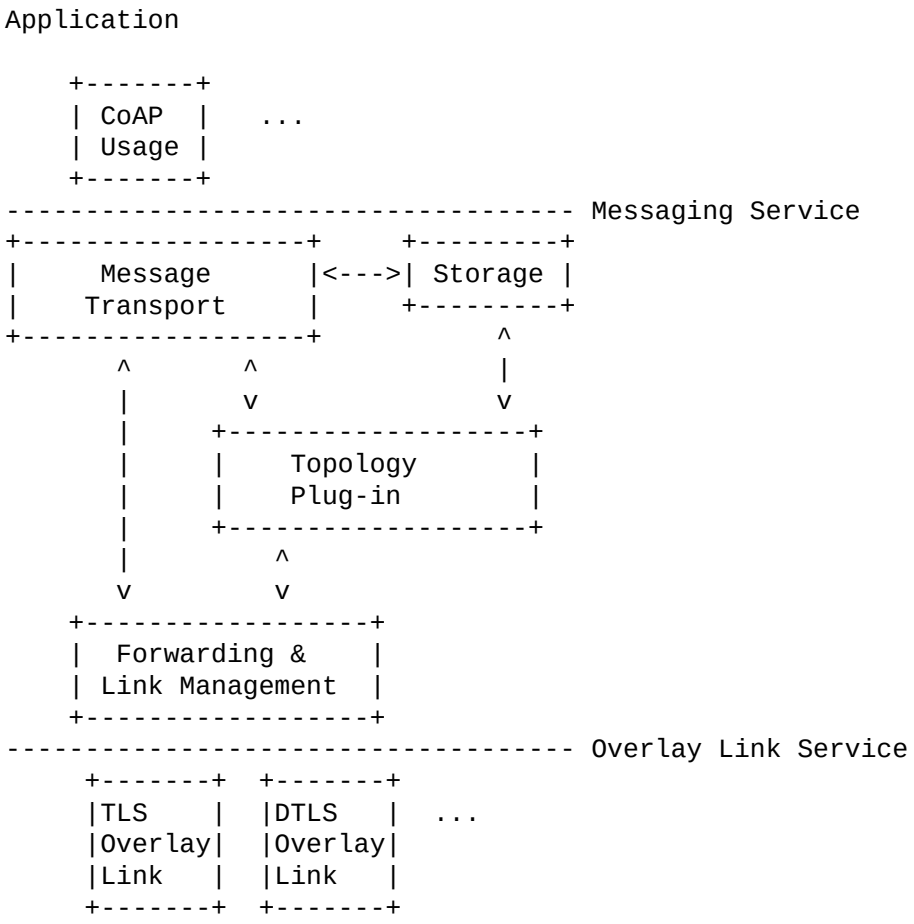

Figure 1: Architecture

The CoAP Usage involves three basic functions:

Registration: CoAP nodes that can use the RELOAD data storage functionality, can store a mapping from their CoAP URI to their Node-ID in the overlay. They can also retrieve the Node-IDs of other nodes. Nodes that are not RELOAD aware can use other mechanisms, for example [CORERESDIR] in their local network.

Lookup: Once a CoAP node has identified the Node-ID for an URI it wishes to retrieve, it can use the RELOAD message routing system to set up a connection that can be used to exchange CoAP messages. Similarly as with the registration, nodes that are not RELOAD aware can use CoAP messages with a RELOAD Node (RN) that will in turn perform the lookup in the overlay.

Caching: Nodes can use the RELOAD overlay as a caching mechanism for information about what CoAP resources are available on the node. This is especially useful for power-constrained nodes that can make their data available in the cache provided by the overlay while in sleep mode.

For instance, a CoAP proxy (See Section 3) could register its Node-ID (e.g. "9996172") and a list of sensors (e.g. "/sensors/temp-1; /sensors/temp-2; /sensors/light, /sensors/humidity") under its URI (e.g. "coap://overlay-1.com/proxy-1/").

When a node wants to discover the values associated with that URI, it queries the overlay for "coap://overlay-1.com/proxy-1/" and gets back the Node-ID of the proxy and the list of its associated sensors. The requesting node can then use the RELOAD overlay to establish a direct connection with the proxy and to read sensor values.

Moreover, the CoAP proxy can store the sensor information in the overlay. In this way, information can be retrieved directly from the overlay without performing a direct connection to the storing proxy.

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 the RELOAD Base Protocol [RFC6940] extensively in this document. Some of those concepts are further described in the "Concepts and Terminology for Peer to Peer SIP" [P2PSIP] document.

3. Architecture

In our architecture we extend the different nodes present in RELOAD (Peer, Client) and add support for sensor devices or other constrained devices. Figure 2 illustrates the overlay topology. The different nodes, according to their functionality, are:

Client
: As specified in [RFC6940], clients are nodes that do not have routing or storage responsibilities in the Overlay.

Peer
: As specified in [RFC6940], peers are nodes in the overlay that can route messages for nodes other than those to which it is directly connected.

Sensor
: Devices capable of measuring a physical quantity. Sensors usually acquire quantifiable information about their surrounding environment such as: temperature, humidity, electric current, moisture, radiation, and so on.

Actuator
: Devices capable of interacting and affecting their environment such as: electrical motors, pneumatic actuators, electric switches, and so on.

Proxy Node
: Devices having sufficient resources to run RELOAD either as client or peer. These devices are located at the edge of the sensor network and, in case of Wireless Sensor Networks (WSN), act as coordinators of the network.

Physical devices can have one or several of the previous functional roles. According to the functionalities that are present in each of the nodes, they can be:

Constrained Node
: A Constrained Node (CN) is a node with limited computational capabilities. CN devices belong to classes of at least C1 and C2 devices as defined in [RFC7228], their main constraint being the implementation of the CoAP protocol. If the CN is wireless, then it will be part of a Low-Rate Wireless Personal Area Network (LR-WPAN), also termed Low-Power and Lossy Network (LLN). Lastly, devices will usually be in sleep mode in order to prevent battery drain, and will not communicate during those periods. A CN is NOT part of the RELOAD overlay, therefore it cannot act as a client, peer, nor proxy. A CN is always either a Sensor or an Actuator. In the latter case, the node is often connected to a continuous energy power supply.

RELOAD Node
: A RELOAD Node (RN) MUST implement the client functionality in the Overlay. Additionally, the node will often be a full RELOAD peer. An RN may also be sensor or actuator since it can have those devices connected to it.

Proxy Node
: A Proxy Node (PN) MUST implement the RN functionality and act as a sink for the LR-WPAN network. The PN connects the short range Wireless Network to the Wide Area Network or the Internet. A Proxy Node fulfills the "Proxy Node" role as described previously in the Architecture.

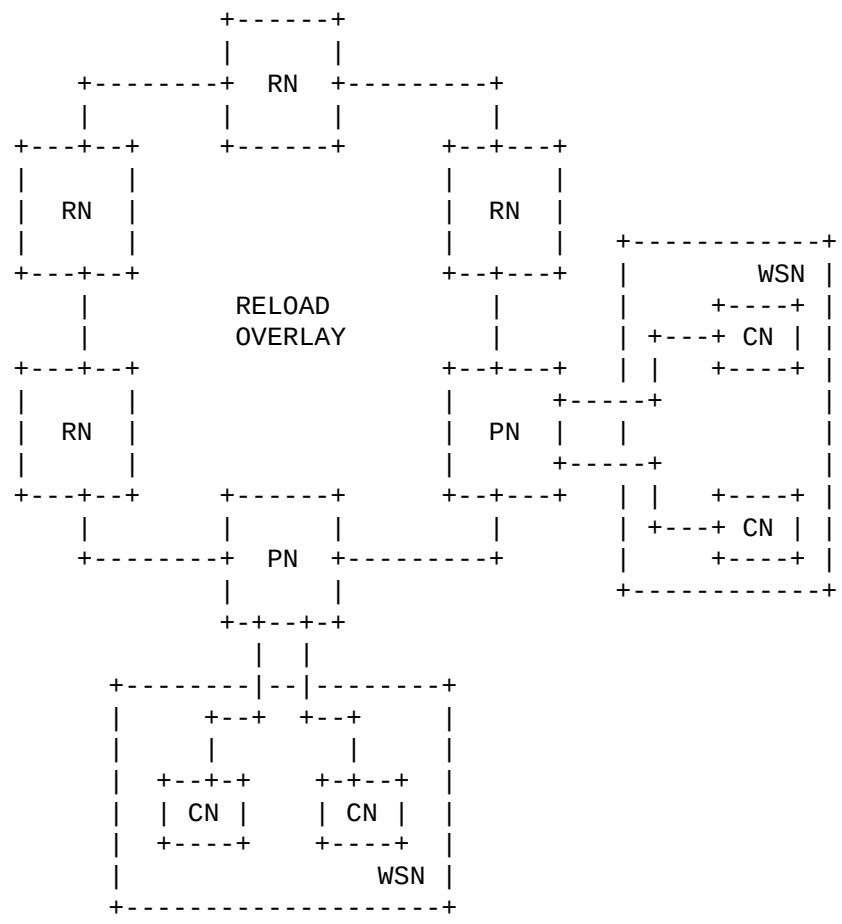

Figure 2: Overlay Topology

4. Registering CoAP URIs

CoAP URIs are typically resolved using a DNS. When CoAP is needed in a RELOAD environment, URI resolution is provided by the overlay as a whole. Instead of registering a URI, a peer stores a CoAPRegistration structure under a hash of its own URI. This uses the CoAP REGISTRATION Kind-ID, which is formally defined in Section 8.1 and uses a DICTIONARY data model.

In this example, a CoAP proxy that is located in an overlay overlay-1.com using a Node-ID "9996172" wants to register four different sensors to the URI "coap://overlay-1.com/proxy-1/.well-known/". We will be using the link format specified in [RFC6690] to store the following mapping in the overlay:

```
 Resource-ID = h(coap://overlay-1.com/proxy-1/.well-known/)
 KEY =  9996172,

 VALUE = [
  </sensors/temp-1>;rt="temperature-c";if="sensor",
  </sensors/temp-2>;rt="temperature-c";if="sensor",
  </sensors/light>;rt="light-lux";if="sensor",
  </sensors/humidity>;rt="humidity-p";if="sensor"
     ]
```

Note that the Resource-ID stored in the overlay is calculated as hash over the URI, that is -- h(URI), which in RELOAD is usually SHA-1.

This would inform any other node performing a lookup for the previous URI "coap://overlay-1.com/proxy-1/.well-known" that the Node-ID value for proxy-1 is "9996172". In addition, this mapping provides relevant information as to the number of sensors (CNs) and the URI path to connect to them using CoAP.

5. Lookup

The RELOAD overlay supports a rendezvous system that can be used for the lookup of other CoAP nodes. This is done by fetching mapping information between CoAP URIs and Node-IDs.

As an example, if a node RN located in the overlay overlay-1.com wishes to read which resources are served at an RN with URI coap://overlay-1.com/proxy-1/, it performs a fetch in the overlay. The Resource-ID used in this fetch is a SHA-1 hash over the URI "coap://overlay-1.com/proxy-1/.well-known/".

After this fetch request, the overlay will return the following result:

```
 Resource-ID = h(coap://overlay-1.com/proxy-1/.well-known/)
 KEY =  9996172,

 VALUE = [
  </sensors/temp-1>;rt="temperature-c";if="sensor",
  </sensors/temp-2>;rt="temperature-c";if="sensor",
  </sensors/light>;rt="light-lux";if="sensor",
  </sensors/humidity>;rt="humidity-p";if="sensor"
  ]
```

The obtained KEY is the Node-ID of the RN responsible of this KEY/VALUE pair. The VALUE is the set of URIs necessary to read data from the CNs associated with the RN.

Using the RELOAD DICTIONARY model allows for multiple nodes to perform a store to the same Resource-ID. This can be used, for example, to perform a store of resources of the same type or with similar characteristics. After performing a lookup, this feature allows the fetching of those multiple RNs that host CNs of the same class.

As an example, provided that the previous peer (9996172) and another peer (9996173) have stored the links to their respective temperature resources in this same Resource-ID (temperature), an RN (e.g., node-A) can do a fetch to the URI "coap://overlay-1.com/temperature/.well-known/", obtaining the following results:

```
Resource-ID = h(coap://overlay-1.com/temperature/.well-known/)

KEY =  9996172,
VALUE = [
 </sensors/temp-1>;rt="temperature-c";if="sensor",
 </sensors/temp-2>;rt="temperature-c";if="sensor",
  ]

KEY = 9996173,
VALUE = [
 </sensors/temp-a>;rt="temperature-c";if="sensor",
       </sensors/temp-b>;rt="temperature-c";if="sensor"
  ]
```

6. Forming a Direct Connection and Reading Data

Once an RN (e.g., node-A) has obtained the lookup information for a node in the overlay (e.g., proxy-1), it can directly connect to that node. This is performed by sending an AppAttach request to the Node-ID obtained during the lookup process.

After the AppAttach negotiation, node-A can access the values of the CNs at proxy-1 using the information obtained during the lookup. Following the example in Section 5, and according to [RFC6690], the requests for accessing the CNs at proxy-1 would be:

```
REQ: GET /sensors/temp-1
REQ: GET /sensors/temp-2
```

Figure 3 shows a sample of a node reading temperature data.

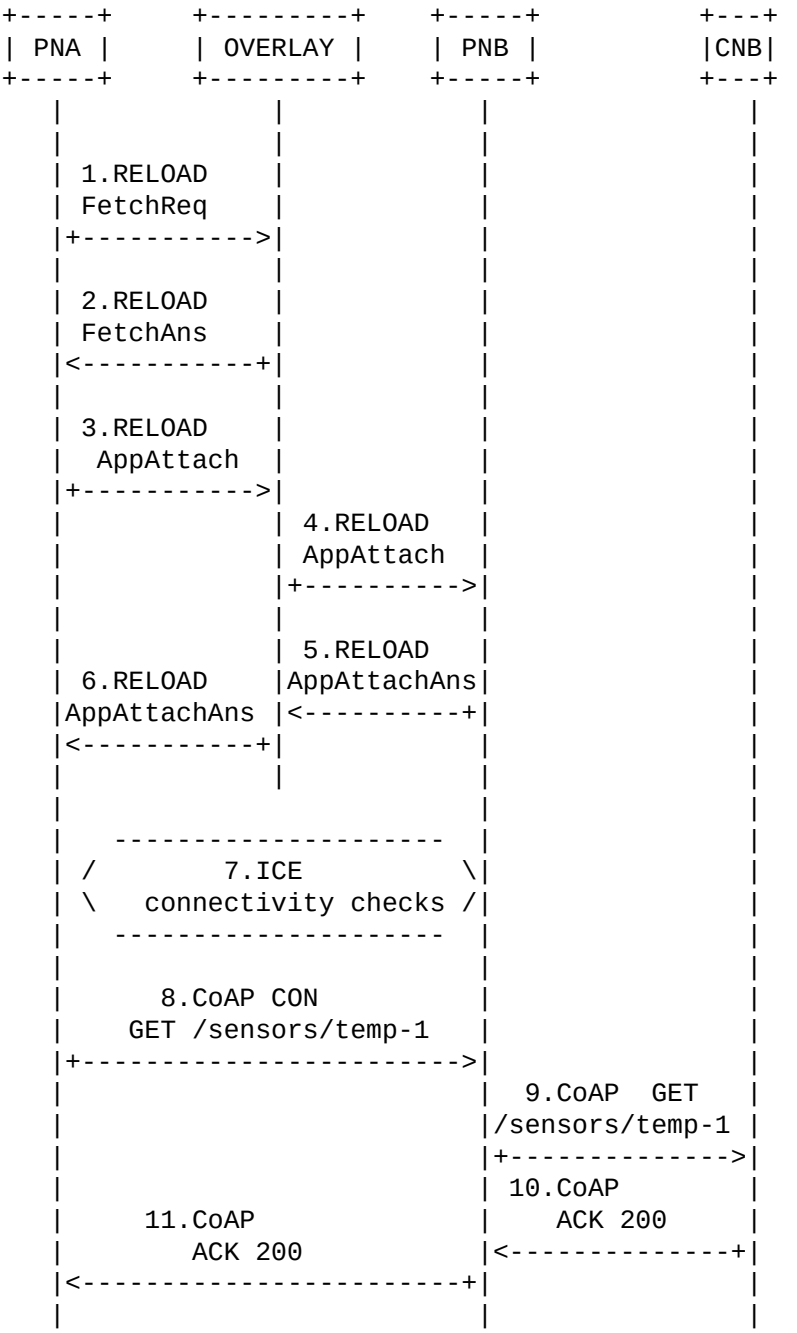

Figure 3: An Example of a Message Sequence

7. Caching Mechanisms

The CoAP protocol itself supports the caching of sensor information in order to reduce the response time and network bandwidth consumption of future, equivalent requests. CoAP caching is specified in Section 5 of [RFC7252]. It consists of reusing stored responses when new requests arrive. This type of storage is done in CoAP proxies.

This CoAP usage for RELOAD proposes an additional caching mechanism for storing sensor information directly in the overlay. In order to do so, it is necessary to define how the data should be stored. Such caching mechanism is primarily intended for CNs with sensor capabilities, not for RN sensors. This is due to the battery constraints of CNs, forcing them to stay in sleep mode for long periods of time.

Whenever a CN wakes up, it sends the most recent data from its sensors to its proxy (PN), which stores the data in the overlay using a RELOAD StoredData structure defined in Section 6 of [RFC6940]. We use the StoredDataValue structure defined in Section 6.2 of [RFC6940], in particular we use the SingleValue format type to store the cached values in the overlay. From that structure length, storage_time, lifetime and Signature are used in the same way. The only difference is DataValue, which in our case can be either a ProxyCache or a SensorCache:

```
enum { reserved (0), proxy_cache(1), sensor_cache(2), (255) }
             CoAPCachingType;
struct {
 CoAPCachingType coap_caching_type;

 select(coap_caching_type) {
  case proxy_cache: ProxyCache proxy_cache_entry;
  case sensor_cache: SensorCache sensor_cache_entry;
  /* extensions */

 }
} CoAPCaching;
```

7.1. ProxyCache

ProxyCache is meant to store values and sensor information (e.g., inactivity time) for all the sensors associated with a certain proxy, as well as their CoAP URIs. SensorCache, on the other hand, is used for storing the information and cached value of only one sensor (CoAP URI is not necessary, as it is the same as the one used for generating the Resource-ID associated to that SensorCache entry).

ProxyCache contains the Node-ID, length, and a series of SensorEntry types.

```
struct {
 Node-ID  Node_ID;
 uint32   length;
 SensorEntry sensors[count];
} ProxyCache;
```

Node-ID
: The Node-ID of the Proxy Node (PN) responsible for different sensor devices;

length
: The length of the rest of the structure;

Sensor-Entry
: List of sensors in the form of SensorEntry types;

SensorEntry contains the coap_uri, sensor_info, and a series of SensorValue types.

```
struct {
 opaque  coap_uri;
 SensorInfo  sensor_info;
 uint32  length;
 SensorValue sensor_value[count];
} SensorEntry;
```

coap_uri
: CoAP name of the sensor device in question;

sensor_info
: contains relevant sensor information;

length
: The length of the rest of the structure;

sensor_value
: contains a list of values stored by the sensor;

7.2. SensorCache

SensorCache: contains the information related to one sensor.

```
struct {
 Node-ID  Node_ID;
 SensorInfo sensor_info;
 uint32  length;
 SensorValue sensor_value[count];
} SensorCache;
```

Node_ID
: identifies the Node-ID of the Proxy Node responsible for the sensor;

sensor_info
: contains relevant sensor information;

length
: The length of the rest of the structure;

sensor_value
: contains a list of values stored by the sensor;

SensorInfo contains relevant sensor information that is dependent on the use case. As an example, we use the sensor manufacturer as relevant information.

```
struct {
 opaque  dev_info;

 /* extensions */

} SensorInfo;
```

dev_info
: Contains specific device information as defined in [RFC6690] -- for example, temperature, luminosity, etc. It can also represent other semantic information about the device.

SensorValue contains the measurement_time, lifetime, and value of the measurement.

```
struct {
 uint32  measurement_time;
 uint32  lifetime;
 opaque  value;

 /* extensions */

} SensorValue;
```

measurement_time
: indicates the moment when the measure was taken, represented as the number of milliseconds elapsed since midnight Jan 1, 1970 UTC not counting leap seconds.

lifetime
: indicates the validity time of that measured value in milliseconds since measurement_time.

value
: indicates the actual value measured. It can be of different types (integer, long, string); therefore, opaque has been used.

8. CoAP Usage Kinds Definition

This section defines the CoAP-REGISTRATION and CoAP-CACHING Kinds.

8.1. CoAP-REGISTRATION Kind

Kind-IDs
: The Resource Name for the CoAP-REGISTRATION Kind-ID is the CoAP URI. The data stored is a CoAPRegistration, which contains a set of CoAP URIs.

Data Model
: The data model for the CoAP-REGISTRATION Kind-ID is dictionary. The dictionary key is the Node-ID of the storing RN. This allows each RN to store a single mapping.

Access Control
: URI-NODE-MATCH. The "coap:" prefix needs to be removed from the COAP URI before matching.

Data stored under the COAP-REGISTRATION Kind is of type CoAPRegistration, defined below.

```
struct {
 Node-ID Node_ID;
 uint16 coap_uris_length;
 opaque coap_uris (0..2^16-1);
} CoAPRegistration;
```

8.2. CoAP-CACHING Kind

Kind-IDs
: The Resource Name for the CoAP-CACHING Kind-ID is the CoAP URI. The data stored is a CoAPCaching, which contains a cached value.

Data Model
: The data model for the CoAP-CACHING Kind-ID is single value.

Access Control
: URI-MATCH. The "coap:" prefix needs to be removed from the COAP URI before matching.

Data stored under the CoAP-CACHING Kind is of type CoAPCaching, defined in Section 7.

9. Access Control Rules

As specified in RELOAD Base [RFC6940], every Kind that is storable in an overlay must be associated with an access control policy. This policy defines whether a request from a given node to operate on a given value should succeed or fail. Usages can define any access control rules they choose, including publicly writable values.

CoAP Usage for RELOAD requires an access control policy that allows multiple nodes in the overlay read and write access. This access is for registering and caching information using CoAP URIs as identifiers. Therefore, none of the access control policies specified in RELOAD Base [RFC6940] are sufficient.

This document defines two access control policies, called URI-MATCH and URI-NODE-MATCH. In the URI-MATCH policy, a given value MUST be written and overwritten if and only if the signer’s certificate contains an uniformResourceIdentifier entry in the subjectAltName Extension [RFC5280] that in canonicalized form hashes to the Resource-ID for the resource. As explained in Section 6.3 of [RFC7252] the "coap" and "coaps" schemes conform to the generic URI, thus they are normalized in the generic form as explained in

Section 6 of [RFC3986]. The hash function used is specified in Section 10.2 of [RFC6940]. The certificate can be generated as specified in Section 9 of [RFC7252], using Certificate mode.

In the URI-NODE-MATCH policy, a given value MUST be written and overwritten if and only if the condition for URI-MATCH is met and, in addition, the dictionary key is equal to the Node-ID in the certificate and that Node-ID is the one indicated in the SignerIdentity value cert_hash.

These Access Control Policies are specified for IANA in Section 11.3.

10. Security Considerations

The security considerations of RELOAD [RFC6940] and CoAP [RFC7252] apply to this specification. RELOAD's security model is based on public key certificates, which are used for signing messages and stored objects. At the connection level, RELOAD can use either TLS or DTLS. In the case of CoAP, several security modes have been defined. Implementations of this specification MUST follow all the security-related rules specified in the RELOAD [RFC6940] and CoAP [RFC7252] specifications.

Additionally, in RELOAD every Kind that is storable in an overlay must be associated with an access control policy. This document specifies two new access control policies, which are specified in Section 9. These policies cover the most typical deployment scenarios.

During the phase of registration and lookup, security considerations relevant to RELOAD apply. A CoAP node that advertises its existence via this mechanism, is more likely to be attacked, compared to a node (especially a sleepy node) that does not advertise its existence. Section 11 of [RFC7252] and Section 13 of [RFC6940] have more information about the kinds of attack and mitigation possible.

The caching mechanism specified in this document is additional to the caching already done in CoAP. Access control is handled by the RELOAD overlay, where the peer storing the data is responsible for validating the signature on the data being stored.

11. IANA Considerations

11.1. CoAP-REGISTRATION Kind-ID

This document introduces a data Kind-ID to the "RELOAD Data Kind-ID" registry:

Kind	Kind-ID	RFC
CoAP-REGISTRATION	0x105	RFC 7650

This Kind-ID was defined in Section 8.1.

11.2. CoAP-CACHING Kind-ID

This document introduces another data Kind-ID to the "RELOAD Data Kind-ID" registry:

Kind	Kind-ID	RFC
CoAP-CACHING	0x106	RFC 7650

This Kind-ID was defined in Section 8.2.

11.3. Access Control Policies

IANA has created a "CoAP Usage for RELOAD Access Control Policy" registry. This registry has been added to the existing RELOAD registry. Entries in this registry are strings denoting access control policies, as described in Section 9. New entries in this registry are to be registered per the Specification Required policy in [RFC5226]. The initial contents of this registry are:

Access Policy	RFC
URI-NODE-MATCH	RFC 7650
URI-MATCH	RFC 7650

This access control policy was described in Section 9.

12. References

12.1. Normative References

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

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <http://www.rfc-editor.org/info/rfc3986>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <http://www.rfc-editor.org/info/rfc5280>.

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format", RFC 6690, DOI 10.17487/RFC6690, August 2012, <http://www.rfc-editor.org/info/rfc6690>.

[RFC6940] Jennings, C., Lowekamp, B., Ed., Rescorla, E., Baset, S., and H. Schulzrinne, "REsource LOcation And Discovery (RELOAD) Base Protocol", RFC 6940, DOI 10.17487/RFC6940, January 2014, <http://www.rfc-editor.org/info/rfc6940>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <http://www.rfc-editor.org/info/rfc7252>.

12.2. Informative References

[CORERESDIR] Shelby, Z., Koster, M., Bormann, C., and P. Stok, "CoRE Resource Directory", Work in Progress, draft-ietf-core-resource-directory-04, July 2015.

[P2PSIP] Bryan, D., Matthews, P., Shim, E., Willis, D., and S. Dawkins, "Concepts and Terminology for Peer to Peer SIP", Work in Progress, draft-ietf-p2psip-concepts-07, May 2015.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 5226, DOI 10.17487/RFC5226, May 2008, <http://www.rfc-editor.org/info/rfc5226>.

[RFC7228] Bormann, C., Ersue, M., and A. Keranen, "Terminology for Constrained-Node Networks", RFC 7228, DOI 10.17487/RFC7228, May 2014, <http://www.rfc-editor.org/info/rfc7228>.

Authors' Addresses

Jaime Jimenez
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: jaime.jimenez@ericsson.com

Jose M. Lopez-Vega
University of Granada
CITIC UGR Periodista Rafael Gomez Montero 2
Granada 18071
Spain

Email: jmlvega@ugr.es

Jouni Maenpaa
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: jouni.maenpaa@ericsson.com

Gonzalo Camarillo
Ericsson
Hirsalantie 11
Jorvas 02420
Finland

Email: gonzalo.camarillo@ericsson.com