Thing-to-Thing Research Group
Internet-Draft
Intended status: Experimental
Expires: November 10, 2020

K. Hartke
Ericsson

May 9, 2020

Resource Discovery in Constrained RESTful Environments (CoRE) using the Constrained RESTful Application Language (CoRAL) draft-hartke-t2trg-coral-reef-04

Abstract

This document explores how the Constrained RESTful Application Language (CoRAL) might be used for two use cases in Constrained RESTful Environments (CoRE): CoRE Resource Discovery, which allows a client to discover the resources of a server given a host name or IP address, and CoRE Resource Directory, which provides a directory of resources on many servers.

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 https://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 November 10, 2020.

Table of Contents

1. Preamble

This document explores how CoRE Resource Discovery [RFC6690] and CoRE Resource Directory [I-D.ietf-core-resource-directory] might look like if based on CoRAL [I-D.ietf-core-coral]. The exploration is done in the style of a self-contained specification rather than a description of changes.

This document doesn't represent a proposal or recommendation for standardization at its current stage.

2. Introduction

Constrained RESTful Environments (CoRE) realize the Representational State Transfer (REST) architectural style [REST] in a suitable form for constrained nodes (e.g., 8-bit microcontrollers with limited RAM and ROM) and constrained networks [RFC7228]. CoRE technologies like the Constrained Application Protocol (CoAP) [RFC7252] are aimed at machine-to-machine (M2M) applications like smart energy and building automation.

The discovery of resources hosted by a constrained server is very important in machine-to-machine applications where no humans are in the loop and static interfaces result in fragility. In the Web, the discovery of resources provided by a Web server is typically based on links in representations of resources pointing at other resources, with search engines providing an entry point to find resources based on queries.

This document applies the idea of using Web Linking [RFC8288] for discovery to Constrained RESTful Environments. The discovery of resources hosted by a constrained Web server, resource metadata, and related resources is called "CoRE Resource Discovery".

The main function of CoRE Resource Discovery is to provide Uniform Resource Identifiers (URIs) [RFC3986] for the resources hosted by a server, complemented by metadata about those resources and possibly links to further resources. In this document, this information is conveyed in the Constrained RESTful Application Language (CoRAL) [I-D.ietf-core-coral].

This document specifies the use of CoRAL in two use cases:

Resource Discovery

: Allows a client to discover the resources of a server, given a server's host name or IP address.

Resource Directory

Allows a client to discover the resources of several servers, given a resource directory's URL.

Allows a server (or a third party acting on behalf of a server) to register its resources with a resource directory, given a resource directory's URL.

2.1. Resource Discovery

In many M2M applications, such as home or building automation, there is a need for local clients and servers to find and interact with each other without human intervention. CoRE Resource Discovery can be used by clients in such environments to discover the resources hosted by the server given a host name or IP address.

In this specification, the discovery is performed by retrieving a CoRAL representation of a well-known resource on the server, called "/.well-known/core". The representation contains a list of links to the resources of interest on the server, which are typically entry points to the different applications hosted by the server. The link targets may be annotated with resource metadata. A client would then find an appropriate resource based on the metadata. Queries based on metadata may also be specified in the query string to filter the result set.

The following example shows a client discovering the resources of a CoAP server by making a GET request to the "/.well-known/core" resource. The client gets a 2.05 (Content) response with a list of links of type <http://coreapps.org/reef#rd-item> to the resources on the server. The links themselves contain metadata about the resources (such as resource type, interface description, available content formats, or even further links to other related resources).

```
=>  0.01 GET
    Uri-Path: .well-known
    Uri-Path: core
    Accept: TBD3
```

```
      <=  2.05 Content
          Content-Format: TBD3

          #using reef = <http://coreapps.org/reef#>
          #using base = <http://coreapps.org/base#>
          #using iana = <http://www.iana.org/assignments/relation/>

          reef:rd-item </sensors> {
             reef:ct TBD3
             base:title "Sensor Index"
          }
          reef:rd-item </sensors/temp> {
             reef:rt "temperature-c"
             reef:if "sensor"
             iana:describedby <http://www.example.com/sensors/t123>
             iana:alternate </t>
          }
          reef:rd-item </sensors/light> {
             reef:rt "light-lux"
             reef:if "sensor"
          }
```

The example contains links to three resources of interest on the server: </sensors>, </sensors/temp>, and </sensors/light>. For </sensors>, a content format hint ("ct") and a title ("title") are provided as resource metadata. For both </sensors/temp> and </sensors/light>, a resource type ("rt"), and an interface description ("if") are provided. Additionally, two links are provided that provide further details on </sensors/temp>: a link to a schema describing this resource ("describedby") and a link to a substitute ("alternate").

Common resource metadata are specified in Section 3. The "/.well-known/core" resource and its interface are specified in Section 4.

2.2. Resource Directory

In many deployment scenarios, such as in constrained networks with sleeping servers or in large M2M deployments with bandwidth limited access networks, it is beneficial to deploy resource directory entities that store links to resources stored on other servers. A resource directory can be thought of as a limited search engine for M2M resources.

In this specification, a resource directory provides the same lookup interface as a "/.well-known/core" resource, except that it provides links to resources on potentially many, many different servers. Whereas a "/.well-known/core" resource is populated by the hosting

server, the resource directory provides a registration interface that allows any server (or third party acting on behalf of a server) to register its resources.

The registration interface is a collection resource with the common operations of create, read, update, and delete. The items of the collection are groups of links of type <http://coreapps.org/reef#rd-item> that are to be made available in the lookup interface.

The following example shows a client registering a group of links with a resource directory by making a POST request to the collection resource. The client receives a 2.01 (Created) response with the location of the created collection item. The client can later use this location to update or delete the whole group of links at once.

```
=>  0.02 POST
    Uri-Path: path
    Uri-Path: to
    Uri-Path: resource
    Uri-Path: directory
    Uri-Path: registrations
    Content-Format: TBD3

    #using reef = <http://coreapps.org/reef#>

    #base <coap://[2001:db8:3::124]/>
    reef:rd-item </light/left> { reef:rt "light-lux" reef:ct 0 }
    reef:rd-item </light/middle> { reef:rt "light-lux" reef:ct 0 }
    reef:rd-item </light/right> { reef:rt "light-lux" reef:ct 0 }

<=  2.01 Created
    Location-Path: path
    Location-Path: to
    Location-Path: resource
    Location-Path: directory
    Location-Path: registrations
    Location-Path: 42
```

The following example shows a client performing a lookup on the resource directory by making a GET request to the resource directory resource. The client receives a 2.05 (Content) response with a combined view of all groups of links registered earlier, filtered by a query.

```
=>  0.01 GET
    Uri-Path: path
    Uri-Path: to
    Uri-Path: resource
    Uri-Path: directory
    Uri-Path: lookup
    Uri-Query: rt=light-lux
    Accept: TBD3

<=  2.05 Content
    Content-Format: TBD3

    #using reef = <http://coreapps.org/reef#>

    #base <coap://[2001:db8:1::9]/>
    reef:rd-item </1234> { reef:rt "light-lux" }

    #base <coap://[2001:db8:3::124]/>
    reef:rd-item </light/left> { reef:rt "light-lux" reef:ct 0 }
    reef:rd-item </light/middle> { reef:rt "light-lux" reef:ct 0 }
    reef:rd-item </light/right> { reef:rt "light-lux" reef:ct 0 }
```

The resource directory and its lookup and registration interface are specified in Section 5.

2.3. Notational Conventions

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. Resource Metadata

Both "/.well-known/core" resources and resource directories link to resources of interest using the <http://coreapps.org/reef#rd-item> link relation type. Metadata for these resources can be expressed by nesting the metadata inside these links.

In particular, resource metadata takes the shape of nested links that either directly specify a literal value (such as a string or number) or target a related resource identified by a URI; see Section 2.3 of [I-D.ietf-core-coral] for details.

The following link relation types for expressing resource metadata are defined:

<http://coreapps.org/base#language>
The link target is a hint indicating what the (human-spoken) language of the result of dereferencing the link context should be.

<http://coreapps.org/reef#media>
The link target indicates the intended destination medium or media for style information for the link context.

<http://coreapps.org/base#title>
The link target is a label that it can be used as a human-readable identifier for the link context.

Multiple labels can be specified, each as a link with the label as link target. Each link can have a nested link indicating a language tag for the label.

<http://coreapps.org/coap#type>
The link target is a hint indicating what the content format of the result of dereferencing the link context should be.

<http://coreapps.org/reef#rt>
The link target is an application-specific semantic type of the link context.

Multiple resource types can be specified, each as a link with the resource type as link target. The link target MUST NOT contain multiple resource types separated by white space.

<http://coreapps.org/reef#if>
The link target is a specific interface definition that can be used to interact with the link context.

<http://coreapps.org/reef#sz>
The link target is an indication of the maximum size of the resource representation returned by performing a GET on the link context.

<http://coreapps.org/reef#ct>
The link target is a hint about the Content-Formats that the link context returns.

4. Resource Discovery

Given a host name or IP address, a client can discover the resources of a server implementing this section through the use of a well-known resource [RFC8615]. Well-known resources have a path component that

begins with "/.well-known/". This specification defines a new well-known resource for CoRE Resource Discovery: "/.well-known/core".

4.1. Well-known Resource

The "/.well-known/core" resource is offered by servers implementing this specification on the default port appropriate for the protocol for the purpose of resource discovery. It is up to the server to decide which of the resources in its namespace are included; the "/.well-known/core" resource is generally meant to provide entry points to applications hosted by the server.

A client wishing to discover the resources of a server constructs the URI <{scheme}://{host}:{port}/.well-known/core> from the scheme, host name/IP address, and port. The client then retrieves a CoRAL document from this URI, as specified in [I-D.ietf-core-coral]. The document contains a list of links, each from the well-known resource to one resource hosted by the server, along with resource metadata. The client can filter the list using a number of query parameters.

4.1.1. Resource List Representation

A list of resources is represented as a CoRAL document [I-D.ietf-core-coral] containing the following elements:

- For each resource that the server wishes to advertise to the client, a link of type <http://coreapps.org/reef#rd-item> targeting that resource. The link MUST target a resource in the namespace of the server (same origin). The link MAY have nested links providing resource metadata (including, but not limited to, the resource metadata specified in Section 3).

4.2. Interactions

4.2.1. Getting All Resources

A client can get a list of all resources by making a GET request to <{scheme}://{host}:{port}/.well-known/core>. The request MUST include an Accept option with value TBD3.

On success, the server returns a 2.05 (Content) response with a representation of the list of resources (see Section 4.1.1) that the server wishes to advertise to the client.

Example:

```
=>  0.01 GET
    Uri-Path: .well-known
    Uri-Path: core
    Accept: TBD3

<=  2.05 Content
    Content-Format: TBD3

    #using reef = <http://coreapps.org/reef#>
    #using base = <http://coreapps.org/base#>
    #using iana = <http://www.iana.org/assignments/relation/>

    reef:rd-item </sensors> {
       reef:ct 40
       base:title "Sensor Index"
    }
    reef:rd-item </sensors/temp> {
       reef:rt "temperature-c"
       reef:if "sensor"
       iana:describedby <http://www.example.com/sensors/t123>
       iana:alternate </t>
    }
    reef:rd-item </sensors/light> {
       reef:rt "light-lux"
       reef:if "sensor"
    }
```

4.2.2. Getting Resources By Resource Type

A client can filter a list of resources by resource type by making a GET request to <{scheme}://{host}:{port}/.well-known/core?rt={value}>. The request MUST include an Accept option with value TBD3.

On success, the server returns a 2.05 (Content) response with a representation of the list of resources (see Section 4.1.1), but containing only the subset of links that has resource metadata of type <http://coreapps.org/reef#rt> with the specified text value.

Example:

```
   =>  0.01 GET
       Uri-Path: .well-known
       Uri-Path: core
       Uri-Query: rt=temperature-c
       Accept: TBD3

   <=  2.05 Content
       Content-Format: TBD3

       #using reef = <http://coreapps.org/reef#>
       #using iana = <http://www.iana.org/assignments/relation/>

       reef:rd-item </sensors/temp> {
          reef:rt "temperature-c"
          reef:if "sensor"
          iana:describedby <http://www.example.com/sensors/t123>
          iana:alternate </t>
       }
```

4.2.3. Getting Resources By Interface Type

A client can filter a list of resources by interface type by making a GET request to <{scheme}://{host}:{port}/.well-known/core?if={value}>. The request MUST include an Accept option with value TBD3.

On success, the server returns a 2.05 (Content) response with a representation of the list of resources (see Section 4.1.1), but containing only the subset of links that has resource metadata of type <http://coreapps.org/reef#if> with the specified text value.

Example:

```
   =>  0.01 GET
       Uri-Path: .well-known
       Uri-Path: core
       Uri-Query: if=sensor
       Accept: TBD3

   <=  2.05 Content
       Content-Format: TBD3

       #using reef = <http://coreapps.org/reef#>
       #using iana = <http://www.iana.org/assignments/relation/>

       reef:rd-item </sensors/temp> {
          reef:rt "temperature-c"
          reef:if "sensor"
          iana:describedby <http://www.example.com/sensors/t123>
          iana:alternate </t>
       }
       reef:rd-item </sensors/light> {
          reef:rt "light-lux"
          reef:if "sensor"
       }
```

5. Resource Directory

A resource directory provides information about entry points to applications hosted by other servers. It is intended to make discovery operations more efficient than retrieving each "/.well-known/core" of these servers individually.

5.1. Resource Lookups

A client wishing to discover resources using a resource directory needs to be pre-configured with the URI of a resource directory or acquire the URI through some discovery process. The client then retrieves a CoRAL document from this URI, as specified in [I-D.ietf-core-coral]. The document contains a list of links, each from the resource directory to one of the resources in the directory, along with any registered resource metadata. The client can filter the list either by using a number of query parameters or by submitting a filter query.

5.1.1. Filter Query Representation

TODO.

5.2. Resource Registrations

A server (or a third party acting on behalf of a server) can register resources with a resource directory by submitting a CoRAL document, containing the new links to be created at the directory. The directory processes the submitted links in two ways: First, it includes those links in the list of results to client queries. Second, it creates a resource containing the group of submitted links, such that the server (or third party) can easily update or delete the whole group as a single unit at a later time.

5.2.1. Registration Unit Representation

A registration unit is represented as a CoRAL document [I-D.ietf-core-coral] containing the registered resources as top-level element.

A registered resource is represented as a link where the link relation type is <http://coreapps.org/reef#rd-item> and the link target is the registered resource. This link MAY have metadata about the resource (including, but not limited to, of the type specified in Section 3) as nested elements.

5.2.2. Registration Unit List Representation

A list of registration units is represented as a CoRAL document [I-D.ietf-core-coral] containing the units in the list as top-level elements.

Each registration unit is represented as a link where the link relation type is <http://coreapps.org/reef#rd-unit> and the link target is the registration unit URI.

5.3. Interactions

5.3.1. Getting All Resources

A client can get a list of all resources by making a GET request to the lookup URI.

On success, the server returns a 2.05 (Content) response with a representation of the list of resources (see Section 4.1.1).

Example:

```
   =>  0.01 GET
       Uri-Path: path
       Uri-Path: to
       Uri-Path: resource
       Uri-Path: directory
       Uri-Path: lookup
       Accept: TBD3

   <=  2.05 Content
       Content-Format: TBD3

       #using reef = <http://coreapps.org/reef#>
       #using base = <http://coreapps.org/base#>
       #using iana = <http://www.iana.org/assignments/relation/>

       reef:rd-item <coap://[2001:db8:1::1]/sensors> {
          reef:ct 40
          base:title "Sensor Index"
       }
       reef:rd-item <coap://[2001:db8:1::1]/sensors/temp> {
          reef:rt "temperature-c"
          reef:if "sensor"
          iana:describedby <http://www.example.com/sensors/t123>
          iana:alternate </t>
       }
       reef:rd-item <coap://[2001:db8:1::1]/sensors/light> {
          reef:rt "light-lux"
          reef:if "sensor"
       }
```

5.3.2. Getting Resources By Resource Type

A client can filter a list of resources by resource type by making a GET request to the result of resolving the URI <?rt={value}> relative to the lookup URI.

On success, the server returns a 2.05 (Content) response with a representation of the list of resources (see Section 4.1.1), but containing only the subset of links that has resource metadata of type <http://coreapps.org/reef#rt> with the specified text value.

Example:

```
   =>  0.01 GET
       Uri-Path: path
       Uri-Path: to
       Uri-Path: resource
       Uri-Path: directory
       Uri-Path: lookup
       Uri-Query: rt=temperature-c
       Accept: TBD3

   <=  2.05 Content
       Content-Format: TBD3

       #using reef = <http://coreapps.org/reef#>
       #using iana = <http://www.iana.org/assignments/relation/>

       reef:rd-item <coap://[2001:db8:1::1]/sensors/temp> {
          reef:rt "temperature-c"
          reef:if "sensor"
          iana:describedby <http://www.example.com/sensors/t123>
          iana:alternate </t>
       }
```

5.3.3. Getting Resources By Interface Type

A client can filter a list of resources by interface type by making a GET request to the result of resolving the URI <?if={value}> relative to the lookup URI.

On success, the server returns a 2.05 (Content) response with a representation of the list of resources (see Section 4.1.1), but containing only the subset of links that has resource metadata of type <http://coreapps.org/reef#if> with the specified text value.

Example:

```
   =>  0.01 GET
       Uri-Path: path
       Uri-Path: to
       Uri-Path: resource
       Uri-Path: directory
       Uri-Path: lookup
       Uri-Query: if=sensor
       Accept: TBD3

   <=  2.05 Content
       Content-Format: TBD3

       #using reef = <http://coreapps.org/reef#>
       #using iana = <http://www.iana.org/assignments/relation/>

       reef:rd-item <coap://[2001:db8:1::1]/sensors/temp> {
          reef:rt "temperature-c"
          reef:if "sensor"
          iana:describedby <http://www.example.com/sensors/t123>
          iana:alternate </t>
       }
       reef:rd-item <coap://[2001:db8:1::1]/sensors/light> {
          reef:rt "light-lux"
          reef:if "sensor"
       }
```

5.3.4. Getting Resources By Resource Metadata

A client can filter a list of resources by submitting the representation of a metadata filter (see Section 5.1.1) in a FETCH request to the lookup URI.

On success, the server returns a 2.05 (Content) response with a representation of the list of resources (see Section 4.1.1) that match the filter.

Example:

```
   =>  0.05 FETCH
       Uri-Path: path
       Uri-Path: to
       Uri-Path: resource
       Uri-Path: directory
       Uri-Path: lookup
       Content-Format: TODO
       Accept: TBD3

       TODO

   <=  2.05 Content
       Content-Format: TBD3

       #using reef = <http://coreapps.org/reef#>
       #using iana = <http://www.iana.org/assignments/relation/>

       reef:rd-item <coap://[2001:db8:1::1]/sensors/temp> {
          reef:rt "temperature-c"
          reef:if "sensor"
          iana:describedby <http://www.example.com/sensors/t123>
          iana:alternate </t>
       }
```

5.3.5. Getting All Registration Units

A client can list a collection of registration units by making a GET request to the collection URI.

On success, the server returns a 2.05 (Content) response with a representation of the list of all registration units (see Section 5.2.2) in the collection.

Example:

```
=>  0.01 GET
    Uri-Path: path
    Uri-Path: to
    Uri-Path: resource
    Uri-Path: directory
    Uri-Path: registrations
    Accept: TBD3

<=  2.05 Content
    Content-Format: TBD3

    #using reef = <http://coreapps.org/reef#>

    #base </path/to/resource/directory/registrations/>
    reef:rd-unit <1>
    reef:rd-unit <2>
    reef:rd-unit <3>
    reef:rd-unit <4>
```

5.3.6. Creating a Registration Unit

A client can add a new registration unit to a collection of registration units by submitting a representation of the unit (see Section 5.2.1) in a POST request to the registration collection URI.

On success, the server returns a 2.01 (Created) response indicating the registration unit URI of the new registration unit.

Example:

```
=>  0.02 POST
    Uri-Path: path
    Uri-Path: to
    Uri-Path: resource
    Uri-Path: directory
    Uri-Path: registrations
    Content-Format: TBD3

    #using reef = <http://coreapps.org/reef#>

    #base <coap://[2001:db8:4::1]/>
    reef:rd-item </light/left> { reef:rt "light" reef:ct 0 }
    reef:rd-item </light/middle> { reef:rt "light" reef:ct 0 }
    reef:rd-item </light/right> { reef:rt "light" reef:ct 0 }

<=  2.01 Created
    Location-Path: path
    Location-Path: to
    Location-Path: resource
    Location-Path: directory
    Location-Path: registrations
    Location-Path: 42
```

5.3.7. Reading a Registration Unit

A client can read a registration unit by making a GET request to the registration unit URI.

On success, the server returns a 2.05 (Content) response with a representation of the registration unit (see Section 5.2.1).

Example:

```
=>  0.01 GET
    Uri-Path: path
    Uri-Path: to
    Uri-Path: resource
    Uri-Path: directory
    Uri-Path: registrations
    Uri-Path: 42
    Accept: TBD3

<=  2.05 Content
    Content-Format: TBD3

    #using reef = <http://coreapps.org/reef#>

    #base <coap://[2001:db8:4::1]>
    reef:rd-item </light/left> { reef:rt "light" reef:ct 0 }
    reef:rd-item </light/middle> { reef:rt "light" reef:ct 0 }
    reef:rd-item </light/right> { reef:rt "light" reef:ct 0 }
```

5.3.8. Updating a Registration Unit

A client can update a resource registration by submitting the representation of the updated registration (see Section 5.2.1) in a PUT request to the topic URI. Any existing registrations in the registration unit are replaced by this update.

On success, the server returns a 2.04 (Updated) response.

Example:

```
   =>  0.03 PUT
       Uri-Path: path
       Uri-Path: to
       Uri-Path: resource
       Uri-Path: directory
       Uri-Path: registrations
       Uri-Path: 42
       Content-Format: TBD3

       #using reef = <http://coreapps.org/reef#>

       #base <coap://[2001:db8:4::1]/>
       reef:rd-item </light/left> { reef:rt "light" reef:ct 0 }
       reef:rd-item </light/right> { reef:rt "light" reef:ct 0 }

   <=  2.04 Changed
```

5.3.9. Deleting a Registration Unit

A client can delete a registration unit by making a DELETE request on the registration unit URI.

On success, the server returns a 2.02 (Deleted) response.

Example:

```
=>  0.04 DELETE
    Uri-Path: path
    Uri-Path: to
    Uri-Path: resource
    Uri-Path: directory
    Uri-Path: registrations
    Uri-Path: 42

<=  2.02 Deleted
```

6. Security Considerations

TODO.

7. IANA Considerations

7.1. CoRE Dictionary

This document creates a new registry named "CoRAL Dictionary for CoRE" under the Constrained RESTful Environments (CoRE) Parameters" registry [CORE-PARAMETERS] for use with the CoRAL binary format [I-D.ietf-core-coral]. The registry is located at <http://TBD5/>.

[[NOTE TO RFC EDITOR: Please replace all occurrences of "http://TBD5/" in this document with the URI of the new registry.]]

The registry is a mapping between a key and a value. The key is an integer in the range 0 to 2147483647 (2^31-1). The value is either an Internationalized Resource Identifier (IRI) reference, a Boolean value, an integer, a floating-point number, a date/time value, a byte string, or a text string. Both the key and the value are to be denoted in the CoRAL textual format [I-D.ietf-core-coral] and must be unique within the registry. A reference may be provided to offer more information about a value.

The registry policy is Expert Review.

The initial entries in the registry are as follows:

- Key: 0
  Value: <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
  Reference: [W3C.REC-rdf-schema-20140225]

- Key: 1
  Value: <http://www.iana.org/assignments/relation/item>
  Reference: [RFC6573]

- Key: 2
  Value: <http://www.iana.org/assignments/relation/collection>
  Reference: [RFC6573]

- Key: 3
  Value: <http://coreapps.org/collections#create>
  Reference: [I-D.ietf-core-coral]

- Key: 4
  Value: <http://coreapps.org/base#update>
  Reference: [I-D.ietf-core-coral]

- Key: 5
  Value: <http://coreapps.org/collections#delete>
  Reference: [I-D.ietf-core-coral]

- Key: 6
  Value: <http://coreapps.org/base#search>
  Reference: [I-D.ietf-core-coral]

- Key: 7
  Value: <http://coreapps.org/coap#accept>
  Reference: [I-D.ietf-core-coral]

- Key: 8
  Value: <http://coreapps.org/reef#rd-unit>
  Reference: [I-D.hartke-t2trg-coral-reef]

- Key: 9
  Value: <http://coreapps.org/reef#rd-item>
  Reference: [I-D.hartke-t2trg-coral-reef]

- Key: 10
  Value: <http://coreapps.org/base#language>
  Reference: [I-D.ietf-core-coral]

- Key: 11
  Value: <http://coreapps.org/reef#media>
  Reference: [I-D.hartke-t2trg-coral-reef]

- Key: 12
  Value: <http://coreapps.org/base#title>
  Reference: [I-D.ietf-core-coral]

- Key: 13
  Value: <http://coreapps.org/coap#type>
  Reference: [I-D.ietf-core-coral]

- Key: 14
  Value: <http://coreapps.org/reef#rt>
  Reference: [I-D.hartke-t2trg-coral-reef]

- Key: 15
  Value: <http://coreapps.org/reef#if>
  Reference: [I-D.hartke-t2trg-coral-reef]

- Key: 16
  Value: <http://coreapps.org/reef#sz>
  Reference: [I-D.hartke-t2trg-coral-reef]

- Key: 17
  Value: <http://coreapps.org/reef#ct>
  Reference: [I-D.hartke-t2trg-coral-reef]

- Key: 18
  Value: </.well-known/core>
  Reference: [I-D.hartke-t2trg-coral-reef]

- Key: 19
  Value: <http://coreapps.org/base#direction>
  Reference: [I-D.ietf-core-coral]

- Key: 20
  Value: "ltr"
  Reference:

- Key: 21
  Value: "rtl"
  Reference:

- Key: 22
  Value: <http://coreapps.org/coap#method>
  Reference: [I-D.ietf-core-coral]

- Key: 23
  Value: <http://coreapps.org/base#representation>
  Reference: [I-D.ietf-core-coral]

7.2. CoAP Content Format

This document registers a CoAP content format for CoRAL documents in the binary format that use the registry established in Section 7.1. The registration is in accordance with the procedures of RFC 7252 [RFC7252].

- Content Type: application/coral+cbor;dictionary="http://TBD5/"
  Content Coding: identity
  ID: TBD3
  Reference: [I-D.hartke-t2trg-coral-reef]

[[NOTE TO RFC EDITOR: Please replace all occurrences of "TBD3" in this document with the code point assigned by IANA.]]

[[NOTE TO IMPLEMENTERS: Experimental implementations can use content format ID 65088 until IANA has assigned a code point.]]

8. References

8.1. Normative References

[I-D.ietf-core-coral]
Hartke, K., "The Constrained RESTful Application Language (CoRAL)", draft-ietf-core-coral-03 (work in progress), March 2020.

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

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8615] Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", RFC 8615, DOI 10.17487/RFC8615, May 2019, <https://www.rfc-editor.org/info/rfc8615>.

8.2. Informative References

[CORE-PARAMETERS]
IANA, "Constrained RESTful Environments (CoRE) Parameters", <http://www.iana.org/assignments/core-parameters>.

[I-D.ietf-core-resource-directory]
Shelby, Z., Koster, M., Bormann, C., Stok, P., and C. Amsuess, "CoRE Resource Directory", draft-ietf-core-resource-directory-24 (work in progress), March 2020.

[REST] Fielding, R., "Architectural Styles and the Design of Network-based Software Architectures", Ph.D. Dissertation, University of California, Irvine, 2000, <http://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>.

[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, <https://www.rfc-editor.org/info/rfc3986>.

[RFC6573] Amundsen, M., "The Item and Collection Link Relations", RFC 6573, DOI 10.17487/RFC6573, April 2012, <https://www.rfc-editor.org/info/rfc6573>.

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

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

[RFC8288] Nottingham, M., "Web Linking", RFC 8288, DOI 10.17487/RFC8288, October 2017, <https://www.rfc-editor.org/info/rfc8288>.

[W3C.REC-rdf-schema-20140225] Brickley, D. and R. Guha, "RDF Schema 1.1", World Wide Web Consortium Recommendation REC-rdf-schema-20140225, February 2014, <http://www.w3.org/TR/2014/REC-rdf-schema-20140225>.

Acknowledgements

Thanks to Christian Amsuess, Carsten Bormann and Jim Schaad for helpful comments and discussions that have shaped the document.

Author's Address

Klaus Hartke
Ericsson
Torshamnsgatan 23
Stockholm SE-16483
Sweden

Email: klaus.hartke@ericsson.com