Resource Discovery in Constrained RESTful Environments (CoRE)
Using the Constrained RESTful Application Language (CoRAL)
draft-hartke-t2trg-coral-reef-01

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 September 12, 2019.

Copyright Notice

Copyright (c) 2019 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 (https://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

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 there are no humans 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.hartke-t2trg-coral].

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

Resource Discovery
   Allows a client to discover the resources of a server given a host name or IP address.

Resource Directory
   Allows a client to discover the resources of several servers given the URL of a resource directory.
   Allows a server (or a third party acting on behalf of a server) to register resources with a resource directory given a 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, this 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 links
may have nested resource metadata. A client would then find an
appropriate resource based on the metadata. Metadata queries may
also be specified the query string to filter the result set.

The following example shows how a client discovers the resources of a
CoAP server by making a GET request to the "/.well-known/core"
resource. The client receives a 2.05 (Content) response with a list
of links of type <http://TBD6/rd-item>. The links contain nested
elements with resource metadata (such as resource type, interface
description, available content formats, and related resources).

```plaintext
=> 0.01 GET
   Uri-Path: .well-known
   Uri-Path: core
   Accept: TBD3
<= 2.05 Content
   Content-Format: TBD3

   #using <http://TBD6/>
   #using iana = <http://www.iana.org/assignments/relation/>

   rd-item </sensors> {
      ct 40
      title "Sensor Index"
   }

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

   rd-item </sensors/light> {
      rt "light-lux"
      if "sensor"
   }
```

In detail, this representation 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 `<sensors/temp>` as well as `<sensors/light>`, a resource type ("rt") and an interface description ("if") are provided as resource metadata. Additionally, two links are provided with further detail on `<sensors/temp>`: one to a schema describing this resource ("describedby") and one to an substitute for this resource ("alternate").

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 different servers. For populating the resource directory, a registration interface is offered. 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://TBD6/rd-item>` that are to be made available in the lookup interface.

The following example shows how a client registers a group of links with a CoAP 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 use this location later to update the group of links or delete them from the directory.

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

#using <http://TBD6/>

#base <coap://[2001:db8:3::124]/>
rd-item </light/left> { rt "light-lux" ct 0 }
rd-item </light/middle> { rt "light-lux" ct 0 }
rd-item </light/right> { rt "light-lux" ct 0 }
```
The following example shows how a client performs 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.

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

The following example shows how a client performs 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.

=> 2.05 Content
Content-Format: TBD3

#using <http://TBD6/>

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

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

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://TBD6/rd-item> link relation type. Metadata for these resources can be expressed by nesting further links inside these links.
The following link relation types for resource metadata are defined:

<http://TBD6/hreflang>
   The link target is a hint indicating what the (human-spoken) language of the result of dereferencing the link context should be.

<http://TBD6/media>
   The link target indicates the intended destination medium or media for style information for the link context.

<http://TBD6/title>
   The link target is a label that it can be used as a human-readable identifier for the link context.

   The link can have a nested link containing language information.

<http://TBD6/type>
   The link target is a hint indicating what the media type of the result of dereferencing the link context should be.

<http://TBD6/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://TBD6/if>
   The link target is a specific interface definition that can be used to interact with the link context.

<http://TBD6/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://TBD6/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 [I-D.nottingham-rfc5785bis]. 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. How It Works

A server implementing this specification offers a "/.well-known/core" resource on the default port appropriate for the protocol for the purpose of resource discovery. It is up to the server 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 an initial URI from a template using the given host name or IP address. It then retrieves a CoRAL representation, as specified in [I-D.hartke-t2trg-coral]. The representation contains a list of links, each from the well-known resource to one resource, along with resource metadata. The client can filter the list using a number of query parameters.
4.2. API Reference

4.2.1. Get All Resources

Request

Request Method: GET

URI Template: coap://{host-and-port}/.well-known/core

Accept Option: TBD3

Response

When successful, the server returns a 2.05 (Content) response with a representation in content format TBD3 ('application/coral+cbor; dictionary=http://TBD5/'). The representation MUST contain zero or more links of type <http://TBD6/rd-item>, each of which MUST target a resource in the namespace of the server (same origin). The links may have nested elements providing resource metadata.

Example

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

<= 2.05 Content
   Content-Format: TBD3

   #using <http://TBD6/>
   #using iana = <http://www.iana.org/assignments/relation/>

   rd-item </sensors> {
      ct 40
      title "Sensor Index"
   }
   rd-item </sensors/temp> {
      rt "temperature-c"
      if "sensor"
      iana:describedby <http://www.example.com/sensors/t123>
      iana:alternate </t>
   }
   rd-item </sensors/light> {
      rt "light-lux"
      if "sensor"
   }
4.2.2. Get Resources By Resource Type

Request

Request Method: GET

URI Template: coap://{host-and-port}/.well-known/core?rt={value}

Accept Option: TBD3

Response

When successful, same response kind as in Section 4.2.1.

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 <http://TBD6/>
   #using iana = <http://www.iana.org/assignments/relation/>

   rd-item </sensors/temp> {
     rt "temperature-c"
     if "sensor"
       iana:describedby <http://www.example.com/sensors/t123>
       iana:alternate </t>
   }
4.2.3. Get Resources By Interface Type

Request

Request Method: GET

URI Template: coap://{host-and-port}/.well-known/core?if={value}

Accept Option: TBD3

Response

When successful, same response kind as in Section 4.2.1.

Example

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

<= 2.05 Content
   Content-Format: TBD3

#using <http://TBD6/>
#using iana = <http://www.iana.org/assignments/relation/>

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

rd-item </sensors/light> {
   rt "light-lux"
   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 performing a lookup on the "/.well-known/core" of each of these servers.

5.1. How It Works

A client wishing to discover resources using a resource directory needs to be configured with the URI of a resource directory or acquire it through some discovery process. The client then retrieves the representation as specified in [I-D.hartke-t2trg-coral]. The resource directory serves a list of links, each from the resource directory to one of the resources, along with the resource metadata. The client can filter the list using a number of query parameters.

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

5.2.1. Get All Resources

Request

Request Method: GET

URI Template: {resource-directory-location}

Accept Option: TBD3

Response

When successful, same response kind as in Section 4.2.1.

Example

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

<= 2.05 Content
   Content-Format: TBD3

#using <http://TBD6/>
#using iana = <http://www.iana.org/assignments/relation/>

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

Request

Request Method: GET

URI Template: {resource-directory-location}?rt={value}

Accept Option: TBD3

Response

When successful, same response kind as in Section 4.2.1.

Example

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

<= 2.05 Content
   Content-Format: TBD3

   #using <http://TBD6/>
   #using iana = <http://www.iana.org/assignments/relation/>

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

Request

Request Method: GET
URI Template: {resource-directory-location}?if={value}
Accept Option: TBD3

Response

When successful, same response kind as in Section 4.2.1.

Example

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

<= 2.05 Content
   Content-Format: TBD3

#using <http://TBD6/>
#using iana = <http://www.iana.org/assignments/relation/>

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

Request

Request Method: GET

URI Template: {resource-directory-location}

Accept Option: TBD3

Response

When successful, the server returns a 2.05 (Content) response with a representation in content format TBD3. The representation contains or zero or more links with the `<http://TBD6/rd-unit>` link relation type, each of which has a resource registration as the link target.

Example

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

<= 2.05 Content
   Content-Format: TBD3

   #using <http://TBD6/>

   rd-unit </rd/1>
   rd-unit </rd/2>
   rd-unit </rd/3>
   rd-unit </rd/4>
5.2.5. Create Resource Registration

Request

Request Method: POST

URI Template: {resource-directory-location}

Content-Format Option: TBD3

The client submits a representation in content format TBD3. The representation contains zero or more links with the </rd-item> link relation type, each of which has a resource to be registered as the link target.

Response

When successful, the server returns a 2.01 (Created) response indicating the location at which the resource registration was created.

Example

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

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

<= 2.01 Created
   Location-Path: path
   Location-Path: to
   Location-Path: resource
   Location-Path: directory
   Location-Path: 42
5.2.6. Read Resource Registration

Request

Request Method: GET

URI Template: {registration-location}

Accept Option: TBD3

Response

When successful, same response kind as in Section 4.2.1.

Example

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

<= 2.05 Content
   Content-Format: TBD3

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

Request

Request Method: PUT

URI Template: {registration-location}

Content-Format Option: TBD3

The client submits a representation in content format TBD3. The representation contains zero or more links with the \texttt{<http://TBD6/rd-item>} link relation type, each of which has a resource to be registered as the link target. Any existing list of resources at the location is replaced by this update.

Response

When successful, the server returns a 2.04 (Changed) response.

Example

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

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

<= 2.04 Changed
5.2.8. Delete Resource Registration

Request

Request Method: DELETE

URI Template: {registration-location}

Response

When successful, 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: 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.hartke-t2trg-coral]. The registry is located at <http://TBD5/>.

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.hartke-t2trg-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: <urn:TBD1#create>
  - Reference: [I-D.hartke-t2trg-coral]

- **Key**: 4
  - Value: <urn:TBD1#update>
  - Reference: [I-D.hartke-t2trg-coral]

- **Key**: 5
  - Value: <urn:TBD1#delete>
Reference: [I-D.hartke-t2trg-coral]

- **Key: 6**
  - Value: `<urn:TBD1#search>`
  - Reference: [I-D.hartke-t2trg-coral]

- **Key: 7**
  - Value: `<urn:TBD1#accept>`
  - Reference: [I-D.hartke-t2trg-coral]

- **Key: 8**
  - Value: `<http://TBD6/rd-unit>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 9**
  - Value: `<http://TBD6/rd-item>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 10**
  - Value: `<http://TBD6/hreflang>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 11**
  - Value: `<http://TBD6/media>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 12**
  - Value: `<http://TBD6/title>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 13**
  - Value: `<http://TBD6/type>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 14**
  - Value: `<http://TBD6/rt>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 15**
  - Value: `<http://TBD6/if>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 16**
  - Value: `<http://TBD6/sz>`
  - Reference: [I-D.hartke-t2trg-coral-reef]

- **Key: 17**
  - Value: `<http://TBD6/ct>`
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] [I-D.hartke-t2trg-coral]

8. References

8.1. Normative References

[I-D.hartke-t2trg-coral]
Hartke, K., "The Constrained RESTful Application Language (CoRAL)", draft-hartke-t2trg-coral-08 (work in progress), March 2019.

[I-D.nottingham-rfc5785bis]
Nottingham, M., "Well-Known Uniform Resource Identifiers (URIs)", draft-nottingham-rfc5785bis-09 (work in progress), February 2019.


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",

[REST]
Ph.D. Dissertation, University of California, Irvine, 2000,

STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005,

[RFC6690] Shelby, Z., "Constrained RESTful Environments (CoRE) Link Format",
RFC 6690, DOI 10.17487/RFC6690, August 2012,

RFC 7228, DOI 10.17487/RFC7228, May 2014,

[RFC8288] Nottingham, M., "Web Linking",
RFC 8288, DOI 10.17487/RFC8288, October 2017,
Acknowledgements

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

Author’s Address

Klaus Hartke
Ericsson
Torshamngatan 23
Stockholm SE-16483
Sweden

Email: klaus.hartke@ericsson.com