CoRE Working Group
Internet-Draft
Intended status: Standards Track
Expires: 5 September 2024

C. Amsüss

T. Fossati
ARM

4 March 2024

The Constrained RESTful Application Language (CoRAL) draft-ietf-core-coral-06

Abstract

The Constrained RESTful Application Language (CoRAL) defines a data model and interaction model as well as a compact serialization formats for the description of typed connections between resources on the Web ("links"), possible operations on such resources ("forms"), and simple resource metadata.

Note to Readers

This note is to be removed before publishing as an RFC.

The issues list for this Internet-Draft can be found at <https://github.com/core-wg/coral/labels/coral>. Companion material for this Internet-Draft can be found at <https://github.com/core-wg/coral>.

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 5 September 2024.

Copyright Notice

Table of Contents

1. Introduction

The Constrained RESTful Application Language (CoRAL) is a language for the description of typed connections between resources on the Web ("links"), possible operations on such resources ("forms"), and simple resource metadata.

CoRAL is intended for driving automated software agents that navigate a Web application based on a standardized vocabulary of link relation types and operation types. It is designed to be used in conjunction with a Web transfer protocol, such as the Hypertext Transfer Protocol (HTTP) [RFC7230] or the Constrained Application Protocol (CoAP) [RFC7252].

This document defines the CoRAL data model and interaction model as well as a compact serialization format.

1.1. Data and Interaction Model

The data model is similar to the Resource Description Framework (RDF) [W3C.REC-rdf11-concepts-20140225] model, with provisions to enable form based interaction and to express data from Web Linking ([RFC8288]) based models such as [RFC6690]'s Link Format.

The interaction model derives from the processing model of HTML [W3C.REC-html52-20171214] and specifies how an automated software agent can change the application state by navigating between resources following links and performing operations on resources submitting forms.

1.2. 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.

Terms defined in this document appear in _cursive_ where they are introduced (rendered in plain text as the new term surrounded by underscores).

2. Data and Interaction Model

The Constrained RESTful Application Language (CoRAL) is designed for building Web-based applications [W3C.REC-webarch-20041215] in which automated software agents navigate between resources by following links and perform operations on resources by submitting forms.

2.1. Browsing Context

Borrowing from HTML 5 [W3C.REC-html52-20171214], each such agent maintains a _browsing context_ in which the representations of Web resources are processed. (In HTML, the browsing context typically corresponds to a tab or window in a Web browser.)

At any time, one representation in a browsing context is designated the _active_ representation.

2.2. Documents

A resource representation in one of the CoRAL serialization formats is called a CoRAL _document_. The URI that was used to retrieve such a document is called the document's _retrieval context_. This URI is also considered the base URI for relative URI references in the document.

A CoRAL document consists of a list of zero or more statements that can express links or (in a composition of statements) forms. CoRAL serialization formats may contain additional elements for efficiency or convenience, such as an embedded base URI that takes precedence over the document's base URI, or to concisely represent compound statements (e.g., to express forms).

2.3. Data model

The _basic CoRAL information model_ is similar to the Resource Description Framework (RDF) [W3C.REC-rdf11-concepts-20140225] information model: Data is expressed as an (unordered) set of triples (also called statements), consisting of a subject, a predicate and an object. The predicate is always a URI, the subject is a URI or a blank node, and the object is either a URI, a blank node or a litreal. All URIs here are limited to the syntax-based normalized form of [RFC3986] Section 6.2.2.

Blank nodes are unnamed entities. Literals are CBOR objects.

These triples form a directed multigraph with the subject and object being source and destination, and the predicate a description on the edge. That graph is equivalent to the data.

To form a set and a graph, we define an equivalence relation: URIs are only equal to URIs and if they are identical byte-wise. A blank node is only equal to itself. A literal is equal to a different literal if its value is equal to the other literal's value in the CBOR generic data model.

Triples are equivalent to each other if their subject, predicate and object are pair-wise equivalent.

The _CoRAL structured information model_ is a sequence of "passings" of the basic model's edges, starting at a node identifying the document (its retrieval context, typically URI from which it was obtained) where

* each edge is passed at least one time in total,

* each edge is passed at most one time after each passing that ends in its start point (with the obvious exception that edges from the retrieval context can be passed once from the start), and

* between a passing of an edge from A to B and a later passing from B to C, passings can only be along edges that can be reached from B along the graph, until B is the end of a different passing.

For better understanding, think of the structured information model as a sort of tree spanning from the retrieval context, with the oddity that when a node is reached along two different edges (which a normal tree doesn't do), it is up to the builder of the tree whether to describe anything children of the entered node on one parent or on the other parent, on both, or to describe some children at the first and others at a later occasion.

Exceeding the RDF-like model, this represents CoRAL's focus on the discovery of possivble future application states over the description of a graph of resources.

2.3.1. Observations

The structured form of a data set is in general not unique: If a node has more than one child, their sequence can be varied. If a node has more than one parent, its children may be expressed on any non-empty set of its parents to obtain a structured data set that expresses the same data set.

In general, arbitrary basic data can not be expressed in a structured data set, because

* There may not be a tree that covers the directed graph, or the tree's root may not be the retrieval context.

* There may be multiple edges into a blank node.

In particular, the precise data from one structured information document can only be expressed with the same retrieval context. However, statements can be added to make a data set that is expressible elsewhere (this document defines the carries-information-about relation type leading to the http://www.iana.org/assignments/relation/carries-information-about predicate being usable here), and subsets of the data can be taken and expressed.

Forms are not special in the information model, but are merely statements around a blank node. They can be special in serialization formats (which have more efficient notations for them), and are used by the interaction model for special operations.

The structured information model contains more information than the basic information model. [TBD put this into a different context because it's not an observation any more:] Which precise structure is picked is to suit the processing application, typically by profiling the information and its serialization. It is recommended that the information encoded in the structure (including the order) be derived from data available in the general data set, even though the statements that guide the structure are not necessarily encoded in the subset of data that is being structured.

Serializations like the one in Section 3 have even more choices than the structured information model: They can choose to use or not use packed CBOR to compress parts, can spell out URIs in full or use relative references, or can exercise freedoms of the CBOR encoding. Variation there is not to have an influence on the interpretation of a CoRAL document.

2.3.2. Possible variations

* Each URI is tagged with whether it is intended to be dereferenced or used as an identifier.

2.3.3. Examples

This subsection illustrates the information model and serialization based on an example from [RFC6690]:

```
</sensors>;ct=40;title="Sensor Index",
</sensors/temp>;rt="temperature-c";if="sensor",
</sensors/light>;rt="light-lux";if="sensor",
<http://www.example.com/sensors/t123>;anchor="/sensors/temp";rel="describedby",
</t>;anchor="/sensors/temp";rel="alternate"
```

Figure 1: Original example at coap://.../.well-known/core

After an extraction described in Appendix C.2, this list represents the content of the basic information model representing the above. For the basic model, the table is to be considered unsorted in the first step.

Subject	Predicate	Object
coap://.../	rel:hosts	coap://.../sensors
coap://.../sensors	linkformat:ct	40
coap://.../sensors	linkformat:title	"Sensor Index"
coap://.../	http://www.iana.org/assignments/relation/hosts	coap://.../sensors/temp
coap://.../sensors/temp	linkformat:rt	rt:temperature-c
coap://.../sensors/temp	linkformat:if	if:sensor
coap://.../sensors/temp	rel:describedby	http://www.example.com/sensors/t123
coap://.../sensors/temp	rel:alternate	coap://.../t
coap://.../	http://www.iana.org/assignments/relation/hosts	coap://.../sensors/light
coap://.../sensors/light	linkformat:rt	rt:light-lux

coap://.../sensors/light	linkformat:if	if:sensor

Table 1: Basic (and, through the sequence, Strucutred) Information Model extracted from there (using CURIEs: rel = http://www.iana.org/assignments/relation/, linkformat is TBD in the conversion, if, rt is TBD with IANA).

During extraction, some information on item ordering was preserved into the structured data. Note that while the CoRAL structured data preserves some sequence aspects of the Link-Format file (like the order of attributes), others (like the relative order of links from different contexts) are deemed irrelevant and not preserved.

For serialization, the use of the packing described with the conversion results in a binary CBOR file with this CBOR diagnostic notation:

```
[
  [2, simple(10) / item 10 for rel:hosts /, cri"/sensors", [
    [2, 6(2) / item 20 for linkformat:ct /, 40],
    [2, simple(15) / item 15 for linkformat:title /, "Sensor Index"]
  ]],
  [2, simple(10) / item 10 for rel:hosts /, cri"/sensors/temp", [
    [2, 6(1) / item 18 for linkformat:if /, 6(200) / cri"http:TBD...temperature-c
" /],
    [2, 6(-2) / item 19 for linkformat:rt /, 6(250) / cri"http:TBD...sensor" /],
    [2, simple(12) / item 12 for rel:describedby /, cri"http://www.example.com/se
nsors/t123"],
    [2, simple(11) / item 11 for rel:alternate /, cri"/t"]
  ]],
  [2, 10 / item10 for rel:hosts /, cri"/sensors/light", [
    [2, 6(1) / item 18 for linkformat:if /, 6(-201)],
    [2, 6(-2) / item 19 for linkformat:rt /, 6(250)]
  ]]
]
```

Figure 2: Serialized CoRAL file in diagnostic notation.

[TBD: Numbers are made up]

Note that the "temperature-c" interface and "sensor" resource type get code points in the link-format dictionary because they are of reg-name style and thus would be registered as CoRE Parameters, and be included in the packing as well.

2.3.3.1. Literal example

To illustrate non-trivial literals, a link example of [RFC8288] is converted.

(Note that even the conversion scheme hinted at above for [RFC6690] link format makes no claims at being applicable to general purpose web links like the below; this is merely done to demonstrate how literals can be handled. The example even so happens well illustrate that point: General link attributes may only be valid on the target when the link is followed in that direction ("letztes Kapitel" means last chapter), whereas convertible link-format documents use titles that apply to the described resource independent of which link is currently being followed.)

```
 Link: </TheBook/chapter2>;
         rel="previous"; title*=UTF-8'de'letztes%20Kapitel,
```

Figure 3: Original link about a book chapter from RFC8288

The model this would be converted to is:

Subject	Predicate	Object
http://.../	rel:previous	http://.../TheBook/chapter2
http://.../TheBook/chapter2	linkformat:title	"letztes Kapitel" with language tag "de"

Table 2: Information model extracted from above

In CBOR serialization, this produces:

```
[
  [2, 6(...) / rel:previous /, cri"/TheBook/chapter2", [
    [2, simple(15) / item 15 for linkformat:title /, 38(["de", "letztes Kapitel"]
)]
  ]]
]
```

Figure 4: Serialization of the RFC8288-based example

2.4. Serialization Format

The primary serialization format is a compact, binary encoding of links and forms in Concise Binary Object Representation (CBOR) [RFC8949]. This format is intended for environments with constraints on power, memory, and processing resources [RFC7228] and shares many similarities with the message format of CoAP: In place of verbose strings, small numeric identifiers are used to encode link relation types and operation types. Uniform Resource Identifiers (URIs) [RFC3986] are expressed as Constrained Resource Identifier (CRI) references [I-D.ietf-core-href] and thus pre-parsed for easy use with CoAP. As a result, link serializations in CoRAL are often much more compact and easier to process than equivalent serializations in CoRE Link Format [RFC6690].

For easy representation of CoRAL documents in text, CBOR diagnostic notation is used. Along with indentation and comments, the notation introduced in [I-D.bormann-cbor-edn-literals] is used to represent CRIs. This format is not expected to be sent over the network.

[To be discussed: For even better readability, the RDF Turtle [W3C.REC-turtle-20140225] format can be used when only the basic information model content is to be conveyed. When used like this, the conversion according to the RDF appendix is implied.]

2.5. Links

Any statement "links" a resource with a second resource or literal, and is thus also referred to as a link.

In [RFC8288] terminology, a CoRAL link's subject is the _link context_, the predicate is the _link relation type_, and the object is the _link target_.

However, a link in CoRAL does not have target attributes. Instead, a link may have a list of zero or more nested elements. These enable both the description of resource metadata and the chaining of links, which is done in [RFC8288] by setting the anchor of one link to the target of another.

> A link can be viewed as a statement of the form "{link context} has a {link relation type} resource at {link target}" where the link target may be further described by nested elements.

A link relation type identifies the semantics of a link. In HTML and in [RFC8288], link relation types are typically denoted by an IANA-registered name, such as stylesheet or type. In CoRAL, all link relation types are, in contrast, denoted by a Universal Resource Identifier (URI) [RFC3986], such as <http://www.iana.org/assignments/relation/stylesheet> or <http://www.w3.org/1999/02/22-rdf-syntax-ns#type>. This allows for the decentralized creation of new link relation types without the risk of collisions when they come from different organizations or domains of knowledge. URIs can also lead to documentation, schema, and other information about a link relation type. In CoRAL documents, these URIs are only used as identity tokens, though, and are compared with Simple String Comparison as specified in Section 6.2.1 of [RFC3986].

If the link target is a URI and the URI scheme indicates a Web transfer protocol like HTTP or CoAP, an agent can dereference the URI and navigate the browsing context to its target resource; this is called _following the link_. An anonymous resource is a resource that is identified by neither a URI nor a literal representation. The agent can still follow the link, but can not dereference it and is limited in its next steps by the outgoing links that are expressed in the current document.

A link can occur as a top-level element in a document or as a nested element within a link. When a link occurs as a top-level element, the link context implicitly is the document's retrieval context. When a link occurs nested within a link, the link context of the nested link is the link target of the enclosing link.

There are no restrictions on the cardinality of links; there can be multiple links to and from a particular target, and multiple links of the same or different types between a given link context and target. However, the nesting nature of the data model constrains the description of resource relations to a tree: Relations between linked resources can only be described by further nesting links.

2.6. Forms

A _form_ provides instructions to an agent for performing an operation on a resource on the Web. A form has a _form context_, an _operation type_, a _request method_, and a _submission target_. Additionally, a form may be accompanied by a list of zero or more _form fields_.

In the basic information model, the form is identified with an anonymous node. The form context and operation type are the subject and predicate of an incoming link, respectively; request method and submission target of an outgoing link. Form fields are additional links from that form.

> A form can be viewed as an instruction of the form "To perform an {operation type} operation on {form context}, make a {request method} request to {submission target}" where the request may be further described by form fields.

An operation type identifies the semantics of the operation. Operation types are denoted (like link relation types) by a URI.

Form contexts and submission targets are both denoted by a URI. The form context is the resource on which the operation is ultimately performed. To perform the operation, an agent needs to construct a request with the specified method as the request method and the specified submission target as the request URI. Usually, the submission target is the same resource as the form context, but may be a different resource. Constructing and sending the request is called _submitting the form_.

A form can occur as a top-level element in a document or as a nested element within a link. When a form occurs as a top-level element, the form context implicitly is the document's retrieval context. When a form occurs nested within a link, the form context is the link target of the enclosing link.

2.7. Form Fields

Form fields can be used to provide more detailed instructions to agents for constructing the request when submitting a form. For example, a form field could instruct an agent to include a certain payload or header field in the request. A payload could, for instance, be described by form fields providing acceptable media types, a reference to schema information, or a number of individual data items that the agents needs to supply. Form fields can be specific to the Web transfer protocol that is used for submitting the form.

A form field is a pair of a _form field type_ and a _form field value_. Additionally, a form field may have a list of zero or more nested elements that further describe the form field value.

A form field type identifies the semantics of the form field. Form field types are predicates and thus URIs. Form field values are URIs, blank nodes or literals.

2.8. Navigation

An agent begins the interaction with an application by performing a GET request on an _entry point URI_. The entry point URI is the only URI that the agent is expected to know beforehand. From then on, the agent is expected to make all requests by following links and submitting forms that are provided in the responses resulting from the requests. The entry point URI could be obtained through some discovery process or manual configuration.

If dereferencing the entry point URI yields a CoRAL document (or any other representation that implements the CoRAL data and interaction model), the agent makes this document the active representation in the browsing context and proceeds as follows:

1. The first step for the agent is to decide what to do next, i.e., which type of link to follow or form to submit, based on the link relation types and operation types it understands.

   An agent may follow a link without understanding the link relation type, e.g., for the sake of pre-fetching or building a search index. However, an agent MUST NOT submit a form without understanding the operation type.

2. The agent then finds the link(s) or form(s) with the respective type in the active representation. This may yield one or more candidates, from which the agent will have to select the most appropriate one. The set of candidates can be empty, for example, when an application state transition is not supported or not allowed.

3. The agent selects one of the candidates based on the metadata associated them (in the form of form fields and nested elements) and their order of appearance in the document. Examples for relevant metadata could include the indication of a media type for the target resource representation, the URI scheme of a target resource, or the request method of an operation.

4. The agent obtains the _request URI_ from the link target or submission target. Fragment identifiers are not part of the request URI and MUST be separated from the rest of the URI prior to the next step.

5. The agent constructs a new request with the request URI. If the agent is following a link, then the request method MUST be GET. If the agent is submitting a form, then the request method MUST be the one supplied by the form.

   The agent SHOULD set HTTP header fields and CoAP request options according to the metadata (e.g., set the HTTP Accept header field or the CoAP Accept option when a media type for the target resource is provided). Depending on the operation type of a form, the agent may also have to include a request payload that matches the specifications of some form fields.

6. The agent sends the request and receives the response.

7. If a fragment identifier was separated from the request URI, the agent selects the fragment indicated by the fragment identifier within the received representation according to the semantics of its media type.

8. The agent updates the browsing context by making the (selected fragment of the) received representation the active representation.

9. Finally, the agent processes the representation according to the semantics of its media type. If the representation is a CoRAL document (or any other representation that implements the CoRAL data and interaction model), the agent again has the choice of what to do next. Go to step 1.

2.9. History Traversal

A browsing context has a _session history_, which lists the resource representations that the agent has processed, is processing, or will process.

A session history consists of session history entries. The number of session history entries may be limited and dependent on the agent. An agent with severe constraints on memory size might only have enough memory for the most recent entry.

An entry in the session history consists of a resource representation and the representation's retrieval context. New entries are added to the session history as the agent navigates from resource to resource, discarding entries that are no longer used.

An agent can decide to navigate a browsing context (in addition to following links and submitting forms) by _traversing the session history_. For example, when an agent receives a response with a representation that does not contain any further links or forms, it can navigate back to a resource representation it has visited earlier and make that the active representation.

Traversing the history SHOULD take advantage of caches to avoid new requests. An agent may reissue a safe request (e.g., a GET) when it does not have a fresh representation in its cache. An agent MUST NOT reissue an unsafe request (e.g., a PUT or POST) unless it actually intends to perform that operation again.

2.10. Designing interactions in an Open World

CoRAL can be used to build both open world systems ("if something is not said, it may or may not be true") and closed world systems ("if something is not said, it is not true").

In constrained environments (and the web in general), partial representations are often used for efficiency. For example, a device can query another for particular statements using a yet to be defined FETCH version of CoRAL. It is expected that some tools (e.g., server or agent libraries) require the application to be tolerant of unprocessed statements. Furthermore, it can be easier to evolve applications and their packing dictionaries if loss of statements leads to graceful degradation.

Therefore, it is convenient to build applications on open world assumptions. Such applications can only use statements that add possibilities, and none that limit interactions. Any limitations need to be encoded in statements the agent necesarily has to perform an action in the first place, and can then be relaxed in additional statements.

For example, an application built with open-world assumptions can not create a form that allows feeding gremlins, and in an additional statement (e.g., a form field) forbid after midnight. Instead, the application needs to describe a limited-feeding form, which can only be used if any of the attached conditions is met; the condition "before midnight" can then be expressed in an additional statement.

3. Binary Format

This section defines the encoding of documents in the CoRAL binary format.

A document in the binary format is encoded in Concise Binary Object Representation (CBOR) [RFC8949].

The CBOR structure of a document is presented in the Concise Data Definition Language (CDDL) [RFC8610]. All CDDL rules not defined in this document are defined in Appendix D of [RFC8610].

The media type of documents in the binary format is application/coral+cbor.

3.1. Data Structure

The data structure of a document in the binary format is made up of three kinds of elements: links, forms (as short hands for the statements they are constructed of), and (as an extension to the CoRAL data model) directives. Directives provide a way to encode URI references with a common base more efficiently.

3.1.1. Documents

A document in the binary format is encoded as a CBOR array that contains zero or more elements. An element is either a link, a form, or a directive.

```
document = [*element]
element = link / form / directive
```

The elements are processed in the order they appear in the document. Document processors need to maintain an _environment_ while iterating an array of elements. The environment consists of two variables: the _current context_ and the _current base_. The current context and the current base are both initially set to the document's retrieval context.

3.1.2. Directives

Directives provide the ability to manipulate the environment while processing elements.

There is a single type of directives available: the Base directive.

```
directive = base-directive
```

It is an error if a document processor encounters any other type of directive.

3.1.2.1. Base Directives

A Base directive is encoded as a CBOR array that contains the unsigned integer 1 and a base URI.

```
base-directive = [1, baseURI]
```

The base URI is denoted by a Constrained Resource Identifier (CRI) reference [I-D.ietf-core-href]. The CRI reference MUST be resolved against the current context (not the current base).

```
baseURI = CRI-Reference
CRI-Reference = <Defined in Section XX of RFC XXXX>
```

The directive is processed by resolving the CRI reference against the current context and assigning the result to the current base.

3.1.3. URIs

URIs in links and forms are encoded as CRI references.

```
URI = CRI-Reference
```

A CRI reference is processed by resolving it to a URI as specified in Section 5.2 of [I-D.ietf-core-href] using the current base.

3.1.4. Links

A link is encoded as a CBOR array that contains the unsigned integer 2, the link relation type, the link target, and, optionally, an array of zero or more nested elements.

```
link = [2, relation-type, link-target, ?[*element]]
```

The link relation type is a URI.

```
relation-type = URI
```

The link target is either a URI, a literal value, or null.

```
link-target = URI / literal / null
literal = bool / int / float / time / bytes / text
```

The nested elements, if any, MUST be processed in a fresh environment. The current context is set to the link target of the enclosing link. The current base is initially set to the link target, if the link target is a URI; otherwise, it is set to the current base of the current environment.

3.1.5. Forms

A form is encoded as a CBOR array that contains the unsigned integer 3, the operation type, the submission target, and, optionally, an array of zero or more form fields.

```
form = [3, operation-type, submission-target, ?[*form-field]]
```

The operation type is a URI.

```
operation-type = URI
```

The submission target is a URI.

```
submission-target = URI
```

The request method is either implied by the operation type or encoded as a form field. If both are given, the form field takes precedence over the operation type. Either way, the method MUST be applicable to the Web transfer protocol identified by the scheme of the submission target.

The form fields, if any, MUST be processed in a fresh environment. The current context is set to an unspecified URI that represents the enclosing form. The current base is initially set to the submission target of the enclosing form.

3.1.6. Form Fields

A form field is encoded as a CBOR sequence that consists of a form field type, a form field value, and, optionally, an array of zero or more nested elements.

```
form-field = (form-field-type, form-field-value, ?[*element])
```

The form field type is a URI.

```
form-field-type = URI
```

The form field value is either a URI, a literal value, or null.

```
form-field-value = URI / literal / null
```

The nested elements, if any, MUST be processed in a fresh environment. The current context is set to the form field value of the enclosing form field. The current base is initially set to the form field value, if the form field value is a URI; otherwise, it is set to the current base of the current environment.

3.2. Dictionary Compression

A document in the binary format MAY reference values from an external dictionary using Packed CBOR [I-D.ietf-cbor-packed]. This helps to reduce representation size and processing cost.

Dictionary references can be used subject to [yet to be defined] profiling.

Implementers should note that Packed CBOR is not designed to be uncompressed, but to be used in a compressed form. In particular, constrained devices may operate without even knowing what a given dictionary entry expands to (as long as they know its meaning) .

3.2.1. Media Type Parameter

The application/coral+cbor media type for documents in the binary format is defined to have a dictionary parameter that specifies the dictionary in use. The dictionary is identified by a URI. For example, a CoRAL document that uses the dictionary identified by the URI <http://example.com/dictionary> would have the following content type:

```
application/coral+cbor;dictionary="http://example.com/dictionary"
```

The URI serves only as an identifier; it does not necessarily have to be dereferencable (or even use a dereferencable URI scheme). It is permissible, though, to use a dereferencable URI and to serve a representation that provides information about the dictionary in a machine- or human-readable way. (The representation format and security considerations of such a representation are outside the scope of this document.)

For simplicity, a CoRAL document can reference values only from one dictionary; the value of the dictionary parameter MUST be a single URI.

The dictionary parameter is OPTIONAL. If it is absent, the default dictionary specified in Appendix B of this document is assumed.

Once a dictionary has made an assignment, the assignment MUST NOT be changed or removed. A dictionary, however, may contain additional information about an assignment, which may change over time.

In CoAP, media types (including specific values for their parameters, plus an optional content coding) are encoded as an unsigned integer called the "content format" of a representation. For use with CoAP, each new CoRAL dictionary therefore needs to have a new content format registered in the CoAP Content Formats Registry [CORE-PARAMETERS].

3.3. Export Interface

The definition of documents, links, and forms in the CoRAL binary format can be reused in other CBOR-based protocols. Specifications using CDDL should reference the following rules for this purpose:

```
CoRAL-Document = document
CoRAL-Link = link
CoRAL-Form = form
```

For each embedded document, link, and form, the CBOR-based protocol needs to specify the document retrieval context, link context, and form context, respectively.

4. Document Semantics

4.1. Submitting Documents

By default, a CoRAL document is a representation that captures the current state of a resource. The meaning of a CoRAL document changes when it is submitted in a request. Depending on the request method, the CoRAL document can capture the intended state of a resource (PUT) or be subject to application-specific processing (POST).

4.1.1. PUT Requests

A PUT request with a CoRAL document enclosed in the request payload requests that the state of the target resource be created or replaced with the state described by the CoRAL document. A successful PUT of a CoRAL document generally means that a subsequent GET on that same target resource would result in an equivalent document being sent in a success response.

An origin server SHOULD verify that a submitted CoRAL document is consistent with any constraints the server has for the target resource. When a document is inconsistent with the target resource, the origin server SHOULD either make it consistent (e.g., by removing inconsistent elements) or respond with an appropriate error message containing sufficient information to explain why the document is unsuitable.

The retrieval context and the base URI of a CoRAL document in a PUT are the request URI of the request.

4.1.2. POST Requests

A POST request with a CoRAL document enclosed in the request payload requests that the target resource process the CoRAL document according to the resource's own specific semantics.

The retrieval context of a CoRAL document in a POST is defined by the target resource's processing semantics; it may be an unspecified URI. The base URI of the document is the request URI of the request.

4.2. Returning Documents

In a response, the meaning of a CoRAL document changes depending on the request method and the response status code. For example, a CoRAL document in a successful response to a GET represents the current state of the target resource, whereas a CoRAL document in a successful response to a POST might represent either the processing result or the new resource state. A CoRAL document in an error response represents the error condition, usually describing the error state and what next steps are suggested for resolving it.

4.2.1. Success Responses

Success responses have a response status code that indicates that the client's request was successfully received, understood, and accepted (2xx in HTTP, 2.xx in CoAP). When the representation in a success response does not describe the state of the target resource, it describes result of processing the request. For example, when a request has been fulfilled and has resulted in one or more new resources being created, a CoRAL document in the response can link to and describe the resource(s) created.

The retrieval context and the base URI of a CoRAL document representing the current state of a resource are the request URI of the request.

The retrieval context of a CoRAL document representing a processing result is an unspecified URI that refers to the processing result itself. The base URI of the document is the request URI of the request.

4.2.2. Redirection Responses

Redirection responses have a response status code that indicates that further action needs to be taken by the agent (3xx in HTTP). A redirection response, for example, might indicate that the target resource is available at a different URI or the server offers a choice of multiple matching resources, each with its own specific URI.

In the latter case, the representation in the response might contain a list of resource metadata and URI references (i.e., links) from which the agent can choose the most preferred one.

The retrieval context of a CoRAL document representing such multiple choices in a redirection response is an unspecified URI that refers to the redirection itself. The base URI of the document is the request URI of the request.

4.2.3. Error Responses

Error response have a response status code that indicates that either the request cannot be fulfilled or the server failed to fulfill an apparently valid request (4xx or 5xx in HTTP, 4.xx or 5.xx in CoAP). A representation in an error response describes the error condition.

The retrieval context of a CoRAL document representing such an error condition is an unspecified URI that refers to the error condition itself. The base URI of the document is the request URI of the request.

5. Usage Considerations

This section discusses some considerations in creating CoRAL-based applications and vocabularies.

5.1. Specifying CoRAL-based Applications

CoRAL-based applications naturally implement the Web architecture [W3C.REC-webarch-20041215] and thus are centered around orthogonal specifications for identification, interaction, and representation:

* Resources are identified by URIs or represented by literal values.

* Interactions are based on the hypermedia interaction model of the Web and the methods provided by the Web transfer protocol. The semantics of possible interactions are identified by link relation types and operation types.

* Representations are CoRAL documents encoded in the binary format defined in Section 3. Depending on the application, additional representation formats may be used.

5.1.1. Application Interfaces

Specifications for CoRAL-based applications need to list the specific components used in the application interface and their identifiers. This should include the following items:

* The Web transfer protocols supported.

* The representation formats used, identified by their Internet media types, including the CoRAL serialization formats.

* The link relation types used.

* The operation types used. Additionally, for each operation type, the permissible request methods.

* The form field types used. Additionally, for each form field type, the permissible form field values.

5.1.2. Resource Identifiers

URIs are a cornerstone of Web-based applications. They enable the uniform identification of resources and are used every time a client interacts with a server or a resource representation needs to refer to another resource.

URIs often include structured application data in the path and query components, such as paths in a filesystem or keys in a database. It is a common practice in HTTP-based application programming interfaces (APIs) to make this part of the application specification, i.e., to prescribe fixed URI templates that are hard-coded in implementations. However, there are a number of problems with this practice [RFC8820].

In CoRAL-based applications, resource names are therefore not part of the application specification --- they are an implementation detail. The specification of a CoRAL-based application MUST NOT mandate any particular form of resource name structure.

[RFC8820] describes the problematic practice of fixed URI structures in more detail and provides some acceptable alternatives.

5.1.3. Implementation Limits

This document places no restrictions on the number of elements in a CoRAL document or the depth of nested elements. Applications using CoRAL (in particular those running in constrained environments) may limit these numbers and define specific implementation limits that an implementation must support at least to be interoperable.

Applications may also mandate the following and other restrictions:

* Use of only either HTTP or CoAP as the supported Web transfer protocol.

* Use of only dictionary references in the binary format for certain vocabulary.

* Use of URI references and CRI references only up to a specific length.

5.2. Minting Vocabulary

New link relation types, operation types, and form field types can be minted by defining a URI that uniquely identifies the item. Although the URI may point to a resource that contains a definition of the semantics, clients SHOULD NOT automatically access that resource to avoid overburdening its server. The URI SHOULD be under the control of the person or party defining it, or be delegated to them.

To avoid interoperability problems, it is RECOMMENDED that only URIs are minted that are normalized according to Section 6.2 of [RFC3986]. This is easily achieved when the URIs are defined in CRI form (in which they also become part of the dictionary), as this avoids many common non-normalized forms of URIs by construction.

Non-normalized forms that are still to be avoided include:

* Uppercase characters in scheme names and domain names

* Explicitly stated HTTP default port (e.g., <http://example.com/> is preferable over <http://example.com:80/>)

* Punycode-encoding of Internationalized Domain Names in URIs

* URIs that are not in Unicode Normalization Form C

URIs that identify vocabulary do not need to be registered. The inclusion of domain names in URIs allows for the decentralized creation of new URIs without the risk of collisions.

However, URIs can be relatively verbose and impose a high overhead on a representation. This can be a problem in constrained environments [RFC7228]. Therefore, CoRAL alternatively allows the use of packed references that abbreviate CBOR data items from a dictionary, as specified in Section 3.2. These impose a much smaller overhead but instead need to be assigned by an authority to avoid collisions.

5.3. Expressing Registered Link Relation Types

Link relation types registered in the Link Relations Registry [LINK-RELATIONS], such as collection [RFC6573] or icon [W3C.REC-html52-20171214], can be used in CoRAL by appending the registered name to the URI <http://www.iana.org/assignments/relation/>:

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

iana:collection </items>
iana:icon       </favicon.png>
```

The convention of appending the relation type name to the prefix <http://www.iana.org/assignments/relation/> to form URIs is adopted from the Atom Syndication Format [RFC4287]; see also Appendix A.2 of [RFC8288].

Note that registered relation type names are required to be lowercase ASCII letters (see Section 3.3 of [RFC8288]).

5.4. Expressing Simple RDF Statements

In RDF [W3C.REC-rdf11-concepts-20140225], a statement says that some relationship, indicated by a predicate, holds between two resources. Existing RDF vocabularies can therefore be a good source for link relation types that describe resource metadata. For example, a CoRAL document could use the FOAF vocabulary [FOAF] to describe the person or software that made it:

```
#using rdf = <http://www.w3.org/1999/02/22-rdf-syntax-ns#>
#using foaf = <http://xmlns.com/foaf/0.1/>

foaf:maker null {
   rdf:type        <http://xmlns.com/foaf/0.1/Person>
   foaf:familyName "Hartke"
   foaf:givenName  "Klaus"
   foaf:mbox       <mailto:klaus.hartke@ericsson.com>
}
```

5.5. Expressing Natural Language Texts

Text strings can be associated with a Language Tag [RFC5646] and a base text direction (right-to-left or left-to-right) by using CBOR tag 38.

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

iana:terms-of-service </tos> {
   base:title 38(["de", "Nutzungsbedingungen"])
   base:title 38(["en-US", "Terms of use"])
   base:title 38(["az", "ltr", "stifad rtlri"])
}
```

[Maturity note: Whether direction will actually be expressed in an updated tag 38, how precisely that is done, or whether a new tag will be allocated for text with direction is currently still under discussion.]

5.6. Embedding Representations in CoRAL

When a document links to many Web resources and an agent needs a representation of each of them, it can be inefficient to retrieve each representations individually. To minimize round-trips, documents can embed representations of resources.

A representation can be embedded in a document by including a link of type <http://coreapps.org/base#representation>:

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

iana:icon </favicon.gif> {
   base:representation
      b64'R0lGODlhAQABAAAAACH5BAEKAAEALAAAAAABAAEAAAIAOw==' {
      http:type "image/gif"
   }
}
```

An embedded representation SHOULD have a nested link of type <http://coreapps.org/http#type> or <http://coreapps.org/coap#type> that indicates the content type of the representation.

The link relation types <http://coreapps.org/base#representation>, <http://coreapps.org/http#type>, and <http://coreapps.org/coap#type> are defined in Appendix A.

6. Security Considerations

CoRAL document processors need to be fully prepared for all types of hostile input that may be designed to corrupt, overrun, or achieve control of the agent processing the document. For example, hostile input may be constructed to overrun buffers, allocate very big data structures, or exhaust the stack depth by setting up deeply nested elements. Processors need to have appropriate resource management to mitigate these attacks.

CoRAL serialization formats intentionally do not feature the equivalent of XML entity references so as to preclude the entire class of attacks relating to them, such as exponential XML entity expansion ("billion laughs") [CAPEC-197] and malicious XML entity linking [CAPEC-201].

Implementers of the CoRAL binary format need to consider the security aspects of decoding CBOR. See Section 10 of [RFC8949] for security considerations relating to CBOR. In particular, different number encodings for the same numeric value are not equivalent in CoRAL (e.g., a floating-point value of 0.0 is not the same as the integer 0).

CoRAL makes extensive use of resource identifiers. See Section 7 of [RFC3986] for security considerations relating to URIs. See Section 7 of [I-D.ietf-core-href] for security considerations relating to CRIs.

The security of applications using CoRAL can depend on the proper preparation and comparison of internationalized strings. For example, such strings can be used to make authentication and authorization decisions, and the security of an application could be compromised if an entity providing a given string is connected to the wrong account or online resource based on different interpretations of the string. See [RFC6943] for security considerations relating to identifiers in URIs and other strings.

CoRAL is intended to be used in conjunction with a Web transfer protocol like HTTP or CoAP. See Section 9 of [RFC7230], Section 9 of [RFC7231], etc., for security considerations relating to HTTP. See Section 11 of [RFC7252] for security considerations relating to CoAP.

CoRAL does not define any specific mechanisms for protecting the confidentiality and integrity of CoRAL documents. It relies on security mechanisms on the application layer or transport layer for this, such as Transport Layer Security (TLS) [RFC8446].

CoRAL documents and the structure of a web of resources revealed from automatically following links can disclose personal information and other sensitive information. Implementations need to prevent the unintentional disclosure of such information. See Section 9 of [RFC7231] for additional considerations.

Applications using CoRAL ought to consider the attack vectors opened by automatically following, trusting, or otherwise using links and forms in CoRAL documents. See Section 5 of [RFC8288] for related considerations.

In particular, when a CoRAL document is the representation of a resource, the server that is authoritative for that resource may not necessarily be authoritative for nested elements in the document. In this case, unless an application defines specific rules, any link or form where the link/form context and the document's retrieval context do not share the same Web Origin [RFC6454] should be discarded ("same-origin policy").

7. IANA Considerations

7.1. Media Type "application/coral+cbor"

This document registers the media type application/coral+cbor according to the procedures of [RFC6838].

Type name:
application

Subtype name:
coral+cbor

Required parameters:
N/A

Optional parameters:
dictionary - See Section 3.2 of [I-D.ietf-core-coral].

Encoding considerations:
binary - See Section 3 of [I-D.ietf-core-coral].

Security considerations:
See Section 6 of [I-D.ietf-core-coral].

Interoperability considerations:
N/A

Published specification:
[I-D.ietf-core-coral]

Applications that use this media type:
See Section 1 of [I-D.ietf-core-coral].

Fragment identifier considerations:
As specified for application/cbor.

Additional information:
Deprecated alias names for this type: N/A
Magic number(s): N/A

File extension(s): .coral.cbor
Macintosh file type code(s): N/A

Person & email address to contact for further information:
See the Author's Address section of [I-D.ietf-core-coral].

Intended usage:
COMMON

Restrictions on usage:
N/A

Author:
See the Author's Address section of [I-D.ietf-core-coral].

Change controller:
IESG

Provisional registration?
No

7.2. CoAP Content Formats

This document registers CoAP content formats for the content types application/coral+cbor and text/coral according to the procedures of [RFC7252].

* Content Type: application/coral+cbor
  Content Coding: identity
  ID: TBD3
  Reference: [I-D.ietf-core-coral]

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

[[NOTE TO IMPLEMENTERS: Experimental implementations may use content format ID 65087 for application/coral+cbor until IANA has assigned code points.]]

8. References

8.1. Normative References

[I-D.bormann-cbor-edn-literals]
Bormann, C., "Application-Oriented Literals in CBOR Extended Diagnostic Notation", Work in Progress, Internet-Draft, draft-bormann-cbor-edn-literals-02, 28 March 2023, <https://datatracker.ietf.org/doc/html/draft-bormann-cbor-edn-literals-02>.

[I-D.ietf-cbor-packed]
Bormann, C. and M. Gütschow, "Packed CBOR", Work in Progress, Internet-Draft, draft-ietf-cbor-packed-12, 2 March 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-cbor-packed-12>.

[I-D.ietf-core-href]
Bormann, C. and H. Birkholz, "Constrained Resource Identifiers", Work in Progress, Internet-Draft, draft-ietf-core-href-14, 9 January 2024, <https://datatracker.ietf.org/doc/html/draft-ietf-core-href-14>.

[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/rfc/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/rfc/rfc3339>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

[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/rfc/rfc3986>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/rfc/rfc4648>.

[RFC5234] Crocker, D., Ed. and P. Overell, "Augmented BNF for Syntax Specifications: ABNF", STD 68, RFC 5234, DOI 10.17487/RFC5234, January 2008, <https://www.rfc-editor.org/rfc/rfc5234>.

[RFC5646] Phillips, A., Ed. and M. Davis, Ed., "Tags for Identifying Languages", BCP 47, RFC 5646, DOI 10.17487/RFC5646, September 2009, <https://www.rfc-editor.org/rfc/rfc5646>.

[RFC6454] Barth, A., "The Web Origin Concept", RFC 6454, DOI 10.17487/RFC6454, December 2011, <https://www.rfc-editor.org/rfc/rfc6454>.

[RFC6657] Melnikov, A. and J. Reschke, "Update to MIME regarding "charset" Parameter Handling in Textual Media Types", RFC 6657, DOI 10.17487/RFC6657, July 2012, <https://www.rfc-editor.org/rfc/rfc6657>.

[RFC6838] Freed, N., Klensin, J., and T. Hansen, "Media Type Specifications and Registration Procedures", BCP 13, RFC 6838, DOI 10.17487/RFC6838, January 2013, <https://www.rfc-editor.org/rfc/rfc6838>.

[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/rfc/rfc8174>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/rfc/rfc8949>.

[Unicode] The Unicode Consortium, "The Unicode Standard, Version 13.0.0", ISBN 978-1-936213-26-9, March 2020, <https://www.unicode.org/versions/Unicode13.0.0/>.

8.2. Informative References

[CAPEC-197]
MITRE, "CAPEC-197: XML Entity Expansion", September 2019, <https://capec.mitre.org/data/definitions/197.html>.

[CAPEC-201]
MITRE, "CAPEC-201: XML Entity Linking", September 2019, <https://capec.mitre.org/data/definitions/201.html>.

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

[FOAF] Brickley, D. and L. Miller, "FOAF Vocabulary Specification 0.99", January 2014, <http://xmlns.com/foaf/spec/20140114.html>.

[HAL] Kelly, M., "JSON Hypertext Application Language", Work in Progress, Internet-Draft, draft-kelly-json-hal-11, 19 October 2023, <https://datatracker.ietf.org/doc/html/draft-kelly-json-hal-11>.

[HTTP-METHODS]
IANA, "Hypertext Transfer Protocol (HTTP) Method Registry", <http://www.iana.org/assignments/http-methods>.

[I-D.ietf-httpapi-linkset]
Wilde, E. and H. Van de Sompel, "Linkset: Media Types and a Link Relation Type for Link Sets", Work in Progress, Internet-Draft, draft-ietf-httpapi-linkset-10, 5 May 2022, <https://datatracker.ietf.org/doc/html/draft-ietf-httpapi-linkset-10>.

[LINK-RELATIONS]
IANA, "Link Relations", <http://www.iana.org/assignments/link-relations>.

[MEDIA-TYPES]
IANA, "Media Types", <http://www.iana.org/assignments/media-types>.

[RFC4287] Nottingham, M., Ed. and R. Sayre, Ed., "The Atom Syndication Format", RFC 4287, DOI 10.17487/RFC4287, December 2005, <https://www.rfc-editor.org/rfc/rfc4287>.

[RFC5789] Dusseault, L. and J. Snell, "PATCH Method for HTTP", RFC 5789, DOI 10.17487/RFC5789, March 2010, <https://www.rfc-editor.org/rfc/rfc5789>.

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

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

[RFC6943] Thaler, D., Ed., "Issues in Identifier Comparison for Security Purposes", RFC 6943, DOI 10.17487/RFC6943, May 2013, <https://www.rfc-editor.org/rfc/rfc6943>.

[RFC7089] Van de Sompel, H., Nelson, M., and R. Sanderson, "HTTP Framework for Time-Based Access to Resource States -- Memento", RFC 7089, DOI 10.17487/RFC7089, December 2013, <https://www.rfc-editor.org/rfc/rfc7089>.

[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/rfc/rfc7228>.

[RFC7230] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing", RFC 7230, DOI 10.17487/RFC7230, June 2014, <https://www.rfc-editor.org/rfc/rfc7230>.

[RFC7231] Fielding, R., Ed. and J. Reschke, Ed., "Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content", RFC 7231, DOI 10.17487/RFC7231, June 2014, <https://www.rfc-editor.org/rfc/rfc7231>.

[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/rfc/rfc7252>.

[RFC8132] van der Stok, P., Bormann, C., and A. Sehgal, "PATCH and FETCH Methods for the Constrained Application Protocol (CoAP)", RFC 8132, DOI 10.17487/RFC8132, April 2017, <https://www.rfc-editor.org/rfc/rfc8132>.

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

[RFC8446] Rescorla, E., "The Transport Layer Security (TLS) Protocol Version 1.3", RFC 8446, DOI 10.17487/RFC8446, August 2018, <https://www.rfc-editor.org/rfc/rfc8446>.

[RFC8820] Nottingham, M., "URI Design and Ownership", BCP 190, RFC 8820, DOI 10.17487/RFC8820, June 2020, <https://www.rfc-editor.org/rfc/rfc8820>.

[UAX31] The Unicode Consortium, "Unicode Standard Annex #31: Unicode Identifier and Pattern Syntax", Revision 33, March 2020, <https://www.unicode.org/reports/tr31/tr31-33.html>.

[UTR36] The Unicode Consortium, "Unicode Technical Report #36: Unicode Security Considerations", Revision 15, September 2014, <https://www.unicode.org/reports/tr36/tr36-15.html>.

[UTS39] The Unicode Consortium, "Unicode Technical Standard #39: Unicode Security Mechanisms", Revision 22, February 2020, <https://www.unicode.org/reports/tr39/tr39-22.html>.

[W3C.REC-html52-20171214]
Danilo, A., Ed., Eicholz, A., Ed., Moon, S., Ed., Faulkner, S., Ed., and T. Leithead, Ed., "HTML 5.2", W3C REC REC-html52-20171214, W3C REC-html52-20171214, 14 December 2017, <https://www.w3.org/TR/2017/REC-html52-20171214/>.

[W3C.REC-rdf-schema-20140225]
Brickley, D., Ed. and R. Guha, Ed., "RDF Schema 1.1", W3C REC REC-rdf-schema-20140225, W3C REC-rdf-schema-20140225, 25 February 2014, <https://www.w3.org/TR/2014/REC-rdf-schema-20140225/>.

[W3C.REC-rdf11-concepts-20140225]
Wood, D., Ed., Lanthaler, M., Ed., and R. Cyganiak, Ed., "RDF 1.1 Concepts and Abstract Syntax", W3C REC REC-rdf11-concepts-20140225, W3C REC-rdf11-concepts-20140225, 25 February 2014, <https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/>.

[W3C.REC-turtle-20140225]
Prud'hommeaux, E., Ed. and G. Carothers, Ed., "RDF 1.1 Turtle", W3C REC REC-turtle-20140225, W3C REC-turtle-20140225, 25 February 2014, <https://www.w3.org/TR/2014/REC-turtle-20140225/>.

[W3C.REC-webarch-20041215]
Jacobs, I., Ed. and N. Walsh, Ed., "Architecture of the World Wide Web, Volume One", W3C REC REC-webarch-20041215, W3C REC-webarch-20041215, 15 December 2004, <https://www.w3.org/TR/2004/REC-webarch-20041215/>.

Appendix A. Core Vocabulary

This section defines the core vocabulary for CoRAL: a set of link relation types, operation types, and form field types.

A.1. Base

Link Relation Types:

<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
: Indicates that the link's context is an instance of the class specified as the link's target, as defined by RDF Schema [W3C.REC-rdf-schema-20140225].

<http://coreapps.org/base#title>
: Indicates that the link target is a human-readable label (e.g., a menu entry).

  The link target MUST be a literal. The text string SHOULD be wrapped in a tag indicating language and, if necessary, direction if applicable.

<http://coreapps.org/base#representation>
: Indicates that the link target is a representation of the link context.

  The link target MUST be a byte string.

  The representation may be a full, partial, or inconsistent version of the representation served from the URI of the resource.

  A link with this link relation type can occur as a top-level element in a document or as a nested element within a link. When it occurs as a top-level element, it provides an alternate representation of the document's retrieval context. When it occurs nested within a link, it provides a representation of link target of the enclosing link.

Operation Types:

<http://coreapps.org/base#update>
: Indicates that the state of the form's context can be replaced with the state described by a representation submitted to the server.

  This operation type defaults to the PUT method [RFC7231] [RFC7252] for both HTTP and CoAP. Typical overrides by a form field include the PATCH method [RFC5789] [RFC8132] for HTTP and CoAP and the iPATCH method [RFC8132] for CoAP.

<http://coreapps.org/base#search>
: Indicates that the form's context can be searched by submitting a search query.

This operation type defaults to the POST method [RFC7231] for HTTP and the FETCH method [RFC8132] for CoAP. Typical overrides by a form field include the POST method [RFC7252] for CoAP.

A.2. Collections

Link Relation Types:

<http://www.iana.org/assignments/relation/item>
Indicates that the link's context is a collection and that the link's target is a member of that collection, as defined in Section 2.1 of [RFC6573].

<http://www.iana.org/assignments/relation/collection>
Indicates that the link's target is a collection and that the link's context is a member of that collection, as defined in Section 2.2 of [RFC6573].

Operation Types:

<http://coreapps.org/collections#create>
Indicates that the form's context is a collection and that a new item can be created in that collection with the state defined by a representation submitted to the server.

This operation type defaults to the POST method [RFC7231] [RFC7252] for both HTTP and CoAP.

<http://coreapps.org/collections#delete>
Indicates that the form's context is a member of a collection and that the form's context can be removed from that collection.

This operation type defaults to the DELETE method [RFC7231] [RFC7252] for both HTTP and CoAP.

A.3. HTTP

Form Field Types:

<http://coreapps.org/http#method>
Specifies the HTTP method for the request.

The form field value MUST be a text string in the format defined in Section 4.1 of [RFC7231]. The possible set of values is maintained in the HTTP Methods Registry [HTTP-METHODS].

A form field of this type MUST NOT occur more than once in a form. If absent, it defaults to the request method implied by the form's operation type.

<http://coreapps.org/http#accept>
Specifies an acceptable HTTP content type for the request payload. There may be multiple form fields of this type. If a form does not include a form field of this type, the server accepts any or no request payload, depending on the operation type.

The form field value MUST be a text string in the format defined in Section 3.1.1.1 of [RFC7231]. The possible set of media types and their parameters is maintained in the Media Types Registry [MEDIA-TYPES].

Link Relation Types:

<http://coreapps.org/http#type>
Specifies the HTTP content type of the link context.

The link target MUST be a text string in the format defined in Section 3.1.1.1 of [RFC7231]. The possible set of media types and their parameters is maintained in the Media Types Registry [MEDIA-TYPES].

A.4. CoAP

Form Field Types:

<http://coreapps.org/coap#method>
Specifies the CoAP method for the request.

The form field value MUST be an integer identifying a CoAP method (e.g., the integer 2 for the POST method). The possible set of values is maintained in the CoAP Method Codes Registry [CORE-PARAMETERS].

A form field of this type MUST NOT occur more than once in a form. If absent, it defaults to the request method implied by the form's operation type.

<http://coreapps.org/coap#accept>
Specifies an acceptable CoAP content format for the request payload. There may be multiple form fields of this type. If a form does not include a form field of this type, the server accepts any or no request payload, depending on the operation type.

The form field value MUST be an integer identifying a CoAP content format. The possible set of values is maintained in the CoAP Content Formats Registry [CORE-PARAMETERS].

Link Relation Types:

<http://coreapps.org/coap#type>
Specifies the CoAP content format of the link context.

The link target MUST be an integer identifying a CoAP content format (e.g., the integer 42 for the content type application/octet-stream without a content coding). The possible set of values is maintained in the CoAP Content Formats Registry [CORE-PARAMETERS].

Appendix B. Default Dictionary

This section defines a default dictionary that is assumed when the application/coral+cbor media type is used without a dictionary parameter.

Key	Value
0	<http://www.w3.org/1999/02/22-rdf-syntax-ns#type>
1	<http://www.iana.org/assignments/relation/item>
2	<http://www.iana.org/assignments/relation/collection>
3	<http://coreapps.org/collections#create>
4	<http://coreapps.org/base#update>
5	<http://coreapps.org/collections#delete>
6	<http://coreapps.org/base#search>
7	<http://coreapps.org/coap#accept>
8	<http://coreapps.org/coap#type>
10	<http://coreapps.org/coap#method>
14	<http://coreapps.org/base#representation>

Table 3: Default Dictionary

Appendix C. Mappings to other formats

While CoRAL has an information model of its own, its data can be converted to different extents with other data formats.

Using these conversions is generally application specific, i.e., this document does not claim equivalence of (say) a given RDF its converted CoRAL document, but applications can choose use these conversions if the limitations described with the conversion are acceptable to them.

C.1. RDF

[TBD: Expand / introduce the common CURIEs used here.]

RDF and the CoRAL Basic Information Model can be interconverted losslessly, as long as some basic restrictions are met:

* All involved IRIs (on the RDF side) and CRIs (on the CoRAL side) can be converted; that means that round-tripping IRIs through CoRAL converts them to the equivalent URIs.

  The precise limitations of what CRIs can not express are described in [I-D.ietf-core-href] and out of scope of this document.

  A possible extension to CoRAL that allows tagged URIs in place of CRIs could remove this limitation. (CRIs that can not be expressed as URIs are not valid anyway).

* A blank node of CoRAL can only have one incoming edge in serialization. RDF documents with multiply connected blank nodes need to undergo skolemization before they can be expressed in CoRAL.

* CoRAL supports arbitrary literal objects, including CBOR tags. For each object that is used in a literal, a mapping to a datatype (typically XSD) needs to be defined.

  When literals are normalized in RDF according to XSD rules, or the literal mappings to RDF datatypes are ambiguous on the CoRAL side, round-tripping CoRAL through RDF can be lossy to the extent of the normalization or ambiguity.

* As always with expressing arbitrary graphs of the Basic Information Model in serialization, if there is no directed tree spanning the directed graph, statements need to be introduced to reach some topics.

Each statement in RDF is mapped to a statement in CoRAL. Any IRI it contains in RDF is mapped to an equivalent CRI in CoRAL and vice versa. Any blank node of RDF is converted to a blank node (serialized as a null) in CoRAL. (Beware that depending on the context established in Section 4, the retrieval context may be a URI or a blank node). Literals are converted as follows:

* CBOR text strings are coverted to RDF string literals without a language tag.

* CBOR literals from the following list are converted to their corresponding text representations of the datatype from the following table:

CDDL	XSD datatype
bool	xsd:boolean
integer	xsd:integer
float	xsd:double
decfrac	xsd:decimal
bytes	xsd:base64Binary or xsd:base64hexBinary (?)
tdate	xsd:date
#6.38([lang: tstr, text: tstr])	rdf:langString with lang as language tag
#6.TBD([lang: tstr, dir: tstr, text: tstr])	i18n:{lang}_{dir}

Table 4: Mapping between CDDL types and XSD datatypes

[TBD: Check compatibilities, give type for at least the basic tags. Directional text might wind up in tag 38,]

* RDF literals are mapped to any CoRAL literal that yields an equivalent RDF literal in the opposite direction.

C.1.1. Example

The FOAF namespace provides this example:

```
<foaf:Person rdf:about="#danbri" xmlns:foaf="http://xmlns.com/foaf/0.1/">
  <foaf:name>Dan Brickley</foaf:name>
  <foaf:homepage rdf:resource="http://danbri.org/" />
  <foaf:openid rdf:resource="http://danbri.org/" />
  <foaf:img rdf:resource="/images/me.jpg" />
</foaf:Person>
```

Figure 5: Original FOAF file at http://.../me.xml

Converted, assuming no particular profiling or dictionary setup (and an ad-hoc table following Section 3.1 of [I-D.ietf-cbor-packed]), this could be:

```
51([[cri'http://danbri.org/'], [<<-3, "xmlns.com", ["foaf", "0.1"], null>>], [],
[
  [2, cri'http://www.iana.org/assignments/relation/carries-information-about', cr
i'/me.xml#danbri',
    [2, cri'http://www.w3.org/1999/02/22-rdf-syntax-ns#type', 6(<<'Person'>>)],
    [2, 6(<<'name'>>), "Dan Brickley"],
    [2, 6(<<'homepage'>>), 6(0)],
    [2, 6(<<'openid'>>), 6(0)],
    [2, 6(<<'img'>>), cri'/images/me.jpg']
  ]
]])
```

Figure 6: Serialized FOAF file at http://.../me.coral

The TBD:talks-about statement is introduced to bridge the gap between the basic and the necessarily structured information model. [TBD: Introduce that somewhere else more generally.]

In this packing, an invalid CRI (with trailing null leaving room for a fragment identifier to be added through packing) is added into the prefixes list. It is not sure whether this particular trick will ever be permitted by any of the profilings, or whether this is better done with base URIs. The mechanism is used because right now it works with the specifications involved without the need for further text, and is likely to be replaced by better mechanisms in later revisions of this document.

C.2. CoRE Link Format

Generic information in Web Links as described in [RFC8288] can not be converted to CoRAL in any practical way: Attributes are not managed, and it is not clear from the syntax whether an attribute is making a statement about the link or its target. (See Section 2.3.3.1 for an example).

Applications that use links with the attribute semantics common in the CoRE ecosystem (typically used with [RFC6690] Link Format) can use this conversion. It defines terms for common properties used for discovering resources, and describes a way to compatibly extend the mapping.

The same mechanism (but probably with a different mapping between names and attributes, and different rules about the necessity of packing entries) can be defined for any data model that builds on [RFC8288] semantics, e.g., the links sent in headers or payloads about [RFC7089] mementos, or applications building on [I-D.ietf-httpapi-linkset].

In several points the mapping describes URIs to necessarily have an entry in the packing table; this refers to the profiling described further down. Parts of a Link Format document that would need an entry but do not have one can not be converted; these are ignored in the conversion unless the converter is configured to be strict and fail the complete conversion in that case.

This mapping from Link Format to CoRAL is performed as follows: * For each relation in a link, a statement is created mapping the link context to the subject, the link target to the object and the relation to the predicate.

If the relation is of ext-rel-type, it is used as a URI as is. Otherwise it is a registered value, prefixed with http://www.iana.org/assignments/relation/ and necessarily packed using table TBD. (This is equivalent to the RPP mechanism for attribute values).

* Each target attribute is converted to one or more statements by the mechanism indicated for the attribute name in the following table. Statements produced from a link have the target as its subject, the attribute name without any trailing asterisk (prefixed with https://TBD/ [to be picked together with IANA as it'll be a registry]) as its predicate, and the object(s) depending on the mechanism.

  Attributes are necessarily listed in this table.

TN	Name	Mechanism
TBD	hreflang	[do we need that?]
TBD	media	[do we need that?]
16	title	string
TBD	type	[do we need that?]
0	rt	WSSP; RPP http://www.iana.org/TBDr/
1	if	WSSP; RPP http://www.iana.org/TBDi/
2	sz	int
3	ct	WSSP; int

Table 5: Initial entries of the target attribute registry (TN = table number)

Available mechanisms are:

* SPSP (space split): Link format values are split at space characters (SP in the RFC6690 ABNF), and all values treated using another mechanism.

* string: The attribute value is stored as a text string literal. If the Link Format attribute is language tagged (i.e. when the attribute name ends with an asterisk and the value is of ext-value shape), the literal is encapsulated in a CBOR language tag (38).

* int: The target attribute is processed as an ASCII encoded number and expressed as an integer literal. A failing conversion is treated like an unknown registered value: It is ignored unless configured otherwise.

* RPP (registered-prefix / packed): The input value (often the result of the SPSP mechanism) is parsed according to the relation-type ABNF production. If it is of ext-rel-type, it is expressed as that URI. If it is prefixed with the string indicated with the mechanism, and necessarily compressed through table TBD.

All currently registered link attributes are used in the CoRE ecosystem as indicating a property of the target that is independent of the link being followed. If this conversion is to be extended to

cover attributes that pertain to the full link being followed (typically along with one or more link relations), the relevant relations are not expressed as a single statement, but as a form, i.e. as two statements linking the context to a blank node and the blank node to the target; the attributes are attached to the blank node. The precise mechanism out of scope for this document, and left to those who first register such an attribute.

Some structure can be carried over from Link Format to the structured model: The sequences of links gets reused, and the set and sequence of attributes in a particular occurrence of a link get applied to the statement produced from the link (or all the statements, if the link has multiple link relations). Statements whose subject is not the document itself are attached to the retrieval context using the necessarily packed http://www.iana.org/assignments/relation/carries-information-about property. Statements about URLs mentioned elsewhere in the document can be expressed there instead.

Link relations of the reg-name form, link attributes, and attribute values from the RPP mechanism MUST be serialized using packed CBOR as initialized in table TBD. No other packing is used. A consumer MAY ignore any items compressed through the dictionary for which it does not know the expanded version: These necessarily represent statements that involve terms the consumer does not understand.

[As an alternative, packing attributes together with their URIs is considered: Rather than [2, 6(/ attr:rt /), 6(/ rt:core.rd /)] we could have 6(rt-core) right away; unregistered values would stay [2, 6(/ attr:rt /), value] or maybe 254([value]) using prefix packing.]

Appendix D. Change Log

This section is to be removed before publishing as an RFC.

Changes from -05 to -06:

* Unmodified resubmission. (Recent work is going on in [I-D.ietf-core-href] and [I-D.ietf-cbor-packed], providing building blocks for CoRAL).

Changes from -04 to -05:

* Literals can no longer have properties. The only use case was annotating languages and directions, and that can be done in CBOR.

* Added section about open and close world modelling.

* Information model merged with the previous data model and interaction section.

Changes from -03 to -04:

* Formalize information model, as basic and structured model.

* Remove textual representation, using CBOR diagnostig notation instead.

* Use Packed CBOR instead of custom dictionaries.

* Give explicit conversions from Link Format and with RDF.

* Remove references to IRIs (outside RDF) as CRIs are closer to URIs.

* Remove requirement for deterministic encoding.

* Many editorial changes.

* Update references.

* Change of authorship.

Changes from -02 to -03:

* Changed the binary format to express relation types, operation types and form field types using [I-D.ietf-core-href] (#2).

* Clarified the current context and current base for nested elements and form fields (#53).

* Minor editorial improvements (#27).

Changes from -01 to -02:

* Added nested elements to form fields.

* Replaced the special construct for embedded representations with links.

* Changed the textual format to allow simple/qualified names wherever IRI references are allowed.

* Introduced predefined names in the textual format (#39).

* Minor editorial improvements and bug fixes (#16 #28 #31 #37 #39).

Changes from -00 to -01:

* Added a section on the semantics of CoRAL documents in responses.

* Minor editorial improvements.

Acknowledgements

The concept and original version of CoRAL (as well as CRIs) was developed by Klaus Hartke. It was heavily inspired by Mike Kelly's JSON Hypertext Application Language [HAL].

The recommendations for minting URIs have been adopted from RDF 1.1 Concepts and Abstract Syntax [W3C.REC-rdf11-concepts-20140225] to ease the interoperability between RDF predicates and link relation types.

Thanks to Carsten Bormann, Jaime Jiménez, Jim Schaad, Sebastian Käbisch, Ari Keränen, Michael Koster, Matthias Kovatsch and Niklas Widell for helpful comments and discussions that have shaped the document.

Authors' Addresses

Christian Amsüss
Email: christian@amsuess.com

Thomas Fossati
ARM
Email: thomas.fossati@arm.com