CDDL 1.1 + 2 plan (standards track)

— Done before IETF 117: CDDL 1.1: Grammar fixes
§2 of draft-bormann-cbor-cddl-2-draft, except for 2.3
Empty files (enabling CDDL 2), non-literal tags, errata fixes

— Done before IETF 117: Mid-2023 Parallel to CDDL 1.1: More control operators
draft-bormann-cbor-cddl-more-control
Additional control operators, another iteration like RFC 9165 (implemented)

— Done before IETF 118: CDDL 2.0:
§4 of draft-bormann-cbor-cddl-2-draft (“modules”)
Draft implementation already available for `import/include`

— Done 2024: CDDL 2.5: §3 of draft-bormann-cbor-cddl-2-draft
("annotations", functionality enabled by that) ➔ Lots of music, lots of fun.
Need to play with this (starting at the IETF 116 Hackathon)
Enables, e.g., §5 of draft-bormann-cbor-cddl-freezer (co-occurrence)

Not on the main line of development:

— (I Mid-2023): §6 of draft-bormann-cbor-cddl-freezer
CDDL-in-JSON format(s) for interchange between tools

— (I, with 2.0) draft-bormann-cbor-rfc-cddl-models
(Builds standard collection of referenceable models)

— (S) §2.3/A.1 of draft-bormann-cbor-cddl-2-draft
literals; develop with draft-bormann-cbor-edn-literals

— (B) draft-bormann-cbor-draft-numbers
(BCP for handling assigned numbers during draft stage)

— (I/S?) draft-bormann-cbor-cddl-csv

CDDL 2.0: Module structure: (See 2023-02-08 interim)

Objectives:
Within a CDDL project:

— Within a project:
  Construct CDDL from multiple files (`;# include`)
— Between projects:
  Reference existing CDDL as libraries (`;# import`)
— Optionally put i*ed CDDL into a namespace (…`as`)
— Optionally limit to specific names (…`from`)

Simple import (intrusively, from "library")

```
$ cddlc -2tcddl -
start = COSE_Key
;# import rfc9052
```

```
start = COSE_Key
COSE_Key = {
  1 => tstr / int,
  ? 2 => bstr,
  ? 3 => tstr / int,
  ? 4 => [+ tstr / int],
  ? 5 => bstr,
  * label => values,
}
label = int / tstr
values = any
```

import as (namespaced from "library")

```
$ cddlc -2tcddl -
start = cose.COSE_Key
;# import rfc9052 as cose
```

```
start = cose.COSE_Key
cose.COSE_Key = {
  1 => tstr / int,
  ? 2 => bstr,
  ? 3 => tstr / int,
  ? 4 => [+ tstr / int],
  ? 5 => bstr,
  * cose.label => cose.values,
}
cose.label = int / tstr
cose.values = any
```

Copy/Paste per explicit names in include

```
$ cddlc -2tcddl -
mydata = {* label => values}
;# include label, values from rfc9052
```

```
mydata = {* label => values}
label = int / tstr
values = any
```

Namespaced "as cose":

```
$ cddlc -2tcddl -
mydata = {* label => values}
;# include cose.label, cose.values from rfc9052 as cose
```

```
mydata = {* label => values}
cose.label = int / tstr
cose.values = any
```

Explicit name plus transitive closure: `import`

```
$ cddlc -2tcddl -
mydata = {Fritz: cose.empty_or_serialized_map}
;# import cose.empty_or_serialized_map from rfc9052 as cose

➔

mydata = {"Fritz" => cose.empty_or_serialized_map}
cose.empty_or_serialized_map = bstr .cbor cose.header_map / bstr .size 0
cose.header_map = {
  cose.Generic_Headers,
  * cose.label => cose.values,
}
cose.Generic_Headers = (
  ? 1 => int / tstr,
  ? 2 => [+ cose.label],
  ? 3 => tstr / int,
  ? 4 => bstr,
  ? (5 => bstr // 6 => bstr),
  )
cose.label = int / tstr
cose.values = any
```

Namespaced `import` with adding unnamespaced alias

```
$ cddlc -2tcddl -
mydata = {Fritz: cose.empty_or_serialized_map}
;# import empty_or_serialized_map from rfc9052 as cose
```

```
mydata = {"Fritz" => cose.empty_or_serialized_map}
empty_or_serialized_map = cose.empty_or_serialized_map
cose.empty_or_serialized_map = bstr .cbor cose.header_map / bstr .size 0
cose.header_map = {
  cose.Generic_Headers,
  * cose.label => cose.values,
}
cose.Generic_Headers = (
  ? 1 => int / tstr,
  ? 2 => [+ cose.label],
  ? 3 => tstr / int,
  ? 4 => bstr,
  ? (5 => bstr // 6 => bstr),
  )
cose.label = int / tstr
cose.values = any
```

Command line Control

```
$ cddlc -2tcddl -icose=rfc9052 -scose.COSE_Key
```

```
-icose=rfc9052 ➔

;# import rfc9052 as cose

-scose.COSE_Key ➔

$.start.$ = cose.COSE_Key
```

```
$.start.$ = cose.COSE_Key
cose.COSE_Key = {
  1 => tstr / int,
  ? 2 => bstr,
  ? 3 => tstr / int,
  ? 4 => [+ tstr / int],
  ? 5 => bstr,
  * cose.label => cose.values,
}
cose.label = int / tstr
cose.values = any
```