A Parametrized Content Format

https://datatracker.ietf.org/doc/draft-fossati-core-parametrized-cf/

CoRE Interim - 2022-07-06

CoAP Content-Format

CoAP squashes the combination of a media type, media type parameters and content coding into a single Content-Format number

This number is carried in the Content-Format and Accept Options

Paraphrasing Uncle Ben from Spider-Man:
"*With great compression comes great inflexibility*"

pəˈramɪtrʌɪzd / pəˈramɪt(ə)rʌɪzd Content-Format (PCF for short)

Concept (natural language)

Extend the familiar `uint16_t` Content-Format with a list of key-val pairs representing the associated media type parameters

Concept (CDD language)

```
parametrized-content-format = [
    content-format
    * [ parameter-name, parameter-value ]
]

content-format = 0..65535

parameter-name = textual / numeric
parameter-value = any
```

Note: `parameter-name` has also a compact representation (`numeric`), which requires a new registry

Relevant excerpts from RFC6838

"

> Parameter names are case-insensitive and no meaning is attached to the order in which they appear. It is an error for a specific parameter to be specified more than once.
> [...]
>
> There is no defined syntax for parameter values. Therefore, registrations MUST specify parameter value syntax. Additionally, some transports impose restrictions on parameter value syntax.
> [...]

"

Applications

Once we have defined it we can use it to create new *parametrised* versions of the `Accept` and `Content-Format` options that carry CBOR-encoded PCF instead of `uint16_t`

Keep the same semantics - e.g., 4.06 (Not Acceptable) and 4.15 (Unsupported Content-Format) would apply as well

Increased flexibility

pContent-Format

Number	C U N R	Name	Format	Length	Default
TBD24		Parametrized Content-Format Option	See below		none

```
bytes .cbor parametrized-content-format
```

(multi-valued) pAccept[1]

Number	C U N R	Name	Format	Length	Default
TBD13	X	Parametrized Multi-Valued Accept Option	See below		none

```
bytes .cbor one-or-more<parametrized-content-format>
```

[1] Carsten asks: "*Should pAccept be* `.cborseq` *rather than* `.cbor`*?*"

Moving to (multi-valued) pAccept

- We wanted to retain the same exact semantics as `Accept`
  - therefore, the critical bit is up
- This implies the request will fail with 4.02 (*maybe* with a problem detail sporting an "Unprocessed CoAP Option" key with value TBD13) if pAccept is not implemented on server-side
- there is no way to soft-fail
  - maybe it is OK

Prior art

RFC 9193

OCF's C-F Version

Comparison with SenML Data Value Content-Format

RFC 9193 defines:

```
Content-Format-Spec:  The string representation of a content format;
   either a Content-Format-String or the (decimal) string
   representation of a Content-Format number.
```

PCF is essentially a third type of `Content-Format-Spec`, roughly a binary version of `Content-Format-String`

How PCF compares to `Content-Format-String`?

— pro: more compact

Comparison with OCF's C-F "version"

OCF have hit the lack of flexibility in content negotiation, but they have worked around it in a different way:

— OCF endpoints exchange a single CBOR based content format `"application/vnd.ocf+cbor"` which is explicitly versioned
— Instead of putting the version in the media type they keep the media type fixed, using a couple of options to do version negotiation

CoAP Option Numbers

Number	Name	Format	Length (bytes)
2049	OCF-Accept-Content-Format-Version	uint	2
2053	OCF-Content-Format-Version	uint	2

OCF encoding

The option value is a two-byte unsigned integer that is used to define the major (MS 5-bits), minor (5-bits) and patch (6-bits) using the simplest ("core") semantic versioning format

Example:

```
"1.1.0" => 00001 00001 000000 => 0x0840
```

Translating OCF into PCF

```
[
  10000,      // application/vnd.ocf+cbor
  [ 0, 2112 ] // version=1.1.0
]

; 9-bytes wire image:

82            # array(2)
   19 2710    # unsigned(10000)
   82         # array(2)
      00      # unsigned(0)
      19 0840 # unsigned(2112)
```

— fits into one Option, instead of two (spare one byte)

— 4 bytes is the price you pay for generalising rather than optimising for one use case

Questions

— Is there any interest for pursuing this feature?
— Is the proposal a good starting point?