Where We Are

- Instance-related drafts are converging
- sdfProtocolMap is coming along nicely
- We can report and update both state and context information
  - Gap: protocol binding information at the instance level
- Goal: Describe how to actually interact with a device
  - need to complement the `sdfProtocolMap` class-level information via instance messages

Example SDF Model using CoAP

(C.f. sdf-protocol-mapping PR #7)

— In a model, protocol information is class information

— How should we specify instance information, e.g., the device's IP address?

```
sdfObject:
  temperatureSensor:
    sdfProperty:
      temperature:
        unit: Cel
        type: number
        sdfProtocolMap:
          coap:
            read:
              method: GET
              href: "/temperature"
              contentType: 60
              # application/cbor
```

Naïve Approach: Replicate Structure of Model in Instance

```
sdfObject:
  temperatureSensor:
    sdfProperty:
      temperature:
        sdfProtocolMap:
          coap:
            read:
              ipAddress: 45.0.13.7
```

Drawbacks:

— Potentially verbose/ redundant

— IP address is likely shared across affordances/resources

Can Context Information Help?

- Maybe include IP address at the instance level via `sdfContext`
- Implicit inheritance may cause problems

```
sdfObject:
  temperatureSensor:
    sdfContext:
      ipAddress:
        type: string
    sdfProperty:
      temperature:
        unit: Cel
        type: number
        sdfProtocolMap:
          coap:
            read:
              method: GET
              href: "/temperature"
              contentType: 60
```

Context Snapshot for that

```
namespace:
  models: "https://example.com/models"
  sensors: "https://example.com/sensors"
defaultNamespace: sensors
sdfInstanceOf:
  model: "models:#/sdfObject/temperatureSensor"
sdfInstance:
  sdfContext:
    ipAddress: 203.0.113.42
```

— With this context snapshot and its model: consumer can access the temperature

Even More Questions

— How to indicate the relationship between `sdfProtocolMap` and `sdfContext` more explicitly?

— How to model security protocols such as OSCORE, EDHOC, or ACE-OAuth?
Include in parameters of transport/transfer protocol secured?

— Does the same work for transport-level security, i.e. TLS and DTLS?

Does Security Deserve its Own Map?

```
sdfObject:
    sdfProperty:
      temperature:
        sdfProtocolMap:
          coap:
            read:
              method: GET
              href: "/temperature"
            security:
              - [ edhoc, oscore ]
        sdfSecurityMap:
          oscore:
            contextEstablishedBy: [ edhoc ]
          edhoc:
            toKey: [ oscore ]
```

Could (D)TLS be Included in the `sdfSecurityMap` as well?

```
sdfObject:
    sdfProperty:
      temperature:
        sdfProtocolMap:
          coap:
            read:
              method: GET
              href: "/temperature"
            security:
              - [ dtls ]
        sdfSecurityMap:
          dtls:
            supportedCipherSuites: [ TLS_PSK_WITH_AES_128_CCM_8 ]
            mode: [ psk ]
```

More Work Needed (T2TRG?)

- In CoRE, security information is often discovered during protocol flow
  - Need to avoid redundancy:
    Descriptive, no intent to control protocol flow
  - Prevent disclosure of sensitive information
  - Which information is actually important for its consumers?
- Should we model information when it is only important for implementers?
- Should we cover authorization and access control (e.g., via ACE) separately?
  - Access control constrained by SDF model
  - SDF providing information for/being patched by a ACL?