OCF Binding Prototype

Düsseldorf Face to Face, July 2017

Michael McCool <michael.mccool@intel.com>

Outline

- Prototype:
  - What was done
  - Architecture
- Issues
- Suggestions
- Next steps

Goals

- Consume OCF metadata
- Output a Thing Description
- Attempt *automatic* translation of metadata

Prototypes

- ocf-bridge (not completed)
  - Consume OCF metadata, expose automatically generated WoT interface, translate dynamically to OCF interactions
- ocf-generator
  - Consume OCF metadata, produce TD describing available resources and interactions
- ocf-client (partial)
  - Consume and test TD generated by ocf-generator; allow interaction with OCF devices through WoT scripting API

Architecture

- Need to extract OCF metadata from multiple sources and integrate
  - Instance metadata: /oic/res/, /oic/d, introspection
  - Type metadata: JSON-schema, RAML
  - Standard metadata: OCF standard
- Some additional hand-written metadata required per OCF “resource type” still needed
  - Annotation metadata

Annotation Metadata

- Semantic tagging
  - For example, oic.rt.light => ocf:Light
- Which resource types should be exposed
  - Some are OCF-specific metadata
- Which combinations of interfaces make sense
- Actions and Events
  - OCF technically only has “properties”
- Names and writable properties
  - Name and “writable” annotations don’t match up exactly with how this is done in TDs

Annotation Example

```
[..
"oic.r.switch.binary": {
      "writable": false,
      "method": "r",
      "protocolContent": {
         "http": "application/json",
         "coap": "application/cbor"
      },
```

```
"outputData": {
      "properties": {
          "ID": {"type": "string"},
          "VALUE": {"type": "boolean"}
       },
       "templateType":
            "application/json",
       "template":
      "{\"id\":\"{{ID}}\",\"value\":{{VA
LUE}}}"
      }
   },...]
```

TD Example (one Interaction)

```
[…{    "@type": ["Property","ocf:Resource"],
        "ocf:di": "47ffbddf-37d1-4f39-95f5-d3ba2fcf2c92",
        "link": [{
        "href":"http://192.168.1.127:8000/api/oic/light?di=47ffbddf-
37d1-4f39-95f5-d3ba2fcf2c92",
                "coap:rt": ["oic.r.switch.binary"],
                "coap:if": ["oic.if.baseline"],
                "ocf:p": {
                        "ocf:bm":3,
                        "ocf:secure":false
                },
                "mediaType": "application/json",
                "driver": "ocf",
                "outputData": {
                        "templateType": ["application/json"],
                        "template": [
                           "{\"id\":{{ID}},\"value\":{{VALUE}}}"
                        ]}
        }],
                "name": "batchlight",
                "ocf:n": "batch light",
                "ocf:icv": "core.1.1.0",
                "writable": false,
                "outputData": {
                    "valueType": {"type":"object"},
                   "properties": {
                        "ID":{"type":"string"},
                        "VALUE":{"type":"boolean"}
                    }
                }
},…]
```

Prototype Limitations

- Some data available at oneIoTA.org in JSON-Schema and RAML was entered manually in annotation metadata
  - Should define a separate program, an “ocf-gatherer”, to collect and translate this information
  - Only needs to be done once per spec, mostly
  - Except for vendor extensions…
- Doesn’t handle CoAP yet
- Doesn’t deal with multiple rt/if combinations
- Includes some unnecessary “OCF internal” resources in the output
- Combines all OCF devices on the local network into one big Thing (leading to potential name collisions)
- Not really aligned with MK’s current P.B. proposal…

Issues

- Names in OCF properties may not map to unique names needed for WoT interface
  - Led to an issue with actually using Scripting API
- OCF di's need to be embedded in URIs but are not consistent from run to run
- Semantic tagging used but context file not defined
  - Need to work on OCF and CoAP semantic models
- Resource directory groups all resources together, need to be re-grouped and split out again as separate Things

Suggestions

- Protocol binding templates as strings
- Separate template media type (which should be encoded as a string, eg JSON) that can be converted to wire media type (which might be binary, eg CBOR)
- Alternatives:
  - Make template always be JSON
  - Get rid of template and make it equivalent to parameter map, eg. make generation of payload the full responsibility of the driver
- Add additional “driver” parameter so that target device ecosystem (eg ocf) can be identified
  - May not be able to infer from protocol, eg http:// or coap://
  - May be special behaviors or exceptions that need to be dealt with in the driver
- Encode raw metadata (optionally…) in TD using ocf: prefix
  - Of possible use to reasoner/planner systems, or to driver

Next Steps

- Update to intercept protocol binding proposal
- Implement protocol binding templates
  - E.g. using moustache and reverse-moustache
  - If we decide we actually need them…
- Implement ocf-gather to ingest existing OCF type metadata
- Additional semantic annotation mechanisms
- Pull request on node-wot
- Try some other IoT standards…

Conclusions

- Automatic mapping possible
  - Mostly; with a small amount of additional metadata to fill in a few blanks
- But OCF is a relatively easy case
  - Web-API-ish, REST-y
- We really need to repeat this exercise with additional standards