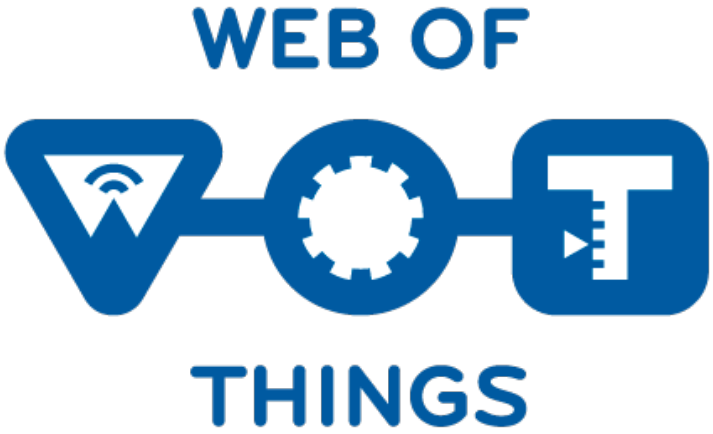

WoT Profiles

Michael Lagally

22 June 2021

Agenda

- Introduction to WoT profiles
- Actions
- HTTP Protocol Binding
- Events
- PRs
- Test Plan (Not discussed)

What is a WoT profile?

- A **WoT Profile** is a normative subset of a *WoT Thing Description* with a normative binding to a selected protocol.
- Profiles guarantee **interoperability** between compliant implementations, multiple profiles are possible.
- The **WoT Profile Specification** defines a **normative** set of *constraints and rules* on the **data model**, **representation format** and **protocol binding**.
- These constraints and rules set limitations and make decisions that reduce the complexity for implementers of the WoT standard.
- The rules are prescriptive, to ensure that compliant implementations satisfy the semantic guarantees implied by them.

Constraints:

Constraints on	*Rationale*	*Example*
vocabulary of Thing Description classes	guaranteed set of metadata fields	Make specific vocabulary terms mandatory, remove others
class relationships	unambiguous structure	limited cardinality, e.g. only one form per operation per interaction affordance.
values of vocabulary terms	simplified processing	Limit the length of characters per string. Always use arrays, where the spec permits a string or an array of strings.
data schemas	simplified processing	Limits on nested objects or arrays of arrays
security	reduced implementation effort	Only a restricted set of security mechanisms
protocol binding	guaranteed protocol semantics	limited protocol(s) and protocol features, Example: predefined mapping of http verbs (GET/PUT) to operation verbs, similar constraints for other protocols.

What are we currently doing?

- Defining a core/baseline profile with a HTTP binding.
- Identifying constraints and rules on the data model.
- Unambiguous interaction semantics for properties, actions and events.
- Constraints on payload formats.
- Protocol binding semantics, e.g. headers, response codes.
- Best practice security requirements.
- Representation (TD Serialisation) format constraints.

Actions

Actions

- The server models interactions that cause state change via actions.
- Actions can have a different **request modes**, they can be synchronous or asynchronous.
- The TD does not have a mechanism to annotate whether an action is sync or async.
  Synchronous action invocations return (*after a while)* with a response (result or timeout) when it is complete. ~~., [the actual operation (e.g. robot arm moving) may still be ongoing.]~~
- **Asynchronous** actions are triggered by the invocation, they return instantaneously with a response.
- This response may contain various data, a handle, an id, invocation status.
- The **consumer** should be able to indicate, whether he wants to wait (sync call with a timeout) or if he wants to perform an async call.

Request modes and timeout

- The request mode can be specified with query parameters "iot.sync" or "iot.async".
- By default if none of the parameters is set then the request is asynchronous.
- These parameters must not be set both.
- The asynchronous mode is a preferrable way of doing invocation as it does not consume as much server resources as compared with the synchronous mode.
- Optionally, a timeout value in milliseconds may be provided with the "iot.timeout" query parameter.

Request modes and timeout

Discussion:
we could use mode= “sync/async, potential future extensions.
There may be implementations that only can be sync or async.
Suggested approach: we force the producer to support both ways.
A server could indicate that it does not support sync calls, this could be indicated via a response code
producer should define whether it is sync or async
consumer code for sync and async is very different
hard real time is not a goal here, focus is green field
sync does not mean that the scripting API needs to be synchronous, the script is still async
we could make all actions async and meet the goals of the charter period
TD annotations for sync/async are planned per issue https://github.com/w3c/wot-thing-description/issues/890
initiate, monitor and cancel are on the charter –
let’s not defer defining basic actions to a future spec but address it in the profile
defining sync / async is easy, however there are several issues that require more thought – defer to 2.0
we should clarify actions based on actual use cases, these were done in Plugfest by various companies, Echonet binding should be considered

HTTP Protocol Binding

HTTP Protocol binding

Methods, Response codes, Headers, Payloads

Methods

- Prohibit all unused methods, e.g. HEAD, CONNECT, …
- Discussion:
  Are directories profile compliant, they use “head” in some cases.
  For now exclude directories from profile considerations.

Response Codes

- An initial list of HTTP response codes has been defined recently.
- Additional codes / clarifiations may be required.

HTTP Protocol binding

Methods, Response codes, Headers, Payloads

Headers

- We have to select/define specific HTTP header fields wrt. content formats, timeouts etc.

Content Type / Content Coding Formats

JSON, XML, Binary?

Discussion:

Start with only JSON, all binary data (e.g. jpeg and png images?) would be base64 encoded
exclude XML, CBOR for now
video streams etc. would be problematic use cases, we would be returning URLs but not streams

Links to html pages, video streams, … are permitted, we are not constraining link targets

We are not constraining video formats, but it is up to the consumer to interpret the format

We could reuse MPEG-DASH, Netflix, … formats, could ask media experts for their opinion
Video streams are out of scope of the core profile

Do we select different data formats for media? No

Events

Invent or align

Consider alignment with standardised Cloud Events

https://cloudevents.io/

https://github.com/cloudevents

Discussion:

Todo: check IP policy of Cloud Events

Also OpenAPI could offer an event model we could adopt
See also https://github.com/w3c/wot-thing-description/issues/893

We could adopt SSE, since there's now wider support in browsers

PRs

PRs

RFC 2119 markup: add css mark into index.html

- https://github.com/w3c/wot-profile/pull/79

Discussion:

We noticed that some span markups are in editors notes – these need to be cleaned up in a further iteration.

We merged the PR, Mizushima-san will provide additional comments.

McCool will do another pass using the test tooling.

References

WoT Architecture task force

- https://www.w3.org/WoT/activities/tf-architecture/

WoT Profile repository

- https://github.com/w3c/wot-profile

Working draft:

- https://www.w3.org/TR/wot-profile/