W3C WoT WG
TD Versioning & Lifecycle
& Extend TD Templates Definitions

Bundang 2018

Sebastian Kaebisch
Versioning

Proposal:
Add a optional **version** term as container to manage the name of the current TD version

```
"version": {"td": "V x.y.z", "iot:firmware":"V x.y.z", "iot:hw": ...,...}
```

Should be standardized in the TD context  Application dependent

**SemVer:** Version number respecting a set of rules and requirements that dictate how version numbers are assigned and incremented ➔ easy for human readability

[https://semver.org/](https://semver.org/)
Versioning & Lifecycle

Detailed lifecycle of a Thing in the industry shows:

- The evolution of the Thing and the introduction of multiple features and updates after the manufacturing phase

► The importance of the versioning of the TD through the lifecycle of a thing
Lifecycle Approach (3)

**Planning phase:** Define technical requirements of the device

- Definition of the elementary version of the TD

**Example:** Thing's name, first set of interaction patterns...

**Design phase:** Development of initial software and device hardware

- Generation of the first version of the TD (all interactions, URI's, etc...)
Lifecycle Approach (4)

Manufacturing phase: Mass production of the device’s hardware and deployment of the software

⇒ Generation of the version V1.0.1 of the TD containing some specific information proper to every single device (ex: Semantic annotation based on application context, etc...)

Maintenance and updates:

⇒ Generation of an enhanced version of the TD Vx.y.z'/Vx.y'.z' supporting new feature (new interactions or additional semantic annotation): Backward compatible changes

Software/Hardware upgrades:

• Making a backward incompatible hardware/software changes resulting in removing some interactions from the previous TD
Versioning Minor vs. Major

TD Version update

+new Property

+new Event
Versioning Minor vs. Major

TD Version update

- remove Property

- remove Action

+ new Property

+ new Action
Let's assume there is a master TD template providing all basic interactions and a huge set of (identical) Things implements these interactions. It would be beneficial to maintain only the master TD and all the Things would inherit the interactions. Thing’s TD only extends its individual information (e.g., id, base URI, semantics like locations, etc) or overwrite master TD information.

**Use case:** Parking slot sensors, room lights, mass production of Things, …