Web of Things (WoT) Thing Description

W3C Proposed Recommendation 30 January 2020

This version:
https://www.w3.org/TR/2020/PR-wot-thing-description-20200130/
Latest published version:
https://www.w3.org/TR/wot-thing-description/
Latest editor's draft:
https://w3c.github.io/wot-thing-description/
Implementation report:
https://w3c.github.io/wot-thing-description/testing/report.html
Previous version:
https://www.w3.org/TR/2019/CR-wot-thing-description-20191106/
Editors:
Sebastian Kaebisch (Siemens AG)
Takuki Kamiya (Fujitsu Laboratories of America)
Michael McCool (Intel)
Victor Charpenay (Siemens AG)
Matthias Kovatsch (Huawei)
Participate:
GitHub w3c/wot-thing-description
File a bug
Commit history
Pull requests
Contributors:
In the GitHub repository
Repository:
We are on GitHub
File a bug

摘要

本文書は、WoT(Web of Things) Thing Description (TD)のための形式モデルおよび共通表現を解説するものである。TD は、モノのメタデータおよびインターフェースを記述し、モノは、WoT 内での対話を提供し、それに参加する物理エンティティまたは仮想エンティティのアブストラクションである。TD は、多様なデバイスの結合、および、多様なアプリの相互運用を可能にする少数のボキャブラリに基づく対話セットを提供する。TD は、デフォルトで JSON-LD 処理も可能な JSON 形式でエンコードされる。後者は、マシン理解可能な方法でモノに関する知識を表すための効果的な基盤を提供する。TD インスタンスは、モノ自体によって提供されるか、または、モノにリソース制限(例えば、制限されたメモリ空間)があるとき、あるいは、WoT 互換性レガシーデバイスの TD によってレトロフィットされているときには、外部で提供される。

本文書のステータス

本項では、発行時点での本文書のステータスを説明する。このため、他文書が本文書よりも新しい場合もある。現行の W3C 出版物一覧および本テクニカルレポートの最新版は、https://www.w3.org/TR/の W3C テクニカルレポート索引に記載されている。

(本仕様書内で黄色表示されている)以下の危険状態の機能は、ＣＲ期間中に報告されるインプリメンテーション経験、及び・あるいは、受け取るコメントが不十分であるために削除されることがある。

- 第 5.3.3.7 項 CertSecurityScheme、第 5.3.3.9 項 PublicSecurityScheme、第 5.3.3.10 項 PoPSecurityScheme 記載のセキュリティスキームに関連する全項。
- 第 5.3.3.11 項 OAuth25securityScheme 内の implicit, password, client フローに関するというボキャブラリ用語とアサーション。
- 前述の 第 5.4 項 デフォルト値定義内の上記項に関連する全デフォルト値。
- 不完全な書き込み拒否を可能にする writeallproperties のためのビヘイビアアサーション

本文書は、WoT ワーキンググループによって推奨候補として発行されたものであり、W3C 推奨に

なることを意図している。

GitHub の問題指摘は本仕様の考察の材料として歓迎される。また、それを public-wot-wg@w3.org(アーカイブ)で我々のメーリングリストにコメントを送ることもできる。

W3C は、本文書が信頼できるものであると考えられていることを示し、開発者コミュニティによるインプリメンテーションを奨励するために、推奨候補を公開するもので、この推奨候補は、2019 年 12 月 4 日以前に提案推奨として更新される予定である。

ワーキンググループのインプリメンテーションレポートを参照いただきたい。

推奨候補としての公開は W3C メンバーによる推奨を意味するものではない。本文書は、常に、更新されたり、他の文書と置き換えられたり、また破棄される可能性もある草案文書であり、“作業中文書”取扱以外で本文書を引用することは不適当である。

本文書は、W3C 特許ポリシーに従い運営しているグループが作成したものである。W3C では、ワーキンググループの成果物に関係する公的な特許公開リストを管理しており、そのページには特許開示にあたっての指示も掲載されている。必須特許請求を含むと信じる特許について実知識を持つ者は、W3C 特許ポリシー第 6 項に従って、その情報を開示しなければならない。

本文書は、2019 年 3 月 1 日の W3C プロセス文書に準拠する。

もくじ

5. IntegerSchema
6. ObjectSchema
7. StringSchema
8. NullSchema

3. セキュリティボキャブラリの定義
    1. SecurityScheme
    2. NOSecurityScheme
    3. BasicSecurithScheme
    4. DigestSecurithScheme
    5. APIKeySecurituScheme
    6. BearerSecurityScheme
    7. CertSecurityScheme
    8. PSKsecurithScheme
    9. PubloicSecurityScheme
    10. PopSecurityScheme
    11. OAuth2SecurithScheme
4. ハイパーメディア制御ボキャブラリの定義
    1. Link
    2. Form
    3. ExpectedResponse

1. デフォルト値の定義

2. **TD 表現フォーマット**

https://www.w3.org/TR/wot-thing-description/

1. JSON タイプへのマッピング
2. デフォルト値の省略
3. 情報モデルのシリアライズ
    1. Thing Root オブジェクト
    2. 人間が読み取り可能メタデータ
    3. Version
    4. securithDefinition to security
    5. properties
    6. actions
    7. events
    8. links
    9. forms
    10. データスキーマ
1. ID

2. **TD コンテキスト拡張子**
    1. 意味論的注釈
    2. プロトコルバインディングの追加
    3. セキュリティ方式の追加

3. **ビヘイビアアサーション**
    1. セキュリティ構成
    2. データスキーマ
    3. プロトコルバインディング
        1. HTTP に基づくプロトコルバインディング

https://www.w3.org/TR/wot-thing-description/

1. ＴＤテンプレート: ランプ
2. ＴＤテンプレート: ブザー

D. **JSON-LD コンテキスト用法**

E. **最近の仕様変更**

1. 第一次推奨候補からの変更点
2. 第三次公開作業ドラフトからの変更点

F. **謝辞**

G. **参考文献**

1. 標準参考文献
2. 参考文献

1. はじめに

本項は標準ではない。

WoT Thing Description (TD)は、W3C Web of Things (W3CWoT)における中心的なビルディングブロックであり、モノ(ウェブサイトの index.html に酷似する)のエントリポイントとみなすことができる。TD インスタンスには、4 つの主要なコンポーネント、すなわち、モノ自体に関する文字メタデータ、モノはどのように使用できるかを示す対話アフォーダンス、マシンが理解できるようにするためにモノと交換されるデータスキーマ、および、ウェブ上の他のモノ、または、文書との形式的または非形式的な関係を表すためのウェブリンクで構成されている。

W3C WoT の対話モデルは、3 タイプの対話アフォーダンスを定義している。すなわち、プロパティ(PropertyAffordance クラス)は、現在の値を取得する、または、動作ステータスを設定するなどパラメータを感知・制御するために使用することができる。アクション(ActionAffordance クラス)は、物理的な(したがって時間のかかる)プロセスの呼び出しをモデル化するが、既存のプラットフォームの RPC のような呼び出しを抽象化するために使用することもできる。イベント(EventAffordance クラス)は、通知、個別イベント、または値のストリームが非同期で受信側に送信される通信のプッシュモデルに使用される。詳細は[WOT-ARCHITECTURE]参照。

一般に、TD は、URI スキーム [RFC3986](例えば、http、coap など,[IANA-URI-SCHEMES])、メディアタイプに基づくコンテンツタイプ[RFC2046](例えば、application/json、application/xml、application/cbor、application/exi など、[IANA-MEDIA-TYPES]、および、セキュリティメカニズム(TD インスタンスのシリアライズは、JSON [RFC8259]に基づいており、ここでは、本仕様書で定義するとおり、JSON 名は TD ボキャブラリ用語を指す。さらに、TD の JSON シリアライズは、拡張および豊富な意味処理を可能にするために、JSON−LD 1.1[JSON-LD11]のシンタックスを遵守している。

例 1 は、TD インスタンスであり、MyLampThing というタイトルのランプというモノを記述するプロパティ、アクション、イベントを持つ対話モデルとなっている。

https://www.w3.org/TR/wot-thing-description/

例 1:TD の例


```
{
    "@context": https://www.w3.org/2019/wot/td/v1,
    "id": "urn:dev:ops:32473-WoTLamp-1234",
                        "title":
                        "MyLampThing",
    "securityDefinitions": {
        "basic_sc": {"scheme": "basic","in":"header"}
    },
    "security": ["basic_sc"],
    "properties": {
        "status" : {
            "type": "string",
            "forms": [{"href": "https://mylamp.example.com/status"}]
        }
    },
    "actions": {
        "toggle" : {
            "forms": [{"href": "https://mylamp.example.com/toggle"}]
        }
    },
    "events":{
        “overheating”:{
```

```
            "data": {"type": "string"},
            "forms": [{
                「href」:“https://mylamp.example.com/oh”,
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

このTDの例からは、タイトルステータスを持つ1つのプロパティアフォーダンスが存在することがわかる。さらに、このプロパティが、(hrefメンバーによってforms構造内でアナウンスされる)URI　https://mylamp.example.com/statusでGETメソッドを使って（安全な形式の)HTTPプロトコルを介してアクセス可能であり、ストリングベースのステータス値を返すということを示す情報が提供されている。GETメソッドの使用は、明示的に述べられていないが、本文書で定義されるデフォルト想定の1つである。

同様に、アクションアフォーダンスが、https://mylamp.example.com/toggle資源上のPOSTメソッドを使って切り替えステータスをトグルするために指定され、ここでもPOSTは、アクションを呼び出すためのデフォルト想定である。

イベントアフォーダンスにより、モノが非同期メッセージのためのメカニズムを送信できるようになる。ここで、ランプの過熱事象が起こった場合に通知されるサブスクリプションは、https://mylamp.example.com/oh上の長いポーリングサブプロトコルを有するHTTPを使用することによって取得することができる。

https://www.w3.org/TR/wot-thing-description/

本例は、また、アクセスのためにユーザ名およびパスワードを必要とする basic セキュリティスキームを指定している。セキュリティスキームには、まず、securityDefinition で名前が与えられ、次に、セキュリティセクションでその名前を指定することによってアクティブ化されることに留意されたい。HTTP プロトコルとの併用で、本例は、HTTP 基本認証の使用を説明している。最上位レベルでの少なくとも 1 つのセキュリティスキームの指定は必須であり、これによってすべてのリソースに関するデフォルトのアクセス要件が与えられる。しかし、セキュリティスキームは、また、モノレベルで与えられる構成をオーバーライドするフォームレベルで与えられる構成を使ってフォームごとに指定することもでき、きめの細かいアクセス制御の指定を可能にしている。アクセス制御機構が何も使用されていないことを示すために特別な nosec セキュリティスキームを使用することも可能である。追加例が後述される。

TD は、いくつかの名前空間にコンテキスト定義を追加する可能性を提供する。このメカニズムは、正式な知識、例えば、アプリケーションの特定のドメインの論理規則が所与の名前空間で見つけることができる場合には、TD インスタンスのコンテンツに追加の意味を結合するために使用することができる。コンテキスト情報は、forms フィールドで宣言された基礎となる通信プロトコルのいくつかの構成およびビヘイビアの指定するときに役立つ。例 2 は、SAREF-Smart Appliance Reference Ontology（スマート家電参照オントロジー）-[SMARTM2M]を参照するものとしてプレフィックス saref を宣言するために、@context に第 2 の定義を導入して、例 1 の TD サンプルを拡張したものである。このＩｏＴオントロジーには、モノの意味論と対話アフォーダンスを持たせる@type フィールドの値として設定することが出来る意味論上ラベルと解釈される用語が含まれる。下記の例では、モノは saref:LightSwitch、 status プロパティは saref:OnOffState、 toggle アクションは saref:ToggleCommand とラベル付けされる。

例 2: 意味論的注釈用の TD コンテキスト拡張子を持つ TD


```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        {"saref":"https://w3id.org/saref#"}
    ],
    "id": "urn:dev:ops:32473-WoTLamp-1234",
```

```
"name": "MyLampThing",
"@type": "saref:LightSwitch",
"securityDefinitions": {"basic_sc": {
    “scheme”: “basic”,
    "in": "header"
}},
"security": ["basic_sc"],
"properties": {
    "status": {
        "type": "string",
        "forms": [{
            "@type": "saref:GetCommand",
            "href":"https://mylamp.example.com/status"
        }]
    }
},
"actions": {
    "toggle": {
        "forms": [{
            "@type": "saref:ToggleCommand",
            "href":"https://mylamp.example.com/toggle"
```

https://www.w3.org/TR/wot-thing-description/

```
            }]
        }
    },
    "events": {
        “overheating” : {
            "data": {"type": "string"},
            "forms": [{
                "@type": "saref:NotifyCommand",
                "href":"https://mylamp.example.com/oh"
            }]
        }
    }
}
```

いくつかの@context 内の宣言メカニズムは、JSON-LD によって指定される。TD インスタンスは、本仕様のバージョン 1.1 [json-ld11] に準拠している。TD インスタンスは、また、RDF ドキュメントとして処理もできる。（意味処理の詳細は、付録 D.JSON-LD コンテキスト用法と名前空間ＩＲＩ内の資料（例： https://www.w3.org/2019/wot/td）参照。)

2. 適合性

標準ではないとされる項と同様に、本仕様書におけるすべてのオーサリングガイドライン、図、例、および注釈は標準ではない。本仕様内のこのようなものはすべて標準である。

本文書内のキーワード“**場合がある**”、”**しなければならない**“、”**してはならない**”、“**推奨される**”、”**べきである**”、**及び、**”**べきではない**”は、このように太字で表示されているときにのみ BCP 14 [RFC2119] [RFC8174]の記述に従い解釈するものとする。

TD インスタンスは、それが TD シリアライズに関する第５項 TD 情報モデル及び第６項 TD 表現フォーマットで　標準ステートメントに従う場合、本仕様に準拠する。

TD インスタンスを確認するための JSON スキーマ[JSON-SCHEMA]は、付録 B. TD インスタンス確認 JSON スキーマで提供されている。

3. 用語

モノ、コンシューマ、TD, 対話モデル、対話アフォーダンス、プロパティ、アクション、イベント、プロトコルバインディング、サービアント、WoT インターフェース、WoT ランタイム等の基本 WoT 用語は、WoT アーキテクチャ仕様の第３項で定義される [WOTARCHITECTURE]。

また、本仕様では、以下の定義をしている。

TD コンテキスト拡張子

追加のボキャブラリ用語を使って TD を拡張するメカニズム。これは、プロトコルバインディング、セキュリティスキーム、データスキーマなどの中核メカニズムに対する意味論的注釈および拡張にとって基本となる。

TD 情報モデル

制約が適用される事前定義のボキャブラリで構築されたクラス定義、したがって、これらのボキャブラリの意味論を定義する。クラス定義は、通常、シグネチャ(ボキャブラリ用語)及びそのシグネチャ内の関数で表現される。TD 情報モデルには、また、クラスに対するグローバル関数として定義されるデフォルト値が含まれる。

https://www.w3.org/TR/wot-thing-description/

TD プロセッサ

所与のフォーマットで TD の何らかの内部表現をシリアライズする、また／あるいは、そのフォーマットからそれをシリアライズ解除することができるシステム。TD プロセッサは、意味的に矛盾する TD、すなわち、モノクラスのインスタンス関係に対する制約を満たすことができない TD を検出しなければならない。そのために、TD プロセッサは、可能なすべてのデフォルト値が割り当てられている TD の標準フォームを計算することができるかもしれない。TD プロセッサは、通常、WoT ランタイムのサブシステムである。ＴＤプロセッサのインプリメンテーションは、(ＴＤドキュメントをシリアライズできる)ＴＤプロデューサのみ、あるいは、（ＴＤドキュメントからシリアライズ解除することができる）ＴＤコンシューマであろう。

TD シリアライズまたは TD ドキュメント

サービアント間で保存および交換できる TD のテキストまたはバイナリ表現。TD シリアライズは、ネットワーク上で交換されるときにメディアタイプによって識別される所与の表現フォーマットに従う。TD のデフォルト表現フォーマットは、本仕様が定義するように JSON ベースである。

ボキャブラリ

名前空間 IRI で識別されるボキャブラリ用語集。

用語・ボキャブラリ用語

文字列。用語がボキャブラリの一部である場合、すなわち、名前空間 IRI とプレフィックスがついている場合、それはボキャブラリ用語と呼ばれる。読みやすくするために、本書に記載されているボキャブラリ用語は、コンパクトな形式で書かれており、完全な IRI とはなっていない。

これらの定義は、第 5.2 項前付けの中でさらに詳細される。

4. 名前空間

本仕様第５項 TD 情報モデルで定義されている本バージョン TD 情報モデルは、以下の IRI によって識別される。

https://www.w3.org/2019/wot/td/v1

URI[RFC3986]でもあるこの IRI[RFC3987]は、JSON−LD コンテキストファイル[json−ld11]を得るためにデリフェレンスすることができ、TD ドキュメント内のコンパクトなストリングを完全な IRI ベースのボキャブラリ用語に拡張することができる。しかしながら、この処理は、JSON ベースの TD ドキュメントを TD プロセッサインプリメンテーションのオプション機能である RDF に変換するときにのみ要求される。

本仕様では、ボキャブラリ用語は常にコンパクトなフォームで表されている。それらの拡張されたフォームは、それらが属するボキャブラリの名前空間 IRI でアクセスすることができる。これらの名前空間は第 5.3 項クラス定義の構造に従う。TD 情報モデルで使用される各ボキャブラリは、以下のように、それ自体の名前空間 IRI を持つ。

ボキャブラリ	**名前空間 IRI**
コア	https://www.w3.org/2019/wot/td#
データスキーマ	https://www.w3.org/2019/wot/jsonschema#
セキュリティ	https://www.w3.org/2019/wot/security#
ハイパーメディアコントロール	https://www.w3.org/2019/wot/hypermedia#

ボキャブラリは互いに独立している。それらは、他の W3C 仕様で再利用・拡張することもある。ボキャブラリの設計を大きく変更するには、必ず新しい年ベースの名前空間 URI の割り当てが必要になる。TD 情報モデルの総体的な一貫性を維持するために、関連する JSON−LD コンテキ

https://www.w3.org/TR/wot-thing-description/

ストファイルは、大きくはない変更、特に、新しい用語の追加を識別するために、あらゆるバージョンがそれ自体の URI (v1、v1.1、v2,)を持つようにバージョン化されることに留意されたい。

いくつかの名前空間 IRI のボキャブラリでは大きくはない変更のみをすることができるので、そのコンテンツは、安全なキャッシュか、または、アプリへの組み込みが可能である。名前空間 IRI の下で比較的静的なコンテンツを公開することの利点の一つは、制約されたデバイス間で交換されるメッセージのペイロードサイズの最適化である。また、プライベートネットワークから公的に利用可能なボキャブラリにアクセスするデバイスが原因となるプライバシー漏洩を回避する(第 9.1 項プライバシーリスクをデリフェレンスするコンテキストも参照)。

5. TD 情報モデル

本項では、TD 情報モデルについて紹介する。TD 情報モデルは、TD およびそのシリアライズ処理のための概念的な基盤となる。これについては、別途、第 6 項 TD 表現フォーマットで説明する。

1. 概要

TD 情報モデルは、以下の独立したボキャブラリで構築される。

- プロパティ、アクション、イベントアフォーダンス、対話モデルを反映する中核 TD ボキャブラリ[WOT-ARCHITECTURE]
- JSON スキーマによって定義された用語(のサブセット) を含むデータ・スキーマ・ボキャブラリ[JSONSCHEMA]
- セキュリティメカニズムとその設定要件を識別する WoT セキュリティボキャブラリ
- ウェブリンクとフォームを使用して RESTful 通信の主な原則をエンコードするハイパーメディア制御ボキャブラリ

これらのボキャブラリのそれぞれは、本質的に、従来のオブジェクト指向の意味でオブジェクトとして解釈される、データ構造を構築するために使用することができる用語セットである。オブ

ジェクトは、クラスのインスタンスであり、プロパティを有する。W3C WoT のコンテキストでは、これらは、モノおよびそれらの対話アフォーダンスを示す。オブジェクトの正式定義は、第 5.2 項前付けに記載されている。TD 情報モデルの主要要素は、第 5.3 項クラス定義の中で提示する。デフォルト値がある場合、TD 内で特定のオブジェクトプロパティを省略してもよい。デフォルトのリストは第５．４項デフォルト値の定義にある。

下記の UML 図は、TD 情報モデルの概要を示している。これは、すべてのクラスとクラス間に存在する関連付けを表として表示している。矢印が示すようにクラス Thing(モノ)から始まる。読みやすくするために、図は、4 つの基本ボキャブラリのおのおの一個とし、4 つの部分に分割されている。

図 1 TD 中核ボキャブラリ

https://www.w3.org/TR/wot-thing-description/

✧ ✧ ✧

図 2 データ スキーマボキャブラリ

✧ ✧ ✧

図 3 WoT セキュリティボキャブラリ

✧ ✧ ✧

図 4 ハイパーメディア制御ボキャブラリ

2. 前付け

ツリーベース文書(すなわち、生 JSON 処理)および豊富な意味論的ウェブツーリング(すなわち、JSON−LD 処理)に関する簡易規則の両方による容易な処理が可能なモデルを提供するために、本文書はそれに応じて TD 情報モデルを構築するための以下のような形式的な前付けを定義する。

本項における全ての定義は、セットであり、これは、直感的に、それ自体がセットであり得る要素の集合である。全ての任意に複雑なデータ構造は、セットとして定義することができる。特に、**オブジェクト**は、以下のように再帰的に定義されるデータ構造である。

- 用語は、ボキャブラリに属している場合も、また、属さない場合もあるオブジェクトである。

- 名前が用語で、値が別のオブジェクトである名前と値のペアのセットもオブジェクトである。

この定義は、オブジェクトが同じ名前で複数の名前-値ペアを含むことを妨げるものではないが、一般に、本仕様では考慮されない。要素が名前として数字のみを有するオブジェクトは、配列と呼ばれる。同様に、名前として(ボキャブラリに属さない)用語のみの要素を有するオブジェクトは、マップと呼ばれる。マップ内のいくつかの名前−値ペアに現れるすべての名前は、マップの範囲内において一意であると想定される。

さらに、オブジェクトは、いくつかのクラスのインスタンスとすることができる。ボキャブラリ用語で表されるクラスは、まず、シグネチャと呼ばれるボキャブラリ用語セットで定義される。シグネチャが空であるクラスは、シンプルタイプと呼ばれる。

クラスのシグネチャは、クラスをさらに定義する 2 つの関数、すなわち、**割当関数**および**型関数**を構築することができる。クラスの割当関数は、入力としてクラスのシグネチャのボキャブラリ用語をうけ、true、 または、false のいずれかを出力として返す。直感的には、割当関数は、クラスの例を挙げるときにシグネチャの要素が必須であるか、または、任意選択であるかを示す。また、クラスの型関数も、クラスのシグニチャのボキャブラリ用語を入力として受け取り、別のクラスを出力として返す。これらの関数は部分的であり、それらのドメインは、定義されているクラスのシグニチャに限定される。

これらの 2 つの関数に基づいて、**インスタンスの関係**は、オブジェクトとクラスとからなるペアに対して定義することができる。この関係は満たすべき制約と定義される。すなわち、以下の 2 つの制約の両方ともが満たされる時にオブジェクトはクラスのインスタンスとなる。

- クラスの割り当て関数が true を返す全ての用語について、オブジェクトは、名前としてボキャブラリ用語を有する名前−値ペアを一個含んでいるか

- オブジェクトの何らかの名前−値ペアの中で名前として使用されるクラスのシグネチャ内の全てのボキャブラリ用語に対して、そのペアの値は、与えられたボキャブラリ用語に対するクラスの型関数が返すクラスのインスタンスであるのか

上記の定義によれば、オブジェクトは、その構造にかかわらず、あらゆる単純型のインスタンスである。しかし、インスタンス関係のもう一つの定義が単純型のために導入されている。つまり、オブジェクトが所与の語彙形式(例えば、boolean(ブール)型の場合は true と false であり、unsignedInt（アンサインドイント）型の場合 1、2、3 など)を持つ用語である場合、そのオブジェクトは単純タイプのインスタンスである。

さらに、**パラメータ化クラス**と呼ばれる追加のクラスを汎用マップおよび配列構造から導出することができる。オブジェクトが、その全ての名前−値ペアの値がこのクラスのインスタンスであるようなマップである場合、そのオブジェクトは、あるクラスのマップ、即ち、あるクラスでパラメータ化されたマップタイプのインスタンスである。同じことが配列にも当てはまる。

最後に、クラスは、前者のあらゆるインスタンスが後者のインスタンスでもある場合、何らかの他のクラスの**サブクラス**である。

上記のすべての定義に従い、TD 情報モデルは、クラス名(ボキャブラリ用語)、シグニチャ(ボキャブラリ用語のセット)、割当関数、および型関数を含むクラス定義のセットとして理解されるべきである。これらのクラス定義は、第 5.3 項クラス定義で表にまとめられている。各表とも、割当列内の値「必須」(あるいは、「オプション」)は、割り当て関数が、対応するボキャブラリ用語に対して true（あるいは、false)を返すということを示している。

慣例により、単純型は小文字で始まる名前で表される。TD 情報モデルは、XML スキーマ [xmlschema11-2-20120405]で定義されている以下の単純型を参照する。すなわち、string, anyURI, dateTime, integer, unsignedInt, double, boolean である。これらの定義(すなわち、それらの語彙形式の仕様)は、本 TD 情報モデルの範囲外である。

さらに、TD 情報モデルは、ボキャブラリ用語のペアに関するグローバル関数を定義する。この関数は、クラス名および別のボキャブラリ用語を入力として受け取り、オブジェクトを返す。返されたオブジェクトが null(ヌル)でない場合、それは、入力クラスのインスタンス内の入力ボキャブラリ用語の何らかの割当の**デフォルト値**を表す。この関数で割当関数に関して上で定義された制約を緩和することが可能となる。すなわち、オブジェクトがすべての必須割当を含む場合、あるいは、デフォルト値が欠けている割当について存在する場合、そのオブジェクトはクラスのインスタンスである。デフォルト値はすべて、表第 5.4 項デフォルト値定義の表に記載されている。第 5.3 項クラス定義の各表では、TD 情報モデルの クラスと ボキャブラリ用語の対応する組み合わせにデフォルト値がある場合、割当列には値「デフォルトあり」と書かれている。

ここで使用される形式化は、抽象データ構造としてのオブジェクトと、モノのような物理世界オブジェクト間で可能性のある関係を考慮していない。しかし、TD 情報モデルに含まれる全てのボキャブラリ用語を RDF リソースとして再解釈し、物理世界のより大きなモデルに結合する可能性(オントロジー)に対しては考慮された。この点については、付録 D.　TD オントロジーで取り扱わ

れる。意味処理の詳細は、付録D.JSON-LDコンテキスト用法と名前空間ＩＲＩ内の資料（例：_ https://www.w3.org/2019/wot/td）参照。

3. クラス定義

TDプロセッサは、TDが第５．３．１項中核ボキャブラリ定義、第５．３．２項データ・スキーマ・ボキャブラリ定義、第５．３．３項セキュリティボキャブラリ定義および第５．３．４ハイパーメディア制御ボキャブラリ定義で定義されているすべてのクラスに対するクラスインスタンス化制約条件を満足**していなければならない**。

1. 中核ボキャブラリ定義

1. Thing

メタデータおよびインターフェースがWoTTDによって記述される物理エンティティまたは仮想エンティティのアブストラクションであり、仮想エンティティは1つまたは複数のモノで構成される。

ボキャブラリ用語	**説明**	**割当**	**型**
@context（@コンテキスト）	JSON-LD キーワード。TD ドキュメント全体で使用される用語と呼ばれる略記名を定義する。	必須	anyURI または配列
@type（@タイプ）	JSON-LD キーワード。オブジェクトに意味タグ(またはタイプ) をラベル付けする。	オプション	string（ストリング）または string（ストリング）の配列
id	URI[RFC3986]形式のモノの識別子(例:　安定した	必須	anyURI または配列

	URI、一時的で不定なＵＲＩ，　ローカルＩＰアドレスを持つＵＲＩ、ＵＲＮ等)。		
title(タイトル)	デフォルト言語で人間が読み取り可能なタイトルを提供する(たとえば、UI 表現のテキスト表示)。	必須	string（ストリング）
titles(タイトル)	複数言語の人間が読み取り可能なタイトルを提供する(例えば、異なる言語での UI 表現のためのテキスト表示)。	オプション	MultiLanguage（複数言語）
description(説明)	デフォルト言語で追加の(人間が読み取り可能な) 情報を提供する。	オプション	string（ストリング）
descriptions(説明)	さまざまな言語で(人間が読み取り可能な)情報をサポートするために使用できる。	オプション	MultiLanguage（複数言語）
version(バージョン)	バージョン情報を提供する。	オプション	VersionInfo（バージョン情報）

https://www.w3.org/TR/wot-thing-description/

created(作成)	TD インスタンスが作成された時点の情報を提供する。	オプション	dateTime（日時）

modified(変更)	TD インスタンスが最後に変更された時点の情報を提供する。	オプション	dateTime（日時）
support(サポート)	TD 管理者に関する情報を URI スキーム(例えば、mailto[RFC6068]、tel [RFC3966]、https)として提供する。	オプション	anyURI
base(ベース)	TD ドキュメント全体にわたってすべての関連 URI 参照に使用さるベース URI を定義する。TD インスタンスでは、すべての関連 URI は、[RFC3986]で定義されているアルゴリズムを使用して、ベース URI に呼応して解読される。 base は、@context で使用される URI や意味処理が T D インスタンスに適用されるときに直接的に関連する Linked Data [LINKED-DATA]グラフ内で使用され	オプション	anyURI

https://www.w3.org/TR/wot-thing-description/

	るＩＲＩに作用しない。		
properties(プロパティ)	モノのすべてのプロパティベースの対話アフォーダンス。	オプション	PropertyAffordance（プロパティアフォーダンス）のマップ
actions(アクション)	モノのすべてのアクションベースの対話アフォーダンス。	オプション	ActionAffordance（アクションフォーダンス）のマップ
events(イベント)	モノのすべてのイベントベースの対話アフォーダンス	オプション	EventAffordance（イベントフォーダンス）のマップ
links(リンク)	指定された TD に関連する任意のリソースへのウェブリンクを提供する。	オプション	Link（リンク）の配列
forms(フォーム)	操作の実行方法を記述するフォームハイパーメディア制御のセット。フォームは、プロトコルバインディングのシリアライズである。本バーションの TD では、モノレベルで記述することができるすべての動作は、モノのプロパティと一度にひとまとめに対話する方法	オプション	Form（フォーム）の配列

に関するものである。

https://www.w3.org/TR/wot-thing-description/

security(セキュリティ)	セキュリティ定義名のセット。セキュリティ定義で定義されている名前から選択される。これら定義名は全て、リソースへアクセスするために満足されていなければならない。	必須	string（ストリング）または string（ストリング）の配列
securityDefinitions (セキュリティ定義)	名前付きセキュリティ構成セット(定義のみ)。security 名前－値ペアで使用しない限り、実際には適用されない。	必須	SecurityScheme（セキュリティスキーム）のマップ

@context 名前-値ペアは、anyURI 型の場合は直接的に、あるいは配列型の場合は最初の要素として、anyURI https://www.w3.org/2019/wot/td/v1 を含んでい**なければならない**。@context が配列である場合、anyURI https://www.w3.org/2019/wot/td/v1 の後に、任意の順序で anyURI 型またはマップ型の要素が続く**可能性もあるが**、@context 配列内のすべての名前-値ペアを持つ Map は一個のみとすることが**推奨される**。値が、anyURI 型の名前空間識別子であり、名前が、その名前空間を表す用語またはプレフィックスである場合、@context 配列に含まれるマップに名前-値ペアが入っている**場合もある**。名前が用語@language であり、値が[BCP47]で定義されている整形式言語タグ(例えば、en、de-AT、gsw-CH、zh-Hans、zh-Hant-HK, sl-neids)である場合、@context 配列に含まれる一個のマップは、TD のデフォルト言語を定義する名前-値ペアを含んでいる**べきである**。

すべての人間が読み取り可能なテキストストリングの基本方向の計算は、以下の一連の規則によって定義される。

- 言語タグが与えられない場合、基本方向は、CLDR 可能性サブタグ[LDML]のような最も強力なヒューリスティックまたは検出アルゴリズムによって推測される**べきである**。

- MultiLanguage(複数言語)マップ以外では、基本方向はデフォルト言語の言語タグから推測できる**かもしれない**。

- MultiLanguage(複数言語)マップの内部では、名前-値ペアの各値の基本方向は、対応する名前で与えられた言語タグから推測される**かもしれない**。

- ある言語を異なる基本方向を持つ複数のスクリプトで書くことができる場合、@language または MultiLanguage(複数言語)マップで与えられる対応する言語タグは、適切な基本方向が推論できるように、スクリプトサブタグを含んでい**なければならない**。例として Azeri があり、これは、ラテン語スクリプトを使用する場合は LTR(az-Latn で指定)、アラビア語スクリプトを使用する場合は RTL(az-Arab で指定)と書かれている。

TD プロセッサは、双方向テキストを処理する際に特殊なケースに気づくべきである。TD プロセッサは、ユーザにストリングを提示するとき、特に周囲のテキストに埋め込むとき(例えば、ウェブユーザインターフェース用)、注意して双方向分離を使用するようにすべきである。混合方向テキストは、言語が適切に識別された場合でも、全言語で起こりうる。

TD プロデューサは、単純なユーザエージェントによってうまく表示できるような方法で混合方向ストリングを提供するよう試みるべきである。例えば、RTL ストリングが LTR ラン(ラテン語スクリプトにおける数字あるいはブランドあるい商用名など)で始まる場合、ストリングの始めに RLM 文字を入れるか、または、双方向制御における逆方向ランをラップすれば、適切な表示を支援することができる。

ウェブ上のストリング:言語と方向のメタデータ[string-meta] は、何らかのガイダンスを提供し、双方向テキストを使用する場合の隠れた危険を多数例証している。

properties, actions, events(プロパティ、アクション、および、イベント)配列で明示的に提供される対話アフォーダンスに加えて、モノは、そのオプションの forms 配列で Form インスタンスによって示されるメタ対話も提供することができる。モノインスタンスの forms 配列が Form インスタンスを含んでいる場合、その名前 op に直接または配列内で割り当てられたストリング値は、readallproperties、writeallproperties、readmultipleproperties、writemultip l eproperties という操作タイプのうちのいずれかでなければならない。(モノインスタンスのフォーム使用例を参照。)

https://www.w3.org/TR/wot-thing-description/

これらのメタ対話のそれぞれのデータスキーマは、単一の ObjectSchema インスタンス内の各 PropertyAffordance インスタンスのデータスキーマを組み合わせることによって構築され、ObjectSchema インスタンスの properties マップは、対応する PropertyAffordance インスタンスの名前によって識別される PropertyAffordance の各データスキーマを含んでいる。

特に指定がない限り(例えば、TD コンテキスト拡張子を介して)、readmultipleproperties 操作の要求データは、期待される PropertyAffordance インスタンス名を含む配列となり、これは、Form インスタンスによって指定されたコンテンツタイプにシリアライズされている。

2. InteractionAffordance

コンシューマに可能な選択肢を示し、コンシューマがそのモノとどのように対話することができるかを示唆するモノのメタデータ。潜在的なアフォーダンスには多くの型があるが、W3C WoT では、3 つの対話アフォーダンス、すなわち、プロパティ、アクション、および、イベントを定義する。

ボキャブラリ用語	説明	割当	型
@type　（タイプ）	JSON-LD キーワード。オブジェクトに意味タグ(またはタイプ) をラベル付けする。	オプション	string(ストリング) または string
title(タイトル)	デフォルト言語に基づいて、人間が読み取り可能なタイトルを提供する(たとえば、UI 表現のテキストを表示)。	オプション	string（ストリング）
titles(タイトル)	複数言語の人間が読み取り可能なタイトルを提供する(例えば、異なる言語での UI 表現のためのテキスト表示)。	オプション	MultiLanguage（複数言語）
description(説明)	デフォルト言語で追加の(人間が読み取り可能な) 情報を提供する。	オプション	string（ストリング）
descriptions(説明)	さまざまな言語で(人間が読み取り可能な)情報をサポートするために使用できる。	オプション	MultiLanguage（複数言語）

https://www.w3.org/TR/wot-thing-description/

forms (フォーム)	操作の実行方法を記述するフォームハイパーメディア制御のセット。フォームは、プロトコルバインディングのシリアライズである。	必須	Form(フォーム)の配列
uriVariables	URI テンプレート変数を、データスキーマ宣言に基づいてコレクションとして定義する。	オプション	DataSchema（データスキーマ）のマップ

クラス対話アフォーダンスには、以下のサブクラスがある。

- PropertyAffordance
- ActionAffordance
- EventAffordance

3. PropertyAffordance

モノの状態を公開する対話アフォーダンス。この状態は検索(読み取り)でき、随意に更新(書き込み)することができる。モノは、変更後に新しい状態をプッシュすることによって、プロパティを観察可能にすることを選択することもできる。

ボキャブラリ用語	説明	割当	型
Observable(観察可能)	モノと仲介者を提供するサービアントが、このプロパティの observeproperty(オブザーブプロパティ) 操作をサポートするプロトコルバインディングを提供すべきかどうかを示すヒント。	オプション	Boolean(ブール)

注

プロパティインスタンスは、クラスのデータスキーマのインスタンスでもある。

したがって、type、unit、readOnly、writeOnly メンバーも含むことができる。

PropertyAffordance は、InteractiveAffordance クラスおよび DataSchema クラスのサブクラスである。Form インスタンスが PropertyAffordance インスタンス内にある場合、op に割り当てられる値は、readproperty、writeproperty、observeproperty、unobserveproperty、または、これらの組み合わせを含む配列のうちの 1 つで**なければならない**。

4. ActionAffordance

状態を操作する(例えば、ランプをオンまたはオフする)、または、モノ上のプロセスをトリガする

https://www.w3.org/TR/wot-thing-description/

(例えば、次第にランプの明るさを落とす)モノの関数の呼び出しを可能にする対話アフォーダンス。

ボキャブラリ用語	説明	割当	型
input(入力)	アクションの入力データスキーマを定義するために使用。	オプション	DataSchema（データスキーマ）
output(出力)	アクションの出力データスキーマを定義するために使用。	オプション	DataSchema（データスキーマ）
safe(安全)	アクションが安全(=true) であるかどうかシグナルする。アクションを呼び出したときに内部状態に変化がないかどうか示すために使用される。その場合、応答は例としてキャッシュすることができる。	デフォルトあり	boolean （ブール）
idemotent(冪等)	アクションが冪等（idempotent） (=ｔ rue) かどうかを示す。同じ入力で、（該当する場合）同じ結果を生じるアクションの呼び出しを繰り返し行えるかどうかを通知する。	デフォルトあり	boolean （ブール）

ActionAffordance は、InteractioniAffordance クラスのサブクラスである。フォームインスタンスがこの ActionAffordance インスタンス内にあるとき、op に割り当てられた値は、invokeaction（アクション呼び出し）で**なければならない**。

5. EventAffordance

イベントソースを記述する対話アフォーダンスであり、非同期的にイベント・データを消費者にプッシュする(例えば、オーバーヒート警報)。

ボキャブラリ用語	**説明**	**割当**	**型**
subscription(サブスクリプション)	サブスクリプション時に渡す必要があるデータを定義する(ウェブフックを設定するためのフィルタあるいはメッセージフォ-マットなど)。	オプション	DataSchema （データスキーマ）
dara(データ)	モノによってプッシュされるイベントインスタンスメッセージのデータスキーマを定義する。	オプション	DataSchema （データスキーマ）
cancellation(キャンセル)	サブスクリプションをキャンセルするために渡す必要があるデータを定義する(ウェブフックを削除するための特定のメッセー	デフォルトあり	DataSchema （データスキーマ）

https://www.w3.org/TR/wot-thing-description/

ジなど)。

EventAffordance は、InteractionAffordance クラスのサブクラスである。Form インスタンスが EventAffordance インスタンス内にあるとき、op に割り当てられた値は、配列内の subscribeevent、unsubscribeevent、または、両方のうちのいずれかでなければならない。

6. VersionInfo

TD ドキュメントのバージョン情報を提供するモノのメタデータ。必要に応じて、ファームウェアおよびハードウェアバージョン(TD 名前空間外における用語定義)などの追加バージョン情報は、TD コンテキスト拡張子メカニズムを介して拡張することができる。

ボキャブラリ用語	**説明**	**割当**	**型**
Instance(インスタンス)	この TD インスタンスのバージョン ID を提供する。	必須	string(ストリング)

VersionInfo クラス のインスタンス内の値は、意味論的バージョニングパターンに従うことが推奨される。ここで、ドットで区切られた 3 つの数字のシーケンスは、メジャーバージョン、マイナーバージョン、パッチバージョンをそれぞれ示す。詳細は[SemVer]を参照。

7. MultiLanguage

[BCP47]に記述されている言語タグによって識別される異なる言語で人間が読み取り可能なテキストを提供するマップ。例えば、TD インスタンス内のコンテナの使用法に関しては、第 6.3.2 項人間が読み取り可能なメタデータ参照。

MultiLanguage マップの各名前は、[BCP47]で定義されているような言語タグでなければならない。

MultiLanguage マップの各値は、string 型でなければならない。

https://www.w3.org/TR/wot-thing-description/

2. データ・スキーマ・ボキャブラリ定義

データスキーマ定義は、JSON スキーマ[JSON-SCHEMA] によって定義された用語の非常に一般的なサブセットを反映している。TD インスタンス内のデータスキーマ定義は、この定義されたサブセットに限定されておらず、第 7 項 TD コンテキスト拡張子で解説されている追加用語のためのＴＤコンテキスト拡張子を使用している JSON スキーマで見られる追加の用語を使用してもよいことに留意されたい。あるいは、これら用語は、ＴＤプロセッサに意味的に無視される。（意味処理の詳細は、付録 D.JSON-LD コンテキスト用法と名前空間ＩＲＩ内の資料（例: https://www.w3.org/2019/wot/td）参照。)

1. DataSchema

使用されるデータフォーマットを記述するメタデータ。確認のために使用することができる。

ボキャブラリ用語	説明
@type　（タイプ）	オブジェクトに意味論タグ(または型)をラベル付けするための JSON-LD キーワード。
title(タイトル)	デフォルト言語に基づいて、人間が読み取り可能なタイトルを提供する(たとえば、UI 表現のテキストを表示)。
titles(タイトル)	複数言語の人間が読み取り可能なタイトルを提供する(例えば、異なる言語での UI 表現のためのテキスト表示)。	オ
description(説明)	デフォルト言語で追加の(人間が読み取り可能な) 情報を提供する。	オ
descriptions(説明)	さまざまな言語で(人間が読み取り可能な)情報をサポートするために使用できる。	オ
type(型)	操作の実行方法を記述するフォームハイパーメディア制御のセット。フォームは、プロトコルバインディングのシリアライズである。	オ
Const	URI テンプレート変数を、データスキーマ宣言に基づいてコレクションとして定義する。	オ
unit(単位)	国際科学、エンジニアリング、ビジネスなどで使用さ	オ

	れる単位情報を提供する。
one of（の一つ）	配列内の指定された一つのスキーマに対してデータが有効であることを保証するために使用される。

https://www.w3.org/TR/wot-thing-description/

enum	配列として提供される制限された値のセット。
readOnly（読取り専用）	プロパティ対話/値が読み取り専用(=true) かそうではないか(=false) を示すヒントとなるブール値。
writeOnly（書き込み専用）	プロパティ対話/値が書き込み専用(=true) かそうではないか(=false)を示すヒントであるブール値。

クラス DataSchema には、以下のサブクラスがある。

- ArraySchema
- BooleanSchema
- NumberSchema
- IntegerSchema
- ObjectSchema
- StringSchema
- NullSchema

ボキャブラリ用語	説明	割当
Items(項目)	配列の特性を定義するために使用される。	オプ
minItems（最小項目）	配列に含めなければならない項目の最小数を定義する。	オプ
maxItems（最大項目)	配列に含める必要がある項目の最大数を定義する。	オプ

フォーマットストリング値は、[JSON-SCHEMA] (特に第 7.3 項定義フォーマット) で定義されている固定値セットとそれに対応するフォーマットルールから判別できる。サービアントは、format 値を使用して、それに応じて追加の確認を実行. **しても良い**。既知の値セットで見つからない値が format に割り当てられる場合、そのような確認は成功**すべきである**。

2. ArraySchema

配列型のデータを記述するメタデータ。このサブクラスは、DataSchema インスタンス内の type に割り当てられた値の array によって示される。

3. BooleanSchema

boolean 型のデータを記述するメタデータ。このサブクラスは、DataSchema インスタンス内の type に割り当てられた値 boolean によって示される。

4. NumberSchema

number 型のデータを記述するメタデータ。このサブクラスは、DataSchema インスタンス内の type に割り当てられた値 number によって示される。

ボキャブラリ用語	説明	割当

https://www.w3.org/TR/wot-thing-description/

minimum(最小)	最小数値を指定する。関連付けられた数値型または整数型のみに適用される。
maximum(最大)	最小数値を指定する。関連付けられた数値型または整数型のみに適用される。

5. IntegerSchema

integer 型のデータを記述するメタデータ。この

サブクラスは、DataSchema インスタンスで type に割り当てられた値 integer によって示される。

ボキャブラリ用語	説明	割当
minimum(最小)	最小数値を指定する。関連付けられた数値型または整数型のみに適用される。	オプ
maximum(最大)	最小数値を指定する。関連付けられた数値型または整数型のみに適用される。	オプ

6. ObjectSchema

object 型のデータを記述するメタデータ。このサブクラスは、DataSchema インスタンス内の type に割り当てられた値 object によって示される。

ボキャブラリ用語	説明	割当	型
properties(プロパティ)	データスキーマのネストされた定義。	オプション	DataSchema のマップ
required(要求される）	オブジェクト型のどのメンバーが必須であるかを定義する。	オプション	string(ストリング)の配列

7. StringSchema

string 型のデータを記述するメタデータ。このサブクラスは、DataSchema インスタンス内の type に割り当てられた値 string によって示される。

8. NullSchema

null 型のデータを記述するメタデータ。このサブクラスは、DataSchema インスタンス内の type に割り当てられた値 null によって示される。このサブクラスは、1 つの許容可能な値、すなわち、null のみを記述する。これは、情報として使われる場合、データも null になれるという oneOf 宣言の一部として使用することができる。

3. セキュリティボキャブラリ定義

本仕様は、W3C WoT のプロトコルバインディングとして適格なプロトコルに直接組み込まれるか、または、これらのプロトコルと組み合わて広く使用されている十分に確立されたセキュリティメ

https://www.w3.org/TR/wot-thing-description/

カニズムから選ばれたメカニズムを提供する。現在の HTTP セキュリティスキームセットは、部分的に OpenAPI 3.0.1 に基づいている([OPENAPI]も参照)。しかしながら本仕様で与えられた HTTP セキュリティスキーム、ボキャブラリ、構文は、OpenAPI と多くの類似点があるが、それらは互換性を持たない。

1. SecurityScheme

セキュリティメカニズムの構成を記述するメタデータ。名前 scheme に割り当てられた値は、第 5 項 TD 情報モデルで定義される標準ボキャブラリ、あるいは、TD コンテキスト拡張子のいずれかで、TD に含まれるボキャブラリの中で定義され**なければならない**。

ボキャブラリ用語	説明
@type（@型）	オブジェクトに意味タグ(またはタイプ) をラベル付けするための JSON-LD キーワード。
ｓ chema(スキーマ)	構成されているセキュリティメカニズムの識別。
description(説明)	デフォルト言語に基づいて、追加の(人間が読み取り可能な)情報を提供する。
descriptions(説明)	さまざまな言語の(人間が読み取り可能な)情報をサポートするために使用できる。
proxy(プロキシ)	このセキュリティ構成がアクセスを提供するプロキシーサーバの URI。指定されていない場合、対応するセキュリティ構成はエンドポイント用である。

クラス SecurityScheme は、以下のサブクラスを有する。

NoSecurityScheme
BasicSecurityScheme
DigestSecurityScheme
APIKeySecurityScheme
BearerSecurityScheme
CertSecurityScheme
PSKSecurityScheme
PublicSecurityScheme
PoPSecurityScheme
OAuth2SecurityScheme

https://www.w3.org/TR/wot-thing-description/

2. NoSecurityScheme

リソースにアクセスするために必要とされる認証または他のメカニズムがないことを示すボキャブラリ用語によって識別される nosec (すなわち、「scheme」:「nosec」)に対応するセキュリティ構成。

3. BasicSecurityScheme

暗号化されていないユーザ名およびパスワードを使用するボキャブラリ用語 basic(すなわち、「scheme」:「basic」)によって識別される基本認証[RFC7617]セキュリティ構成。本スキームは、例えば、TLS のように機密性を提供する他の何らかのセキュリティメカニズムと共に使用されるべきである。

ボキャブラリ用語	説明	割当	型
In(内)	セキュリティ認証情報の場所を指定する。	デフォあり	string (header, query,body, cookie のいずれか)
name(名前)	クエリ、ヘッダー、あるいは、クッキーのパラメータ名。	オプション	string(ストリング)

4. DigestSecurityScheme

ボキャブラリ用語 digest (すなわち、「scheme」:「digest」)によって識別されるダイジェストアクセス認証[RFC7616]セキュリティ構成。本スキームは、基本認証に似ているが、中間者攻撃を回避する機能が追加されている。

https://www.w3.org/TR/wot-thing-description/

ボキャブラリ用語	説明
qop	保護の質。
in (内)	セキュリティ認証情報の場所を指定する。
name(名前)	クエリ、ヘッダー、あるいは、クッキーのパラメータ名。

ッキーのパラメータ名。

5. APIKeySecurityScheme

ボキャブラリ用語 apikey (すなわち、「scheme」:「apikey」)によって識別される API キー認証セキュリティ構成。これは、アクセストークンが不透明であり、標準トークンフォーマットを使用していない場合に用いる。

ボキャブラリ用語	説明
in (内)	セキュリティ認証情報の場所を指定する。
name(名前)	クエリ、ヘッダー、あるいは、ク

6. BearerSecurityScheme

ベアラトークンが OAuth2 とは別個に使用される状況におけるボキャブラリ用語 bearer(すなわち、「scheme」:「bearer」)によって識別されるベアラトークン[RFC6750]セキュリティ構成。本スキームは、ベアラトークンが OAuth2 とは独立して使用される状況のためのものである。oauth2 スキームが指定されている場合は、通常、本スキームを暗示・指定する必要はない。Format に関しては、値 jwt は[RFC7519]への適合性を示し、jws は[RFC7797]への適合性を示し、cwt は[RFC8392]への適合性を示し、jwe は[RFC7516]への適合性を示し、alg の値はこれらの標準と整合して解釈される。ベアラトークンのための他のフォーマットおよびアルゴリズムは、ボキャブラリ拡張において指定**もできる**。

ボキャブラリ用語	**説明**
authorization(許可)	許可サーバの URI。
alg(アルゴリズム)	エンコーディング、暗号化、またはダイジェストアルゴリズム。
format(フォーマット)	セキュリティ認証情報のフォーマットを指定する。
in（内）	セキュリティ認証情報の場所を指定する。
name(名前)	クエリ、ヘッダー、あるいいは、クッキーのパラメータ名。

7. CertSecurityScheme

本項は危険な状態にある。

ボキャブラリ用語 cert (すなわち、「scheme」:「cert」)によって識別される[X509V3]に準拠する証明書ベースの非対称キーセキュリティ構成。

ボキャブラリ用語	**説明**	**割当**
identity（ID）	選択または確認に使用できる情報を提供する識別子。	オプ

8. PSKSecurityScheme

ボキャブラリ用語 psk (すなわち、「scheme」:「psk」)によって識別される事前共有キー認証セ

https://www.w3.org/TR/wot-thing-description/

キュリティ構成。

ボキャブラリ用語	説明
identity（ID）	選択または確認に使用できる情報を提供する識別子。

9. PublicSecurityScheme

本項は危険な状態にある。

ボキャブラリ用語 public (すなわち、「scheme」:「public」)によって識別される生の公開キー非対称鍵セキュリティ構成。

ボキャブラリ用語	説明
identity（ID）	選択または確認に使用できる情報を提供する識別子。

10.PoPSecurityScheme

本項は危険な状態にある。

ボキャブラリ用語 pop(すなわち、「scheme」:「pop」)によって識別される PoP(Proof-of-possession)トークン認証セキュリティ構成。jwt は[RFC7519]との適合性を示し、jws は[RFC7797]との適合性を示し、cwt は[RFC8392]との適合性を示し、jwe は[RFC7516]との適合性を示し、alg の値はこれらの標準と整合して解釈される。PoP トークンのための他のフォーマットおよびアルゴリズムは、ボキャブラリ拡張において指定**もできる**。

ボキャブラリ用語	説明
authorization(許可)	許可サーバの URI。
alg(アルゴリズム)	エンコーディング、暗号化、またはダイジェストアルゴリズム。
format(フォーマット)	セキュリティ認証情報のフォーマットを指定する。
in（内）	セキュリティ認証情報の場所を指定する。	デフ
name(名前)	クエリ、ヘッダー、あるいは、クッキーのパラメータ名。	オフ

11.OAuth2SecurityScheme

本項は危険な状態にある。

ボキャブラリ用語 oauth2 (すなわち、「scheme」:「oauth2」) によって識別される [RFC6749] および [RFC8252] に準拠するシステムの OAuth2 認証セキュリティ構成。implicit フローについては、authorization が含ま**れなければならない**。password と client フローについて、token が含ま**れなければならない**。code フローについては、authorization と token の両方が含まれな**ければならない**。SecurityScheme に scope が定義されていない場合、それらは空であるとみなされる。

https://www.w3.org/TR/wot-thing-description/

ボキャブラリ用語	説明
authorization(許可)	許可サーバの URI。
token(トークン)	トークンサーバの URI.
refresh(リフレッシュ	リフレッシュサーバの URI
scopes(範囲)	配列として提供される許可範囲 ID セット。これらは、クライアントがどのリソースにどのようにアクセスできるかを識別するために、許可サーバによって返されるフォームに関連付けられたトークンとして提供される。フォームに関連付けられている値は、そのフォーム上でアクティブな OAuth2SecurityScheme で定義される値の中から選択する必要がある。
flow(フロー)	許可フロー

4. ハイパーメディア制御ボキャブラリ定義

本モデルは、(タイプされた) Web リンクおよびモノが公開される Web フォームの表現を提供する。Link クラス定義は、Web Linking [RFC8288] で定義されている非常に一般的な用語のサブセットを反映している。定義された用語は、例えば、Switch というモノによって制御される Lamp というものなどの別のモノとの関係を記述するために使用することができる。Form クラスは、モノ(及び他のウェブリソース)の状態を操作するために新たに導入されたハイパーメディア制御に対応する。

1. Link(リンク)

リンクは、「リンクコンテキストは、リンクターゲットにリレーション型リソースを有する」という形式のステートメントとみなすことができ、ここでオプションのターゲット属性によって、リソースをさらに記述することができる。

ボキャブラリ用語	説明
href	リンクのターゲット IRI、または、フォームの送信ターゲット。
type(型)	リンクをデリフェレンスした結果のメディアタイプ[RFC2046]が何であるべきかを示すヒントを提供するターゲット属性。
rel	リンクリレーション型はリンクの意味を識別する。
anchor(アンカー)	所与のＵＲＩあるいはＩＲＩを使って、リンクコンテキスト(デフォールトで、それ自体がその id により識別されるモノ)を無効にする。

ステートメントとみなすことができ、オプションのフォームフィールドは、必要な要求をさらに記述することができる。TD では、フォームコンテキストは、Properties、Actions、および Events などの周囲のオブジェクト、または、メタ対話のためのモノ自体となる。

2. Form

フォームは、「フォームコンテキスト上でオペレーションタイプ操作を実行するために、送信ターゲットへ要求メソッドを要求する」という

https://www.w3.org/TR/wot-thing-description/

ボキャブラリ用語	説明	割当
op	フォームで記述された操作を実行する意味・意図を示す。例えば、Property 対話は、get および set 動作を可能にする。プロトコルバインディングは、ゲットオペレーションのためのフォームと、セットオペレーションのための別のフォームを含むことができる。op 属性は、どのフォームがどのフォームのためのものであるかを示し、クライアントが必要とされる動作のための正しいフォームを選択することができるようにする。op には、操作の意味意図をそれぞれが表す 1 つまたは複数の対話動詞を割り当てることができる。	デフォルトあり
href	リンクのターゲット IRI、またはフォームの送信ターゲット。	必須
contentType（コンテンツタイプ）	メディアタイプ (例: 'text/plain') およびメディアタイプ [RFC2046]の潜在パラメータ(例: 'charset=utf-8') に基づいて、コンテンツタイプを割り当てる。	デフ
contentCoding（コンテンツコーディング）	コンテンツコーディング値は、表現に使用されている、あるいは、適用可能な符号化変換を示す。コンテンツコーディングは、主に、そのメディアタイプのアイデンティティを失うことなく、かつ、情報を失うことなく、表現が圧縮、または、そうでなければ有用に変換を可能にするために使用される。コンテンツコーディングの例は、「gzip」、「deflate」など。	オプ

subptotocol(サブプロトコル)	複数のオプションがある場合に、特定プロトコルに対して対話が実行されるそのメカニズムを示す。例えば、HTTP およびイベントの場合、これは、ロングポーリング(longpoll)、ウェブサブ[websub](websub),サーバが送信したイベント[eventsource](sse)などいくつか利用可能なメカニズムのうちのどれを非同期通知に使用すべきかを示す。サブプロトコルの選択に制限はなく、他のメカニズムもこのサブプロトコル用語によってアナウンスできることに留意されたい。	オ
security(セキュリティ)	securityDefinitions で定義されている。定義から選択されたセキュリティ定義名セット。リソースへのアクセスのためにはこれら全てが満たされなければならない。	オ
scopes(範囲)	配列として提供される許可範囲 ID セット。これらは、クラ	オ

https://www.w3.org/TR/wot-thing-description/

イアントがどのリソースにどのようにアクセスできるかを識別するために、許可サーバによって返される、フォームに関連付けられたトークンとして提供される。フォームに関連付けられている値は、そのフォーム上でアクティブなOAuth2SecuritySchemeで定義される値の中から選択する必要がある。

response(応答)	本オプション用語は、たとえば、以下の場合に使用することができる。出力通信メタデータが入力メタデータとは異なる(例えば、出力コンテンツ型が入力コンテンツ型とは異なる)。応答名には、応答メッセージにのみ有効なメタデータが含まれる。

フォームの可能な操作型リストは確定されたものである。本仕様書の本バージョンでは、[WOT- ARCHITECTURE]に記述された WoT 対話モデルをインプリメンテーションするために必要な周知の型のみを含んでいる。本標準の将来バージョンはこのリストを拡張したものになるかもしれない。が、操作型は、サービアントが任意に設定する**べきではない**。

オプションの response 名前-値ペアは、期待される応答メッセージのメタデータを提供するために使用することができる。中核ボキャブラリでは、そのペアはコンテンツタイプ情報を持つのみであるが、TD コンテキスト拡張子を適用することができる。応答名-値ペアが与えられていない場合、応答のコンテンツタイプは、フォームインスタンスに割り当てられたコンテンツタイプと同じであると仮定しなければならない。ExpectedResponse クラス内の contentType は、デフォルト値を持っていないということに留意されたい。例えば、フォームのコンテンツタイプの値が application/xml である場合、応答のコンテンツタイプの仮定値も application/xml となる。

いくつかの使用事例では、入出力データは、異なる形式、例えば、JSON を受け入れるが画像を返すアクションで表される事もある。そのような場合、オプションの response 名前-値ペアは、期待される応答のコンテンツタイプを記述することができる。期待される応答のコンテンツタイプがフォームのコンテンツタイプと異なる場合、フォームインスタンスは、名前 response を持つ名前-値ペアを含んでい**なければならない**。例えば、ActionAffordance は、その入力データとして application/json のみを受け入れることができ、一方、その出力データとして image/jpeg コンテンツタイプで応答する。その場合、コンテンツタイプは異なり、response 名前-値ペアが、応答コンテンツタイプ(image/jpeg)情報をコンシューマに提供するために使用されなければならない。

contentCoding プロパティとして可能な値は、例えば、IANA HTTP コンテンツコーディングレジストリで見つけることができる。

3. ExpectedResponse

期待される応答メッセージを記述する通信メタデータ。

ボキャブラリ用語	説明	割当
contentType（コンテンツタイプ）	メディアタイプ(例: ’ text/plain’)およびメディアタイプ[RFC2046]の潜在パラメータ(例: 'charset=utf-8') に基づいて、コンテンツタイプを割り当てる。	必須

https://www.w3.org/TR/wot-thing-description/

5.4 デフォルト値定義

TD 内の割当が欠落している場合、TD プロセッサは、第 5.4 項デフォルト値定義内の表に示されているデフォルト値割当に従わな**ければならない**。

下表は、TD 情報モデルで定義されているデフォルト値のすべてである。

クラス	**ボキャブラリ用語**
Form	readOnly
DataSchema	writeOnly
DataSchema	safe
ActionAffordance	idempotent
Form	op
Form	op
Form	op

BasicSecurityScheme	in	head
DigestSecurityScheme	in	head
BearerSecurityScheme	in	head
PoPSecurityScheme	in	head
本機能は危険な状態である。		
APIKeySecurityScheme	in	quer
DigestSecurityScheme	qop	auth
BearerSecurityScheme	alg	ES2
PoPSecurityScheme	alg	ES2
本機能は危険な状態である。		
BearerSecurityScheme	format	jwt
PoPSecurityScheme	format	jwt
本機能は危険な状態である。		

https://www.w3.org/TR/wot-thing-description/

6. TD 表現形式

WoTTD は、モノを表し、第 5 項 TD 情報モデルに基づいてモデル化され、構成されている。本項では、TD 情報モデルが定義するクラス Thing のインスタンスのシリアライズであるモノの JSON ベースの表現形式を定義している。

TD プロセッサは、第 6.1 項 JSON タイプへのマッピングと第 6.3 項情報モデルシリアライズに記載されている規則に従って、JSON フォーマット[RFC8259]に TD をシリアルライズする、また・あるいは、そのフォーマットから TD をシリアライズ解除することができ**なければならない**。

TD 情報モデルの JSON シリアライズは、意味評価を簡素化するために、JSON-LD 1.1[jsonld11]の構文と整合化されている。従って、TD 表現フォーマットは、生の JSON として、または、第 D 項 RDF への変換でさらに詳述するように、JSON−LD 1.1 プロセッサを用いて処理することができる。意味処理の詳細は、付録 D.JSON-LD コンテキスト用法と名前空間ＩＲＩ内の資料（例: https://www.w3.org/2019/wot/td）参照。

相互運用可能な国際化をサポートするために、TD は、オープンエコシステムのための RFC8259[RFC8259]の第８項で定義されている要件に従ってシリアライズされ**なければならない**。要約すると、以下が要求される:

- TD は UTF-8[RFC3629]を使って符号化され**なければならない**。
- インプリメンテーションは、TD ドキュメントの先頭に BOM（byte order mark）(U+FEFF)を追加し**てはならない**。
- TD プロセッサは、BOM をエラーとして扱うのではなく、その存在を無視**してもよい**。

1. JSON タイプへのマッピング

TD 情報モデルは、モデルオブジェクトと JSON タイプ間で容易にマッピングできるように構築される。すべてのクラスは JSON オブジェクトへのマップを例に挙げ、クラスインスタンスの各名前−値ペアは JSON オブジェクトのメンバーである。

第 5.3 項クラス定義(すなわち、string、anyURI、dateTime、integer、unsignedInt、double、boolean)で言及されているシンプルタイプはすべて、下記の規則に従って、基本 JSON タイプ(string、number、boolean)へマップする。これらの規則、名前-値ペアに適用される。

- string 型または anyURI 型の値は、JSON ストリングとしてシリアライズされ**なければならない**。
- dateTime 型の値は、[RFC3339]で指定された"日-時"形式で JSON ストリングとしてシリライズされなければならない。例には、2019-05-24T13:12:45Z および 2015-0711T09:32:26+08:00 などが

ある。dateTime 型の値は、オフセットの代わりに UTC タイムゾーンを表すリテラル z を使用**すべきである**。

- integer 型または unsignedInt 型の値は、小数部または指数部のない JSON 数としてシリアライズし**なければならない**。
- double 型の値は、JSON 番号としてシリアライズし**なければならない**。
- boolean 型の値は、JSON ブールとしてシリアライズし**なければならない**。

TD 情報モデルの複合型(すなわち、配列、マップ、およびクラスインスタンス)はすべて、以下の規則に従って、構成された JSON 型(配列およびオブジェクト)にマッピングされる。

型 Array の値は、JSON 配列としてシリアライズされ**なければならない**。名前-値ペアの各値は、そのペアの数値名で順序付けられた JSON 配列の要素とする。

型 Map の値は、JSON オブジェクトとしてシリアライズされなければならず、名前-値ペアの各値は、JSON オブジェクトのメンバーとする。

クラスインスタンスは JSON オブジェクトとしてシリアライズされ**なければならない**。これは、個々に第 6.3 項 情報モデルシリアライズで与えられている詳細な規則に従って行われなければならない。

2. デフォルト値の省略

TD シリアライズでは、第 5.4 項デフォルト値の定義の表のようにデフォルト値が定義されているボキャブラリ用語を省略してもよい。

以下の例は、デフォルト値(=チェックがついたチェックボックス) を持つメンバーも含めるためのチェックボックスを持つ例 1 の TD インスタンスである。これらのメンバーは、TD シリアライズを単純化するために省略することができる(=チェックがついていないチェックボックス)。TD プロセッサは、あたかも所与のデフォルト値によって明らかに存在するかのように、全く同じようにこれらの省略されたメンバーを解釈することに留意されたい。

例 3

デフォルト値で

{

"@context": https://www.w3.org/2019/wot/td/v1,

"id": "urn:dev:ops:32473-WoTLamp-1234",

"title": "MyLampThing",

"securityDefinitions": {

"basic_sc": {

“Scheme”:”basic”,、

```
            "in": "header"
        }
    },
    “security”: [
        "basic_sc"
    ],
"properties": {
    "status": {
        "type": "string",
        "readOnly": false,
        "writeOnly": false,
        "observable": false,
        "unit": "percent",
        "minimum": 0,
        "maximum": 100,
        "forms": [
            {
                "op": "readproperty",
                "href": "https://mylamp.example.com/status",
                "contentType": "application/json",
                "htv:methodName": "GET"
            }
        ]
    },
    "color": {
        "type": "object",
        "properties": {
            "r": {
                "type": "integer",
                "minimum": 0,
                "maximum": 255
            },
            "g": {
                "type": "integer",
                "minimum": 0,
                "maximum": 255
            },
            "b": {
                "type": "integer",
                "minimum": 0,
                "maximum": 255
            }
        },
        "forms": [
            {
                "op": "readproperty",
                "href": "https://mylamp.example.com/color",
                "contentType": "application/json",
                "htv:methodName": "GET"
            },
            {
                "op": "writeproperty",
                "href": "https://mylamp.example.com/color",
                "contentType": "application/json",
                "htv:methodName": "PUT"
            }
        ]
    }
},
"actions": {
    "toggle": {
        "forms": [
            {
                "op": "invokeaction",
                "href": "https://mylamp.example.com/toggle",
                "htv:methodName": "POST"
            }
        ]
    }
},
"events": {
    "overheating": {
        "data": {
            "type": "string"
        },
        "forms": [
            {
                "op": "subscribeevent",
                "href": "https://mylamp.example.com/oh",
                "contentType": "text/plain",
                "subprotocol": "longpoll"
            }
        ]
    }
}
```

https://www.w3.org/TR/wot-thing-description/

```
r
i
t
e
O
n
l
y
"
:
f
a
l
s
e
,
"
f
o
r
m
s
"
:
[
{
    "op": [
        "
        r
        e
        a
        d
        p
        r
        o
        p
        e
        r
        t
        y
        "
        ,
        "
        w
        r
        i
        t
        e
        p
        r
        o
        p
        e
        r
        t
        y
        "
    ],
    “h
    re
    f”
    :”
    ht
    tp
    s:/
    /
    m
    yl
    a
    m
    p.
    ex
    a
    m
```

```
                pl
                e.
                co
                m
                /
                st
                at
                us
                」
                ,

                "c
                on
                te
                nt
                T
                yp
                e"
                :
                "a
                pp
                lic
                ati
                on
                /js
                on
                "

            }]

        }

    },

    "actions": {

        "toggle": {
```

```
                "
                s
                a
                f
                e
                "
                :

                f
                a
                l
                s
                e
                ,

                "
                i
                d
                e
                m
                p
                o
                t
                e
                n
                t
                "
                :

                f
                a
                l
                s
                e
                ,

                "
```

https://www.w3.org/TR/wot-thing-description/

```
forms": [
    {
        "op": "invokeaction",
        「href」:"https://mylamp.example.com/toggle2",
        "contentType": "application/json",
    }]
},
"events": {
"overheating": {
    "data
```

a
"
:

{

"
t
y
p
e
"
:

"
s
t
r
i
n
g
"
,

"
r
e
a
d
O
n
l
y
"

:

f
a
l
s
e
,

"
w
r
i
t
e
O
n
l
y
"

:

f
a
l
s
e

},

"forms": [{

"op": "subscribeevent",

“h
re
f”
:h
tt

https://www.w3.org/TR/wot-thing-description/

ps
://
m
yl
a
m
p.
ex
a
m
pl
e.
co
m
/
oh
,,

oc
ol
":
"l
on
gp
ol
l"

}]

}

}

}

"c
on
te
nt
T
yp
e"
:
"a
pp
lic
ati
on
/js
on
",

"s
ub
pr
ot

使用されるプロトコルバインディングに応じて、追加のプロトコル固有のボキャブラリ用語が適用されることがあるということに留意されたい。これらは、関連するデフォルト値を持っており、したがって、本サブ項で説明されているように省略することもできる。さらなる情報は、第 8.3 項プロトコルバインディングに記載されている。

3. 情報モデルのシリアライズ

1. モノのルートオブジェクト

TD は、型 Thing のオブジェクトをルートとするデータ構造である。その代り、TD の JSON シリアライズは、TD 情報モデルから構築された構文ツリーのルートである JSON オブジェクトである。

TD シリアライズのルート要素は、@context という名前のメンバーと、https://www.w3.org/2019/wot/td/v1 と等しいか、それを含むストリングまたは配列型の値を含む JSON オブジェクトで**なければならない**。

一般に、この URI は、本仕様が定義する TD 表現フォーマットバージョンを識別するために使用される。JSON-LD 処理[json-ld11] の場合、この URI は TD コンテキストファイルを指定する。配列型の@context は、TD コンテキスト拡張子を示す(詳細は第 7 項ＴＤコンテキスト拡張子を参照)。

例 4

```
{
    "@context": https://www.w3.org/2019/wot/td/v1,
    ...
}
```

名前が Thing のシグニチャ内のボキャブラリ用語である場合、Thing のインスタンスの名前-値ペアは、ルートオブジェクトの JSON メンバーとしてシリアライズされな**ければならない**。すべての必須およびオプションメンバーを含むシリアライズされたルートオブジェクトの TD スニペットを以下のとおり:

例 5: モノシリアライズの例

```
{
    "@context": https://www.w3.org/2019/wot/td/v1,
    "@type": "Thing",
    "id": "urn:dev:ops:32473-Thing-1234",
    "title": "MyThing",
    "titles": {...},
    "description": "Human readable information.",
    "descriptions": {...},
    "support": mailto:support@example.com,
    "version" : {...},
    "created" : "2018-11-14T19:10:23.824Z",
    "modified" : "2019-06-01T09:12:43.124Z"、 "securityDefinitions": {...}、 "security": ...,
    "base" :https://servient.example.com/,
    "properties" : {...},
    "actions": {...},
    “events”: {...},
    “Link”: {...},
    "forms": {...}
}
```

クラス Thing のインスタンス内の version、securityDefinitions、properties、actions、events に割当てられたすべての値は、JSON オブジェクトとしてシリアライズされ**なければならない**。

クラス Thing のインスタンス内の links と forms に割り当てられた値はすべて、第 6.3.8 項 links と第 6.3.9 項 forms での定義のように JSON オブジェクトを含む JSON 配列としてシリアライズされなければならない。

ClassThing のインスタンス内で security に割り当てられた値は、JSON ストリングとして、または、要素が JSON スト

リングの JSON 配列としてシリアライズされ**なければならない**。

2. 人間が読み取り可能なメタデータ

title および description という名前の JSON メンバーは、人間が読み取り可能なメタデータを表示する TD ドキュメント内で使用される。これらは、

TD ドキュメントを確認する開発者のためのコメントとして、または、ユーザインターフェースのための表示テキストとして使用することができる。

第 5.3.1.1 項 Thing で定義されているように、人間が読み取り可能なメタデータを表示するために使用される基本テキストの方向は、最初の強力な規則などのヒューリスティックスを使用して推定するか、言語情報から推論することができる。TD ドキュメントでは、デフォルト言語は、@context 内の@language に割り当てられた値によって定義され、これは、必要に応じてスクリプトサブタグと共に、ベーステキスト方向を決定するために使用することができる。しかし、人間が読み取り可能なテキストを解釈するとき、各人間が読み取り可能なストリング値は、独立して処理され**なければならない**。言い換えれば、TD プロセッサは、1 つのストリングから別のストリングへの方向を変化させること、または、TD 内の他の場所から別のストリングの方向を推論することはできない。

注

ウェブ上のストリング[string-meta]は、基本テキストの方向を決定する手段として、強力な最初の推論、また、言語ベースの推論の両方を示唆している。TD フォーマットが JSON−LD 1.1[json−ld11]に基づいており、これは、現在、明示的な方向メタデータを欠いているので、これらのアプローチは、現在、本公開時点で適切であると考えられている。しかしながら、JSON-LD 1.1 が[string-meta]が推奨するような明示的な基本方向メタデータのサポートを採用する場合、TD フォーマットは、その機能を利用するために更新されるべきである。

title および description を使用する TD スニペットを以下のとおりである。デフォルト言語は、@context 配列内の JSON オブジェクト内の @language メンバーの定義によって en に設定される。

例 6

```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        {"@language" : "en"}
    ],
    "title": "MyThing",
    "description":"human readable information.",
    ...
    "properties": {
        "on": {
            "title" : "On/Off",
            "type": "boolean",
            "forms": [...]
        },
        "status": {
            "title" : "Status",
            "type": "object",
            ...
            "forms": [...]
        }
```

```
        },

        ...

    }
```

titles および descriptions という名前の JSON メンバーは、一個の TD ドキュメント内において複数言語で人間が読み取り可能なメタデータを提供するために TD ドキュメント内で使用される。MultiLanguage マップのすべての名前-値ペアは、JSON オブジェクトのメンバーとしてシリアライズされ**なければならない**。ここで、名前は[BCP47]によって定義された周知の言語タグであり、値は、そのタグによって示された言語の人間が読み取り可能なストリングである。詳細は第 5.3.1.7 項 MultiLanguage 参照。TD ドキュメント内のすべての MultiLanguage オブジェクトには、同じ言語メンバーセットが含まれる**べきである**。

異なるレベルで titles および descriptions 使用する TD スニペットは以下のとおりである。

例 7

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "title": "MyThing",
    「タイトル」:{
        "en":"MyThing",
        "de": "MeinDing",
        "ja" : "私のモノ",
        "zh-Hans" : "我的东西",
        "zh-Hant" : "我的東西"
    },
    "descriptions": {
        「en」:
        「人間が読める情報」、"de": "Mensch enlesbare Information
```

m
a
t
i
o
n
e
n
.
"
、
"
j
a
"
:
"
人
間
が
読
む
こ
と
が
で
き
る
情
報
"
,

"zh-Hans" : "人们可阅读的信息",

"zh-Hant" : "人們可閱讀的資訊"

},

...

"pro
p
e
r
t
i
e
s
"
:
{
"
o
n
"
:
{

「タイトル」:{

「en」:「On/Off」、

「de」:「An/Aus」、

"ja": "オンオフ",

"zh-Hans": "开关",

https://www.w3.org/TR/wot-thing-description/

```
                "zh-Hant": "開關"},
            "type": "boolean",
            "forms": [...]
        },
        "status": {
            「タイトル」:{
                「en」:「Status」、
                「de」:「Zustand」、
                "ja": "状態",
                "zh-Hans": "状态",
                "zh-Hant": "狀態"},
            "type": "object",
            ...
            "forms": [...]
        }
    },
    ...
}
```

TD インスタンスは、title および description の使用を titles および descriptions と組み合わせることもできる。title と titles、あるいは、description と descriptions が同じ JSON オブジェクト内に存在する場合、title および description の値はデフォルトテキストとして見る**ことができる**。title と titles、あるいは、description と descriptions が TD ドキュメントに存在する場合、各 title および description メンバーは、それぞれ対応する title および description メンバーを持っている**べきである**。デフォルトテキストの言語は、デフォルト言語で示され、これは、通常、TD インスタンスの作成者によって設定される。

例 8

```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        {"@language" : "de"}
    ],
    "title": "MyThing",
    "titles": {
        "en":"MyThing",
        "de": "MeinDing",
        "ja" : "私のモノ",
        "zh-Hans" : "我的东西",
        "zh-Hant" : "我的東西"
    },
    "description": "Menschenlesbare Informationen."、"descriptions": {
        「en」
```

https://www.w3.org/TR/wot-thing-description/

:「人間が読める情報」、"de": "Menschenlesbare Information.", "ja": "人間が読むことができる情報", "zh-Hans" : "人们可阅读的信息", "zh-Hant" : "人們可閱讀的資訊"

```
    },
    ...
  "properties": {
    "on": {
      「タイトル」: 「An/Aus」、
      「タイトル」: {
        「en」:「On/Off」、
        「de」:「An/Aus」、
        "ja": "オンオフ",
        "zh-Hans": "开关",
        "zh-Hant": "開關"},
      "type": "boolean",
      "forms": [...]
    },
    "status": {
      「タイトル
```

https://www.w3.org/TR/wot-thing-description/

」 ...

: }

「

Z

u

s

t

a

n

d

」

、

「

タ

イ

ト

ル

」

:

{

「en」:「Status」、

「de」:「Zustand」、

"ja": "状態",

"zh-Hans": "状态",

"zh-Hant": "狀態"},

"type": "object",

...

"forms": [...]

}

},

もう一のデフォルト言語の設定用法は、HTTP の Accept−Language ヘッダーフィールドなどの言語ネゴシエーションメカニズムを介することである。デフォルト言語がネゴシエートされていた場合、ネゴシエーションの結果と返されたコンテンツのデフォルト言語を示すために、@language メンバーが存在しな**ければならない**。デフォルト言語のネゴシエーションが成功した場合、TD ドキュメントは、titles および descriptions メンバー内の MultiLanguage オブジェクトよりも優先して、適切で整合する title および description メンバー値を持っている**べきである**。しかしながら、モノは、そのような動的に生成された TD をサポートしない、また、(例えば、リソース制約のために)言語ネゴシエーションをサポートしないと選択して**もよいこと**に留意されたい。

3. version

名前が VersionInfo のシグニチャに含まれるボキャブラリ用語である場合、VersionInfo のインスタンスの名前-値ペアは、すべて、名前としてボキャブラリ用語を持つ JSON メンバーとしてシリアライズされ**なければならない**。

バージョン情報オブジェクトの TD スニペットは以下のとおりである。

例 9

```
{
    ...

    ...

}
```

Version メンバーは、TD コンテキスト拡張子に基づく追加のアプリケーションおよび/またはデバイス固有のバージョン情報のコンテナである。詳細については、第 7.1 項意味論的注釈を参照。

4. securityDefinitions と security

Thing インスタンスでは、securityDefinitions に割り当てられる値は、SecurityScheme のインスタンスのマップである。SecurityScheme インスタンスのマップのすべての名前-値ペアは、マップをシリアライズした結果の JSON オブジェクトのメンバーとしてシリアライズされ**なければならない**。ペアの名前は JSON ストリングとして、SecurityScheme のインスタンスであるペアの値は JSON オブジェクトとしてシリアライズされ**なければならない**。

SecurityScheme の中の一つのサブクラスのインスタンスの名前-値ペアすべては、その名前がそのサブクラスのシグニチャまたは SecurityScheme のシグニチャに含まれるボキャブラリ用語である場合、SecurityScheme サブクラスのインスタン

スを名前としてボキャブラリ用語でシリアライズした結果である JSON オブジェクトのメンバーとしてシリアライズされ**なければならない**。

以下の TD スニペットは、基本的なユーザ名/パスワード認証をヘッダーで指定する単純なセキュリティ構成である。in に与えられる値は、実際には、デフォルト値(header)であり、省略することもできる。名前付きセキュリティ構成は、securityDefinitions マップで指定されなければならない。その定義は、security メンバーにその JSON 名を組み込むことによってアクティブ化されなければならない。その JSON 名は 1 つの定義だけがアクティブ化された時にストリング型となることができる。

例 10

```
...
"sec
    u
    r
    i
    t
    y
    D
    e
    f
    i
    n
    i
    t
    i
    o
    n
    s
    "
    :
    {
    "
    b
    a
    s
    i
    c
    _
    s
    c
    "
    :
    {
        「スキーム」:「基本」、
        "in": "header"
    }
    },
    ...
```

これは、より複雑な例であり、モノのベアラトークン認証と組み合わされたプロキシのダイジェスト認証を示す TD スニペットである。digest スキームでは、in のデフォルト値(すなわち、header)は省略されるが、依然として適用される。ユーザ名/パスワードおよびトークンなどの対応するプライベートセキュリティ構成は、正常に対話するためにコンシューマ内で構成されなければならないことに留意されたい。複数のセキュリティ定義をアクティブ化すると、security メンバーは配列となる。

例 11

```
...
```

"securityDefinitions":

{

"proxy_sc":

{

「スキーム」:「ダイジェスト」、

"proxy": "https://portal.example.com/"

},

"bearer_sc":

{

"in":"header",

「スキーム」:「ベアラ」、

"format": "jwt",

alg:"ES256"、

"authorization": "https://servient.example.com:8443/"

}

},

...

TD でのセキュリティ構成は必須である。セキュリティ定義は少なくとも 1 つ、モノレベルで(すなわち、TD ルートオブジェクトで)security 配列を通してアクティブ化されな**ければならない**。この構成は、モノと対話するために必要なデフォルトのセキュリティメカニズムと見なすことができる。また、セキュリティ定義は、モノレベルでアクティブ化されたすべての定義を無効にする(すなわち、完全に置き換える)フォームオブジェクトに security メンバーを含めることによって、フォームレベルでのアクティブ化も**できる**。

nosec セキュリティスキームは、セキュリティが必要とされない場合に提供される。モノの最小セキュリティ構成は、以下の例に示すように、モノレベルでの nosec セキュリティスキームのアクティブ化である。

https://www.w3.org/TR/wot-thing-description/

例 12

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-Thing-1234",
    "title": "MyThing",
    「description」: 「人間が読める情報」、
    「対応」: 「https://servient.example.com/contact」、
    "securityDefinitions": {"nosec_sc": {"scheme": "nosec"}}、"security": "nosec_sc",
    "properties": {...},
    "actions": {...},
    「events」: {...}、
    「リンク」: [...]
}
```

より複雑な例として、我々が、すべての対話アフォーダンスが、認証が必要とされないものを除いて、基本認証を要求するモノを持っていると仮定してみよう。status プロパティおよび toggle アクションについては、basic 認証が要求され、モノレベルで定義される。しかしながら、overheating イベントに関しては、認証は必要なく、従って、セキュリティ構成は、フォームレベルでオーバーライドされる。

例 13

```
{
    ...
"sec
    u
    r
    i
    t
    y
    D
    e
    f
    i
    n
    i
    t
    i
    o
    n
    s
    "
    :
    {
    "
    b
    a
    s
    i
    c
    _
    s
    c
    "
    :
    {
    "
    s
    c
    h
    e
    m
    e
    "
    :
    "
    b
    a
    s
    i
    c
    "
    }
    ,
        "nosec_sc": {"scheme": "nosec"}
    },
    "security": ["basic_sc"],
    ...
"pro
    p
    e
    r
```

```
t
i
e
s
"
:

{
"
s
t
a
t
u
s
"
:

{

        ...

        "forms": [{

            "href":"https://
            mylamp.example.com/status"

        }]

    }

},

"actions": {

"
t
o
g
g
l
e
"
:

{

        ...

        "forms": [{

            "href":"https://
            mylamp.example.com/toggle"

        }]

    }

},

"events": {

    「過熱」:{

        ...

        "forms": [{

            「
            hr
            ef
            」
            :
            「
            ht
            tp
            s:/
            /
            m
            yl
```

```
                a
                m
                p.
                ex
                a
                m
                pl
                e.
                co
                m
                /
                oh
                」
                、
                "s
                ec
                ur
                it
                y"
                :
                ["
                no
                se
                c_
                sc
                "]

            }]

        }

    }

}
```

セキュリティ構成は、同じ対話アフォーダンス内の異なるフォームに対して指定することもできる。これは、例えば、HTTP 及び CoAP [RFC7252]のような異なるセキュリティメカニズムをサポートする複数のプロトコルをサポートするデバイスに対して要求される。また、代替の認証メカニズムが許可される場合にも有用である。ここで、プロパティアフォーダンスをアクティブ化する 3 つの方法を示す TD スニペットを紹介する。基本認証を使用した HTTPS、ダイジェスト認証を使用した HTTPS、ベアラトークン認証を使用したＨＴＴＰである。言い換えれば、複数のフォーム内で異なるセキュリティ構成を使用すれば、「OR」方式でセキュリティメカニズムを組み合わせられる。対照的に、複数のセキュリティ構成を同じ security メンバーに構成すると、それらを「AND」方式で組み合わせるということになるが、その場合、それらはすべて、対話アフォーダンスのアクティブ化を可能にするために満足される必要があるためである。モノレベルで (デフォルト)構成を 1 つアクティブ化することは、依然として、必須であることに留意されたい。

https://www.w3.org/TR/wot-thing-description/

例 14

```
{
    ...
    "securityDefinitions": {
        "basic_sc": {"scheme": "basic"},
        "digest_sc": {"scheme":
        "digest"、"qop":
        "auth"、"in":
        "header"}、"psk_sc":
        {"scheme": "psk"}
    },
    "security": ["basic_sc"],
    ...
"properties": {
    "status": {
        ...
        "forms": [{
            "href":"https://
            mylamp.example.com/status"
        }, {
            「href」:「https://mylamp.example.com/status」:
            「
```

```
                    」
                    、
                    "s
                    ec
                    ur
                    it
                    y"
                    :
                    ["
                    di
                    ge
                    st
                    _s
                    c"
                    ]

                }, {

                    "href":
                    "coaps://m
                    ylamp.exa
                    mple.com:
                    5684/status
                    ","security
                    ":
                    ["psk_sc"]

                }]

            }

        },

        ...

    }
```

もう一つより複雑な例として、OAuth2 はスコープを利用する。トークン内に現れる可能性があり、そのリソース(または W3C WoT の場合は対話アフォーダンス)へのアクセスを可能にするために、リソース内の対応する識別子と一致しなければならない識別子である。例えば、以下の例では、status プロパティは、スコープ limited を含むベアラトークンを使用するコンシューマが読み取ることができるが、configure アクションは、special スコープを含むトークンを用いることによって呼び出すことができるのみである。スコープは、ロールと同一ではないが、しばしばロールに関連付けられ、例えば、おそらく、管理ロール内のスコープのみが、「特別な」対話を実行することを許可される。トークンは複数のスコープを持つことができる。本例では、管理者には、おそらく、limited で sepcial なスコープ両方を持つトークンが発行され、一方、通常のユーザには limited スコープを持つトークンのみが発行される。

https://www.w3.org/TR/wot-thing-description/

例 15

```
{
    ...
"sec
    u
    r
    i
    t
    y
    D
    e
    f
    i
    n
    i
    t
    i
    o
    n
    s
    "
    :
    {
    "
    o
    a
    u
    t
    h
    2
    _
    s
    c
    "
    :
    {
        "scheme": "oauth2",
        ...
        "flow": "implicit",
        "authorization"
        :
        "https://exampl
        e.com/authoriz
        ation","scopes"
        :
        ["limited"、"sp
        ecial"]
    }
    },
    "security": ["oauth2_sc"],
    ...
"pro
    p
    e
    r
    t
    i
    e
    s
    "
    :
    {
    "
    s
    t
    a
```

```
t
u
s
"
:

{

        ...

        "forms": [{

            「
            hr
            ef
            」
            :
            「
            ht
            tp
            s:/
            /
            sc
            op
            es
            .e
            xa
            m
            pl
            e.
            co
            m
            /
            st
            at
            us
            」
                                    、
                                    "s
                                    co
                                    pe
                                    s"
                                    :
                                    ["l
                                    i
                                    m
                                    ite
                                    d"
                                    ]

        }]

    }

},

"action": {

    "configure": {

        ...

        "forms": [{

            「
            hr
            ef
            」
            :
            「
            ht
            tp
            s:/
            /
            sc
```

https://www.w3.org/TR/wot-thing-description/

```
                op
                es
                .e
                xa
                m
                pl
                e.
                co
                m
                /
                co
                nf
                ig
                ur
                e
                」
                、
                "s
                co
                pe
                s"
                :
                ["
                sp
                ec
                ial
                "]

            }]

        }

    },

    ...

}
```

5. properties

Thing インスタンス内の properties に割り当てられる値は、PropertyAffordance のインスタンスのマップである。PropertyAffordance インスタンスのマップの名前-値ペアは、すべて、マップをシリアライズした結果の JSON オブジェクトのメンバーとしてシリアライズされな**ければならない**。ペアの名前は JSON ストリングとして、PropertyAffordance インスタンスであるペアの値は JSON オブジェクトとしてシリアライズされなければならない。

PropertyAffordance インスタンスの名前-値ペアは、すべて、その名前が PropertyAffordance、InteractionAffordance、あるいは、DataSchema のシグニチャ(の 1 つ)に含まれるボキャブラリ用語である場合、名前としてボキャブラリ用語のついた PropertyAffordance インスタンスをシリアライズした結果得られる JSON オブジェクトのメンバーとしてシリアライズされなけ**ればならない**。DataSchema インスタンスのシリアライズの詳細については、第 6.3.10 項参照データスキーマ参照。

PropertyAffordance インスタンス内の forms に割り当てられた値は、第 6.3.9 項 forms で定義されているように 1 つ以上の JSON オブジェクトシリアライズを含む JSON 配列としてシリアライズされ**なければならない**。

2 つのプロパティアフォーダンスのスニペットを以下に示す。

例 16: プロパティのシリアライズ例

```
    ...
"properties":
    {
    "on":
    {
        "type": "boolean",
        "forms": [...]
    },
    "status": {
        "type":
        "object",
        "properties":
        {
            "明るさ": {"type":
            "number"、"minimum":0.0,
```

「最大」:100.0

},

"rgb":{"type":"array"、"items":{"type":"number"、"minimum":0,「最大」:255},"minItems": 3、"maxItems": 3}},

```
        },

        ...

    "required": ["brightness"、 "rgb"],
}
    "forms": [...]
```

6. actions

Thing インスタンスでは、actions に割り当てられる値は、ActionAffordance のインスタンスのマップである。ActionAffordance インスタンスのマップの名前-値ペアは、すべて、マップをシリアライズした結果の JSON オブジェクトのメンバーとしてシリアライズされ**なければならない**。ペアの名前は JSON ストリングとして、ActionAffordance インスタンスであるペアの値は JSON オブジェクトとしてシリアライズされ**なければならない**。

ActionAffordance インスタンスの名前-値ペアはすべて、その名前が ActionAffordance、あるいは、InteractionAffordance のシグニチャ(の 1 つ)に含まれるボキャブラリ用語である場合、名前としてボキャブラリ用語のついた ActionAffordance インスタンスをシリアライズした結果得られる JSON オブジェクトのメンバーとしてシリアライズされな**ければならない**。

ActionAffordance インスタンスで output と input に割り当てられる値は、JSON オブジェクトとしてシリアライズされ**なければならない**。これらは、クラス dataschema に依存し、そのシリアライズは、第 6.3.10 項データスキーマで定義される。

https://www.w3.org/TR/wot-thing-description/

ActionAffordance のインスタンスで forms に割り当てられる値は、第 6.3.9 項 forms で定義されているように 1 つ以上の JSON オブジェクトシリアライズ含む JSON 配列としてシリアライズされな**ければならない**。

アクションアフォーダンスの TD スニペットを以下に示す。

例 17: アクションのシリアライズ例

```
...
"actions": {
    "fade" : {
        「タイトル」:「フェードイン/フェードアウト」、
        "description": "スムーズなフェードインとフェードアウトアニメーション"、"input": {
            "type": "object"、
            "properties": {
                "from": {
                    "type": "integer"
```

e
g
e
r
"
、
"
m
i
n
i
m
u
m
"
:
0
,
「最大」:100
},
"to": {
"
t
y
p
e
"
:
"
i
n
t
e
g
e
r
"
、
"
m
i
n
i
m
u
m
"
:
0
,
「最大」:100
},
"duration": {"type": "number"}
},
"required": ["to","duration"],
},
}

https://www.w3.org/TR/wot-thing-description/

```
},                                                    "output": {"type": "string"},

...                                                   "forms": [...]
```

7. events

Thing インスタンスでは、events に割り当てられる値は、EventAffordance のインスタンスのマップである。EventAffordance インスタンスのマップの名前-値ペアは、すべて、マップをシリアライズした結果の JSON オブジェクトのメンバーとしてシリアライズされなければならない。ペアの名前は JSON ストリングとして、EventAffordance インスタンスであるペアの値は JSON オブジェクトとしてシリアライズされ**なければならない**。

EventAffordance インスタンスの名前-値ペアはすべて、その名前が EventAffordance、あるいは、InteractionAffordance のシグニチャ(の 1 つ)に含まれるボキャブラリ用語である場合、名前としてボキャブラリ用語のついた EventAffordance インスタンスをシリアライズした結果得られる JSON オブジェクトのメンバーとしてシリアライズされな**ければならない**。

EventAffordance インスタンスで subscription、data、および、cancellation に割り当てられる値は、JSON オブジェクトとしてシリアライズされ**なければならない**。これらはクラス DataSchema に依存し、そのシリアライズは第 6.3.10 項データスキーマで定義される。

EventAffordance のインスタンスで forms に割り当てられる値は、第 6.3.9 項 forms で定義されているように 1 つ以上の JSON オブジェクトシリアライズを含む JSON 配列としてシリアライズされ**なければならない**。

イベントオブジェクトの TD スニペットを以下に示す。

例 18: イベントのシリアライズ例

```
...
"events": {
"overheated": {
"data": {
"type": "string"
},
"forms": [...]
}
},
...
```

イベントアフォーダンスは、既存の(例えば WebSub[websub])または顧客向けイベントメカニズム(例えば Webhoks)を採用するために、柔軟に定義されている。このため、所望のメカニズムに従って、subscription および cancellation を定義することができる。詳細は [WoT-BindingTemplates]参照。例 A.3 Webhook イベント例 は、Webhook を説明するためにイベントがどのように subscription および

cancellation を使用できるかを例示している。

8. links

link インスタンスの名前-値ペアはすべて、その名前が link のシグニチャに含まれるボキャブラリ用語である場合、名前としてボキャブラリ用語のついた link インスタンスをシリアライズした結果得られる JSON オブジェクトのメンバーとしてシリアライズされな**ければならない**。

links 配列内のリンクオブジェクトの TD スニペットを以下に示す。

例 19: リンクのシリアライズ例

```
...
"links": [{
    "rel": "controlledBy",
    「href」:「https://servient.example.com/things/lampController」、"type": "application/td+json"
}]
...
```

9. forms

form インスタンスの名前-値ペアはすべて、その名前が Form のシグニチャに含まれるボキャブラリ用語である場合、名前としてボキャブラリ用語のついた Form インスタンスをシリアライズした結果得られる JSON オブジェクトのメンバーとしてシリアライズされなければならない

必要に応じて、フォームオブジェクトは、プレフィックスで識別されるプロトコル固有のボキャブラリ用語で補足され**てもよい**。第 8.3 項プロトコルバインディングも参照。

forms 配列内のフォームオブジェクトの TD スニペットを以下に示す。

例 20: Form のシリアライズ例

```
...
"forms": [{
    "op": "writeproperty",
    「hr
    ef
    」:「ht
    tp
    ://
    m
    yt
    e
    m
    p.
    ex
    a
    m
    pl
    e.
    co
    m
    :5
    68
    3/
    te
    m
    p
    」、"c
    on
    te
    nt
    T
    yp
    e":"a
    pp
    lic
    ati
    on
    /js
    on
    "、"h
    tv
    :
    m
    et
    ho
    d
    N
    a
    m
    e":"P
    O
    S
    T"
```

}]

...

href には、http://192.168.1.25/left?p=2 & d=1 の p や d などのダイナミック変数を含む URI を入れることもできる。その際、URI は、[RFC6570]http://192.168.1.25/left{?p,d}で定義されているようにテンプレートとして定義することができる。

そのような場合、URI テンプレート変数は、JSON 名として関連付けられた(一意の)変数名を持つ JSON オブジェクトベースの uriVariables メンバーに集められな**ければならない**。

Form インスタンス中の uriVariables に割り当てられるマップ中の各値のシリアライズは、クラス DataSchema に依存しな**ければならない**。そのシリアライズは、第 6.3.10 項データスキーマで定義される。

URI テンプレートと uriVariables を使用した TD スニペットを以下に示す。

例 21

```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        {"eg": "http://www.example.org/iot"}
    ],
    ...
    "actions": {
        "LeftDown": {
            ...
            "uriVariables": {
                "p" : {"type": "integer"、"minimum": 0、"maximum": 16、"@type": "eg:SomeK
                "d" : {"type": "integer"、"minimum": 0、"maximum": 1、"@type": "eg:Direct
                                                                    ...
                                }
        },
```

```
},

...

},

"forms": [{

   "href" :"http
   ://192.168.1.
   25/left{?
   p,d}","htv:m
   ethodName"
   : "GET"

}]
```

https://www.w3.org/TR/wot-thing-description/

contentType メンバーは、「; 文字」で区切られた属性-値ペアとしてメディアタイプパラメータを含むメディアタイプ[RFC2046]を割り当てるために使用される。例:

例 22

```
...

   「contentType」:
「text/plain; charset=utf-8」

...
```

いくつかの使用事例では、対話アフォーダンスのフォームメタデータは、要求を記述するだけでなく、期待される応答のためのメタデータも提供する。例えば、アクション takePhoto は、要求ペイロードの JSON(すなわち、”contentType”:”application/json”)を使用してカメラのパラメータ設定(アパーチャ優先順位、タイマなど)を送るための input スキーマを定義する。このアクションの出力は撮影された写真であり、これは、例えば、JPEG フォーマットで可能となる。そのような場合、response メンバーは、応答ペイロードの表現フォーマット(例えば、”contentType”:”image/jpeg”)を示すために使用される。ここでは、コンテンツタイプが表現フォーマットを完全に指定するので output スキーマは必要とされない。

Form インスタンスで response に割り当てられる値は、それが存在する場合、JSON オブジェクトでな**ければならない**。応答オブジェクトは、それが存在する場合、ExpectedResponse のクラス定義の中で定義されている contentType メンバーを含んでい**なければならない**。

上記のアクション takePhoto に基づいて、response メンバーを持つ form スニペットを以下に示す。

例 23

```
{
    ...
    "actions": {
        "takePhoto": {
            ...
            "forms": [{
                "op": "invokeaction",
                「href」: 「http://camera.example.com/api/i
```

/snapshot」、"contentType": "application/json"

```
                    "application/json"
                }
            }]
        }
    },
    ...
}
```

"application/json"

```
                "application/json"
            }
        }]
    }
},
...
}
```

"application/json"

```
            "application/json"
        }
    }]
}
},
...
}
```

"application/json"

```
        "application/json"
    }
}]
}
},
...
}
```

"application/json"

```
    "application/json"
}
}]
}
},
...
}
```

"application/json"

```
"application/json"
}
}]
}
},
...
}
```

"application/json"

```
application/json"
}
}]
}
},
...
}
```

"application/json"

```
pplication/json"
}
}]
}
},
...
}
```

"application/json"

```
plication/json"
}
}]
}
},
...
}
```

"application/json"

```
lication/json"
}
}]
}
},
...
}
```

"application/json"

```
ication/json"
}
}]
}
},
...
}
```

"application/json"

```
cation/json"
}
}]
}
},
...
}
```

"application/json"

```
ation/json"
}
}]
}
},
...
}
```

"application/json"

```
tion/json"
}
}]
}
},
...
}
```

"application/json"

```
ion/json"
}
}]
}
},
...
}
```

"application/json"

```
on/json"
}
}]
}
},
...
}
```

"application/json"

```
n/json"
}
}]
}
},
...
}
```

"application/json"

```
/json"
}
}]
}
},
...
}
```

"application/json"

```
json"
}
}]
}
},
...
}
```

"application/json"

```
son"
}
}]
}
},
...
}
```

"application/json"

```
on"
}
}]
}
},
...
}
```

"application/json"

```
n"
}
}]
}
},
...
}
```

"application/json"

```
"
}
}]
}
},
...
}
```

"application/json"

```
}
}]
}
},
...
}
```

"application/json"

```
}]
}
},
...
}
```

"application/json"

```
}
},
...
}
```

"application/json"

```
},
...
}
```

"application/json"

```
...
}
```

"application/json"

```
}
```

"application/json"

forms がトップレベルに存在する場合、それは、モノが提供するメタ対話を記述するために使用することができる。例えば、操作タイプ「readallproperties」および「writeallproperties」は、コンシューマがすべてのプロパティを一度に読み取りと書き込みをすることができるモノとのメタ対話のためのものである。以下の事例では、forms メンバーが TD ルートオブジェクトに含まれ、コンシューマが一つのプロトコルトランザクションでモノのすべてのプロパティ(すなわち、on, brightness, timer)を読み取るか、または、書き込むために、送信対象 https://mylamp.example.com/allproperties を使用することができる。

```
"forms": [{
    "op": "readallproperties",
    "href": "https://mylamp.example.com/allproperties",
    "contentType": "application/json"
},{
    "op": "writeallproperties",
    "href": "https://mylamp.example.com/allproperties",
    "contentType": "application/json"
}],
"properties": {
    "status": {
        "forms": [{
            "href": "https://mylamp.example.com/status",
            "contentType": "application/json"
        }],
        "type": "string",
        "enum": ["green", "yellow", "red"]
    }
}
```

例 24

```
{
    ...
"pro
    p
    e
    r
    t
    i
    e
    s
    "
    :
    {
    "
    o
    n
    "
    :
    {
        "type": "boolean",
        "forms": [...]
    },
    「
    明
    る
    さ
    」
    :
    {
    「
    タ
    イ
    プ
    」
    :
    「
    番
    号
    」
    、
        "forms": [...]
    },
    "timer": {
        "type": "integer",
        "forms": [...]
    }
},
...
"forms": [{
    "op":
    "readallproperties",
    「
    h
    r
    e
    f
    」
    :
    「
    h
    t
    t
    p
    s
    :
    /
    /
    m
    y
    l
    a
    m
    p
    .
    e
    x
    a
    m
    p
    l
    e
    .
    c
    o
    m
    /
    a
    l
    l
    p
    r
    o
    p
    e
    r
    t
    i
    e
    s
```

」

、

"

c

o

n

t

e

n

t

T

y

p

e

"

:

"

a

p

p

l

i

c

a

t

i

o

n

/

j

s

o

n

"

、

"

h

t

v

:

m

e

t

h

o

d

N

a

m

e

"

:

"

G

E

T

"

}, {

"op":
"writeallproperties",

「

h

r

e

f

」

:

「

h

t

t

p

s

:

/

/

m

y

l

a

m

p

.

e

x

a

m

p

l

e

.

c

o

m

/

a

l

l

p

r

o

p

e

r

t

https://www.w3.org/TR/wot-thing-description/

ies」、"contentType": "application/json"、"

htv:methodName": "PUT"

}]

}

オペレーションタイプ writeallproperties の場合、コンシューマは書き込み可能なプロパティ**すべて**と（新しく）割り当てられた値（例：ペイロード内）を提供することが求められる。さもなければ、モノは不整合を避けるためにこの呼び出しを拒否してよい。

10.データスキーマ

DataSchema クラスで定義された WoTTD のデータスキーマは、JSON スキーマ用語のサブセット[JSON-SCHEMA] のに基づいている。したがって、モノとやり取りされるデータを検証するために TD データスキーマのシリアライズを JSON スキーマバリデーターのインプリメンテーションに直接与えることができる。

データスキーマのシリアライズは、PropertyAffordance インスタンス、ActionAffordance インスタンスで input と output に

割り当てられる値、EventAffordance インスタンスで subscription, data, cancellation に割り当てられる値、および、(フォームオブジェクトが URI テンプレートを使用する場合)InteractionAffordance のサブクラスのインスタンスで uriVariables に割り当てられる値に適用される。

DataSchema の一つのサブクラスインスタンスの名前-値ペアはすべて、その名前がそのサブクラスのシグニチャ、あるいは、DataSchema のシグニチャに含まれるボキャブラリ用語である場合、名前としてボキャブラリ用語のついた DataSchema のサブク

ラスインスタンスをシリアライズした結果得られる JSON オブジェクトのメンバーとしてシリアライズされ**なければならない**。

ObjectSchema インスタンス内の properties に割り当てられる値は、JSON オブジェクトとしてシリアライズされなけ**ればならない**。

DataSchema インスタンスの enum、required、および、oneOf に割り当てられる値は、JSON 配列としてシリアライズされな**ければならない**。

ArraySchema のインスタンス内の items に割り当てられる値は、JSON オブジェクトまたは JSON オブジェクトを含む JSON 配列としてシリアライズさ**れなければならない**。

TD スニペットデータスキーマメンバーを以下に示す。周囲のオブジェクトは、データスキーマオブジェクト(例えば、input、output 用)又は付加的なメンバーを含むプロパティオブジェクトであってもよいことに留意されたい。

例 25: DataSchema のシリアライズ例

```
...
"type":
"object",
"properties":
{
    "status": {
        「タイトル」:「ステータス」、
        "type": "string",
        "enum":
        ["On"、"Off"、"Error"]
    },
    「明るさ」: {
        "title": "Brightness value"、"type": "number"、"minimum": 0.0,
        「最大」:100.0
    },
    "rgb": {
        "title":
        "RGB color value"、"type":
```

e
"
:

"
a
r
r
a
y
"
,

「アイテム」:{

"
t
y
p
e
"
:

"
n
u
m
b
e
r
"
、
"
m
i
n

i
m
u
m
"
:

0
,

「最大」:255

},

"minItems": 3、

"maxItems": 3

}

},

...

readOnly および writeOnly という用語は、読み取り対話(すなわち、プロパティを読み取る時)においてどのデータ項目をやり取りするか、および書き込み対話(すなわち、プロパティを書き込む時)においてどのデータ項目をやりとりするかを知らせるために使用することができる。これは、従来型でないモノのプロパティが、読み取りおよび書き込みのために異なるデータを示すとき

の回避策として使用することができ、これは、TD を用いて既存のデバイスまたはサービスを増補するときに発生する可能性がある。

readOnly および writeOnly を使用した TD スニペットを以下に示す。

例 26

```
...
"properties": {
    "status": {
        "description": "Read or Write
On/Off status.",
        "type": "object",
        "properties": {
            "latestStatus":
```

```
{
        "type": "string",
        "enum": ["On", "Off"]
    },
    "newStatusValue": {
        "type": "string",
        "enum": ["On", "Off"]
    },
    "readOnly": true
}
```


https://www.w3.org/TR/wot-thing-description/

f
"
]
、
"
w
r
i
t
e
O
n
l
y
"
:

t
r
u
e

}

},

形式: [...]

}

}

...

ティを更新するには、ペイロード内の newStatusValue メンバーを介して新しい値が提供されなければならない。

追加機能として、TD インスタンスでは、データスキーマ内の unit メンバーが使用できる。これによって、測定単位をデータアイテムに関連付けることができる。そのストリング値は、自由に選択することができる。しかしながら、周知のボキャブラリで定義されている単位を選択することが推奨される。例については、第 7 項 TD コンテキスト拡張子参照。

6.4 Identification

TD の JSON ベースのシリアライズは、メディアタイプ application/td+json、または、CoAP コンテンツフォーマット ID T.B.D.によって識別される(第 10 項 IANA 考慮事項を参照)。

注: CoAP コンテンツフォーマット

CoAP ベースの WoT インプリメンテーションでは、正式コンテンツフォーマット ID が割り当てられるまで、試験的なコンテンツフォーマット 65100 を使用することができる。

7. TD コンテキスト拡張子

status プロパティが読み込まれると、ペイロード内の latestStatus メンバーを使用してステータスデータが返される。status プロパ

本項は標準ではない。

第５項 TD 情報モデルの標準ボキャブラリ定義に加えて、WoTTD は、追加の名前空間からコンテキスト知識を追加する機能を提供している。本メカニズムは、TD インスタンスを追加の(例えば、ドメイン固有の)意味論で強化するために使用することができる。また、将来、追加のプロトコルバインディングや新しいセキュリティスキームをインポートするために使用することもできる。

そのような TD コンテキスト拡張子について、TD は、JSON−LD [json−ld11]で周知の @context メカニズムを使用する。TD コンテキスト拡張子を使用する場合、クラス Thing の @context 値は、JSON-LD コンテキストファイルを識別する anyURI 型の追加要素を持つ配列、あるいは、第 5.3.1.1 項で定義されている通り、名前空間 IRI を含むマップである。.

第 6.1 項 JSON 型へのマッピングにある複合型のシリアライズ規則は、拡張@context 名前-値ペアのシリアライズを定義している。TD コンテキスト拡張子を持つスニペットを以下に示す。

実施例 27

```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        {
            「iot」:「http://example.org/iot」、
            "cov":
                "http://www.example.org/coap-binding#"
        },
        "https://schema.org/"
    ],
    ...
}
```

1. 意味論的注釈

TD コンテキスト拡張子は、TD インスタンスへのボキャブラリ用語追加を可能にする。含まれる名前空間が、RDF スキーマまたは OWL によって提供される定義などのクラス定義に基づく場合、それらを使用して、インスタンスをそのような外部クラス定義に関連付けることによって、TD のクラスインスタンスに意味論的に注釈を付けることができる。これは、@type 名前-値ペアにクラス名を割り当てるか、または、複数の関連付け/注釈の配列値にクラス名を入れて行う。第 6.1 項 JSON 型へのマッピング内のシリアライズ規則に従って、@type は、JSON ストリングまたは JSON 配列として

シリアライズされる。@type は、ノード型を設定するために使用される JSON-LD キーワード [json-ld11] である。

TD コンテキスト拡張子は、TD の任意の Class インスタンス内に追加の名前−値ペアおよび明確に定義された値を含めることも可能にする。これらのペアおよび値は、含まれるボキャブラリ用語で定義され、それぞれ、対応する JSON オブジェクト内の追加メンバーまたは既存メンバーの値としてシリアライズされる。例として、モノの追加バージョンメタデータ、または、データアイテムの測定単位がある。

一例として、以下に示す TD スニペットは、モノのハードウェアおよびファームウェアのバージョン番号を追加することによってバージョン情報コンテナを拡張し、モノや例 2 と OM（測定単位オントロジー [RIJGERSBERG]）でも使用されているデータスキーマユニット: SAREF 用に外部のボキャブラリの値を使用する。これらのボキャブラリは、例として使用されており、家庭自動化領域では特にその他のボキャブラリが存在するかもしれない。

例 28

```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        {
            「v」: 「http://www.example.org/versioningTerms#」、
            "saref": "https://w3id.org/saref#",
            「om」:「http://www.wurvoc.org/vocabularies/om-1.8/」
        }
    ],
    "@type
```

https://www.w3.org/TR/wot-thing-description/

"

:

"

Thing"、"version":

{

「インスタンス」:「1.2.1」、

「v:firmware」:「0.9.1」、

"v:hardware": "1.0"

},

...

"properties":

{

"temperature":

{

"@type": "saref:Temperature"、"description": "Temperature value of the weather station"、"type": "number"、"minimum": -32.5,

「最大」:55.2、

"unit":"om:degree_Celsius"、"forms":[...]

},

```
            ...
        },
        ...
    }
```

多くの場合、TD コンテキスト拡張子は、対話中に（レスポンスのペイロード内で）データ交換により表示される物理世界のオブジェクトの状態情報の意味論的処理を可能にするために使用されるかもしれない。たとえば、RDF 内のこの状態情報の意味論的説明は、TD ドキュメントに埋め込むことができ、データスキーマの項目は、物理世界のオブジェクトの RFD モデル状態の特定部分の参照を促す注釈を個々につけることができる。

下記の TD スニペットは、ランプの上他ウィを説明するために SAREF を使用している。
SSN（Semantic Sensor Network Ontology 意味論的センサーネットワークオントロジー）[Vocab-SSN]から取り入れた外部ボキャブラリ用語 ssn:forProperty は、status プロパティのデータスキーマを物理世界オブジェクトの実際のオン／オフ状態にリンクするために使用されている。

https://www.w3.org/TR/wot-thing-description/

例 29

```
{

"@context": [

"https://www.w3.org/2019/
wot/td/v1",

{

"saref":
"https://w3id.org/saref#",

"ssn":
"http://www.w3.org/ns/ssn/"

}

],

"id": "urn:dev:ops:32473-
WoTLamp-1234",

"@type": "saref:LightSwitch",

"saref:hasState": {

"@id": "urn:dev:ops:32473-
WoTLamp-1234/state",

"@type": "saref:OnOffState"

},

...

"properties": {

"status": {

"ssn:forProperty":
"urn:dev:ops:32473-
WoTLamp-1234/state",

"type": "string",

"forms": [{"href":
"https://mylamp.example.com/
status"}]

},

"fullStatus": {

"ssn:forProperty":
"urn:dev:ops:32473-
WoTLamp-1234/state",

"type": "object",

"properties": {

"statusString": { "type":
"string" },

"statusCode": { "type":
"number" },

"statusDescription": { "type":
"string" }

},

"forms": [{"href":
"https://mylamp.example.com/
status?full=true"}]

},

...

},

...

  }
```

例２では、モノの状態は、それ自体の status アフォーダンスによって与えられ、起こりうる状態変化は、toggle アフォーダンスで与えられる。つまり、物理世界オブジェクトの状態は、直接、モノの対話アフォーダンスを提供するということである。この設計はシンプルなケースにおいては十分である。より複雑なケースでは、しかし、複数のアフォーダンスが同じ物理的状態に使用できることもある。上記の例では、fullStatus プロパティが、ランプの状態をより多くの言葉で表現する他の方法を提供している。

2. プロトコルバインディングの追加

本項は標準ではない。

TD の TD コンテキスト拡張子を使って通信メタデータを補足することができ、または、Form インスタンスを表す JSON オブジェクトにシリアライズされた追加のボキャブラリ用語で新しいプロトコルバインディングを追加することができる。(第 8.3 項プロトコルバインディングも参照)。

以下の TD の例では、仮想 CoAP プロトコルバインディングを使用する。というのは、このようなプロトコルバインディングが本仕様書執筆時点では存在しないためである。この TD コンテキスト拡張子は、名前空間例 http://www.example.org/coap-binding#を介してアクセス可能

な RDF1.0[HTTP-in-RDF10]内の HTTP ボキャブラリに類似した CoAP RDF ボキャブラリがあると仮定している。補足された cov:methodName メンバーは、どの CoAP メソッドが適用されなければならないかをコンシューマに指示する(例えば、CoAP メソッドコード 0.01 の場合は GET、CoAP メソッドコード 0.02 の場合は POST、または CoAP メソッドコード 0.07 の場合は iPATCH)。

例 30: TD コンテキスト拡張によるフォームの特殊化

```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        {"cov": "http://www.example.org/coap-binding#"}
    ],
    ...
"properties": {
    {
        "brightness": {
            "description": "The current brightness setting", "type": "integer", "minimum": -64,
            「最大」:64、
            "forms": [{
                "op": "readproperty",
                "href": "coap://example.org/brightness",
                "cov:methodName": "GET"
            }, {
                "op": "writeproperty",
```

```
"                                   o
h                                   v:
re                                  m
f"                                  et
:                                   h
"                                   o
c                                   d
o                                   N
a                                   a
p:                                  m
//                                  e
e                                   ":
x                                   "
a                                   P
m                                   O
pl                                  S
e.                                  T
o                                   "
r
g:                              }]
6
1                           },
6
1                           ...
6/                      },
a
pi                      ...
/b                  }
ri
g
ht
n
e
ss
",
"
c
```

第 5.3.3 項セキュリティボキャブラリ定義に含まれていない新しいセキュリティスキームは、TD コンテキスト拡張子メカニズムを使用してインポートすることができる。本例では、http://www.example.org/ace-security#で名前空間が本例のために定義する[ACE]に基づく仮想 ACE セキュリティ方式を使用している。このような追加のセキュリティスキームは、クラスセキュリティスキームのサブクラスでなければならないことに留意されたい。

3. セキュリティスキームの追加

https://www.w3.org/TR/wot-thing-description/

{ "https://www.w3.org/2019/wot/td/v1",

例 31 @context: [{

...

「cov」:「http://www.example.org/coap-binding#」、"ace": "http://www.example.org/ace-security#"

}],

"securityDefinitions": {"ace_sc": {

"scheme": "ace:ACESecurityScheme",

...

"ace:as": "coaps://as.example.com/token","ace:audience": "coaps://rs.example.com","ace:scopes": ["limited"、 "special"]、 "ace:cnonce": true

}

},

"security": ["ace_sc"]、 "properties": {

"status": {

...

"forms": [{

"op": "readproperty",

"href": "coaps://rs.example.com/status","contentType": "application/cbor"、 "cov:methodName": "GET",

"ace:scopes": ["limited"]

```
            }]

        }

    },

    "action": {

        "configure": {

            ...

            "forms": [{

                "op": "invokeaction",

                "href":
                "coaps://rs.example.com/configure","contentType":
                "application/cbor"、 "cov:methodName": "POST",

                "ace:scopes": ["special"]

            }]

        }

    },

    ...

}
```

第 5.3.3 項セキュリティボキャブラリ定義の中で定義されているセキュリティスキーム全てが既に TD コンテキストの一部であり、TD コンテキスト拡張子を使って含める必要はないということに留意された。

https://www.w3.org/TR/wot-thing-description/

8. ビヘイビアのアサーション

以下のアサーションは、TD の表現または情報モデルとは対照的に、WoT システムのコンポーネントのビヘイビアに関するものである。しかしながら、TD は記述的であり、特に、前から存在するネットワークインターフェースを記述するために使用されることもあるということに留意されたい。この場合、そのようなすでに存在するインターフェースのビヘイビアを制約するアサーションを行うことはできない。代わりに、アサーションは、そのようなインターフェースを正確に表すために、TD に対する制約となると解釈されなければならない。

1. セキュリティ構成

安全な相互運用を可能にするために、セキュリティ構成はモノの要件を正確に反映しなければならない。

- モノが対話のために特定のアクセスメカニズムを要求する場合、そのメカニズムは TD のセキュリティ構成の中で指定され**なければならない**。
- モノが対話のために特定のアクセスメカニズムを要求しない場合、そのメカニズムは TD のセキュリティ構成の中で指定さ**れてはならない**。

2. データスキーマ

TD で提供されるデータスキーマは、TD で指定された対話の中で記述されたモノが返し、また、受け入れるデータペイロードを正確に表すべきである。一般に、コンシューマは、WoTTD に与えられていないものを生成セず、厳密にデータスキーマに従うべきである。が、WoTTD に明示的に与えられていないモノからの追加データを受け入れるべきである。一般的に、モノは、WoTTD によって記述されるが、コンシューマは、モノと対話するときに WoTTD に従わざるを得ない。

- WoTTD に記述されている別のターゲットのモノと対話するときに、コンシューマとして動作するモノは、その対話で与えられたデータスキーマに従って編成されたデータを生成しな**ければならない**。
- WoTTD は、各対話によって返され、また、受け入れられたデータを正確に記述しな**ければならない**。
- モノは、その WoTTD で与えられたデータスキーマにそのようなデータが記述されていない場合でも、対話から追加データを返して**もよい**。これは、返されるデータ内に追加のプロパティあるいはアイテムがある可能性がある場合に、ObjectSchema と ArraySchema（items が DataSchema の列にある場合）に適用される。これは、「JASON-SCHEMA」内に定義されるとおりに、”additioalProperties”: true あるいは“additionalItems”: true があるかのように挙動する。
- 別のモノと対話するときにコンシューマとして動作するモノは、ターゲットのモノの TD で与えられたデータスキーマに記述されていない追加データを確実に受け取らな**ければならない**。これは、返されるデータ内に追加のプロパティあるい

はアイテムがある可能性がある場合に、ObjectSchema と ArraySchema（items が DataSchema の列にある場合）に適用される。これは、「JASON-SCHEMA」内に定義されるとおりに、”additioalProperties”: true あるいは“additionalItems”: true があるかのように挙動する。

- 別のモノと対話するときにコンシューマとして動作するモノは、そのモノの TD で与えられたデータスキーマに記述されていないデータを生成し**てはならない**。

- 別のモノと対話するときにコンシューマとして動作するモノは、URI テンプレート、ベース URI、および、ターゲットのモノの TD で与えられる href パラメータに従って URI を生成し**なければならない**。

- WoTTD 内の URI テンプレート、ベース URI、および href メンバーは、モノの WoT インターフェースを正確に記述し**なければならない**。

3. プロトコルバインディング

プロトコルバインディングは、対話アフォーダンスから、HTTP [RFC7231]、CoAP [RFC7252]、MQTT [MQTT]などの特定のプロトコルの具体的メッセージへのマッピングである。対話アフォーダンスのプロトコルバインディングは、第 6.3.9 項 forms で定義されているような形式でシリアライズされる。

WoTTD 内のすべてのフォームには、href メンバーが与える送信ターゲットが入っていなければならない。この送信ターゲットの URI スキームは、モノがどのプロトコルバインディングを実装しているか[WoTArchitecture]を示す。例えば、ターゲットが http または https で始まる場合、コンシューマは、モノが HTTP ベースのプロトコルバインディングを実装していることを推測することができ、フォームインスタンス内の HTTP 固有の用語を期待すべきである(第 8.3.1 項 HTTP ベースのプロトコルバインディングを参照)。

- WoTTD 内のすべてのフォームは、その href メンバーの URI スキームが示すプロトコルバインディングの要件に従わな**ければならない**。

- WoT TD 内のすべてのフォームは、対話内でモノが受け入れる要求(要求ヘッダーなどがあれば)を正確に記述しな**ければならない**。

1. HTTP ベースのプロトコルバインディング

デフォルトに従い、TD は、RDF1.0　[HTTP-in-RDF10]内の HTTP ボキャブラリの HTTPRDF ボキャブラリ定義を入れることによって、HTTP ベースのプロトコルバインディングをサポートする。このボキャブラリは、http://www.w3.org/2011/http#を指すプレフィックス htv を使って TD インスタンス内で直接的に使用することができる。さらに、HTTP ベースのプロトコルバインディングの詳細に関しては、[WOT-BINDING-TEMPLATE]を参照。

HTTP ベースのプロトコルバインディングを実装するモノと対話するために、コンシューマは、フォームを送信するときにどの HTTP メソッドを使用するかを知っている必要がある。一般的なケースでは、TD は、メソッドを示す用語、すなわち htv:methodName を明示的に含むことができる。簡潔にするために、HTTP ベースのプロトコルバインディングは、各操作タイプのデフォルト値を定義し、これは、また、モノが期待するメソッド(例えば、読み取りのための GET、書き込みのための PUT)の収束を目的とする。
HTTP ベースのプロトコルバインディングを表すフォームの中でメソッドが示されていない場合、デフォルト値は下表のように仮定されな**ければならない**。

ボキャブラリ用語	**デフォルト値**
htv:methodName	GET
htv:methodName	PUT
htv:methodName	ポスト

例えば、第１項はじめにの例では、この形式のオペレーションタイプと HTTP 方法は入っていない。以下のデフォルト値は、例１の形式のためであると考えるべきである。

例 32

HTTP ベースのプロトコルバインディングデフォルト値を使用

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "title": "MyLampThing",
    "securityDefinitions": {
```

https://www.w3.org/TR/wot-thing-description/

n
s
"
:

{

"basic_sc": {

「スキーム」:「基本」、

"in": "header"

}

},

「セキュリティ」: [

"basic_sc"

],

"pro
p
e
r
t
i
e
s
"
:

{
"
s
t
a
t
u
s
"
:

{

"
t
y
p
e
"
:

"
s
t
r
i
n
g
"
、
"
f
o
r
m
s
"
:

[

{

"op": "readproperty",

「href」:
「https://
mylamp.examp
le.com/
status」、"htv:

```
                    methodName":
                    "GET"

                },

                {

                    "op": "writeproperty",

                    「href」:
                    「https://
                    mylamp.examp
                    le.com/
                    status」、"htv:
                    methodName":
                    "PUT"

                }

            ]

        }

    },

    "actions": {

        "toggle": {

"
f
o
r
m
s
"
:

[

{
                    "op": "invokeaction",

                    「href」:
                    「https://
                    mylamp.examp
                    le.com/
                    toggle」、"htv
                    :methodName":
                    "POST"

                }

            ]

        }

    },

    "events": {

"ov
        e
        r
        h
        e
        a
        t
        i
        n
        g
        "
        :

        {
        "
        d
        a
        t
```

https://www.w3.org/TR/wot-thing-description/

```
a
"
:

{

    "type": "string"

},

"forms": [

{

「
href」:「https://myl
amp.example.com/oh」、"subprotocol":"long
poll"

}

]
```

}

https://www.w3.org/TR/wot-thing-description/

```
                }
    }
```

2. その他のプロトコルバインディング

モノが実装できるプロトコルバインディングの数は制限されていない。他のプロトコルバインディング(例えば、CoAP、MQTT、または、OPC UA のための)は、RDF 1.0[HTTP−in−RDF10]の HTTP ボキャブラリと同様のプロトコルボキャブラリ、または、デフォルト値定義を含む仕様など別個の文書で標準化される事になっている。このようなプロトコルは、コンテキスト拡張メカニズムの使用によって TD に単純に結合することができる(第 7 項 TD コンテキスト拡張子を参照)。

IoT プラットフォームとエコシステムの説明方法に関する情報に関しては、[WOT-BINDING-TEMPLATE]を参照。

9. セキュリティとプライバシーに関する考慮事項

本項は標準ではない。

一般に、WoT システムを保護するために取られるセキュリティ対策は、システムが直面する可能性がある脅威および攻撃者、ならびに、保護する必要がある資産の価値に依存する。さらに、プライバシーリスクは、識別可能な人とモノの関連性、および、直接的な情報とそのような関連性から入手できる推測情報に依存することになる。様々な状況に適応させることができる脅威モデルを含め、WoT に関するセキュリティおよびプライバシーの考慮事項の詳細な考察は、参考文献[WOT-SECURITY-CONSIDERATIONS]に記載されている。本項では、セキュリティとプライバシーリスクと、WoTTD に直接関連する実行可能な軽減対策についてのみ説明する。

WoTTD は、安全なネットワークインターフェースと安全でないネットワークインターフェースの両方を記述することができる。TD が既存のネットワークインターフェースに組み込まれる場合、そのネットワークインターフェースのセキュリティ状態に変化は期待できない。

WoTTD の使用は、以下の項で挙げられるセキュリティおよびプライバシーリスクを紹介している。各リスク説明の後、いくつかの実行可能な軽減対策を提案する。

1. プライバシーリスクをデリフェレンスするコンテキスト

JSON-LD [json-ld11] ドキュメントの@context メンバーで指定されたボキャブラリファイルのデリフェレンスはプライバシーリスクになりうる。WoT の場合、攻撃者は、そのようなデリフェレンスによって生成されたネットワークトラフィックを観察することができ、特にドメイン固有のボキャブラリが使用される場合、デバイスに関する情報を推論するために、宛先 IP アドレスなどのデリフェレンスのメタデータを使用することができる。これは、たとえ接続が暗号化さ

れていてもリスクとなり、DNS プライバシーリークにつながる。

軽減対策:

ボキャブラリファイルの実際のデリフェレンスは避ける。ボキャブラリファイルは、可能な限りキャッシュされるべきである。理想的には、(既知の)ボキャブラリの識別子としてのみ機能する@context メンバー内の URI で、それを変更不能にし、解釈デバイスに組み込み、全くデリフェレンスができないようにする。これには、既存の URI が変更不能データを参照できることを保証するために更新には新しい URI の使用が必須であるため、厳密なバージョン制御が必要となる。コンテキストファイルが TD 内のメタデータを解釈するシステムにローカルで利用可能になる見込みを高めることが可能な場合には必ず、周知の標準ボキャブラリファイルを使用する。

2. 変更不能識別子のプライバシーリスク

識別子（id）を含んでいる TD は、識別可能な人と関連付けられているモノを説明することができる。このような識別子は、トラッキングなどのさまざまなリスクを呈する。しかし、その識別子も変異することができない場合、デバイスが他の人に譲渡あるいは販売され、既知の ID がその人をトラッキングするために使用されるため、トラッキングリスクは増幅する。

軽減対策:

すべての識別子は可変でなければならず、モノの id を更新するメカニズムでなければならない。具体的には、モノの id は、ハードウェアに固定されるべきではない。しかしながら、これは、識別子が固定された URI であるという Linked　Data（リンクドデータ）の理想と矛盾する。多くの状況では、モノが再初期化される場合、識別子への更新が許容される。この場合、ソフトウェアエンティティとして、古いモノが存在しなくなり、新しいモノが作成される。これは、例えば、デバイスが新しい所有者に売られると、十分にトラッキングチェーンを遮断することができる。あるいは、デバイスの動作状態中により頻繁な変更が望まれる場合、変更が行われたときに識別子の変更を許可されたユーザのみに通知するメカニズムを導入することができる。しかしながら、いくつかのクラスのデバイス、例えば、医療デバイスは、いくつかの管轄区域において法律によって不変の ID を必要とすることがあるということに留意されたい。この場合、そのような不変の識別子を含む TD などのファイルへのアクセスを保証するために特別な注意が払われるべきである。また、できる限り、 そのような場合、“真に”変異不可能な識別子を TD 内で共有しないことが望ましい。

3. 指紋プライバシーリスク

上述したように、TD 内の id メンバーは、プライバシーリスクを引き起こしうる。しかしながら、その追跡リスクを軽減するために説明の通りに

https://www.w3.org/TR/wot-thing-description/

id が更新されたとしても、指紋を介して、TD を特殊な物理デバイスに関連付け、そこから、指紋を使って識別可能な人にたどり着くことが依然として可能でる。

特定のデバイスインスタンスが指紋で識別できない場合、一連の対話など TD 内の情報からデバイスタイプを推測し、医療状態など、識別可能な人に関する個人情報を推測するために使用することができる。

軽減対策:

許可されたユーザのみが、モノの TD へのアクセスを提供されるべきである。また、許可レベルに必要な情報量と使用例のみが提供されるべきである。TD が、たとえば、認証を要求するディレクトリサービスを通してなど安全で機密性のある経路で許可されたユーザのみに配信できるならば、外部の許可のない人たちは指紋のための TD へのアクセス権を持たない。このリスクをさらに軽減するために、TTD の特定の使用には不必要な情報は、可能な限り、省くべきである。たとえば、コンシューマがモノの状態を保存しないデバイスへの特別な接続に関しては、id は省くことができる。コンシューマがその使用のためにある対話が必要でない場合、それを省くことができる。コンシューマが、ある対話を使用する許可を得ていない場合、それも省くことができる。コインシュー間が、人間が読み取り可能な情報、たとえば、タイトルや説明を表示する能力を持たない場合、省くことができる。あるいは、ゼロ長ストリングに変えることができる。

4. グローバル一意識別子プライバシーリスク

グローバル一意識別子は、第三者が識別子を知っていることになるので、一元的権限者がこれらを生成／配布する必要がある場合プライバシーリスクを呈する。

軽減対策:

TD 内の id フィールドは、意識してグローバル一意にする必要はない。一元的登録を必要としない配布方法で適切な ID を生成するために利用できる暗号メカニズムが複数存在する。これらが、通常、同一識別子を生成する可能性は非常に低い。また、これがシステム設計で考慮する必要がある。たとえば、必要に応じ、同一の ID を検知し、ID の再生成を行う。ID の範囲も、また、グローバルである必要はない。あるコンテキストでモノを判別する識別子の使用が好ましい。例としては、家庭あるいは工場内。

5. TD 傍受と改竄セキュリティリスク

コンテキストファイルの傍受および改竄は、ボキャブラリの解釈を変更して攻撃を容易にするために使用され得る。

軽減対策:

理想的には、コンテキストファイルは、認証されたチャネルを介してのみ取得されるということであるが、デリフェレンスされた場合、傍受および変更に対して脆弱である HTTP URL を使って多くのコンテキストが表示されるということは注目に値する(かつ、残念なこ

とである)。しかし、コンテキストファイルが変更不能でキャッシュされ、可能な限りデリフェレンスが回避されれば、このリスクは低減できる。

6. 個人情報プライバシーリスクの推測

たくさんの場所で、ユーザのプライバシーを保護するために、個人情報、すなわち、特定の個人に関連付けることができる情報を処理するための法的要件がある。このような情報は、もちろん、IoT デバイスが直接生成することができる。しかしながら、IoT デバイスの存在及びメタデータ(TD に格納されたデータの種類)は、個人情報を持っているか、又は、推論するために使用することもできる。この情報は、特定の個人がある種のデバイスを所有しているという事実と同じくらい単純であり得る。そして、その個人に関する追加の推論につながり得る。

軽減対策:

個人デバイスに関連付けられた TD は、それが個人情報を含んでいるかのように扱う。本原則の適用例として、ユーザ同意の取得方法を考えてみてください。モノが生成する個人データの使用に対する同意は、モノが、データを消費するシステムと組み合わされるときに取得されることが多く、これは、TD がデバイスにアクセスするためにローカルディレクトリまたは TD を消費するシステムに登録されるときにも頻繁に発生する。この場合、モノから送信されるデータの使用同意は、モノの TD にアクセスするための同意と組み合わせることができる。第 2 の例として、TD が個人情報を含むと考える場合、個人情報は無期限に保持されるべきではなく、同意が与えられた目的以外に使用されるべきでない。

10.IANA の考慮事項

1. application/td+json メディアタイプ登録

型名:

アプリケーション

サブ型名:

td+json

必須パラメータ:

なし

オプションパラメータ:

なし

エンコーディングに関する考慮事項:

RFC 6839、第 3.1 項を参照。

セキュリティ考慮事項:

RFC 8259 を参照。

WoT TD は、モノのメタデータの純粋なデータ交換フォーマットのためのものであるため、シリアライズは、構文解析される JavaScript の eval() 関数などのコード実行メカニズムをすり抜けて**はならない**。(無効な)文書は、実行されると、システムのセキュリティを危うくする予想外の副次的影響をもたらす可能性があるコードを含むこともある。

WoTTD は JSON-LD 1.1 プロセッサで評価することができる。JSON-LD 1.1 プロセッサは、通常、自動的にリモートコンテキスト(TD コンテキスト拡張子、第 7 項 TD コンテキスト拡張子を参照)へのリンクに従い、各コンシューマからの明示的なリクエストを得ずに、ファイルが転送される。リモートコンテキストが第 3 者から提供されると、第 3 者が、プライバシーの懸念につながる使用パターンまたは同様の情報を収集することができるようになる。リソースが制約されたデバイス上のインプリメンテーションには、(JSON-LD 処理とは対照的に)未加工の JSON 処理実行が期待されるが、一般的には、インプリメンテーションは、サポートされているコンテキスト拡張子の精査済みバージョンを静的にキャッシュする**べきであり、**リモートコンテキストへのリンクに従う**べきではない**。サポートされているコンテキスト拡張子は、その代わり、安全なソフトウェア更新メカニズムで管理することができる。

HTTP などの安全でない接続でウェブからロードされるコンテキスト拡張子(第 7 項 TD コンテキスト拡張子を参照)には、セキュリティを危険にさらしうる方法で TD 情報モデルを変更するように攻撃者が変更するリスクがある。このため、コンシューマは、システムがリモートコンテキストを使用できるようにする前に、再度、リモートコンテキストを精査し、キャッシュするべきである。

JSON-LD 処理には、通常、長い IRI[RFC3987]を短い用語で置き換えるということを鑑み、WoTTD は、JSON-LD 1.1 プロセッサを使用して処理されるときにかなり拡張することがあり、最悪の場合には、結果として得られるデータは、受信者のリソース全てを消費することもある。コンシューマは、何らかの TD メタデータを相当の疑いを持って扱う**べきである**。

相互運用性の考慮事項:

RFC 8259 を参照。

適合コンテンツと非適合コンテンツの両方を処理するための規則は、本仕様で定義される。

公開されている仕様:

https://w3c.github.io/wot-thing-description

本メディアタイプを使用するアプリケーション:

W3C WoT 内の全参加エンティティ、すなわち、Web of Things (WoT)アーキテクチャで定義されているモノ、コンシューマ、および仲介者。

フラグメント識別子の考慮事項:

RFC 6839、第 3.1 項を参照。

追加情報

マジックナンバー**:**

適用外

ファイル拡張子:

.jsontd

Macintosh ファイルタイプコード:

テキスト

詳細情報に関する連絡先とメールアドレス:

Matthias Kovatsch <w3c@kovatsch.net>

意図されたアプリケーション:

共通

使用上の制約事項:

なし

著者:

WoTTD 仕様は、Web of Things Working Group の成果物である。

変更管理者:

W3C

2. CoAP コンテンツフォーマット登録

IANA は、Constrained RESTful Environments (CoRE)パラメータ登録 [RFC7252]内の CoAP コンテンツフォーマットサブレジストリのメディアタイプに対し簡潔な CoAP コンテンツフォーマット ID を割り当てている。WoTTD のコンテンツフォーマット ID は、256 から 9999 までの(t.b.d.)である(IETF レビューまたは IESG 承認 l)。

メディアタイプ:

application/td+json

https://www.w3.org/TR/wot-thing-description/

エンコーディング:

-

ID:

T.B.D.

参考:

[「Web of Things (WoT)Thing Description」、2019 年 5 月]

A. TD インスタンスの例

本項は標準ではない。

A.1 CoAP プロトコルバインディングを使用した MyLampThing の例

モノの特徴リスト:

- タイトル: MyLampThing
- コンテンツ拡張子: なし
- 提供されるアフォーダンス:プロパティ一

つ、アクション一つ、 イベント一つ

- セキュリティ: PSKSecurityScheme（PSK セキュリティスキーム）

- プロトコルバインディング: TLS の CoAP [RFC7252]

- コメント: 第 7.2 項 プロトコルバインディングの追加を参照

例 33: CoAP プロトコルバインディングを使用した MyLampThing

```
{
"@context": [
"https://www.w3.org/2019/wot/td/v1",
{
"cov": "http://www.example.org/coap-binding#"
}
],
"id": "urn:dev
```

v
:
o
p
s
:
3
2
4
7
3
-
W
o
T
L
a
m
p
-
1
2
3
4
"
、
"
t
i
t
l
e
"
:
"
M
y
L
a
m
p
T
h
i
n
g
"
,
"description" : "MyLampThing uses JSON serialization"、"securityDefinitions": {"psk_sc": {"scheme": "psk"}}、"security": ["psk_sc"],
"properties" :

{
"status":
{
「description」:「ランプの現在のステータスを表示します」、
「type」:「string」、
"forms": [{
"href": "coaps://mylamp.example.com/status",
"covmethodName": "GET"
}]
}
},
"actions": {
"toggle": {
"d

https://www.w3.org/TR/wot-thing-description/

```
s
"
:

[
{

    "h
    re
    f":
    "c
    oa
    ps
    ://
    m
    yl
    a
    m
    p.
    ex
    a
    m
    pl
    e.
    co
    m
    /t
    og
    gl
    e"
    ,"
    co
    v:
    m
    et
    ho
    d
    N
    a
    m
    e"

e
s
c
r
i
p
t
i
o
n
"

:

"
ラ
ン
プ
の
オ
ン
/
オ
フ
を
切
り
替
え
る
"
、
"
f
o
r
m
```

:
"P
O
S
T"

}]

}

},

"events": {

「過熱」:{

「description」:「ランプが臨界温度(過熱)」、「データ」:{「type」:「string」}、

"forms": [{

"h
re
f":
"c
oa
ps
://
m
yl
a
m
p.
ex
a
m
pl
e.
co
m
/o
h"
,"
co
v:
m
et
ho
d
N
a
m
e"
:
"
G
E
T"
,

"subprotocol" : "cov:observe"

}]

}

}

}

A.2 MQTT プロトコルバインディングを使用した MyLightSensor の例

モノの特徴リスト:

- タイトル: MyLampSensor
- コンテンツ拡張子: なし
- 提供されるアフォーダンス: イベント一つ
- セキュリティ: なし
- プロトコルバインディング: MQTT [MQTT]
- コメント: MQTT クライアントは、アドレス 192.168.1.187:1883 の背後で実行されている MQTT ブローカーによって、トピック/lightSensor に光センサデータ(数字はテキスト形式でシリアライズされている)を頻繁に発行する。

例 34: MQTT プロトコルバインディングを使用した MyLightSensor


```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "title": "MyLightSensor",
    "id": "urn:dev:ops:32473-WoTLightSensor-1234",
    "securityDefinitions": {"nosec_sc": {"scheme":
```

```
"nosec"}}、"security": ["nosec_sc"],

"events": {

"lightSensor"、"forms": [

{

"data": {

"type": "integer"

}、"href": "mqtt://192.168.1.187:1883/lightSensor"、"contentType" : "text/plain"

}

]

}

}
```


https://www.w3.org/TR/wot-thing-description/

A. 3 Webhook イベントの例

モノの特徴リスト:

- タイトル: WebhookThing
- コンテキスト拡張子: HTTP プロトコルバインディング 補足(TD コンテキストにすでに含まれている htv プレフィックス)を使用する。
- 提供されるアフォーダンス: 1 イベント
- セキュリティ: なし
- プロトコルバインド: HTTP
- コメント: WebhookThing は、Webhook メカニズムを使用して、最新の温度値を定期的にコンシューマにプッシュするイベントアフォーダンス temperature を提供し、モノは、コンシューマが提供するコールバック URI に POST 要求を送信する。これを説明するために、subscription メンバーは、subscribeevent フォームを使って送信されなければならない書き込み専用パラメータ callbackURL を定義する。読み取り専用パラメータ subscriptionID は、そのサブスクリプションが返信する。WebhookThing は、data によって定義されたペイロードを有するこのコールバック URI に定期的に**ポスト**する。サブスクライブを解除するには、コンシューマは、URI テンプレートを利用する unsubscribeevent フォームを送信しなければならない。uniVariables メンバーは、subscriptionID ストリングを入れるようにコンシューマに通知する。これは、適切な意味注釈を含めるために、TD コンテキスト拡張子を使ってさらに自動化することができる。あるいは、subscription と同様に cancellation メンバーを使ったサブスクライブ解除を想定し、これを、サブスクライブ解除するためのペイロードで**ポスト**要求を記述する unsubscribeevent フォームと組み合わせることができる。

例 35: サブスクリプションおよびキャンセルを伴う温度イベント


```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-Thing-1234",
    "title": "WebhookThing",
    "description": "Webhook-based Event with subscription and unsubscribe form.","securityDefinitions": {"nosec_sc": {"scheme": "nosec"}},
    "security": "nosec_sc",
    "events": {
        "temperature": {
            "securityDefinitions": {"nosec_sc": {"scheme": "nosec"}},
            "security": ["nosec_sc"],
            "forms": [{
                "href": "https://mylamp.example.com/temperature",
                "subscribe": "https://mylamp.example.com/temperature/subscribe",
                "unsubscribe": "https://mylamp.example.com/temperature/unsubscribe",
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

"description": "周期的な温度値の更新を提供します。""subscription": {

"type": "object",

"properties": {

"callback
URL": {

"type": "string",

"format": "uri",

"description": "Webhook no "writeOnly" のためにサブスクライバーによって提供されたコールバック URL: true

},

"subs

c
r
i
p
t
i
o
n
I
D
"
:

{
"
t
y
p
e
"
:

"
s
t
r
i
n
g
"
,

「description」:
「readOnly」で提供さ
れるキャンセル用の
一意のサブスクリプ
ション ID: true

}

}

},

"data": {

"type":
"number"、"description": "コー
ルバック URL に送信される
最新温度値。

},

"can
c
e
l
l
a
t
i
o
n
"
:

{
"

https://www.w3.org/TR/wot-thing-description/

type": "object",
"properties": {
"subscriptio
nID": {
"type": "integer",
「description」:「サブスクリプションを取り消すために必要なサブスクリプションID」、"writeOnly": true
}
}
},
"uriVariables": {

"subscriptionID": {"type":
"string"}

},

"forms": [

{

"op": "subscribeevent",

"href":
"http://192.168.0.124:8080/
events/temp/subscribe"、"c
ontentType":
"application/json",

"htv:methodName":
"POST"

},

{

"op": "unsubscribeevent",

"href":
"http://192.168.0.12
4:8080/events/temp/
{subscriptionID}"、
"htv:methodName":
"DELETE"

https://www.w3.org/TR/wot-thing-description/

```
                }
            ]
        }
    }
}
```

B. TD インスタンス確認のための JSON スキーマ

本項は標準ではない。

以下は、JSON ベースのフォーマットでシリアライズされた TD インスタンスを構文的に確認するための JSON スキーマ[JSON-SCHEMA]文書である。

注

本文書によって定義された TD は、JSON-LD [json-ld11]からわかる@context メカニズムを使用することによって外部ボキャブラリを追加することができる。また、この外部ボキャブラリ中の用語は、第 5 項 TD 情報モデルで定義されている用語に加えて使用することができる。このため、下記 JSON スキーマは意図的にその点に関して厳密には記載されていない。外部ボキャブラリが使用されていない場合により厳密な確認を実行するために、異なるスコープ/レベルで additionalProperties スキーマプロパティ true の値を false に置き換えることができる。

注

一部の JSON スキーマ確認ツールでは、iri ストリングフォーマットをサポートしていないということに注意する。

TD インスタンスを確認するための以下の JSON スキーマは、デフォルト値を持つ用語が存在することを要求しない。したがって、デフォルト値を持つ用語は任意選択である。(第 5.4 項 デフォルト値定義も参照)

```
{
"title": "WoT TD Schema - 16 October 2019",
"description": "JSON Schema for validating TD
  instances against the TD model. TD instances can
  be with or without terms that have default values",
"$schema ":
"http://json-schema.org/draft-07/schema#",
"definitions": {
"thing-context-w3c-uri": {
```

```
"type": "string",
"enum": [
"https://www.w3.org/2019/wot/td/v1"
]
},
"thing-context": {
"oneOf": [{
"type": "array",
"items": {
"anyOf": [{
"$ref": "#/definitions/anyUri"
},
{
"type": "object"
}
]
},
                "contains": {
"$ref": "#/definitions/thing-context-w3c-uri"
}
},
{
"$ref": "#/definitions/thing-context-w3c-uri"
}
]
},
"type_declaration": {
"oneOf": [{
"type": "string"
},
{
"type": "array",
"items": {
"type": "string"
}
}
]
},
"property_element": {
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
```

https://www.w3.org/TR/wot-thing-description/

```
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"title": {
"$ref": "#/definitions/title"
},
"titles": {
"$ref": "#/definitions/titles"
},
"uriVariables": {
"type": "object",
"additionalProperties": {
"$ref": "#/definitions/dataSchema"
}
},
"forms": {
"type": "array",
"minItems": 1,
"items": {
"$ref": "#/definitions/form_element_property"
}
},
"observable": {
"type": "boolean"
},
"writeOnly": {
"type": "boolean"
},
"readOnly": {
"type": "boolean"
},
"oneOf": {
"type": "array",
                                    "items": {
"$ref": "#/definitions/dataSchema"
}
},
"unit": {
"type": "string"
},
"enum": {
"type": "array",
"minItems": 1,
"uniqueItems": true
},
"format": {
"type": "string"
},
"const": {},
"type": {
"type": "string",
```

```
"enum": [
"boolean",
"integer",
"number",
"string",
"object",
"array",
"null"
]
},
"items": {
"oneOf": [{
"$ref": "#/definitions/dataSchema"
},
{
"type": "array",
"items": {
"$ref": "#/definitions/dataSchema"
}
}
]
},
"maxItems": {
"type": "integer",
"minimum": 0
},
"minItems": {
"type": "integer",
"minimum": 0
},
"minimum": {
"type": "number"
},
"maximum": {
"type": "number"
},
"properties": {
"additionalProperties": {
"$ref": "#/definitions/dataSchema"
}
},
"required": {
"type": "array",
"items": {
"type": "string"
}
```

https://www.w3.org/TR/wot-thing-description/

```
}
},
"required": [
"forms"
],
"additionalProperties": true
},
"action_element": {
"type": "object",
"properties": {
"description": {
"type": "string"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"title": {
"$ref": "#/definitions/title"
},
"titles": {
"$ref": "#/definitions/titles"
},
"uriVariables": {
"type": "object",
"additionalProperties": {
"$ref": "#/definitions/dataSchema"
}
},
"@type": {
"$ref": "#/definitions/type_declaration"
},
"forms": {
"type": "array",
"minItems": 1,
"items": {
"$ref": "#/definitions/form_element_action"
}
},
"input": {
"$ref": "#/definitions/dataSchema"
},
"output": {
"$ref": "#/definitions/dataSchema"
},
"safe": {
"type": "boolean"
},
"idempotent": {
"type": "boolean"
}
},
"required": [
```

```
"forms"
],
"additionalProperties": true
},
"event_element": {
"type": "object",
"properties": {
"description": {
"type": "string"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"title": {
"$ref": "#/definitions/title"
},
"titles": {
"$ref": "#/definitions/titles"
},
"uriVariables": {
"type": "object",
"additionalProperties": {
"$ref": "#/definitions/dataSchema"
}
},
"@type": {
"$ref": "#/definitions/type_declaration"
},
"forms": {
"type": "array",
"minItems": 1,
"items": {
"$ref": "#/definitions/form_element_event"
}
},
"subscription": {
"$ref": "#/definitions/dataSchema"
},
"data": {
"$ref": "#/definitions/dataSchema"
},
"cancellation": {
"$ref": "#/definitions/dataSchema"
},
"type": {
"not": {}
```

https://www.w3.org/TR/wot-thing-description/

```
},
"enum": {
"not": {}
},
"const": {
"not": {}
}
},
"required": [
"forms"
],
"additionalProperties": true
},
"form_element_property": {
"type": "object",
"properties": {
"href": {
"$ref": "#/definitions/anyUri"
},
"op": {
"oneOf": [{
"type": "string",
"enum": [
"readproperty",
"writeproperty",
"observeproperty",
"unobserveproperty"
]
},
{
"type": "array",
"items": {
"type": "string",
"enum": [
"readproperty",
"writeproperty",
"observeproperty",
"unobserveproperty"
]
}
}
]
},
"contentType": {
"type": "string"
},
"security": {
"type": "array",
"items": {
"type": "string"
}
},
```

```
"scopes": {
"type": "array",
"items": {
"type": "string"
}
},
"subProtocol": {
"type": "string",
"enum": [
"longpoll",
"websub",
"sse"
]
},
"response": {
"type": "object",
"properties": {
"contentType": {
"type": "string"
}
}
}
},
"required": [
"href"
],
"additionalProperties": true
},
"form_element_action": {
"type": "object",
"properties": {
"href": {
"$ref": "#/definitions/anyUri"
},
"op": {
"oneOf": [{
"type": "string",
"enum": [
"invokeaction"
]
},
{
"type": "array",
"items": {
"type": "string",
"enum": [
```

https://www.w3.org/TR/wot-thing-description/

```
"invokeaction"
]
}
}
]
},
"contentType": {
"type": "string"
},
"security": {
"type": "array",
"items": {
"type": "string"
}
},
"scopes": {
"type": "array",
"items": {
"type": "string"
}
},
"subProtocol": {
"type": "string",
"enum": [
"longpoll",
"websub",
"sse"
]
},
"response": {
"type": "object",
"properties": {
"contentType": {
"type": "string"
}
}
},
"required": [
"href"
],
"additionalProperties": true
},
"form_element_event": {
"type": "object",
"properties": {
"href": {
"$ref": "#/definitions/anyUri"
},
"op": {
"oneOf": [{
"type": "string",
```

```
"enum": [
"subscribeevent",
"unsubscribeevent"
                    ]
},
{
"type": "array",
"items": {
"type": "string",
"enum": [
"subscribeevent",
"unsubscribeevent"
]
}
}
]
},
"contentType": {
"type": "string"
},
"security": {
"type": "array",
"items": {
"type": "string"
}
},
"scopes": {
"type": "array",
"items": {
"type": "string"
}
},
"subProtocol": {
"type": "string",
"enum": [
"longpoll",
"websub",
"sse"
]
},
"response": {
"type": "object",
"properties": {
"contentType": {
"type": "string"
}
```

https://www.w3.org/TR/wot-thing-description/

```
}
}
},
"required": [
"href"
],
"additionalProperties": true
},
"form_element_root": {
"type": "object",
"properties": {
"href": {
"$ref": "#/definitions/anyUri"
},
"op": {
"oneOf": [{
"type": "string",
"enum": [
"readallproperties",
"writeallproperties",
"readmultipleproperties",
"writemultipleproperties"
]
},
{
"type": "array",
"items": {
"type": "string",
"enum": [
"readallproperties",
"writeallproperties",
"readmultipleproperties",
"writemultipleproperties"
]
}
}
]
},
"contentType": {
"type": "string"
},
"security": {
"type": "array",
"items": {
"type": "string"
}
},
"scopes": {
"type": "array",
"items": {
"type": "string"
}
```

```
},
"subProtocol": {
"type": "string",
"enum": [
"longpoll",
"websub",
"sse"
]
},
"response": {
"type": "object",
"properties": {
"contentType": {
"type": "string"
}
}
},
"required": [
"href"
],
"additionalProperties": true
},
"description": {
"type": "string"
},
"title": {
"type": "string"
},
"descriptions": {
"type": "object"
},
"titles": {
"type": "object"
},
"dataSchema": {
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
},
"title": {
"$ref": "#/definitions/title"
```

https://www.w3.org/TR/wot-thing-description/

```
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"titles": {
"$ref": "#/definitions/titles"
},
"writeOnly": {
"type": "boolean"
},
"readOnly": {
"type": "boolean"
},
"oneOf": {
"type": "array",
"items": {
"$ref": "#/definitions/dataSchema"
}
},
"unit": {
"type": "string"
},
"enum": {
"type": "array",
"minItems": 1,
"uniqueItems": true
},
"format": {
"type": "string"
},
"const": {},
"type": {
"type": "string",
"enum": [
"boolean",
"integer",
"number",
"string",
"object",
"array",
"null"
]
},
"items": {
"oneOf": [{
"$ref": "#/definitions/dataSchema"
},
{
"type": "array",
"items": {
"$ref": "#/definitions/dataSchema"
}
```

```
}
]
},
"maxItems": {
"type": "integer",
"minimum": 0
},
"minItems": {
"type": "integer",
"minimum": 0
},
"minimum": {
"type": "number"
},
"maximum": {
"type": "number"
},
"properties": {
"additionalProperties": {
"$ref": "#/definitions/dataSchema"
}
},
"required": {
"type": "array",
"items": {
"type": "string"
}
}
},
"link_element": {
"type": "object",
"properties": {
"anchor": {
"$ref": "#/definitions/anyUri"
},
"href": {
"$ref": "#/definitions/anyUri"
},
"rel": {
"type": "string"
},
"type": {
"type": "string"
}
},
```

https://www.w3.org/TR/wot-thing-description/

```
"required": [
"href"
],
"additionalProperties": true
},
"securityScheme": {
"oneOf": [{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
                    "$ref":
                    "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"nosec"
]
}
},
"required": [
"scheme"
]
},
{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"basic"
```

```
]
},
"in": {
"type": "string",
"enum": [
"header",
"query",
"body",
"cookie"
]
},
"name": {
"type": "string"
}
},
"required": [
"scheme"
]
},
{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"cert"
]
},
"identity": {
"type": "string"
}
},
"required": [
"scheme"
```

https://www.w3.org/TR/wot-thing-description/

```
]
},
{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"digest"
]
},
"qop": {
"type": "string",
"enum": [
"auth",
"auth-int"
]
},
"in": {
"type": "string",
"enum": [
"header",
"query",
"body",
"cookie"
]
},
"name": {
"type": "string"
}
},
"required": [
"scheme"
]
},
{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
```

```
},
"description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"bearer"
]
},
"authorization": {
"$ref": "#/definitions/anyUri"
},
"alg": {
"type": "string",
"enum": [
"MD5",
"ES256",
"ES512-256"
]
},
"format": {
"type": "string",
"enum": [
"jwt",
"jwe",
"jws"
]
},
"in": {
"type": "string",
"enum": [
"header",
"query",
"body",
"cookie"
]
},
"name": {
"type": "string"
```

https://www.w3.org/TR/wot-thing-description/

```
}
},
"required": [
"scheme"
]
},
{
        "type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"psk"
]
},
"identity": {
"type": "string"
}
},
"required": [
"scheme"
]
},
{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
```

```
"enum": [
"public"
]
},
"identity": {
"type": "string"
}
},
"required": [
"scheme"
]
},
{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
                "description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"oauth2"
]
},
"authorization": {
"$ref": "#/definitions/anyUri"
},
"token": {
"$ref": "#/definitions/anyUri"
},
"refresh": {
"$ref": "#/definitions/anyUri"
},
"scopes": {
"type": "array",
"items": {
"type": "string"
```

https://www.w3.org/TR/wot-thing-description/

```
}
},
"flow": {
"type": "string",
"enum": [
"implicit",
"password",
"client",
"code"
]
}
},
"required": [
"scheme",
"flow"
]
},
{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"apikey"
]
},
"in": {
"type": "string",
"enum": [
"header",
"query",
"body",
"cookie"
]
},
"name": {
"type": "string"
}
},
"required": [
```

```
"scheme"
]
},
{
"type": "object",
"properties": {
"@type": {
"$ref": "#/definitions/type_declaration"
},
"description": {
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"proxy": {
"$ref": "#/definitions/anyUri"
},
"scheme": {
"type": "string",
"enum": [
"pop"
]
},
"authorization": {
"$ref": "#/definitions/anyUri"
},
"format": {
"type": "string",
"enum": [
"jwt",
"jwe",
"jws"
]
},
"alg": {
"type": "string",
"enum": [
"MD5",
"ES256",
"ES512-256"
]
},
"in": {
"type": "string",
"enum": [
```

https://www.w3.org/TR/wot-thing-description/

```
                "header",
"query",
"body",
"cookie"
]
},
"name": {
"type": "string"
}
},
"required": [
"scheme"
]
}
]
},
"anyUri": {
"type": "string",
"format": "iri-reference"
}
},
"type": "object",
"properties": {
"id": {
"type": "string",
"format": "uri"
},
"title": {
"$ref": "#/definitions/title"
},
"titles": {
"$ref": "#/definitions/titles"
},
"properties": {
"type": "object",
"additionalProperties": {
"$ref": "#/definitions/property_element"
}
},
"actions": {
"type": "object",
"additionalProperties": {
"$ref": "#/definitions/action_element"
}
},
"events": {
"type": "object",
"additionalProperties": {
"$ref": "#/definitions/event_element"
}
},
"description": {
```

```
"$ref": "#/definitions/description"
},
"descriptions": {
"$ref": "#/definitions/descriptions"
},
"version": {
"type": "object",
"properties": {
"instance": {
"type": "string"
}
},
"required": [
"instance"
]
},
"links": {
"type": "array",
"items": {
"$ref": "#/definitions/link_element"
}
},
"forms": {
"type": "array",
"minItems": 1,
"items": {
"$ref": "#/definitions/form_element_root"
}
},
"base": {
"$ref": "#/definitions/anyUri"
},
"securityDefinitions": {
"type": "object",
"minProperties": 1,
"additionalProperties": {
"$ref": "#/definitions/securityScheme"
}
},
"support": {
"$ref": "#/definitions/anyUri"
},
"created": {
"type": "string"
},
"modified": {
```


https://www.w3.org/TR/wot-thing-description/

```
"type": "string"

},

"security": {

"type": "array",

"minItems": 1,

"items": {

"type": "string"

}

},

"@type": {

"$ref": "#/definitions/type_declaration"

},

"@context": {

"$ref": "#/definitions/thing-context"

}

},

"required": [

"title",

"security",

"securityDefinitions",

"@context"

],

"additionalProperties": true

}
```

C. TD テンプレート

本項は標準ではない。

TD テンプレートは、モノのクラスの記述であり、クラウドサーバによる何千ものデバイスの共通処理を可能にするために、モノのグループ全体で共有されるプロパティ、アクション、イベント、および共通メタデータを記述しており、モノそれぞれで利用するものではない。TD テンプレートは、第 5 項 TD 情報モデルからの同じ中核ボキャブラリおよび情報モデルを使用する。

TD テンプレートを使って次のことが可能になる。

- クラウドサービスによる複数のモノの管理
- まだ開発されていないデバイス/モノのシミュレーション
- 共通のモノモデルを共有する異なるメーカーのデバイス間で共通のアプリ
- 複数のモデルをモノと組み合わせる

TD テンプレートは、デバイスとのインターフェース及び可能な対話(プロパティ、アクション、

イベント)の論理的記述であるが、シリアル番号、GPS 位置、セキュリティ情報、具体的なプロトコルエンドポイントなどデバイス固有の情報を含まない。

TD テンプレートは、特定のエンドポイントへのプロトコルバインディングを含まず、特定のセキュリティメカニズムを定義していないので、フォームおよびセキュリティ定義とセキュリティキーは存在してはならない。

同じ TD テンプレートは、複数のベンダの複数のモノによって実装することができる。モノは複数の TD テンプレートを実装し、追加のメタデータ(ベンダ、位置、セキュリティ)を定義し、具体的なプロトコルへのバインディングを定義することができる。共通のモノに組み合わされている異なる TD テンプレートからのプロパティ、アクション、イベントの間の衝突を回避するために、これらの識別子はすべて、モノの中で唯一無二でなければならない。

あるクラスのデバイスの共通の TD テンプレートは、ベンダを超えてアプリを書くことを可能にし、アプリ開発者にとってより魅力的な市場を生み出す。具体的な TD は、複数の TD テンプレートを実装することができ、したがって、機能ブロックを結合されたデバイスに集約することができる。

クラウドベンダーのビジネスモデルは、通常、何千もの同一のデバイス管理することで構築される。同一 TD テンプレートのデバイスすべては、クラウドアプリにより同じ方法で管理することができる。インターフェースとインスタンスが別々に扱われる場合、複数のシミュレートされたデバイスを作成することは容易である。

しかし、TD テンプレートは、いくつかの任意のボキャブラリ用語と必須のボキャブラリ用語が存在しない TD のサブセットであるので、TD と同じ方法・同じフォーマットでシリアライズすることができる。TD テンプレートインスタンスは、いくつかの必須用語が欠落しているため、TD インスタンスと同じ方法で確認することができないことに留意されたい。

C. 1 TD テンプレートの例

本項では、ランプの TD テンプレートとブザーの TD テンプレートを紹介する。

C.1.1 TD テンプレート: ランプ

例 36: JSON でシリアライズされた MyLampTD テンプレート

```
{
    "@co
    ntext"
```

:
["http
s://ww
w.w3.
org/20
19/wo
t/td/v1
"]、"
@type
" :
"Thin
gTem
plate",

"
t
i
t
l
e
"
:

"
L
a
m
p

T
h
i
n
g

T
e
m
p
l
a

t
e
"

、
"
d
e
s
c
r
i
p
t
i
o
n
"

:

"
L
a
m
p

T
h
i
n
g

T
e
m
p
l
a
t
e
"

、
"
p
r
o
p
e
r
t
i
e
s
"
:

{

"status": {

「description」:「ランプの現在のステータス(on|off)」、「type」:「string」、

"readOnly": true

}

},

"actions": {

"toggle": {

「説明」:「ランプのオン/オフを切り替える」

}

},

"events": {

「過熱」:{

"description" :"ランプが臨界温度に達する(過熱)"、"data": {"type": "string"}

}

}

}

C.1.2 TD テンプレート: ブザー

例 37: JSON でシリアライズ MyBuzzerTD テンプレート

{

"@context": ["https://www.w3.org/2019/wot/td/v1"],

"@type" : "ThingTemplate",

"title": "Buzzer Thing Description Template",

"description" : "Thing Description Template of a buzzer that makes noise for 10 seconds",

"actions": {

"buzz": {

"description" : "buzz for 10 seconds"

}

}

}

C. JSON-LD　コンテキスト用法

本項は標準ではない。

現行の仕様書は、TD 情報モデルを異なるボキャブラリに関する一連の制約、すなわち、ボキャブラリ用語として取り入れている。本項では、これらの制約の機会読み取り可能定義が、TD ドキュメントの必須@context を使用して以下にクライアントアプリに結合されたのかを簡単に説明する。

TD ドキュメントから TD 情報モデルへのアクセスは、２段階で行われる。まず、クライアントは、IRI への JSON ストリングからのマッピングを検索する。気尾のマッピングは、後述のとおり JSON-LD コインテキストを定義されている。そこで、クライアントは、それらをデレファランスして IRI 上で定義されている制約にアクセスすることができる。制約はクライアントプログラムが簡単に解読できる RDF 形式で論理原理として定義されている。

第５項 TD 情報モデル内で参照されているボキャブラリ用語はすべて、TD ドキュメント内で（コンパクトな）JSON ストリングとしてシリアライズされている。しかしながら、これら用語のはそれぞれ、第一の LINKED Data 原理[LINKED-DATA]どおり、完全な IRI で明確に識別できる。JSON キーから IRI へのマッピングは、TD の @context 値が示すものである。たとえば、https：//www.w3.org/2019/wot/td/v1 のファイルは、以下のマッピングを含んでいる（さらに存在する中で）；

properties -> https://www.w3.org/2019/wot/td#hasPropertyAffordance

object -> https://www.w3.org/2019/wot/json-schema#ObjectSchema

basic -> https://www.w3.org/2019/wot/security#BasicSecurityScheme

href -> https://www.w3.org/2019/wot/hypermedia#hasTarget

この JSON ファイ ry は JSON-LD 1.1 シンテックス[JSON-LD11]に続くものである。多数の JSON-LD ライブラリが自動的に TD の@context を処理し、その中の JSON ストリングすべてを拡張することができる。

TD のボキャブラリ用語すべてが IRI に拡張されると、第二ステップは、そのボキャブラリ用語を参照する TD 情報も出るのフラグメントを取得するためにこの IRI をデレファランスする。たとえば、

https://www.w3.org/2019/wot/json-schema#ObjectSchema デレファランスすると、ObjectSchema という用語はクラスであり、もっと正確にすると DataSchema のサブクラスであるという RDF ドキュメントとなる。このような理論原理は、さまざまな複雑な形式を使った RDF で表現される。サブクラス関係は、RDF Schema 原理「RDF-SCHEMA」として表現される。さらに、これら原理は、さま座万形式でシリアライズされることがある。ここでは、Turtle　形式[TURTL]でシリアライズされている。

<https://www.w3.org/2019/wot/json-schema#ObjectSchema>

a rdfs:Class .

<https://www.w3.org/2019/wot/json-schema#ObjectSchema>

rdfs:subClassOf <https://www.w3.org/2019/wot/json-schema#DataSchema> .

https://www.w3.org/TR/wot-thing-description/

デフォルトで、ユーザエージェントが内容交渉をしない場合、人間が読み取り可能な HTML 資料が RDF ドキュメントの代わりに返される。内容を交渉するためには、クライアントは、自身のリクエスト内に HTTP ヘッダー Accept: text/turtle を入れなければならない。

E.. 最近の仕様変更

E.1 第１次推奨候補からの変更点

- 一般
- TD プロセッサとの明確化された定義と、すべてのインプリメンテーションが正確に想定するように、インプリメンテーションは TD プロデューサあるいは TD コンシューマであるという関連テキスト（インプリメンテーションに影響なし）
- モノ id の明確にされた一意性と可能な値。特に、ローカル IP アドレスを持つ URI あるいは一時的で変異する URI のようなプライバシー保護する値
- 更新されたレファランス
- 意味論注釈の見直しテキスト
- バグ修正のために更新された例２
- Name は、第５．３．１．１ Thing クラス定義への前回変更を反映するために title へ改名された。
- @type は、forms から改名された。
- バグ修正のために更新された例３
- op は、デフォルト値を無効にするために status プロパティの form に追加された。
- [“readproperty”, “writeproperty”]のデフォルト値は　両方が GET 法にしてしまったであろう。にこれは意図的ではなかった。
- 小さな編集上の見直しと修正
- 用語法
- 第３項用語法　葉基準ではない。
- TD 情報モデル
- 第５．３．１．１ Thing で、モノのボキャブラリ用語 id は、オプションとなった。また、id の一意性の表現は、必要がないため削除された。
- 図１のラベル位置調整
- アサーション・強調部分でない大文字 MAY を修正。（データスキーマの TD コンテキスト拡張子；インプリメンテーションに影響なし）
- 図２“JSON スキーマボキャブラリ”のタイトルは、“データスキーマボキャブラリ”に変更。
- OAuth2 認証キュリティ構成のボキャブラリ用語は、現在、デフォルト値はない。よって、第５．４項デフォルト値定義か

ら削除された。

- TD 表形式

- 第６．３．９項　forms で、オペレーションタイプ writeallproperties 取り扱い時のコンシューマとモノに対する期待事項が明確化された。

- TD コンテキスト拡張子

- 見直しされた例２８と例２９

　　モノを拡張クラスで意味論的に注釈するために saref:TemperatureSensor を使用する。

　　例中で整合性を取るためにプロパティを注釈するために SAREF を使用する。

　　より新しい ontology-of-units-of measure(om)を使用する。

- 例２との設計上の違いを明確にするために例２９を更新した。

- ビヘイビアアサーション

- 第８．２項 Data Schemas はデータスキーマに説明されていないにもかかわらず、モノが追加データを返し、コンシューマが追加データを受け取らなければならないときの詳細な状況を説明している。

- 第８．３．１項 HTTP ベースのプロトコルバインディングは、“HTTP プロトコルバインディング”から改名された。

- セキュリティとプライバシーに関する考慮事項

- 一意でなくオプションのモノのボキャブラリ用語 id と整合させるために第９項セキュリティとプライバシーに関する考慮事項を更新。

　　使用例に必要のないときに ID をフィルタリングするといった軽減対策を追加。

　　デバイスタイプを特徴で識別するといったリスクを追加。

　　グローバル一意識別子リスクを追加し、これが不要であり、配布された暗号メカニズムあるいはローカルな範囲内の id を使用することができるということを指摘している。

- 付録

- 付録 C　TD テンプレートは、“モノテンプレート”から改名。

- 付録 D JDON-LD コンテキスト用法は、RDF 内の TD 情報検索のための２段階ステップを説明している。

- WoT Architecture 仕様[WOT-ARCHITECTURE]書参照は、有益である。

E. 2 第 3 次公開作業ドラフトからの変更点

https://www.w3.org/TR/wot-thing-description/

第 3 次公開作業ドラフトからの変更点は、Candidate Recommendation 推奨候補内で説明されている。

E. 謝辞

編集者は、御寄稿、ガイダンスおよび専門知識を御提供頂いた Michael Koster, Michael Lagally, Kazuyuki Ashimura, Ege Korkan, Daniel Peintner, Toru Kawaguchi, María Poveda, Dave Raggett, Kunihiko Toumura, Takeshi Yamada, Ben Francis, Manu Sporny, Klaus Hartke, Addison Phillips, Jose M. Cantera, Tomoaki Mizushima, Soumya Kanti Datta and Benjamin Klotz に謝意を表します。

また、本文書改良を可能にしたサポート、技術的入力、および提案を頂いた W3C スタッフ及び W3C WoT インタレストグループ (WoT IG)およびワーキンググループ (WoT WG)の他のすべての現役および前参加者に対し謝意を表します。

E. 参考文献

G.1 標準参考文献

[BCP47]

言語を識別するためのタグ。A。フィリップス; M。Davis.IETF.2009 年 9 月。IETF Best Current Practice.URL: https://tools.ietf.org/html/bcp47

[eventsource]

Server-Sent Events. Ian Hickson. W3C. 3 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/eventsource/

[RFC2046]

Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types. N. Freed; N. Borenstein. IETF. November 1996. Draft Standard. URL: https://tools.ietf.org/html/rfc2046

[RFC2119]

要求レベルを示すために RFC で使用されるキーワード。S. ブラッドナー。IETF.1997 年 3 月。ベストカレントプラクティス.URL: https://tools.ietf.org/html/rfc2119

[RFC3339]

インターネット上の日付と時刻: タイムスタンプ。G. Klyne; C。Newman.IETF.2002 年 7 月。提案規格。URL: https://tools.ietf.org/html/rfc3339

[RFC3629]

UTF-8、ISO 10646 の変換形式。F. Yergeau.IETF.2003 年 11 月。インターネット標準。

URL: https://tools.ietf.org/html/rfc3629

[RFC3986]

URI(Uniform Resource Identifier):汎用構文。T. Berners-Lee; R。Fielding; L。Masinter.IETF.2005 年 1 月。インターネット標準。URL: https://tools.ietf.org/html/rfc3986

[RFC3987]

Internationalized Resource Identifiers (IRIs). M. Duerst; M. Suignard. IETF. January 2005. Proposed Standard. URL: https://tools.ietf.org/html/rfc3987

[RFC6570]

URI テンプレート。J. Gregorio; R。Fielding; M。Hadley; M。Nottingham; D。Orchard.IETF.2012 年 3 月提案規格。URL: https://tools.ietf.org/html/rfc6570

[RFC6749]

OAuth 2.0 許可フレームワーク。D. Hardt、Ed.。IETF.2012 年 10 月。提案規格。

URL: https://tools.ietf.org/html/rfc6749

[RFC6750]

The OAuth 2.0 Authorization Framework: Bearer Token Usage. M. Jones; D. Hardt. IETF. October 2012. Proposed Standard. URL: https://tools.ietf.org/html/rfc6750

[RFC7252]

The Constrained Application Protocol (CoAP).Z. Shelby; K。Hartke; C。Bormann.IETF.2014 年 6 月。

提案規格。URL: https://tools.ietf.org/html/rfc7252

[RFC7516]

JSON Web 暗号化(JWE)。M. Jones; J。Hildebrand.IETF.2015 年 5 月。提案規格。URL: https://tools.ietf.org/html/rfc7516

[RFC7519]

JSON Web トークン(JWT)。M. Jones; J。Bradley; N。Sakimura.IETF.2015 年 5 月。提案規格。

URL: https://tools.ietf.org/html/rfc7519

[RFC7616]

HTTP Digest Access Authentication. R. Shekh-Yusef, Ed.; D. Ahrens; S. Bremer. IETF. September 2015. Proposed Standard. URL: https://httpwg.org/specs/rfc7616.html

[RFC7617]

The 'Basic' HTTP Authentication Scheme. J. Reschke. IETF. September 2015. Proposed Standard. URL: https://httpwg.org/specs/rfc7617.html

[RFC7797]

JSON Web Signature (JWS) Unencoded Payload Option.M. Jones.IETF.2016 年 2 月。提案規格。URL: https://tools.ietf.org/html/rfc7797

[RFC8174]

RFC 2119 Key Words の大文字と小文字のあいまいさ (Ambiguity of Uppercase vs Lowercase)。B. Leiba.IETF.2017 年 5 月。ベストカレントプラクティス.URL: https://tools.ietf.org/html/rfc8174

[RFC8252]

ネイティブアプリ用の OAuth 2.0。W. Denniss; J。Bradley.IETF.2017 年 10 月。ベストカレントプラクティス.URL: https://tools.ietf.org/html/rfc8252

[RFC8259]

The JavaScript Object Notation (JSON) Data Interchange Format.T. Bray、Ed.。IETF.2017 年 12 月インターネット標準。URL: https://tools.ietf.org/html/rfc8259

[RFC8288]

Web リンク。M. Nottingham.IETF.2017 年 10 月。提案規格。URL: https://tools.ietf.org/html/rfc8288

[RFC8392]

CBOR ウェブトークン(CWT)。M. Jones; E。Wahlstroem; S。Erdtman; H。Tschofenig.IETF.2018 年 5 月。

提案規格。URL: https://tools.ietf.org/html/rfc8392

【websub】

WebSub. Julien Genestoux; Aaron Parecki. W3C. 23 January 2018. W3C Recommendation. URL: https://www.w3.org/TR/websub/

[X509V3]

https://www.w3.org/TR/wot-thing-description/

ITU-T Recommendation X.509 version 3 (1997).「情報技術-オープンシステム相互接続-ディレクトリ認証フレームワーク」ISO/IEC 9594-8:1997.。ITU。

[xmlschema11-2-20120405]

W3C XML スキーマ定義言語(XSD) 1.1 パート 2: データ型。David Peterson; Sandy Gao; Ashok Malhotra; Michael Sperberg-McQueen; Henry Thompson; Paul V。Biron et al。W3C。5 April 2012.W3C 勧告。URL: https://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/

G.2 参考文献

[ACE]

Authentication and Authorization for Constrained Environments (ACE) using the OAuth 2.0 Framework (ACE-OAuth). L. Seitz; G. Selander; E. Wahlstroem; S. Erdtman; H. Tschofenig. IETF. 27 March 2019. Internet-Draft. URL: https://tools.ietf.org/html/draft-ietf-ace-oauth-authz-24

[HTTP-in-RDF10]

HTTP Vocabulary in RDF 1.0.Johannes Koch; Carlos A。Velasco; Philip Ackermann.W3C。2017 年 2 月 2 日。W3C 注意。URL: https://www.w3.org/TR/HTTP-in-RDF10/

[IANA-MEDIA-TYPES]

メディアタイプ。IANA.URL: https://www.iana.org/assignments/media-types/

[IANA-URI-SCHEMES]

Uniform Resource Identifier (URI) Schemes. IANA. URL: https://www.iana.org/assignments/uri-schemes/uri-schemes.xhtml

[jSON-ld11]

JSON-LD 1.1.Gregg Kellogg; Pierre-Antoine Champin.W3C。2019 年 5 月 10 日。W3C Working Draft.

URL: https://www.w3.org/TR/json-ld11/

[jSON-SCHEMA]

JSON Schema Validation: A Vocabulary for Structural Validation of JSON. Austin Wright; Henry Andrews; Geraint Luff. IETF. 19 March 2018. Internet-Draft. URL: https://tools.ietf.org/html/draft-handrews-json-schema-validation-01

[LDML]

Unicode Technical Standard #35: Unicode Locale Data Markup Language (LDML).Davis とマークする。

CLDR コントリビュータ。URL: https://unicode.org/reports/tr35/

【LINKED-DATA】

リンクされたデータ設計の問題。Tim Berners-Lee.W3C.2006 年 7 月 27 日。W3C-内部ドキュメント。URL: https://www.w3.org/DesignIssues/LinkedData.html

[MQTT]

MQTT バージョン 5.0。Andrew Banks; Ed Briggs; Ken Borgendale; Rahul Gupta.OASIS。2018 年 10 月 31 日。候補 OASIS 規格 01。URL: http://docs.oasis-open.org/mqtt/mqtt/v5.0/cos01/mqttv5.0-cos01.html

【OPENAPI】

OpenAPI 仕様: バージョン 3.0.1。Darrel Miller; Jason Harmon; Jeremy Whitlock; Kris Hahn; Marsh Gardiner; Mike Ralphson; Rob Dolin; Ron Ratovsky; Tony Tam.OpenAPI Initiative、Linux Foundation.2017 年 12 月 7 日。URL: https://swagger.io/specification/

[RDF-SCHEMA]

RDF Schema 1.1. Dan Brickley; Ramanathan Guha. W3C. 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/rdf-schema/

[RFC3966]

The tel URI for Telephone Numbers. H. Schulzrinne. IETF. December 2004. Proposed Standard. URL: https://tools.ietf.org/html/rfc3966

[RFC6068]

The 'mailto' URI Scheme. M. Duerst; L. Masinter; J. Zawinski. IETF. October 2010. Proposed Standard. URL:

https://tools.ietf.org/html/rfc6068

[Rijgersberg]

Ontology of Units of Measure and Related Concepts. Hajo Rijgersberg; Mark van Assem; Jan Top. Semantic Web journal, IOS Press. 2013. URL: http://www.semantic-web-journal.net/content/ontology-units-measure-and-related-concepts

[SemVer]

セマンティックバージョン 2.0.0。Tom Preston-Werner.2017 年 12 月 26 日。URL: https://semver.org/

[smartM2M]

ETSI TS 103 264 V2.1.1 (2017-03): SmartM2M; Smart Appliances; Reference Ontology and oneM2M Mapping.ETSI。2017 年 3 月公開。URL: http://www.etsi.org/deliver/etsi_ts/103200_103299/103264/02.01.01_60/ts_103264v020101p.pdf

[string-meta]

Web 上のストリング: Language and Direction Metadata。Addison Phillips; Richard Ishida.W3C.2019 年 4 月 16 日。W3C Working Draft.URL: https://www.w3.org/TR/string-meta/

[TURTLE]

RDF 1.1 Turtle. Eric Prud'hommeaux; Gavin Carothers. W3C. 25 February 2014. W3C Recommendation. URL: https://www.w3.org/TR/turtle/

[vocab-ssn]

Semantic Sensor Network Ontology の略語。Armin Haller; Krzysztof Janowicz; Simon Cox; Danh Le Phuoc; Kerry Taylor; Maxime Lefrançois.W3C.2017 年 10 月 19 日。W3C 勧告。URL: https://www.w3.org/TR/vocab-ssn/

[WOT-ARCHITECTURE]

1

[WoT-Binding-Templates]

Web of Things (WoT) Protocol Binding Templates.Michael Koster.W3C。2018 年 4 月 5 日。W3C 注意。

URL: https://www.w3.org/TR/wot-binding-templates/

[WOT-SECURITY-CONSIDERATION]

Web of Things (WoT) Security and Privacy Considerations.; Michael McCool; Elena Reshetova.W3C.2019 年 3 月 URL: https://w3c.github.io/wot-security/