W3C Recommendation

【注意】 この文書は、W3CのWeb of Things (WoT) Thing Description W3C Recommendation 9 April 2020の和訳である。
この文書の正式版はW3Cのサイト上にある英語版であり、この文書には翻訳に起因する誤りがありえる。誤訳、誤植などのご指摘は、翻訳チームのGitHub Issueまでお願いしたい。

First Update: 2020年06月19日 | Last Update: 2021年5月13日

✧ ✧ ✧

Web of Things (WoT) Thing Description

W3C勧告 2020年4月9日

本バージョン:
https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/
最新公開バージョン:
https://www.w3.org/TR/wot-thing-description/
最新編集者草案:
https://w3c.github.io/wot-thing-description/
実装報告書:
https://w3c.github.io/wot-thing-description/testing/report.html
旧バージョン:
https://www.w3.org/TR/2020/PR-wot-thing-description-20200130/
編集者:
Sebastian Kaebisch (Siemens AG)
Takuki Kamiya (Fujitsu Laboratories of America)
Michael McCool (Intel)
Victor Charpenay (Siemens AG)
Matthias Kovatsch (Huawei)
参加可能:
GitHub w3c/wot-thing-description
File a bug
Commit history
Pull requests
貢献者:
In the GitHub repository
リポジトリ:
We are on GitHub
File a bug

公開以後に報告されたエラーや問題がないか**正誤表**を確認のこと。

翻訳版も参照のこと。

概要

この文書では、Web of Things (WoT) Thing Descriptionの形式モデルと共通表現について記述している。Thing Descriptionでは、Thingのメタデータとインターフェースを記述する。このとき、Thingとは、Web of Thingsに対して相互作用を提供し、Web of Thingsに加えられる、物理的または仮想的なエンティティーを抽象化したものである。Thing Descriptionは、様々なデバイスを統合し、様々なアプリケーションの相互運用を可能にする小さな語彙に基づく一連の相互作用を提供する。デフォルトでは、Thing DescriptionはJSON形式でエンコードされ、JSON-LDの処理も可能である。後者は、機械が理解できる方法でThingに関する知識を表現するための強力な基盤を提供する。Thing Descriptionのインスタンスは、Thing自身が提供できる。あるいは、Thingにリソース上の

制限がある場合 (例えば、メモリ空間が限られている場合) や、Web of Thingsと互換性のある旧式デバイスがThing Descriptionで改造されている場合には外部で提供できる。

この文書のステータス

この項は、この文書の公開時のステータスについて記述している。他の文書がこの文書に取って代わることがありえる。現行のW3Cの刊行物およびこの技術報告の最新の改訂版のリストは、https://www.w3.org/TR/のW3C技術報告インデックスにある。

> ステータスの更新 (2020年6月) : (参加可能セクション中の) 「Commit history」へのリンクは、GitHubブランチの名称変更によりリンク切れになっていたため、2020年6月23日にin-placeで (=/TRエリアにあるHTMLを直接) 修正された。

この文書は、Web of Thingsワーキンググループによって勧告として公開された。

この仕様の議論にはGitHubのIssueをお勧めする。または、メーリングリストにコメントを送信することもできる。コメントはpublic-wot-wg@w3.org (アーカイブ) に送信いただきたい。

ワーキンググループの実装報告書を参照いただきたい。

この文書は、W3C会員、ソフトウェア開発者、他のW3Cグループ、および他の利害関係者によりレビューされ、ディレクターによりW3C勧告として承認されたものである。安定した文書であり、参考資料として用いること、別の文書で引用することができる。勧告の作成におけるW3Cの役割は、その仕様への関心を引いて、広く普及させていくことにある。これにより、ウェブの機能および相互運用性の向上につながる。

この文書は、W3C特許方針の下で活動しているグループによって作成された。W3Cは、このグループの成果物に関連するあらゆる特許の開示の公開リストを維持し、このページには特許の開示に関する指示も含まれている。不可欠な請求権 (Essential Claim(s)) を含んでいると思われる特許に関して実際に知っている人は、W3C特許方針の6項に従って情報を開示しなければならない。

この文書は、2019年3月1日のW3Cプロセスドキュメントに準拠している。

目次

1. **はじめに**

2. **適合性**

3. **用語**

4. **名前空間**

5. **TD情報モデル**
5.1 概要
5.2 予備事項
5.3 クラスの定義
5.3.1 コア語彙の定義
5.3.1.1 `Thing`
5.3.1.2 `InteractionAffordance` (相互作用のアフォーダンス)
5.3.1.3 `PropertyAffordance` (Propertyのアフォーダンス)
5.3.1.4 `ActionAffordance` (Actionのアフォーダンス)
5.3.1.5 `EventAffordance` (Eventのアフォーダンス)
5.3.1.6 `VersionInfo` (バージョン情報)
5.3.1.7 `MultiLanguage` (多言語)
5.3.2 データスキーマ語彙の定義
5.3.2.1 `DataSchema`
5.3.2.2 `ArraySchema` (配列スキーマ)
5.3.2.3 `BooleanSchema` (ブールスキーマ)
5.3.2.4 `NumberSchema` (数値スキーマ)

5.3.2.5 IntegerSchema (整数スキーマ)
5.3.2.6 ObjectSchema (オブジェクトスキーマ)
5.3.2.7 StringSchema (文字列スキーマ)
5.3.2.8 NullSchema (ヌル (Null) スキーマ)
5.3.3 セキュリティ語彙の定義
5.3.3.1 SecurityScheme (セキュリティスキーム)
5.3.3.2 NoSecurityScheme (セキュリティスキームなし)
5.3.3.3 BasicSecurityScheme (基本セキュリティスキーム)
5.3.3.4 DigestSecurityScheme (ダイジェストセキュリティスキーム)
5.3.3.5 APIKeySecurityScheme (APIキーセキュリティスキーム)
5.3.3.6 BearerSecurityScheme (ベアラーセキュリティースキーム)
5.3.3.7 PSKSecurityScheme (PSKセキュリティスキーム)
5.3.3.8 OAuth2SecurityScheme (OAuth2セキュリティスキーム)
5.3.4 ハイパーメディア制御語彙の定義
5.3.4.1 Link (リンク)
5.3.4.2 Form (フォーム)
5.3.4.3 ExpectedResponse (予期される応答)
5.4 デフォルト値の定義

6. TD表現形式
6.1 JSONの型へのマッピング
6.2 デフォルト値の省略
6.3 情報モデルのシリアライゼーション
6.3.1 Thingのルートオブジェクト
6.3.2 人間が読めるメタデータ
6.3.3 version (バージョン)
6.3.4 securityDefinitions (セキュリティ定義) とsecurity (セキュリティ)
6.3.5 properties (プロパティー)
6.3.6 actions (アクション)
6.3.7 events (イベント)
6.3.8 links (リンク)
6.3.9 forms (フォーム)
6.3.10 データスキーマ
6.4 識別

7. TDコンテキスト拡張
7.1 セマンティックなアノテーション
7.2 プロトコルバインディングの追加
7.3 セキュリティスキームの追加

8. 動作の言明
8.1 セキュリティ構成情報
8.2 データスキーマ
8.3 プロトコルバインディング
8.3.1 HTTPに基づくプロトコルバインディング
8.3.2 その他のプロトコルバインディング

9. セキュリティとプライバシーに関する留意点
9.1 プライバシーのリスクをフェッチするコンテキスト
9.2 不変な識別子に関するプライバシーのリスク
9.3 指紋に関するプライバシーのリスク
9.4 グローバルに一意な識別子に関するプライバシーのリスク
9.5 TDの傍受と改ざんに関するセキュリティのリスク
9.6 コンテキストの傍受と改ざんに関するセキュリティのリスク
9.7 個人識別可能情報の推測に関するプライバシーのリスク

10. IANAに関する留意点
10.1 application/td+jsonメディアタイプの登録
10.2 CoAPコンテンツ形式の登録

A. **Thing Descriptionのインスタンスの例**
A.1 CoAPプロトコルバインディングを用いたMyLampThingの例
A.2 MQTTプロトコルバインディングを用いたMyIlluminanceSensorの例
A.3 Webhook Eventの例

B. **TDインスタンス検証用JSONスキーマ**

C. **Thing Descriptionテンプレート**
C.1 Thing Descriptionテンプレートの例
C.1.1 Thing Descriptionテンプレート: 照明
C.1.2 Thing Descriptionテンプレート: ブザー

D. **JSON-LDコンテキストの使用法**

E. **最近の仕様変更**
E.1 勧告案からの変更
E.2 第2勧告候補からの変更
E.3 最初の勧告候補からの変更

F. **謝辞**

G. **参考文献**
G.1 規範的な参考文献
G.2 参考情報の参考文献

1. はじめに §

この章は参考情報である。

WoT Thing Description (TD) は、W3CのWeb of Things (WoT) の中心的な構成要素であり、Thingのエントリポイントと考えることができる (ウェブサイトの*index.html*とよく似ている)。TDのインスタンスには四つの主要コンポーネントがある。それらは、Thing自身に関するテキスト形式のメタデータ、Thingの使用方法を示す一連の相互作用のアフォーダンス、Thingと交換されるデータの機械が理解できるスキーマ、そして最後に、ウェブ上の他のThingやドキュメント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ある。

W3C WoTの相互作用モデルでは、3種類の相互作用のアフォーダンスを定義している。Property (`PropertyAffordance`クラス) は、現在の値の取得や操作状態の設定などのパラメータの検知と制御に使用できる。Action (`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も使用できる。Event (`EventAffordance`クラス) は、通知、離散的なイベント、または値のストリームが受信者に非同期で送信される通信のプッシュモデルに用いられる。詳細に関しては、 [WOT-ARCHITECTURE] を参照していただきたい。

一般的に、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 [RFC2046] に基づくコンテンツタイプ (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のメカニズム (認証、認可、機密性など) で識別される様々なプロトコルバインディングに関するメタデータを提供する。TDのインスタンスのシリアライゼーションは、JSON [RFC8259] に基づいており、JSONキー名は、この仕様書で定義しているTD 語彙の用語を指す。さらに、TDのJSONシリアライゼーションは、JSON-LD 1.1 [JSON-LD11] の構文に従い、拡張機能と豊かなセマンティックの処理を可能にする。

例1は、Property、Action、Eventを用いて、*MyLampThing*というタイトルの照明であるThingの相互作用モデルを記述したTDのインスタンスである。

例1: Thing Descriptionのサンプル

```
{
    "@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、*status*というタイトルを持つ一つのPropertyのアフォーダンスが存在することが分かる。さらに、(`forms` 構造内で`href`メンバーによって示される) `https://mylamp.example.com/status` というURIでGETメソッドを用いて (セキュアな形式の) HTTPプロトコルを介してこのPropertyにアクセスでき、文字列ベースの状態の値を返すということを示す情報が提供されている。GETメソッドの使用については明示的に述べられていないが、それは、この文書で定義しているデフォルト時の解釈 (default assumption) の一つである。

同様に、 `https://mylamp.example.com/toggle` というリソースにおいて、POSTメソッドを用いてスイッチの状態を切り替えるActionのアフォーダンスが指定されており、この場合も、POSTはActionを呼び出すためのデフォルト時の解釈である。

Eventのアフォーダンスにより、Thingが非同期メッセージを送信するメカニズムが有効になる。 ここでは、`https://mylamp.example.com/oh` に対するロングポーリングサブプロトコル (long polling subprotocol) とともにHTTPを用いることにより、発生する可能性のある、照明の過熱という事象が発生したときの通知を受け取るよう登録できる。

この例では、アクセスにユーザ名とパスワードを求める`basic`なセキュリティスキームも指定している。セキュリティスキームは、まず`securityDefinitions`で名前が与えられ、次に`security`の部分でその名前を指定することでアクティブ化されることに注意していただきたい。この例では、HTTPプロトコルの使用と組み合わせて、HTTP基本認証の使用方法を示している。少なくとも一つのセキュリティスキームを最上位レベルで指定することは必須であり、それが、すべてのリソースに対するデフォルトのアクセス要件となる。しかし、セキュリティスキームはフォームごとに指定することもできる。フォームのレベルで指定した構成は、`Thing`レベルで指定した構成を上書きするため、きめ細かいアクセス制御の指定が可能である。`nosec`という特殊なセキュリティスキームを用いて、アクセス制御メカニズムが用いられていないことを示すこともできる。後ほど、追加の例を示す。

Thing Descriptionは、名前空間にコンテキストの定義を追加する可能性を提供する。形式的に定義された知識 (例えば、特定のアプリケーション領域の論理規則) が、与えられた名前空間の配下に存在する場合は、このメカニズムを用いてThing Descriptionインスタンスの内容に追加のセマンティクスを組み込むことができる。コンテキスト情報は、`forms`フィールドで宣言されている基礎となる通信プロトコルの一部の構成と動作を指定するのにも役立つ。例2では、`@context`に2番目の定義を導入して、SAREF (Smart Appliance Reference Ontology)

[SMARTM2M] への参照として接頭辞`saref`を宣言することにより、例1のTDサンプルを拡張している。このIoTオントロジーには、`@type`フィールドの値として設定できるセマンティックラベルとして解釈される用語が含まれており、Thingのセマンティクスとその相互作用のアフォーダンスを提供する。下記の例では、Thingは`saref:LightSwitch`、`status` Propertyは`saref:OnOffState`、`toggle` Actionは`saref:ToggleCommand`でラベル付けされている。

例2: セマンティックアノテーション用のTDコンテキスト拡張を用いたThing Description

```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        { "saref": "https://w3id.org/saref#" }
    ] ,
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "title": "MyLampThing",
    "@type": "saref:LightSwitch",
    "securityDefinitions": {"basic_sc": {
        "scheme": "basic",
        "in": "header"
    }},
    "security": ["basic_sc"] ,
    "properties": {
        "status": {
            "@type": "saref:OnOffState",
            "type": "string",
            "forms": [{
                "href": "https://mylamp.example.com/status"
            }]
        }
    },
    "actions": {
        "toggle": {
            "@type": "saref:ToggleCommand",
            "forms": [{
                "href": "https://mylamp.example.com/toggle"
            }]
        }
    },
    "events": {
        "overheating": {
            "data": {"type": "string"},
            "forms": [{
                "href": "https://mylamp.example.com/oh"
            }]
        }
    }
}
```

`@context`内の宣言メカニズムは、JSON-LDで指定される。TDインスタンスは、その仕様のバージョン1.1 [json-ld11] に準拠している。したがって、TDインスタンスはRDFドキュメントとして処理することもできる (セマンティックな処理の詳細に関しては、付録§ D. SON-LDコンテキストの使用法やhttps://www.w3.org/2019/wot/tdなどの名前空間IRI下のドキュメントを参照)。

2. 適合性 §

非規定的と記している項と同じく、この仕様のすべての作成ガイドライン、図、例、注は、参考情報である。この仕様のその他の部分はすべて規定的である。

この文書の「することができる／してもよい (*MAY*)」、「しなければならない (*MUST*)」、「してはならない (*MUST NOT*)」、「推奨される (*RECOMMENDED*)」、「すべきである／する必要がある (*SHOULD*)」、「すべきでない／する必要がない (*SHOULD NOT*)」というキーワードは、ここで示しているように、すべて大文字で表示されている場合にのみ、BCP 14 [RFC2119] [RFC8174] で記述されているように解釈されるべきである。

Thing Descriptionインスタンスは、Thing Descriptionのシリアライゼーションに関する§ 5. TD情報モデルと§ 6. TD表現形式の規範的なステートメントに従っていれば、この仕様に準拠する。

Thing Descriptionインスタンスを検証するためのJSONスキーマ [JSON-SCHEMA] は、付録§ B. TDインスタンス検証用JSONスキーマで提供している。

3. 用語 §

この章は参考情報である。

Thing、***Consumer***、***Thing Description*** (***TD***)、***相互作用モデル*** (Interaction Model)、***相互作用のアフォーダンス*** (Interaction Affordance)、***Property***、***Action***、***Event***、***プロトコルバインディング*** (Protocol Binding)、***Servient***、***WoT インターフェース*** (WoT Interface)、***WoT ランタイム*** (WoT Runtime) などの基本的なWoT用語は、WoT アーキテクチャ仕様 [WOT-ARCHITECTURE] の第3章で定義している。

さらに、この仕様では次の定義を導入している。

TD コンテキスト拡張 **(TD Context Extension)**
: 語彙用語の追加により、Thing Description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コアなメカニズムに対するセマンティックアノテーションと拡張の基礎である。

TD 情報モデル **(TD Information Model)**
: 制約が適用される定義済み語彙で構築されるクラス定義の集合。したがって、その語彙のセマンティクスを定義する。クラス定義は通常、シグネチャ (一組の語彙用語) と、そのシグネチャに関する関数群で表現される。TD情報モデルには、クラスに関するグローバル関数として定義されているデフォルト値も含まれている。

TD プロセッサ **(TD Processor)**
: 特定の形式でThing Descriptionの内部表現をシリアライズおよび/または、その形式から逆シリアライズすることができるシステム。TDプロセッサは、セマンティック的に矛盾するThing Description、つまり、`Thing` クラスのインスタンス関係の制約を満たすことができないThing Descriptionを検出しなければならない。その目的のために、TDプロセッサは、すべての可能なデフォルト値が割り当てられている正規形のThing Descriptionを計算できる。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プロセッサの実装は、TDの作成者のみ (TDドキュメントへのシリアライズが可能) またはTDの消費者のみ (TDドキュメントからの逆シリアルライズが可能) である。

TD シリアライゼーション **(TD Serialization) または*TD ドキュメント*** **(TD Document)**
: Servient間で保存および交換できるThing Descriptionのテキストまたはバイナリ表現。TDシリアライゼーションは特定の表現形式に従い、ネットワーク上で交換される際にメディアタイプによって識別される。Thing Descriptionのデフォルト表現形式は、この仕様で定義しているJSONベースである。

語彙 **(Vocabulary)**
: 名前空間IRIで識別される語彙用語のコレクション。

用語 **(Term) および *語彙用語*** **(Vocabulary Term)**
: 文字列。用語が語彙の一部である場合、つまり、名前空間IRIが接頭辞として付与されている場合は、語彙用語と呼ばれる。読みやすくするために、この文書で示している語彙用語は、完全なIRIとしてではなく、常にコンパクトな形式で記述している。

これらの定義については、§ 5.2 予備事項でさらに展開する。

4. 名前空間 §

この仕様の§ 5. TD情報モデルで定義しているTD情報モデルのバージョンは、次のIRIで識別される。

```
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] を取得するために逆参照でき、TDドキュメントのコンパクトな文字列をIRIベースの完全な語彙用語に展開できる。しかし、この処理は、JSONベースのTDドキュメントを、TDプロセッサ実装のオプション機能であるRDFに変換する場合にのみ必要である。

現在の仕様では、語彙用語は常にコンパクトな形式で示される。その展開形式には、それが属している語彙の名前空間IRI下でアクセスできる。この名前空間は、§ 5.3 クラスの定義の構造に従う。TD情報モデルで用いられる個々の語彙は、次のような独自の名前空間IRIを持っている。

語彙	*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/json-schema#`
セキュリティ	`https://www.w3.org/2019/wot/security#`
ハイパーメディア制御	`https://www.w3.org/2019/wot/hypermedia#`

上記の語彙は互いに独立している。これらは、他のW3C仕様で再利用したり、拡張したりすることができる。語彙の設計に互換性のない (breaking) 変更を加えるたびに、年ベースの新しい名前空間URIの割り当てが必要になる。TD情報モデルの全般的な一貫性を維持するために、互換性のある (non-breaking) 変更 (特に新しい用語の追加) も識別できるように、関連するJSON-LDコンテキストファイルは、すべてのバージョンが独自のURI (`v1`、`v1.1`、`v2`、...) を持つようにバージョン付けされていることに注意していただきたい。

ある名前空間IRI下の語彙は、互換性のある変更しか行えないため、その内容を安全にキャッシュしたりアプリケーションに埋め込んだりすることができる。名前空間IRI下で比較的静的な内容を公開する利点の一つは、制約のある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を最適化することである。これにより、公開されている語彙に対して、プライベートネットワークからアクセスするデバイスにより引き起こされるプライバシーの漏えいも回避される (§ 9.1 プライバシーのリスクを逆参照するコンテキストも参照)。

5. TD情報モデル §

この項では、TD情報モデルについて紹介する。TD情報モデルは、「§ 6. TD表現形式」で別個に説明している、Thing Descriptionの処理とそのシリアライゼーションの概念的な基盤として機能する。

5.1 概要 §

TD情報モデルは、次の独立した語彙に基づいて構築される。

- *コア*TD 語彙。Property、Action、Eventの相互作用のアフォーダンスの相互作用モデルを反映する [WOT-ARCHITECTURE] 。
- *データスキーマ*語彙。JSONスキーマ [JSON-SCHEMA] で定義されている用語 (のサブセット) を含む。
- *WoTセキュリティ*語彙。セキュリティメカニズムとその構成要件を識別する。
- *ハイパーメディア制御*語彙。ウェブリンクとフォームを用いたRESTfulな通信の主要な原則をエンコードする。

これらの語彙はそれぞれ基本的に、従来のオブジェクト指向の意味のオブジェクトとして解釈されるデータ構造の構築に使用できる用語の集合である。オブジェクトはクラスのインスタンスであり、プロパティーを持つ。W3C WoTのコンテキストでは、Thingとその相互作用のアフォーダンスを示す。オブジェクトの正式な定義を § 5.2 予備事項で示している。TD情報モデルの主要な要素は、§ 5.3 クラスの定義で示している。デフォルト値が存在している場合は、オブジェクトプロパティーをTDで省略できる。デフォルトのリストは、§ 5.4 デフォルト値の定義で示す。

次に示すUML図は、TD情報モデルの概要を示している。すべてのクラスを表として表しており、`Thing`というクラスから始まる、クラス間に存在する関連付けを有向の矢印として表している。読みやすくするために、図は、四つの基本語彙ごとに一つずつの、四つの部分に分割している。

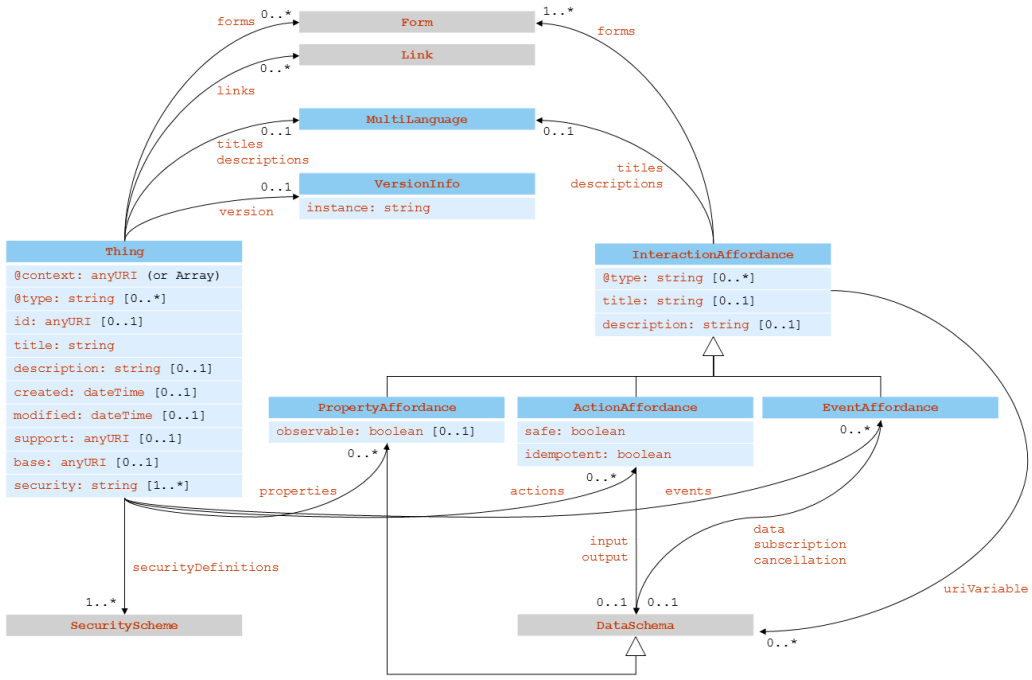

図1 TDコア語彙

✧ ✧ ✧

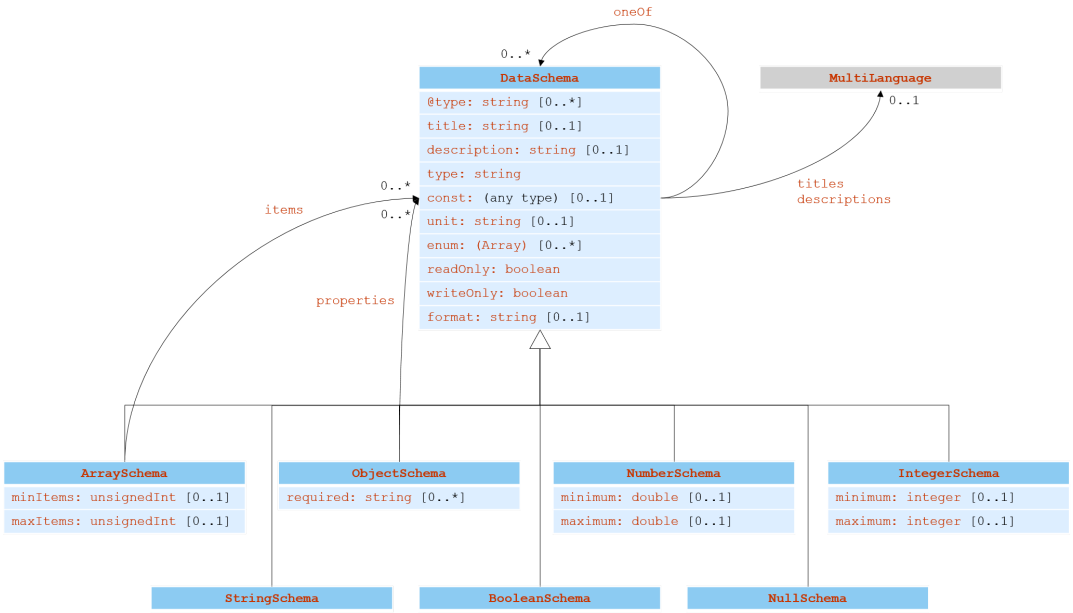

図2 データスキーマ語彙

✧ ✧ ✧

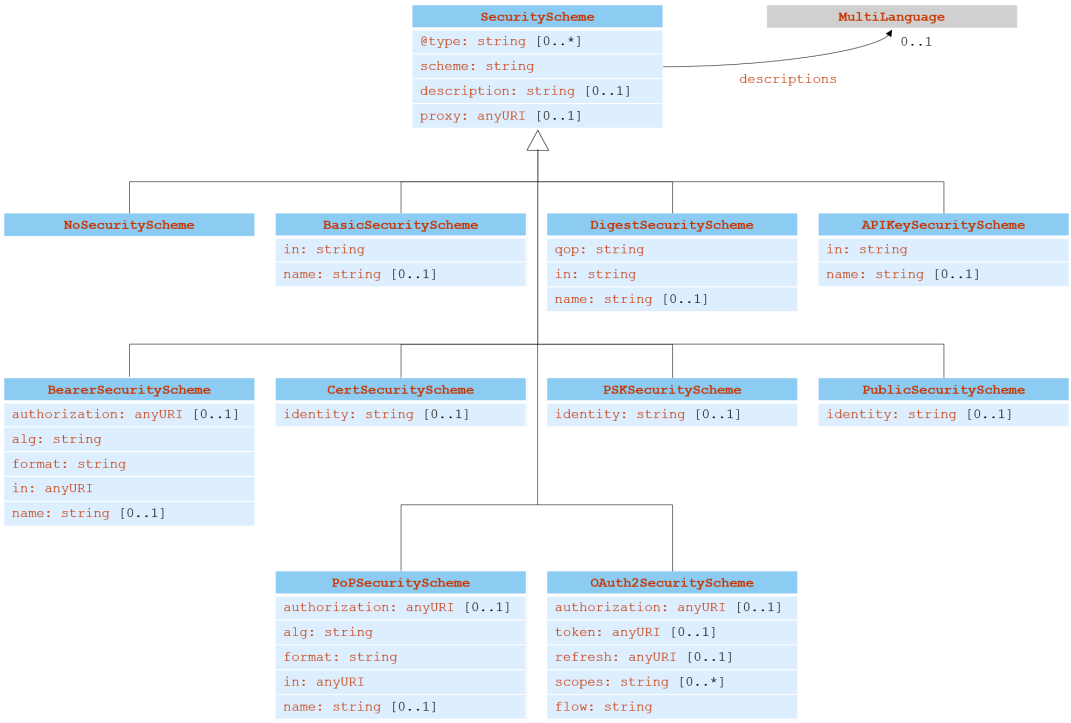

図3 WoTセキュリティ語彙

✧ ✧ ✧

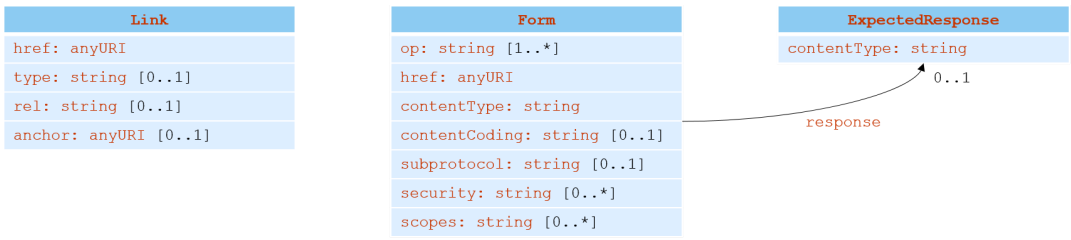

図4 ハイパーメディア制御語彙

5.2 予備事項 §

ツリーベースのドキュメントに関するシンプルな規則 (つまり、未加工のJSONの処理) と、豊かなセマンティックウェブツール (つまり、JSON-LD処理) の両方で容易に処理できるモデルを提供するために、この文書では、次の形式的な予備事項を定義し、それに応じてTD情報モデルを構築する。

この項のすべての定義は*集合*を意味しており、それは直感的に、それ自身が集合になり得る要素のコレクションである。任意の複雑なデータ構造はすべて集合として定義できる。特に、***オブジェクト***は、次のとおりに再帰的に定義されたデータ構造である。

- 用語は、語彙に属していてもいなくても、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-値のペアも、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の複数の名前-値のペアを含むことを妨げないが、これは一般的にこの仕様では検討しない。要素の名前が数字のみであるオブジェクトを***配列***と呼ぶ。同様に、要素が用語 (いかなる語彙にも属さない) のみを名前として持つオブジェクトを***マップ***と呼ぶ。マップ内の名前-値のペアに現れるすべての名前は、そのマップの範囲内で一意であると見なされる。

さらに、オブジェクトは***クラス***のインスタンスになりえる。語彙用語で示されるクラスは、***シグネチャ***と呼ばれる一組の語彙用語で最初に定義される。シグネチャが空のクラスは、***単純型***と呼ばれる。

クラスのシグネチャにより、クラスを詳細に定義した二つの関数 (***割当関数***と***型関数***) を構築できる。クラスの割当関数は、クラスのシグネチャの語彙用語を入力として受け取り、`true`または`false`を出力として返す。直感的には、割当関数は、クラスをインスタンス化するときにシグネチャの要素が必須かオプションかを示す。クラスの型関数は、クラスのシグネチャの語彙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、その定義域は、定義されているクラスのシグネチャに制限される。

これらの二つの関数に基づいて、オブジェクトとクラスで構成されるペアに対して***インスタンス関係***を定義できる。この関係は、満たすべき制約として定義されている。つまり、次の二つの制約が*両方*とも満たされる場合、オブジェクトはクラスのインスタンスになる。

- クラスの割当関数が`true`を返すすべての用語に対し、語彙用語を名前とする名前-値のペアがオブジェクトに含まれている。
- オブジェクトの名前-値のペアで名前として用いられるクラスのシグネチャのすべての語彙用語に対し、そのペアの値は、特定の語彙用語のクラスに対する型関数によって返されるクラスのインスタンスである。

上記の定義によれば、オブジェクトは、その構造に関係なく、すべての単純型のインスタンスになる。代わりに、インスタンス関係の別の定義が単純型に導入されている。オブジェクトは、特定の字句形式を持つ用語である場合、単純型のインスタンスである (例えば、`boolean`型の場合は`true`、`false`で、`unsignedInt`型の場合は`1`、`2`、`3`...など)。

さらに、***パラメータ化されたクラス***と呼ばれる追加のクラスを、汎用的なマップと配列の構造から派生させることができる。オブジェクトは、含んでいるすべての名前-値の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、何らかのクラスのマップ、つまり、何らかのクラスで*パラメータ化された*マップ型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、前者のすべてのインスタンスが後者のインスタンスでもある場合、クラスは他のクラスの***サブクラス***になる。

上記のすべての定義を前提とすると、TD情報モデルは、クラス名 (語彙用語)、シグネチャ (語彙用語の集合)、割当関数、および型関数を含む、クラス定義の集合と理解される。これらのクラスの定義は、§ 5.3 クラスの定義の表として提供している。各表では、割り当ての列の「必須」の値 (または「オプション」) は、割当関数が対応する語彙用語に対して`true` (または`false`) を返すことを示している。

慣例により、単純型は小文字で始まる名前で示される。TD情報モデルは、XMLスキーマ [XMLSCHEMA11-2-20120405] で定義されている、`string`、`anyURI`、`dateTime`、 `integer`、`unsignedInt`、`double`、`boolean`という単純型を参照する。それらの定義 (つまり、字句形式の仕様) は、TD情報モデルの範囲外である。

さらに、TD情報モデルは、語彙用語のペアにおけるグローバル関数を定義する。この関数は、クラス名と別の語彙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`ではない場合、それは入力クラスのインスタンスの入力語彙用語の割り当てに対する***デフォルト値***を表す。この関数により、上記の割当関数で定義している制約を緩和できる。すべての必須の割り当てが含まれている場合、*または*欠落している割り当てにデフォルト値が存在する場合、オブジェクトはクラスのインスタンスである。すべてのデフォルト値を§ 5.4 デフォルト値の定義の表で示している。§ 5.3 クラスの定義の各表には、TD情報モデルのクラスと語彙用語の対応する組み合わせにデフォルト値が使用可能な場合に、割り当て列に「デフォルトあり」という値が含まれている。

ここで紹介している形式化では、抽象的なデータ構造としてのオブジェクトと、Thingなどの物理世界のオブジェクトとの、ありえ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すべての語彙用語を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 (オントロジー) に統合する可能性には注意を払った。セマンティックな処理の詳細に関しては、§ D. JSON-LDコンテキストの使用法や、https://www.w3.org/2019/wot/tdなどの名前空間IRI下にあるドキュメントを参照していただきたい。

5.3 クラスの定義 §

TDプロセッサは、§ 5.3.1 コア語彙の定義、§ 5.3.2 データスキーマ語彙の定義、§ 5.3.3 セキュリティ語彙の定義、および§ 5.3.4 ハイパーメディア制御語彙の定義で定義しているすべてのクラスのクラスインスタンス化の制約を満たさなければならない (*MUST*)。

5.3.1 コア語彙の定義 §

5.3.1.1 *Thing* §

WoT Thing Descriptionでメタデータとインターフェースが記述されている、ある物理エンティティーまたは仮想エンティティーの抽象化。一方、仮想エンティティーは、一つ以上のThingの合成物である。

語彙用語	*説明*	*割り当て*	*型*
`@context`	TDドキュメント全体で用いる、用語と呼ばれる省略名を定義するためのJSON-LDキーワード。	必須	`anyURI`または配列
`@type`	セマンティックタグ (または型) でオブジェクトをラベル付けするためのJSON-LDキーワード。	オプション	`string`または`string`の配列
`id`	URI [RFC3986] 形式のThingの識別子 (例えば、安定したURI、一時的かつ可変なURI、ローカルなIPアドレスを持つURI、URNなど)。	オプション	`anyURI`
`title`	デフォルトの言語に基づいて、人間が読めるタイトルを提供する (例えば、UI表現用のテキストを表示)。	必須	`string`
`titles`	多言語の人間が読めるタイトルを提供する (例えば、様々な言語でUI表現のテキストを表示)。	オプション	`MultiLanguage`
`description`	デフォルト言語に基づいて、追加の (人間が読める) 情報を提供する。	オプション	`string`
`descriptions`	様々な言語の (人間が読める) 情報をサポートするために使用できる。	オプション	`MultiLanguage`
`version`	バージョン情報を提供する。	オプション	`VersionInfo`
`created`	TDのインスタンスがいつ作成されたかの情報を提供する。	オプション	`dateTime`
`modified`	TDのインスタンスがいつ最終的に変更されたかの情報を提供する。	オプション	`dateTime`
`support`	TDの保守担当者に関する情報を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`

語彙用語	*説明*	*割り当て*	*型*
`base`	TDドキュメント全体のすべての相対URI参照に用いる基底URIを定義する。TDのインスタンス内では、[RFC3986] で定義されているアルゴリズムを用いて、すべての相対URIが基底URIに対して相対的に解決される。 `base`は、`@context`内で用いられるURIと、TDのインスタンスにセマンティックな処理が適用されるときに関連があるリンクトデータ [LINKED-DATA] のグラフ内で用いられるIRIには影響を与えない。	オプション	`anyURI`
`properties`	当該Thingの、Propertyベースの相互作用のアフォーダンスのすべて。	オプション	`PropertyAffordance`のマップ
`actions`	当該Thingの、Actionベースの相互作用のアフォーダンスのすべて。	オプション	`ActionAffordance`のマップ
`events`	当該Thingの、Eventベースの相互作用のアフォーダンスのすべて。	オプション	`EventAffordance`のマップ
`links`	指定されたThing Description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`の配列
`forms`	操作の実行方法を記述する、フォームのハイパーメディア制御の集合。フォームは、プロトコルバインディングのシリアライゼーションである。このバージョンのTDでは、Thingのレベルで記述できるすべての操作は、ThingのProperty と一度にまとめて相互作用する方法に関するものである。	オプション	`Form`の配列
`security`	セキュリティ定義名の集合。`securityDefinitions`で定義されているものから選定する。リソースにアクセスするためには、これらすべてが満たされていなければならない。	必須	`string`または`string`の配列
`securityDefinitions`	名前付きセキュリティ構成情報 (定義のみ) の集合。`security`の名前-値のペアで名前が用いられていなければ、実際には適用されない。	必須	`SecurityScheme`のマップ

翻訳者のメモ

上記表中の「`security`」において，小文字の「must」があるが，Assertionとして抽出しないでよいか。

`@context`の名前-値のペアには、`anyURI`型の場合は直接、配列型の場合は最初の要素として、anyURIである `https://www.w3.org/2019/wot/td/v1` を含めなければならない (*MUST*)。`@context`が配列である場合は、anyURIである `https://www.w3.org/2019/wot/td/v1` の後に`anyURI`型またはマップ型の要素を任意の順序で置くことができるが (*MAY*)、`@context`配列内には、すべての名前-値のペアを持つマップを一つだけ含めることが推奨される (*RECOMMENDED*)。`@context`配列に含まれるマップには、名前-値のペアを含めることができる (*MAY*)。ここで、(名前-値ペアの) 値は`anyURI`型の名前空間識別子であり、名前はその名前空間を示す用語または

接頭辞である。`@context`配列に含まれる一つのマップには、Thing Descriptionのデフォルト言語を定義する名前-値のペアを含めるべきである (*SHOULD*)。ここで、名前は`@language`という用語で、値は [BCP47] で定義されている整形式の言語タグである (例えば、`en`、`de-AT`、`gsw-CH`、`zh-Hans`、`zh-Hant-HK`、`sl-nedis`)。

人間が読めるすべてのテキスト文字列の基本書字方向の計算は、次の一連の規則によって定義される。

- 言語タグが指定されていない場合は、基本書字方向は、最初の強い文字 (first-strong) のヒューリスティクスまたはCLDR Likely Subtags [LDML] などの検出アルゴリズムによって推測されるべきである (*SHOULD*)。
- `MultiLanguage`マップの外部では、基本書字方向は、デフォルト言語の言語タグから推測できる (*MAY*)。
- `MultiLanguage`マップの内部では、名前-値のペアの各値の基本書字方向は、対応する名前で指定されている言語タグから推測できる (*MAY*)。
- 異なる基本書字方向を持つ複数のスクリプトで言語を記述できる場合、`@language`または`MultiLanguage`マップで指定されている対応する言語タグには、適切な基本書字方向を推測できるように、文字のサブタグを含めなければならない (*MUST*)。例は、アゼルバイジャン語 (Azeri) で、ラテン文字を用いる場合 (`az-Latn`を用いて指定) はLTRで、アラビア文字を用いる場合 (`az-Arab`を用いて指定) はRTLで記述される。

TDプロセッサは、双方向テキストを処理する際に、いくつかの特殊なケースに注意すべきである。ユーザに文字列を提示する際に、特に周囲のテキストに埋め込むときには (例えば、ウェブユーザインターフェース用に)、bidi分離 (bidi isolation) を用いるように注意すべきである。言語が正しく識別されたとしても、方向が混在する文はどの言語においても発生しえる。

TDの作成者は、未熟なユーザエージェントが正常に表示できる方法で、方向が混合する文字列を提供するように努めるべきである。例えば、RTLの文字列がLTR進行 (ラテン文字で書かれた数字、ブランド名や商号など) で始まる場合、文字列の先頭にRLM文字を含めるか、逆方向の進行をbidi制御でラップすることで適切な表示をサポートできる。

「ウェブの文字列: 言語と方向のメタデータ」 [string-meta] は、双方向テキストを用いるにあたってのガイダンスを提供するとともに、いくつかの落とし穴について例示している。

`properties` (プロパティー)、`actions` (アクション)、および`events` (イベント) の配列で明示的に提供される相互作用のアフォーダンスに加えて、Thingは、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相互作用も提供できる。Thingのインスタンスの`forms`配列に`Form`インスタンスが含まれている場合、`op`という名前に直接的に、または配列内で割り当てられている文字列の値は、`readallproperties`、`writeallproperties`、`readmultipleproperties`、または`writemultipleproperties`のいずれかの*操作型*の一つでなければならない (*MUST*)。 (Thingのインスタンス内の`form`の使用例を参照。)

これらのメタ相互作用ごとのデータスキーマは、各`PropertyAffordance`インスタンスのデータスキーマを一つの`ObjectSchema`インスタンスに結合することで構築される。ここで、`ObjectSchema`インスタンスの`properties`マップは、対応する`PropertyAffordances`インスタンスの名前で識別される`PropertyAffordances`の各データスキーマを含んでいる。

特に指定されていない場合 (例えば、TDコンテキスト拡張により)、`readmultipleproperties`操作のリクエストデータは、`Form`インスタンスで指定されているコンテンツタイプにシリアライズされている、意図された`PropertyAffordances`インスタンス名を含む配列である。

5.3.1.2 `InteractionAffordance`相互作用のアフォーダンス §

可能な選択肢をConsumerに提示するThingのメタデータで、これにより、ConsumerがThingと相互作用を行える方法を提案する。潜在的なアフォーダンスには多くの種類があるが、W3C WoTでは、Property、Action、Eventという3種類の相互作用のアフォーダンスを定義している。

語彙用語	*説明*	*割り当て*	*型*
`@type`	セマンティックタグ (または型) でオブジェクトをラベル付けするJSON-LDキーワード。	オプション	`string`または`string`の配列

語彙用語	*説明*	*割り当て*	*型*
`title`	デフォルトの言語に基づいて、人間が読めるタイトルを提供する (例えば、UI表現用のテキストを表示)。	オプション	`string`
`titles`	多言語の、人間が読めるタイトルを提供する (例えば、様々な言語でUI表現のテキストを表示)。	オプション	`MultiLanguage`
`description`	デフォルト言語に基づいて、追加の (人間が読める) 情報を提供する。	オプション	`string`
`descriptions`	様々な言語の (人間が読める) 情報をサポートするために使用できる。	オプション	`MultiLanguage`
`forms`	操作の実行方法を記述する、フォームのハイパーメディア制御の集合。フォームは、プロトコルバインディングのシリアライゼーションである。	必須	`Form`の配列
`uriVariables`	DataSchema宣言に基づいて、URIテンプレート変数をコレクションとして定義する。	オプション	`DataSchema`のマップ

`InteractionAffordance`というクラスには、次のサブクラスがある。

- `PropertyAffordance`
- `ActionAffordance`
- `EventAffordance`

5.3.1.3 `PropertyAffordance` (Propertyのアフォーダンス) §

Thingの状態を公開する相互作用のアフォーダンス。そして、この公開された状態は、取得 (読み取り) することができる。また、オプションで更新 (書き込み) することができる。Thingは、変更後の新しい状態をプッシュすることにより、Propertyを監視可能にすることも選択できる。

翻訳者のメモ

Thingから公開された state (状態) を誰が取得・更新するのかも明記すべき。基本的には、Consumerもしくは Intermediaryが取得・更新するものであると考えられる。

語彙用語	*説明*	*割り当て*	*型*
`observable`	Thingを提供するServientとIntermediaryが、このPropertyの`observeproperty`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`

注

Propertyインスタンスは、DataSchemaというクラスのインスタンスでもある。したがって、`type`、`unit`、`readOnly`、`writeOnly`などのメンバーを含めることができる。

`PropertyAffordance`は、`InteractionAffordance`クラスと`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`に割り当てられている値は、`readproperty`、`writeproperty`、`observeproperty`、`unobserveproperty`、またはこれらの用語の組み合わせを含んでいる配列のいずれかでなければならない (*MUST*)。

5.3.1.4 `ActionAffordance` (Actionのアフォーダンス) §

状態を操作したり (例えば、照明のオン/オフを切り替える)、Thingにおけるプロセスを始動させる (例えば、時間の経過とともに照明を暗くする) といった、Thingの機能の呼び出しを可能にする相互作用のアフォーダンス。

語彙用語	説明	割り当て	型
`input`	Actionの入力データスキーマを定義するために用いられる。	オプション	`DataSchema`
`output`	Actionの出力データスキーマを定義するために用いられる。	オプション	`DataSchema`
`safe`	Actionが安全か (=true) 否かを通知する。Actionを呼び出す際に、(当該Actionの呼び出しによって) 内部状態 (リソースの状態を参照) が変更されることがありえるかありえないかを通知するために用いられる。その場合 (=当該Actionの呼び出しによって、内部状態が変更されることがない場合)、応答を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`
`idempotent`	Actionが冪等か (=true) 否かを示す。同じ入力に基づいて当該のActionを繰り返し呼び出した場合の結果 (結果が存在している場合) が常に同じであるかどうかを通知する。	デフォルトあり	`boolean`

翻訳者のメモ

上記の表中、「safe」の説明において、「Used to signal if there is no internal state (cf. resource state) is changed when invoking an Action.」とあるが、「there is no internal state is canged」となっており、「is」が二回使われているのはおかしいので，「there is」を削除すべき。

`ActionAffordance`は、`InteractionAffordance`クラスのサブクラスである。Formインスタンスが`ActionAffordance`インスタンス内にある場合、opに割り当てられている値は、`invokeaction`でなければならない (*MUST*)。

5.3.1.5 `EventAffordance` (Eventのアフォーダンス) §

イベント (例えば、オーバーヒートの警報) の出どころを記述している相互作用のアフォーダンスで、当該イベントのデータをConsumerに非同期でプッシュする。

翻訳者のメモ

上記箇所の英語原文では「event data」に，出どころを記述している当該イベントのデータであることを示すための「the」があるべき．

語彙用語	説明	割り当て	型
`subscription`	登録時に渡す必要があるデータ (例えば、Webhookを設定するためのフィルターやメッセージ形式) を定義する。	オプション	`DataSchema`
`data`	ThingがプッシュするEventの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`
`cancellation`	登録を中止するために渡す必要があるデータ (例えば、Webhookを削除するための特定のメッセージ) を定義する。	オプション	`DataSchema`

`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場合、`op`に割り当てられている値は、`subscribeevent`、`unsubscribeevent`、または配列内の両方の用語のいずれかでなければならない (*MUST*)。

5.3.1.6 `VersionInfo` (バージョン情報) §

TDドキュメントに関するバージョン情報を提供するThingのメタデータ。必要に応じて、ファームウェアやハードウェアのバージョン (TD名前空間外の用語定義) などの付加的なバージョン情報をTDコンテキスト拡張のメカニズムを介して拡張できる。

語彙用語	*説明*	*割り当て*	*型*
`instance`	このTDのインスタンスのバージョン表示を提供する。	必須	`string`

`VersionInfo`クラスのインスタンス内の値は、ドットで区切られた三つの数字の列がそれぞれ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示す、セマンティックバージョニングのパターンに従うことが推奨される。詳細に関しては、 [SEMVER] を参照のこと。

5.3.1.7 `MultiLanguage` (多言語) §

[BCP47] で記述されている言語タグによって識別される様々な言語の、人間が読めるテキストの集合を提供するマップ。Thing Descriptionのインスタンスにおけるこのコンテナの使用例については、§ 6.3.2 人間が読めるメタデータを参照していただきたい。

`MultiLanguage`マップのそれぞれの名前は、 [BCP47] で定義されている言語タグでなければならない (*MUST*)。`MultiLanguage`マップのそれぞれの値は、`string`型でなければならない (*MUST*)。

5.3.2 データスキーマ語彙の定義 §

データスキーマ語彙定義は、JSONスキーマ [JSON-SCHEMA] で定義されている用語の非常に一般的なサブセットを反映したものである。Thing Descriptionのインスタンス内のデータスキーマ定義はこの定義済みのサブセットに限定されておらず、§ 7. TDコンテキスト拡張に記述されているとおり、追加の用語のためにTDコンテンツ拡張を用いて、JSONスキーマに含まれる追加の用語を使用できること、そして、そうでない場合 (=コンテクスト拡張が利用されていない場合) は、これらの用語は、TDプロセッサによってセマンティック的に無視されることに注意が必要である。 (セマンティックな処理の詳細に関しては、§ D. JSON-LDコンテキストの使用法や、https://www.w3.org/2019/wot/tdなどの名前空間IRI下のドキュメントを参照)。

> 翻訳者のメモ
>
> 上記で「otherwise」の意味するところがわかりにくかったため、「(=コンテクスト拡張が利用されていない場合)」という補足を加えた。

データスキーマは、データ形式に含まれるデータの抽象的な表記法である。TDでは、具体的なデータ形式は、コンテンツタイプを用いてForm (§ 5.3.4.2 `Form` (フォーム) を参照) で指定される。Formのインスタンス内のコンテンツタイプの値が`application/json`であれば、JSONスキーマプロセッサでデータスキーマを直接処理できる。そうでない場合は、Web of Things (WoT) バインディングテンプレート [WOT-BINDING-TEMPLATES] により、データスキーマの、XML [xml] などの他のコンテンツタイプへの、利用可能なマッピングが定義される。フォームのインスタンス内のコンテンツタイプが`application/json`でなく、当該コンテンツタイプにマッピングが定義されていない場合は、当該コンテンツタイプに対するデータスキーマの指定には意味がない。

5.3.2.1 `DataSchema` §

利用されているデータ形式を記述するメタデータ。バリデーションのために使用できる。

語彙用語	説明	割り当て	型
@type	セマンティックタグ (または型) でオブジェクトをラベル付けするJSON-LDキーワード。	オプション	stringまたはstringの配列
title	デフォルトの言語に基づいて、人間が読めるタイトルを提供する (例えば、UI表現用のテキストを表示)。	オプション	string
titles	多言語の人間が読めるタイトルを提供する (例えば、様々な言語でUI表現のテキストを表示)。	オプション	MultiLanguage
description	デフォルト言語に基づいて、追加の (人間が読める) 情報を提供する。	オプション	string
descriptions	様々な言語の (人間が読める) 情報をサポートするために使用できる。	オプション	MultiLanguage
type	JSONスキーマと互換性のあるJSONベースのデータ型の割り当て (ブーリアン、整数、数値、文字列、オブジェクト、配列、ヌル (null) のうちのいずれか)。	オプション	string (object、array、string、number、integer、 boolean、nullのうちのいずれか)
const	定数値を提供する。	オプション	任意の型
unit	例えば、国際的な科学、工学、ビジネスなどで用いられている単位の情報を提供する。	オプション	string
oneOf	当該データが、配列内で指定されたスキーマのいずれかに対して有効であることを保証するために用いられる。	オプション	DataSchemaの配列

語彙用語	*説明*	*割り当て*	*型*
`enum`	配列として提供される、制限された値の集合。	オプション	任意の型の配列
`readOnly`	プロパティーの相互作用/値が読み取り専用か (=true) 否か (=false) を示すヒントであるブーリアン値。	デフォルトあり	`boolean`
`writeOnly`	プロパティーの相互作用/値が書き込み専用か (=true) 否か (=false) を示すヒントであるブーリアン値。	デフォルトあり	`boolean`
`format`	「date-time」、「email」、「uri」などの形式パターンに基づくバリデーションを可能にする (以下も参照のこと)。	オプション	`string`

`DataSchema`というクラスには、次のサブクラスがある。

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

`format`文字列の値は、 [JSON-SCHEMA] (特に、7.3 定義フォーマット) で定義されている一定の値の集合と、それらに対応するフォーマットの規則から判別される。Servientは、`format`の値を用いて、それに応じた追加のバリデーションを実行できる (*MAY*)。既知の値の集合中に含まれない値が`format`に割り当てられていれば、そのようなバリデーションは成功すべきである (*SHOULD*)。

5.3.2.2 `ArraySchema` (配列スキーマ) §

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`インスタンスの中の`type`に割り当てられている`array`という値により示される。

語彙用語	*説明*	*割り当て*	*型*
`items`	配列の特性を定義するために用いられる。	オプション	`DataSchema`または`DataSchema`の配列
`minItems`	配列内になくてはならない要素の最小の数を定義する。	オプション	`unsignedInt`

語彙用語	説明	割り当て	型
maxItems	配列内になくてはならない要素の最大の数を定義する。	オプション	unsignedInt

5.3.2.3 BooleanSchema (ブーリアンスキーマ) §

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インスタンスのtypeに割り当てられているbooleanという値により示される。

5.3.2.4 NumberSchema (数値スキーマ) §

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インスタンスのtypeに割り当てられているnumberという値により示される。

語彙用語	説明	割り当て	型
minimum	最小の数の値を指定する。関連づけられた数値型または整数型にのみ適用される。	オプション	double
maximum	最大の数の値を指定する。関連づけられた数値型または整数型にのみ適用される。	オプション	double

5.3.2.5 IntegerSchema (整数スキーマ) §

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インスタンスの中のtypeに割り当てられているintegerという値により示される。

語彙用語	説明	割り当て	型
minimum	最小の数の値を指定する。関連づけられた数値型または整数型にのみ適用される。	オプション	integer
maximum	最大の数の値を指定する。関連づけられた数値型または整数型にのみ適用される。	オプション	integer

5.3.2.6 ObjectSchema (オブジェクトスキーマ) §

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インスタンスの中のtypeに割り当てられているobjectという値により示される。

語彙用語	説明	割り当て	型
properties	入れ子になったデータスキーマの定義。	オプション	DataSchemaのマップ
required	オブジェクト型のどのメンバーが必須かを定義する。	オプション	stringの配列

5.3.2.7 StringSchema (文字列スキーマ) §

`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`インスタンスの`type`に割り当てられている`string`という値で示される。

5.3.2.8 `NullSchema` (ヌル (Null) スキーマ) §

`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`インスタンスの`type`に割り当てられている`null`という値で示される。このサブクラスは、一つの許容値、つまり`null`のみを記述する。これは`oneOf`宣言の一部として使用でき、その場合、データが`null`にもなりえることを示すために用いる。

5.3.3 セキュリティ語彙の定義 §

この仕様は、W3C WoTのプロトコルバインディングとして適しているプロトコルに直接組み込まれる、またはそのプロトコルで広く用いられる、確立されたセキュリティメカニズムの選択について規定している。現在のHTTPセキュリティスキームは、部分的にOpenAPI 3.0.1に基づいている ([OPENAPI] も参照)。しかし、この仕様で示しているHTTPセキュリティスキーム、語彙、および構文は、OpenAPIと多くの類似点を共有しているものの、互換性はない。

5.3.3.1 `SecurityScheme` (セキュリティスキーム) §

セキュリティメカニズムの構成を記述するメタデータ。`scheme`という名前に割り当てられる値は、Thing Descriptionに含まれる語彙内、つまり、§ 5. TD情報モデルで定義される標準的な語彙内、またはTDコンテキスト拡張に含まれる語彙内のいずれか、により定義されなければならない (*MUST*)。

語彙用語	*説明*	*割り当て*	*型*
`@type`	セマンティックタグ (または型) でオブジェクトをラベル付けするJSON-LDキーワード。	オプション	`string`または`string`の配列
`scheme`	設定されているセキュリティメカニズムの識別。	必須	`string` (例えば、`nosec`、`basic`、`digest`、`bearer`、`psk`, `oauth2`、または`apikey`)
`description`	デフォルト言語に基づいて、追加の (人間が読める) 情報を提供する。	オプション	`string`
`descriptions`	様々な言語の (人間が読める) 情報をサポートするために使用できる。	オプション	`MultiLanguage`
`proxy`	このセキュリティ構成情報がアクセスを提供するプロキシサーバーのURI。指定されていない場合、対応するセキュリティ構成情報はエンドポイント用である。	オプション	`anyURI`

`SecurityScheme`というクラスには次のサブクラスがある。

- NoSecurityScheme
- BasicSecurityScheme
- DigestSecurityScheme
- APIKeySecurityScheme
- BearerSecurityScheme
- PSKSecurityScheme
- OAuth2SecurityScheme

5.3.3.2 NoSecurityScheme (セキュリティスキームなし) §

nosecという語彙用語 (つまり、"scheme": "nosec") で識別されるセキュリティ構成情報で、リソースへのアクセスに認証やその他のメカニズムが必要ないことを示す。

5.3.3.3 BasicSecurityScheme (基本セキュリティスキーム) §

basicという語彙用語 (つまり、"scheme": "basic") で識別される、暗号化されていないユーザ名とパスワードが用いられる基本認証 [RFC7617] セキュリティ構成情報。このスキームは、例えば、TLSなどの機密性を提供する他のセキュリティメカニズムとともに用いるべきである。

語彙用語	*説明*	*割り当て*	*型*
in	セキュリティ認証情報の場所を指定する。	デフォルトあり	string (header、query、body、cookieのうちの一つ)
name	クエリ、ヘッダー、またはクッキーのパラメータの名前。	オプション	string

5.3.3.4 DigestSecurityScheme (ダイジェストセキュリティスキーム) §

digestという語彙用語 (つまり、"scheme": "digest") で識別されるダイジェストアクセス認証 [RFC7616] セキュリティ構成情報。このスキームは基本認証に似ているが、中間者攻撃 (man-in-the-middle attack) を回避する機能が追加されている。

語彙用語	*説明*	*割り当て*	*型*
qop	保護の品質。	デフォルトあり	string (auth、auth-intのうちの一つ)
in	セキュリティ認証情報の場所を指定する。	デフォルトあり	string (header、query、body、cookieのうちの一つ)
name	クエリ、ヘッダー、またはクッキーのパラメータの名前。	オプション	string

5.3.3.5 APIKeySecurityScheme (APIキーセキュリティスキーム) §

apikeyという語彙用語 (つまり、"scheme": "apikey") で識別されるAPIキー認証セキュリティ構成情報。これは、アクセストークンが不透明で、標準的なトークン形式を用いていない場合である。

語彙用語	説明	割り当て	型
in	セキュリティ認証情報の場所を指定する。	デフォルトあり	string (header、query、body、cookieのうちの一つ)
name	クエリ、ヘッダー、またはクッキーのパラメータの名前。	オプション	string

5.3.3.6 BearerSecurityScheme (ベアラーセキュリティースキーム) §

ベアラートークンがOAuth2と関わりなく用いられる状況のために、bearerという語彙用語 (つまり、"scheme": "bearer") で識別されるベアラートークン [RFC6750] セキュリティ構成情報。oauth2スキームが指定されている場合は、一般的に、このスキームを指定する必要はなく、暗黙的に指定されている。formatでは、jwtという値は [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] との適合性を示す。algの値はこれらの標準における解釈と整合的に取り扱われる。ベアラートークンのその他の形式とアルゴリズムは、語彙拡張で指定してもよい (*MAY*)。

語彙用語	説明	割り当て	型
authorization	認可サーバーのURI。	オプション	anyURI
alg	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。	デフォルトあり	string (例えば、MD5、ES256、またはES512-256)
format	セキュリティ認証情報の形式を指定する。	デフォルトあり	string (例えば、jwt、cwt、jwe、またはjws)
in	セキュリティ認証情報の場所を指定する。	デフォルトあり	string (header、query、body、cookieのうちの一つ)
name	クエリ、ヘッダー、またはクッキーのパラメータの名前。	オプション	string

5.3.3.7 PSKSecurityScheme (PSKセキュリティスキーム) §

pskという語彙用語 (つまり、"scheme": "psk") で識別される事前共有キー認証セキュリティ構成情報。

語彙用語	説明	割り当て	型
identity	選択や確認に使用できる情報を提供する識別子。	オプション	string

5.3.3.8 OAuth2SecurityScheme (OAuth2セキュリティスキーム) §

oauth2という語彙用語 (つまり、"scheme": "oauth2") で識別される、 [RFC6749] と [RFC8252] に適合するシステムのためのOAuth2認証セキュリティ構成情報である。codeのフローでは、authorizationとtokenの両方が含まれていなければならない (*MUST*)。SecuritySchemeでscopesが定義されていなければ、それは空と見なされる。

語彙用語	説明	割り当て	型
authorization	認可サーバーのURI。	オプション	anyURI
token	トークンサーバーのURI。	オプション	anyURI
refresh	更新サーバーのURI。	オプション	anyURI
scopes	配列として提供される認可範囲識別子の集合。これらは、クライアントがアクセスできるリソースとその方法を識別するために、認可サーバーによって返されたトークンで提供され、フォームに関連付けられる。フォームに関連付けられる値は、そのフォームでアクティブなOAuth2SecuritySchemeで定義されているものから選択すべきである。	オプション	stringまたはstringの配列
flow	認可フロー。	必須	string (例えば、code)

5.3.4 ハイパーメディア制御語彙の定義 §

現在のモデルは、Thingが公開する (型付きの) ウェブリンクとウェブフォームの表現を提供する。Linkクラスの定義は、ウェブリンク [RFC8288] で定義されている用語の非常に一般的なサブセットを反映したものである。定義された用語は、例えば、*照明*という*Thing*が*スイッチ*という*Thing*によって制御されるなど、別のThingとの関係を記述するために使用できる。Formクラスは、Thing (および、その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のフォームに対応している。

5.3.4.1 Link (リンク) §

リンクは、「***リンクコンテキスト***の***リンクターゲット***には***関係型***のリソースがある」という形のステートメントとして見ることができ、オプションの***ターゲット属性***でリソースを詳細に記述できる。

語彙用語	説明	割り当て	型
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	anyURI
type	リンクの逆参照の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。	オプション	string
rel	リンク関係型はリンクのセマンティクスを識別する。	オプション	string
anchor	指定されたURIまたはIRIでリンクコンテキスト (デフォルトでは、そのidで識別されるThing自体) を上書きする。	オプション	anyURI

5.3.4.2 Form (フォーム) §

フォームは、「***フォームコンテキスト***で***操作型***の操作を実行するために、***送信ターゲット***に***リクエストメソッド***のリクエストを行う」というステートメントとして見ることができ、オプションの***フォームフィールド***で、必要なリクエストを詳細に記述できる。Thing Descriptionでは、***フォームコンテキスト***は、Property、Action、Eventなどの周囲のオブジェクト、またはメタ相互作用のThing自体である。

語彙用語	*説明*	*割り当て*	*型*
op	フォームで記述された操作を実行するセマンティックな意図を示す。例えば、Propertyの相互作用では、getとsetの操作が可能である。プロトコルバインディングには、get操作用のフォームと、別のset操作用フォームを含むことができる。op属性は、フォームの対象を示し、必要な操作に適したフォームをクライアントが選択できるようにする。opには、それぞれに操作のセマンティックな意図を表す一つ以上の相互作用の動詞を割り当てることができる。	デフォルトあり	stringまたはstringの配列 (readproperty、writeproperty、observ writemultiplepropertiesのうちの一つ)
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	anyURI
contentType	メディアタイプ (例えば、text/plain) と、そのメディアタイプの潜在的なパラメータ (例えば、charset=utf-8) [RFC2046] に基づいてコンテンツタイプを割り当てる。	デフォルトあり	string
contentCoding	コンテンツコーディングの値は、表現に適用された、または適用できるエンコーディング変換を示す。コンテンツコーディングは、その基礎となるメディアタイプが同一のままで、かつ、情報を失うことなく、表現を圧縮したり、そうでない場合は便利に変換したりするために主に用いる。コンテンツコーディングの例には、「gzip」、「deflate」などがある。	オプション	string

語彙用語	*説明*	*割り当て*	*型*
`subprotocol`	複数のオプションがある場合に、特定のプロトコルで相互作用が達成される厳密なメカニズムを示す。例えば、HTTPやEventの場合、ロングポーリング (`longpoll`)、WebSub [websub] (`websub`)、サーバー送信Event (`sse`) [html] (EventSourceとしても知られている) などの非同期通知に使用可能ないくつかのメカニズムのどれを用いるかを示す。サブプロトコルの選択に制限はなく、このサブプロトコル用語で他のメカニズムも告知できることに注意していただきたい。	オプション	`string` (例えば、`longpoll`、`websub`、または`sse`)
`security`	セキュリティ定義名の集合。`securityDefinitions`で定義されているものから選定する。リソースにアクセスするためには、これらすべてが満たされていなければならない。	オプション	`string`または`string`の配列
`scopes`	配列として提供される認可範囲識別子の集合。これらは、クライアントがアクセスできるリソースとその方法を識別するために、認可サーバーによって返されたトークンで提供され、フォームに関連付けられる。フォームに関連付けられる値は、そのフォームでアクティブな`OAuth2SecurityScheme`で定義されているものから選択すべきである。	オプション	`string`または`string`の配列
`response`	このオプションの用語は、例えば、出力の通信メタデータが入力のメタデータと異なる場合 (例えば、出力コンテンツタイプが入力コンテンツタイプと異なる場合) に使用できる。応答の名前には、応答メッセージにのみ有効なメタデータが含まれる。	オプション	`ExpectedResponse`

`contentCoding`プロパティーの可能な値は、例えば、IANA HTTPコンテンツコーディングレジストリにある。

フォームの可能な操作型のリストは固定されている。このバージョンの仕様の時点では、[WOT-ARCHITECTURE] で記述されているWoT相互作用モデルを実装するために必要な既知の型のみが含まれている。この標準の将来のバージョンでは、このリストは拡張される可能性があるが、操作型は、サービエントが任意に設定すべきではない (*SHOULD NOT*)。

オプションの`response`の名前-値のペアを用いて、予期される応答メッセージのメタデータを提供できる。コア語彙では、コンテンツタイプ情報のみが含まれているが、TDコンテキスト拡張を適用できる。`response`の名前-値のペアが提供されていない場合は、応答のコンテンツタイプがFormインスタンスに割り当てられているコンテンツタイプと等しいと想定しなければならない (*MUST*)。`ExpectedResponse`クラス内の`contentType`にはデフォルト値がないことに注意していただきたい。例えば、フォームのコンテンツタイプの値が`application/xml`の場合、想定される応答のコンテンツタイプの値も`application/xml`になる。

ユースケースによっては、例えば、JSONを受け入れるけれども画像を返すActionなど、入力と出力のデータが異なる形式で表される場合がある。そのようなケースでは、オプションの`response`の名前-値のペアで、予期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。予期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には、`Form`インスタンスに`response`という名前を持つ名前-値のペアを含めなければならない (*MUST*)。例えば、`ActionAffordance`は、入力データでは`application/json`のみを受け入れ、出力データでは`image/jpeg`というコンテンツタイプで応答する可能性がある。その場合、コンテンツタイプが異なり、`response`の名前-値のペアを用いて、応答コンテンツタイプ (`image/jpeg`) の情報をConsumerに提供しなければならない。

5.3.4.3 `ExpectedResponse` (予期される応答) §

予期される応答メッセージを記述する通信メタデータ。

語彙用語	*説明*	*割り当て*	*型*
`contentType`	メディアタイプ (例えば、`text/plain`) と、そのメディアタイプの潜在的なパラメータ (例えば、`charset=utf-8`) [RFC2046] に基づいてコンテンツタイプを割り当てる。	必須	`string`

5.4 デフォルト値の定義 §

TDの割り当てが欠落している場合、TDプロセッサは、§ 5.4 デフォルト値の定義の表で表されているデフォルト値の割り当てに従わなければならない (*MUST*)。

次の表で、TD情報モデルで定義しているすべてのデフォルト値を示す。

クラス	*語彙用語*	*デフォルト値*	*説明*
`Form`	`contentType`	`application/json`	
`DataSchema`	`readOnly`	`false`	
`DataSchema`	`writeOnly`	`false`	
`ActionAffordance`	`safe`	`false`	
`ActionAffordance`	`idempotent`	`false`	
`Form`	`op`	`readproperty`と`writeproperty`の要素を持つ`string`の配列	`PropertyAffordance`のインスタンス内で定義されている場合
`Form`	`op`	`invokeaction`	`ActionAffordance`のインスタンス内で定義されている場合

クラス	語彙用語	デフォルト値	説明
`Form`	`op`	`subscribeevent`	`EventAffordance`のインスタンス内で定義されている場合
`BasicSecurityScheme`	`in`	`header`	
`DigestSecurityScheme`	`in`	`header`	
`BearerSecurityScheme`	`in`	`header`	
`APIKeySecurityScheme`	`in`	`query`	
`DigestSecurityScheme`	`qop`	`auth`	
`BearerSecurityScheme`	`alg`	`ES256`	
`BearerSecurityScheme`	`format`	`jwt`	

6. TD表現形式 §

WoT Thing DescriptionはThingを表し、§ 5. TD情報モデルに基づいてモデル化され構造化される。この項では、TD情報モデルで定義している`Thing`というクラスのインスタンスのシリアライゼーションである、ThingのJSONベースの表現形式を定義する。

TDプロセッサは、§ 6.1 JSONの型へのマッピングと§ 6.3 情報モデルのシリアライゼーションで記述している規則に従って、Thing DescriptionのJSON形式 [RFC8259] へのシリアライズおよび/またはその形式からのThing Descriptionの逆シリアライズができなければならない (*MUST*)。

TD情報モデルのJSONシリアライゼーションは、セマンティックの評価を合理化するために、JSON-LD 1.1 [json-ld11] の構文との整合化を図っている。したがって、TD表現形式は未加工のJSONとして処理するか、JSON-LD 1.1プロセッサで処理するかのいずれかが可能である (セマンティックな処理の詳細に関しては、§ D. JSON-LDコンテキストの使用法や、https://www.w3.org/2019/wot/tdなどの名前空間IRI下のドキュメントを参照)。

相互運用可能な国際化をサポートするには、オープンエコシステムに関するRFC8259 [RFC8259] の8.1項で定義されている要件に従ってTDをシリアライズしなければならない (*MUST*)。要約すると、これには次が必要である。

- TDはUTF-8 [RFC3629] を用いてエンコードしなければならない (*MUST*)。
- 実装では、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 (*MUST NOT*)。
- TDプロセッサは、BOM (byte order mark) の存在をエラーとして扱うのではなく、無視することができる (*MAY*)。

6.1 の型へのマッピング §

TD情報モデルは、モデルのオブジェクトとJSONの型の間に簡単なマッピングが存在するように構築される。すべてのクラスのインスタンスはJSONオブジェクトにマッピングされ、クラスのインスタンスの個々の名前-値のペアは、JSONオブジェクトのメンバーである。

§ 5.3 クラス定義で言及しているすべての単純型 (つまり、`string`、`anyURI`、`dateTime`、`integer`、`unsignedInt`、`double`、および`boolean`) は、以下に列挙している規則に従って、プリミティブなJSONの型 (文字列、数値、ブール値) にマッピングされる。これらの規則は、名前-値のペアの値に適用される。

- `string`型または`anyURI`型の値は、JSON文字列としてシリアライズしなければならない (*MUST*)。
- `dateTime`型の値は、 [RFC3339] で指定されている「日時」形式に従ってJSON文字列としてシリアライズしなければならない (*MUST*)。例には、`2019-05-24T13:12:45Z`や`2015-07-11T09:32:26+08:00`が含まれる。`dateTime`型の値は、オフセットではなくUTCタイムゾーンを表す`Z`というリテラルを用いるべきである (*SHOULD*)。

- `integer`型または`unsignedInt`型の値は、小数部または指数部のないJSON数値としてシリアライズしなければならない*MUST*)。
- `double`型の値は、JSON数値としてシリアライズしなければならない (*MUST*)。
- `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 (*MUST*)。

TD情報モデルのすべての複合型 (つまり、配列、マップ、およびクラスのインスタンス) は、以下に列挙している規則に従って、構造化されたJSONの型 (配列およびオブジェクト) にマッピングされる。

- 配列型の値は、名前-値のペアの各値を、JSON配列の要素として、ペアの数値名順に並べて、JSON配列としてシリアライズしなければならない (*MUST*)。
- マップ型の値は、名前-値の各ペアを、JSONオブジェクトのメンバーにして、JSONオブジェクトとしてシリアライズしなければならない (*MUST*)。
- クラスのインスタンスは、§ 6.3 情報モデルのシリアライゼーションで個別に指定している詳細な規則に従って、JSONオブジェクトとしてシリアライズしなければならない (*MUST*)。

6.2 デフォルト値の省略 §

Thing Descriptionのシリアライゼーションでは、§ 5.4 デフォルト値の定義の表で列挙している、デフォルト値が定義されている語彙用語を省略できる。

次の例は、例1のTDのインスタンスに、デフォルト値でメンバーも含めるチェックボックスを付けたものである (チェックボックスにチェックあり)。これらのメンバーは、TDシリアライゼーションを簡素化するために省略できる (チェックボックスにチェックなし)。TDプロセッサは、これらの省略されたメンバーを、特定のデフォルト値で明示的に存在しているのと完く同じように解釈することに注意していただきたい。

例3

☑ *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": {
        "basic_sc": {
            "scheme": "basic",
            "in": "header"
        }
    },
    "security": [
        "basic_sc"
    ] ,
    "properties": {
        "status": {
            "type": "string",
            "readOnly" : false,
            "writeOnly" : false,
            "forms": [{
                "op": [
                    "readproperty",
                    "writeproperty"
                ] ,
                "href": "https://mylamp.example.com/status",
                "contentType": "application/json"
            }]
        }
    },
    "actions": {
        "toggle": {
            "safe": false,
            "idempotent": false,
            "forms": [{
                "op": "invokeaction",
                "href": "https://mylamp.example.com/toggle",
                "contentType": "application/json"
            }]
        }
    },
    "events": {
        "overheating": {
            "data": {
                "type": "string",
                "readOnly" : false,
                "writeOnly" : false
            },
            "forms": [{
                "op": "subscribeevent",
                "href": "https://mylamp.example.com/oh",
                "contentType": "application/json",
                "subprotocol": "longpoll"
            }]
        }
    }
}
```

用いられてい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語彙用語が適用されている場合があることに注意していただきたい。デフォルト値が関連付けられている場合もあるため、この小項目で説明しているように省略することができる。詳細な情報は、§ 8.3 プロトコルバインディングにある。

6.3 情報モデルのシリアライゼーション §

6.3.1 Thingのルートオブジェクト §

Thing Descriptionは、Thing型のオブジェクトをルートとするデータ構造である。次に、Thing DescriptionのJSONシリアライゼーションはJSONオブジェクトであり、それは、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。

TDシリアライゼーションのルート要素は、`@context`という名前を持つメンバーと、`https://www.w3.org/2019/wot/td/v1` と等しいか、それぞれに含む文字列型または配列型の値を持つメンバーを含むJSONオブジェクトでなければならない (*MUST*)。

一般的に、このURIは、この仕様で定義しているTD表現形式のバージョンを識別するために用いる。JSON-LD処理 [json-ld11] の場合、このURIはThing Descriptionのコンテキストファイルを指定する。配列型の`@context`は、TDコンテキスト拡張を示す (詳細に関しては、§ 7. TDコンテキスト拡張を参照)。

例4

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

Thingのインスタンスの名前-値のペアはすべて、名前がThingのシグネチャ内の語彙用語である場合、ルートオブジェクトのJSONメンバーとしてシリアライズしなければならない (*MUST*)。

すべての必須およびオプションの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の断片を以下に示す。

例5: Thingのシリアライゼーションのサンプル

```
{
    "@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": {...},
    "links": [...] ,
    "forms": [...]
}
```

Thingというクラスのインスタンス内の`version`、`securityDefinitions`、`properties`、`actions`、および`events`に割り当てられているすべての値は、JSONオブジェクトとしてシリアライズしなければならない (*MUST*)。

`links`に割り当てられているすべての値、およびThingというクラスのインスタンス内の`forms`は、それぞれ§ 6.3.8 `links` (リンク) と§ 6.3.9 `forms` (フォーム) で定義しているJSONオブジェクトを含むJSON配列としてシリアライズしなければならない (*MUST*)。

Thingというクラスのインスタンス内のsecurityに割り当てられている値は、JSON文字列またはJSON文字列を要素とするJSON配列としてシリアライズしなければならない (*MUST*)。

6.3.2 人間が読めるメタデータ §

titleおよびdescriptionという名前のJSONメンバーは、人間が読めるメタデータを提供するためにTDドキュメント内で用いる。これらは、TDドキュメントの検査を行う開発者用のコメントとして、またはユーザインターフェース用の表示テキストとして使用できる。

§ 5.3.1.1 Thingで定義しているように、人間が読めるメタデータを表示するために用いられるテキストの基本書字方向は、最初の強い文字 (first-strong) の規則などのヒューリスティクスを用いて推定するか、言語情報から推測することができる。TDドキュメントでは、デフォルトの言語は@context内の@languageに割り当てられている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ともに、テキストの基本書字方向を決定するために使用できる。しかし、人間が読めるテキストを解釈するときは、人間が読める各文字列値を個別に処理しなければならない (*MUST*)。言い換えれば、TDプロセッサは、ある文字列から別の文字列へと方向の変更を繰り越したり、ある文字の方向をTD内の他の場所から推測したりすることはできない。

> 注
>
> ウェブ上の文字列 [STRING-META] は、テキストの基本書字方向を決定する手段として、最初の強い文字 (strong-first) と言語ベースの両方の推論を提案している。Thing Descriptionの形式がJSON-LD 1.1 [json-ld11] に基づいていて、現在のところ明示的な方向のメタデータが欠落していることを考えると、これらのアプローチはこの公開時点では適切であると考えられる。しかし、JSON-LD 1.1が [STRING-META] で推奨されている明示的な基本書字方向のメタデータのサポートを採用した場合、その機能を利用するためにThing Descriptionの形式を更新すべきである。

titleおよびdescriptionを用いたTDの断片を以下に示す。デフォルト言語は、@context配列のJSONオブジェクト内の@languageメンバーの定義により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、一つのTDドキュメント内で複数の言語で人間が読めるメタデータを提供するためにTDドキュメント内で用いる。MultiLanguageマップのすべての名前-値のペアは、JSONオブジェクトのメンバーとしてシリアライズしなければならず (*MUST*)、その名前は [BCP47] で定義されている整形式の言語タグであり、値はタグで示されている言語の人間が読める文字列である。詳細に関して

は、§ 5.3.1.7 MultiLanguage (多言語) を参照していただきたい。TDドキュメント内のすべてのMultiLanguageオブジェクトには、同じ言語メンバーの集合が含まれているべきである (*SHOULD*)。

様々なレベルでtitlesとdescriptionsを用いたTDの断片を以下に示す。

例7

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "title": "MyThing",
    "titles": {
        "en":"MyThing",
        "de": "MeinDing",
        "ja" : "私の物",
        "zh-Hans" : "我的东西",
        "zh-Hant" : "我的東西"
    },
    "descriptions": {
        "en":"Human readable information.",
        "de": "Menschenlesbare Informationen.",
        "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": {
            "titles": {
                "en": "On/Off",
                "de": "An/Aus",
                "ja": "オンオフ",
                "zh-Hans": "开关",
                "zh-Hant": "開關" },
            "type": "boolean",
            "forms": [...]
        },
        "status": {
            "titles": {
                "en": "Status",
                "de": "Zustand",
                "ja": "状態",
                "zh-Hans": "状态",
                "zh-Hant": "狀態" },
            "type": "object",
            ...
            "forms": [...]
        }
    },
    ...
}
```

TDのインスタンスは、titleとdescriptionの使用をtitlesとdescriptionsと組み合わせることもできる。titleおよびtitlesまたはdescriptionおよびdescriptionsが同じJSONオブジェクト内に存在している場合、titleとdescriptionの値はデフォルトのテキストとして表示されるかもしれない (*MAY*)。titleおよびtitlesまたはdescriptionおよびdescriptionsがTDドキュメント内に存在する場合、個々のtitleとdescriptionのメンバーは、それぞれ対応するtitlesとdescriptionsのメンバーを持っているべきである (*SHOULD*)。デフォルトのテキストの言語は、デフォルトの言語で示される。これは通常、Thing Descriptionのインスタンスの作成者が設定する。

例8

```
{
    "@context": [
        "https://www.w3.org/2019/wot/td/v1",
        { "@language" : "de" }
    ] ,
    "title": "MyThing",
    "titles": {
        "en":"MyThing",
        "de": "MeinDing",
        "ja" : "私の物",
        "zh-Hans" : "我的东西",
        "zh-Hant" : "我的東西"
    },
    "description": "Menschenlesbare Informationen.",
    "descriptions": {
        "en":"Human readable information.",
        "de": "Menschenlesbare Informationen.",
        "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": {
            "title" : "An/Aus",
            "titles": {
                "en": "On/Off",
                "de": "An/Aus",
                "ja": "オンオフ",
                "zh-Hans": "开关",
                "zh-Hant": "開關" },
            "type": "boolean",
            "forms": [...]
        },
        "status": {
            "title" : "Zustand",
            "titles": {
                "en": "Status",
                "de": "Zustand",
                "ja": "状態",
                "zh-Hans": "状态",
                "zh-Hant": "狀態" },
            "type": "object",
            ...
            "forms": [...]
        }
    },
    ...
}
```

デフォルト言語を設定する別の可能性は、HTTPの`Accept-Language`ヘッダーフィールドなどの言語交渉メカニズムを用いることである。デフォルト言語の交渉が行われた場合、交渉の結果と、返されるコンテンツの対応するデフォルト言語を示す`@language`メンバーが存在しなければならない (*MUST*)。デフォルト言語の交渉が正常に行われた場合は、TDドキュメントは`titles`と`descriptions`のメンバーの`MultiLanguage`オブジェクトよりも優先して、メンバーである`title`と`description`に適切な一致する値を含めるべきである (*SHOULD*)。しかし、Thingは、そのような動的に生成されるTDのサポートも、言語交渉のサポートも行わないことを選択できる (*MAY*) (例えば、リソースの制約のために) ことに注意していただきたい。

6.3.3 `version` (バージョン) §

名前が、VersionInfoのシグネチャに含まれている語彙用語であるVersionInfoのインスタンスのすべての名前-値のペアは、語彙用語を名前として持つJSONメンバーとしてシリアライズしなければならない (*MUST*)。

バージョン情報オブジェクトのTDの断片を以下に示す。

例9

```
{
    ...
    "version": { "instance": "1.2.1" },
    ...
}
```

versionのメンバーは、TDコンテキスト拡張に基づく追加のアプリケーション固有および/またはデバイス固有のバージョン情報用のコンテナとして意図されている。詳細に関しては、§ 7.1 セマンティックなアノテーションを参照していただきたい。

6.3.4 securityDefinitions (セキュリティ定義) とsecurity (セキュリティ) §

Thingのインスタンスでは、securityDefinitionsに割り当てられている値は、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-値のペアは、マップのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (MUST)。ペアの名前はJSON文字列としてシリアライズしなければならず (*MUST*)、ペアの値 (SecuritySchemeのインスタンス) は、JSONオブジェクトとしてシリアライズしなければならない (*MUST*)。

SecuritySchemeのサブクラスのうちの一つのインスタンスのすべての名前-値のペアは、名前がそのサブクラスのシグネチャまたはSecuritySchemeのシグネチャに含まれている語彙用語である場合、語彙用語を名前として用いて、SecuritySchemeサブクラスのインスタンスのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。

次のTDの断片は、ヘッダーで基本的なユーザ名/パスワード認証を指定するシンプルなセキュリティ構成情報を示している。inに指定される値は、実際にはデフォルト値 (header) であり、省略できる。名前付きセキュリティ構成情報はsecurityDefinitionsマップで指定しなければならない。その定義は、securityメンバーのそのJSON名によってアクティブ化しなければならず、それは、一つの定義のみがアクティブ化される場合には文字列型でありえる。

例10

```
...
"securityDefinitions": {
    "basic_sc": {
        "scheme": "basic",
        "in": "header"
    }
},
"security": "basic_sc",
...
```

ここで、より複雑な例として、Thingにおけるベアラートークン認証と組み合わせたプロキシにおけるダイジェスト認証を示すTDの断片を示す。digestスキームでは、inのデフォルト値 (つまり、header) は省略されるが、適用はされる。正常な相互作用を行うためには、Consumer内で、ユーザ名/パスワードやトークンなどの対応する非公開のセキュリティ構成情報を設定しなければならないことに注意していただきたい。複数のセキュリティ定義をアクティブ化すると、securityメンバーは配列になる。

例11

```
...
"securityDefinitions": {
    "proxy_sc": {
        "scheme": "digest",
        "proxy": "https://portal.example.com/"
    },
    "bearer_sc": {
        "in":"header",
        "scheme": "bearer",
        "format": "jwt",
        "alg": "ES256",
        "authorization": "https://servient.example.com:8443/"
    }
},
"security": ["proxy_sc", "bearer_sc"] ,
...
```

TDのセキュリティ構成情報は必須である。Thingのレベル (つまり、TDルートオブジェクト) のsecurity配列を介して少なくとも一つのセキュリティ定義をアクティブ化しなければならない (*MUST*)。この構成は、Thingとの相互作用に必要なデフォルトのセキュリティメカニズムと見ることができる。セキュリティ定義は、フォームオブジェクトにsecurityメンバーを含めることにより、フォームのレベルでアクティブ化することもでき (*MAY*)、これは、Thingのレベルでアクティブ化されるすべての定義を上書き (つまり、完全に置き換え) する。

セキュリティが必要ない場合、nosecセキュリティスキームが提供される。Thingの最小限のセキュリティ構成情報は、次の例に示すように、Thingのレベルでのnosecセキュリティスキームのアクティブ化である。

例12

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "id": "urn:dev:ops:32473-Thing-1234",
    "title": "MyThing",
    "description": "Human readable information.",
    "support": "https://servient.example.com/contact",
    "securityDefinitions": { "nosec_sc": { "scheme": "nosec" }},
    "security": "nosec_sc",
    "properties": {...},
    "actions": {...},
    "events": {...},
    "links": [...]
}
```

より複雑な例を示すために、認証が不要なものを除き、すべての相互作用のアフォーダンスで基本認証を必要なThingがあると仮定する。`status` Propertyと`toggle` Actionの場合、`basic`認証が必要であり、Thingのレベルで定義される。しかし、`overheating` Eventの場合は、認証は必要ないため、セキュリティ構成情報はフォームのレベルで上書きされる。

例13

```
{
    ...
    "securityDefinitions": {
        "basic_sc": {"scheme": "basic"},
        "nosec_sc": {"scheme": "nosec"}
    },
    "security": ["basic_sc"] ,
    ...
    "properties": {
        "status": {
            ...
            "forms": [{
                "href": "https://mylamp.example.com/status"
            }]
        }
    },
    "actions": {
        "toggle": {
            ...
            "forms": [{
                "href": "https://mylamp.example.com/toggle"
            }]
        }
    },
    "events": {
        "overheating": {
            ...
            "forms": [{
                "href": "https://mylamp.example.com/oh",
                "security": ["nosec_sc"]
            }]
        }
    }
}
```

セキュリティ構成情報は、同じ相互作用のアフォーダンス内の様々なフォーム用に指定することもできる。これは、様々なセキュリティメカニズムをサポートする、HTTPやCoAP [RFC7252] などの、複数のプロトコルをサポートするデバイスに必要でありえる。これは、代替認証メカニズムが許されている場合にも役立つ。ここでは、基本認証を用いたHTTPS経由、ダイジェスト認証、ベアラートークン認証という、Propertyのアフォーダンスをアクティブ化する三つの可能な方法を示すTDの断片を示す。言い換えると、複数のフォーム内で様々なセキュリティ構成情報を用いると、セキュリティメカニズムを「OR」方式で組み合わせることができる。対照的に、同じ`security`メンバーに複数のセキュリティ構成情報を配置すると、それらは「AND」方式で結合される。これは、その場合には、相互作用のアフォーダンスのアクティブ化を可能にするために、それらすべてを満たす必要があるためである。Thingのレベルで一つの (デフォルトの) 構成をアクティブ化することは依然として必須であることに注意していただきたい。

例14

```
{
    ...
    "securityDefinitions": {
        "basic_sc": { "scheme": "basic" },
        "digest_sc": { "scheme": "digest" },
        "bearer_sc": { "scheme": "bearer" }
    },
    "security": ["basic_sc"] ,
    ...
    "properties": {
        "status": {
            ...
            "forms": [{
                "href": "https://mylamp.example.com/status"
            }, {
                "href": "https://mylamp.example.com/status",
                "security": ["digest_sc"]
            }, {
                "href": "https://mylamp.example.com/status",
                "security": ["bearer_sc"]
            }]
        }
    },
    ...
}
```

別のより複雑な例として、OAuth2は範囲を用いる。これはトークンに現れる識別子であり、そのリソース (または、W3C WoTの場合は相互作用のアフォーダンス) 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。例えば、下記では、`limited`という範囲を含むベアラートークンを用いてConsumerが`status` Propertyを読むことができるが、`configure` Actionは、`special`の範囲を含むトークンでのみ呼び出すことができる。範囲は役割と同一ではないが、多くの場合それに関連付けられている。例えば、おそらく管理的な役割を持つもののみが「特別な」相互作用を実行する認可を与えられる。トークンは複数のスコープを持つことができる。この例では、管理者にはおそらく`limited`と`special`の両方の範囲を持つトークンが発行されるが、通常のユーザには`limited`の範囲のトークンのみが発行される。

例15

```
{
    ...
    "securityDefinitions": {
        "oauth2_sc": {
            "scheme": "oauth2",
            ...
            "flow": "code",
            "authorization": "https://example.com/authorization",
            "token": "https://example.com/token",
            "scopes": ["limited", "special"]
        }
    },
    "security": ["oauth2_sc"] ,
    ...
    "properties": {
        "status": {
            ...
            "forms": [{
                "href": "https://scopes.example.com/status",
                "scopes": ["limited"]
            }]
        }
    },
    "actions": {
        "configure": {
            ...
            "forms": [{
                "href": "https://scopes.example.com/configure",
                "scopes": ["special"]
            }]
        }
    },
    ...
}
```

6.3.5 `properties` (プロパティー) §

`Thing`のインスタンスの`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すべての名前-値のペアは、マップのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。ペアの名前はJSON文字列としてシリアライズしなければならず (*MUST*)、ペアの値 (`PropertyAffordance`のインスタンス) はJSONオブジェクトとしてシリアライズしなければならない (*MUST*)。

`PropertyAffordance`のインスタンスのすべての名前-値のペアは、名前が`PropertyAffordance`、`InteractionAffordance`、または`DataSchema`のシグネチャ (の一つ) に含まれている語彙用語である場合、語彙用語を名前として用いて、`PropertyAffordance`インスタンスのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。`DataSchema`インスタンスのシリアライズの詳細に関しては、§ 6.3.10 データスキーマを参照していただきたい。

`PropertyAffordance`のインスタンスの`forms`に割り当てられている値は、§ 6.3.9 `forms` (フォーム) で定義している一つ以上のJSONオブジェクトのシリアライゼーションを含むJSON配列としてシリアライズしなければならない (*MUST*)。

二つのPropertyアフォーダンスの断片を以下に示す。

例16: Propertyのシリアライゼーションのサンプル

```
...
"properties": {
    "on": {
        "type": "boolean",
        "forms": [...]
    },
    "status": {
        "type": "object",
        "properties": {
            "brightness": {
                "type": "number",
                "minimum": 0.0,
                "maximum": 100.0
            },
            "rgb": {
                "type": "array",
                "items" : {
                    "type" : "number",
                    "minimum": 0,
                    "maximum": 255
                },
                "minItems": 3,
                "maxItems": 3
            }
        },
        "required": ["brightness", "rgb"] ,
        "forms": [...]
    }
},
...
```

6.3.6 `actions` (アクション) §

`Thing`のインスタンスでは、`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すべての名前-値のペアは、マップのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。ペアの名前はJSON文字列としてシリアライズしなければならず (*MUST*)、ペアの値 (`ActionAffordance`のインスタンス) はJSONオブジェクトとしてシリアライズしなければならない (*MUST*)。

`ActionAffordance`のインスタンスのすべての名前-値のペアは、名前が`ActionAffordance`または`InteractionAffordance`のシグネチャ (の一つ) に含まれている語彙用語である場合、語彙用語を名前として用いて、`ActionAffordance`インスタンスのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。

`ActionAffordance`のインスタンス内で`input`および`output`に割り当てられている値は、JSONオブジェクトとしてシリアライズしなければならない (*MUST*)。それらは、クラス`DataSchema`に依拠しており、そのシリアライゼーションは§ 6.3.10 データスキーマで定義している。

`ActionAffordance`のインスタンス内の`forms`に割り当てられている値は、§ 6.3.9 `forms` (フォーム) で定義している一つ以上のJSONオブジェクトのシリアライゼーションを含むJSON配列としてシリアライズしなければならない (*MUST*)。

Actionのアフォーダンスの TDの断片を以下に示す。

例17: Actionのシリアライゼーションのサンプル

```
...
"actions": {
    "fade" : {
        "title": "Fade in/out",
        "description": "Smooth fade in and out animation.",
        "input": {
            "type": "object",
            "properties": {
                "from": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                },
                "to": {
                    "type": "integer",
                    "minimum": 0,
                    "maximum": 100
                },
                "duration": {"type": "number"}
            },
            "required": ["to","duration"] ,
        },
        "output": {"type": "string"},
        "forms": [...]
    }
},
...
```

6.3.7 `events` (イベント) §

`Thing`のインスタンスでは、`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すべての名前-値のペアは、マップのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。ペアの名前はJSON文字列としてシリアライズしなければならず (*MUST*)、ペアの値 (`EventAffordance`のインスタンス) はJSONオブジェクトとしてシリアライズしなければならない (*MUST*)。

`EventAffordance`のインスタンスのすべての名前-値のペアは、名前が`EventAffordance`または`InteractionAffordance`のシグネチャ (の一つ) に含まれている語彙用語である場合、語彙用語を名前として用いて、`EventAffordance`インスタンスのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。

`EventAffordance`のインスタンス内で`subscription`、`data`、そして`cancellation`に割り当てられている値は、JSONオブジェクトとしてシリアライズしなければならない (*MUST*)。それらは、クラス`DataSchema`に依拠しており、そのシリアライゼーションは§ 6.3.10 データスキーマで定義している。

`EventAffordance`のインスタンス内の`forms`に割り当てられている値は、§ 6.3.9 `forms` (フォーム) で定義している一つ以上のJSONオブジェクトのシリアライゼーションを含むJSON配列としてシリアライズしなければならない (*MUST*)。

Eventオブジェクトの TDの断片を以下に示す。

例18: Eventのシリアライゼーションのサンプル

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

既存の (例えば、WebSub [websub]) または顧客指向の出来事メカニズム (例えば、Webhooks) を採用するために、Eventのアフォーダンスは柔軟な方法で定義されている。このため、`subscription`と`cancellation`は、望ましいメカニズムに従って定義できる。詳細に関しては、 [WOT-BINDING-TEMPLATES] を参照していただきたい。§ A.3 Webhook Eventの例の例は、Eventが`subscription`と`cancellation`を用いてWebhookを記述する方法を示している。

6.3.8 `links` (リンク) §

`Link`のインスタンスのすべての名前-値のペアは、名前が`Link`のシグネチャに含まれている語彙用語である場合、語彙用語を名前として用いて、`Link`インスタンスのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。

`links`配列内のリンクオブジェクトのTDの断片を以下に示す。

例19: リンクのシリアライゼーションのサンプル

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

6.3.9 `forms` (フォーム) §

`Form`のインスタンスのすべての名前-値のペアは、名前が`Form`のシグネチャに含まれている語彙用語である場合、語彙用語を名前として用いて、`Form`インスタンスのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。

必要に応じて、フォームオブジェクトは、接頭辞で識別されるプロトコル固有の語彙用語で補完できる (*MAY*)。§ 8.3 プロトコルバインディングも参照していただきたい。

`forms`配列内のフォームオブジェクトのTDの断片を以下に示す。

例20: フォームのシリアライゼーションのサンプル

```
...
"forms": [{
    "op": "writeproperty",
    "href" : "http://mytemp.example.com:5683/temp",
    "contentType": "application/json",
    "htv:methodName": "POST"
}]
...
```

`href`には、http://192.168.1.25/left?p=2&d=1のpやdなどの動的変数を含むURIを含めることもできる。その場合、URIは [RFC6570] で定義されているテンプレート (`http://192.168.1.25/left{?p,d}`) として定義できる。

このようなケースでは、URIテンプレート変数は、関連する (一意な) 変数名をJSON名として持つJSONオブジェクトベースの`uriVariables`メンバーで収集されなければならない (*MUST*)。

`Form`のインスタンス内の`uriVariables`に割り当てられているマップの各値のシリアライゼーションは、クラス`DataSchema`に依拠しなければならず (*MUST*)、そのシリアライゼーションは§ 6.3.10 データスキーマで定義している。

URIテンプレートと`uriVariables`を用いたTDの断片を以下に示す。

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:Some
                "d" : { "type": "integer", "minimum": 0, "maximum": 1, "@type": "eg:Direc
            },
            "forms": [{
              "href" : "http://192.168.1.25/left{?p,d}",
              "htv:methodName": "GET"
            }]
        },
        ...
    },
    ...
}
```

`contentType`メンバーは、`;`という文字で区切られた属性-値の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用いる。例は次のとおり。

例22

```
...
"contentType" : "text/plain; charset=utf-8",
...
```

ユースケースによっては、相互作用のアフォーダンスのフォームメタデータはリクエストを記述するだけでなく、予期される応答に関するメタデータも提供する。例えば、`takePhoto` Actionは、リクエストペイロードにJSONを用いて (つまり、`"contentType": "application/json"`)、カメラのパラメータ設定 (絞り優先、タイマーなど) を送信する`input`スキーマを定義しする。このアクションの出力は、撮影された写真であり、例えば、JPEG形式で利用可能である。このような場合、応答ペイロードの表現形式を示すために`response`メンバーが用いられる (例えば、`"contentType": "image/jpeg"`)。ここでは、コンテンツタイプで表現形式が完全に指定されるため、`output`スキーマは必要ない。

`Form`のインスタンス内の`response`に割り当てられている値は、存在している場合、JSONオブジェクトでなければならない (*MUST*)。応答オブジェクトには、存在している場合、`ExpectedResponse`のクラス定義で定義されている`contentType`メンバーが含まれていなければならない (*MUST*)。

上記の`takePhoto` Actionに基づいて、`response`メンバーを持つ`form`の断片を以下に示す。

例23

```
{
    ...
    "actions": {
        "takePhoto": {
            ...
            "forms": [{
                "op": "invokeaction",
                "href": "http://camera.example.com/api/snapshot",
                "contentType": "application/json",
                "response": {
                    "contentType": "image/jpeg"
                }
            }]
        }
    },
    ...
}
```

`forms`が最上レベルに存在している場合は、Thingが提供するメタ相互作用を記述するために使用できる。例えば、「readallproperties」および「writeallproperties」という操作型は、Consumerがすべてのプロパティーを一度に読み書きできるThingとのメタ相互作用のためのものである。下記の例では、`forms`メンバーがTDのルートオブジェクトに含まれており、Consumerは、送信ターゲット `https://mylamp.example.com/allproperties` を用いて、1回のプロトコルトランザクションでThingのすべてのProperty (つまり、`on`、`brightness`、`timer`) の読み書きの両方を行うことができる。

例24

```
{
    ...
    "properties": {
        "on": {
            "type": "boolean",
            "forms": [...]
        },
        "brightness": {
            "type": "number",
            "forms": [...]
        },
        "timer": {
            "type": "integer",
            "forms": [...]
        }
    },
    ...
    "forms": [{
        "op": "readallproperties",
        "href": "https://mylamp.example.com/allproperties",
        "contentType": "application/json",
        "htv:methodName": "GET"
    }, {
        "op": "writeallproperties",
        "href": "https://mylamp.example.com/allproperties",
        "contentType": "application/json",
        "htv:methodName": "PUT"
    }]
}
```

操作型`writeallproperties`の場合、Consumerがすべての書き込み可能な (`readOnly`でない) プロパティーと (新しい) 割り当てられた値 (例えば、ペイロード内) を提供することが予期される。同様に、`writemultipleproperties`操作型では、Consumerが書き込み可能な (`readOnly`ではない) プロパティーを提供

することが予期される。Thing側では、readmultiplepropertiesおよびreadallpropertiesの操作型の場合に、Thingは読み取り可能な (writeOnlyではない) プロパティーを返すことが予期される。

6.3.10 データスキーマ §

DataSchemaクラスを通じて定義されるWoT Thing Descriptionのデータスキーマは、JSONスキーマ用語 [JSON-SCHEMA] のサブセットに基づいている。したがって、TDデータスキーマのシリアライゼーションをJSONスキーマ検証ソフトの実装に直接フィードして、Thingと交換されるデータを検証できる。

データスキーマのシリアライゼーションは、PropertyAffordanceインスタンス、ActionAffordanceインスタンス内のinputおよびoutputに割り当てられている値、EventAffordanceインスタンス内のsubscription、data、cancellationに割り当てられている値、InteractionAffordanceのサブクラスのインスタンス内のuriVariablesに割り当てられている値 (フォームオブジェクトがURIテンプレートを用いる場合) に適用される。

DataSchemaのサブクラスのうちの一つのインスタンスのすべての名前-値のペアは、名前がそのサブクラスのシグネチャまたはDataSchemaのシグネチャに含まれている語彙用語である場合、語彙用語を名前として用いて、DataSchemaサブクラスのインスタンスのシリアライズから生じるJSONオブジェクトのメンバーとしてシリアライズしなければならない (*MUST*)。

ObjectSchemaのインスタンス内のpropertiesに割り当てられている値は、JSONオブジェクトとしてシリアライズしなければならない (*MUST*)。

DataSchemaのインスタンス内のenum、required、oneOfに割り当てられている値は、JSON配列としてシリアライズしなければならない (*MUST*)。

ArraySchemaのインスタンス内のitemsに割り当てられている値は、JSONオブジェクトまたはJSONオブジェクトを含んだJSON配列としてシリアライズしなければならない (*MUST*)。

TDの断片の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およびoutput用) またはPropertyオブジェクトであり、追加のメンバーが含まれていることに注意していただきたい。

例25: DataSchemaのシリアライゼーションのサンプル

```
...
"type": "object",
"properties": {
    "status": {
        "title": "Status",
        "type": "string",
        "enum": ["On", "Off", "Error"]
    },
    "brightness": {
        "title": "Brightness value",
        "type": "number",
        "minimum": 0.0,
        "maximum": 100.0
    },
    "rgb": {
        "title": "RGB color value",
        "type": "array",
        "items" : {
            "type" : "number",
            "minimum": 0,
            "maximum": 255
        },
        "minItems": 3,
        "maxItems": 3
    }
},
...
```

`readOnly`および`writeOnly`という用語は、読み取りの相互作用 (つまり、Propertyの読み取り時) で交換されるデータ項目と、書き込みの相互作用 (つまり、Propertyの書き込み時) で交換されるデータ項目を示すために使用できる。これは、規定的ではないThingのPropertyが読み取りと書き込みで異なるデータを示す場合の回避策として使用でき、それは、Thing Descriptionで既存のデバイスまたはサービスを拡張する場合に当てはまる。

`readOnly`と`writeOnly`を用いたTDの断片を以下に示す。

例26

```
...
"properties": {
    "status": {
        "description": "Read or write On/Off status.",
        "type": "object",
        "properties": {
            "latestStatus": {
                "type": "string",
                "enum": ["On", "Off"] ,
                "readOnly": true
            },
            "newStatusValue": {
                "type": "string",
                "enum": ["On", "Off"] ,
                "writeOnly": true
            }
        },
        forms: [...]
    }
}
...
```

`status` Propertyが読み取られると、ペイロードの`latestStatus`メンバーを用いて状態のデータが返される。`status` Propertyを更新するには、ペイロードの`newStatusValue`メンバーを通じて新しい値を提供しなければならない。

追加機能として、Thing Descriptionインスタンスにより、データスキーマ内で`unit`メンバーを使用できるようになる。これを使用して、測定単位をデータ項目に関連付けることができる。その文字列の値は自由に選択できる。しかし、よく知られている語彙で定義されている単位を選択することをお勧めする。例に関しては、§ 7. TDコンテキスト拡張を参照していただきたい。

6.4 識別 §

Thing DescriptionのJSONベースのシリアライゼーションは、`application/td+json`というメディアタイプまたは`432`というCoAPコンテンツ形式IDによって識別される (§ 10. IANAに関する留意点を参照)。

7. TDコンテキスト拡張 §

この項は非規範的である。

§ 5. TD情報モデルの標準的な語彙の定義に加えて、WoT Thing Descriptionは追加の名前空間からコンテキストの知識を追加する可能性を提供する。このメカニズムを用いて、追加の (例えば、ドメイン固有の) セマンティクスでThing Descriptionのインスタンスを充実させることができる。また、将来的に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も使用できる。

そのようなTDコンテキスト拡張の場合、Thing DescriptionはJSON-LD [json-ld11] で知られている`@context`メカニズムを用いる。TDコンテキスト拡張を用いる場合、`Thing`というクラスの`@context`の値は、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か、§ 5.3.1.1 `Thing`で定義している名前空間IRIを含んでいるマップである。

§ 6.1 JSONの型へのマッピングの複合型のシリアライゼーション規則は、拡張された`@context`名前-値のペアのシリアライゼーションを定義する。TDコンテキスト拡張の断片を以下に示す。

例27

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

7.1 セマンティックなアノテーション §

TDコンテキスト拡張により、Thing Descriptionインスタンスに語彙用語を追加できる。含まれている名前空間が、RDFスキーマやOWLによって提供されるものなどのクラス定義に基づいている場合、インスタンスをそのような外部のクラス定義に関連付けることにより、Thing Descriptionの任意のクラスのインスタンスにセマンティックにアノテーションを付与するためにそれを使用できる。これは、`@type`の名前-値のペアにクラス名を割り当てるか、複数の関連付け/アノテーションの配列の値にクラス名を含めることによって行われる。§ 6.1 JSONの型へのマッピングのシリアライゼーション規則に従って、`@type`はJSON文字列かJSON配列としてシリアライズされる。`@type`は、ノードの型を設定するために用いられるJSON-LDキーワード [json-ld11] である。

TDコンテキスト拡張により、Thing Descriptionのクラスのインスタンス内に追加の名前-値のペアと明確に定義された値を含めることもできる。このペアと値は、含まれている語彙用語を通じて定義され、それぞれ対応するJSONオブジェクトの追加メンバーまたは既存のメンバーの値としてシリアライズされる。例は、Thingの追加バージョンのメタデータまたはデータアイテムの測定単位である。

例として、以下に示すTDの断片は、Thingのハードウェアとファームウェアのバージョン番号を追加することでバージョン情報のコンテナを拡張し、Thingとデータスキーマの単位に外部語彙の値を用いている。SAREFは例2でも用いており、OMは測定単位のオントロジー [RIJGERSBERG] である。これらの語彙は例として用いている。 — 特にホームオートメーションの領域には他の語彙が存在している場合がある。

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.ontology-of-units-of-measure.org/resource/om-2/"
        }
    ] ,
    "version": {
        "instance": "1.2.1",
        "v:firmware": "0.9.1",
        "v:hardware": "1.0"
    },
    ...
    "@type": "saref:TemperatureSensor",
    "properties": {
        "temperature": {
            "description": "Temperature value of the weather station",
            "type": "number",
            "minimum": -32.5,
            "maximum": 55.2,
            "unit": "om:degree_Celsius",
            "forms": [...]
        },
        ...
    },
    ...
}
```

多くの場合、TDコンテキスト拡張を用いて、データスキーマの一部にアノテーションを付与し、物理世界のオブジェクトの状態情報をセマンティックに処理できる。これは、相互作用中に交換されるデータによって表される (例えば、応答のペイロードで)。例えば、RDFのこの状態情報のセマンティックな記述は、TDドキュメントに埋め込むことができ、データスキーマの部分は、物理世界のオブジェクトのRDFでモデル化された状態の特定の部分を参照するように個々にアノテーションを付与することができる。

以下のTDの断片では、SAREFを用いて照明の状態を記述している。SSN (セマンティックセンサーネットワークオントロジー [VOCAB-SSN]) から取得した外部の語彙用語である`ssn:forProperty`は、`status` Propertyのデータスキーマを物理世界の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用いられている。

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、Thingの状態は、`status`アフォーダンス自体によって示され、可能な状態の変化は`toggle`アフォーダンスによって示されている。言い換えれば、物理世界のオブジェクトの状態は、Thingの相互作用のアフォーダンスを直接提供する。この設計は、シンプルな場合には満足のいくものである。しかし、より複雑なケースでは、同じ物理的な状態に対していくつかのアフォーダンスが利用できる場合がありる。上記の例では、`fullStatus` Propertyで、照明の状態のより詳細な別の表現を提供している。

7.2 プロトコルバインディングの追加 §

Thing DescriptionでTDコンテキスト拡張を用いると、通信メタデータを補完したり、`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語彙用語を通じて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。 (§ 8.3 プロトコルバインディングも参照)。

この仕様の執筆時点では、そのようなプロトコルバインディングは利用できないため、次のTDの例では、架空のCoAPのプロトコルバインディングを用いている。このTDコンテキスト拡張は、例の名前空間`http://www.example.org/coap-binding#`からアクセスできる*RDF 1.0のHTTP語彙* [HTTP-in-RDF10] に似た*RDF語彙のCoAP*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メソッドを適用しなければならないかをConsumerに指示する (例えば、CoAPメソッドコード0.01には`GET`、CoAPメソッドコード0.02には`POST`、またはCoAPメソッドコード0.07には`iPATCH`)。

例30: TDコンテキスト拡張を介したフォームの特殊化

```
{
    "@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,
            "maximum": 64,
            "forms": [{
                "op": "readproperty",
                "href": "coap://example.org:61616/api/brightness",
                "cov:methodName": "GET"
            }, {
                "op": "writeproperty",
                "href": "coap://example.org:61616/api/brightness",
                "cov:methodName": "POST"
            }]
        },
        ...
    },
    ...
}
```

7.3 セキュリティスキームの追加 §

最後に、§ 5.3.3 セキュリティ語彙の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コンテキスト拡張のメカニズムを用いてインポートできる。この例では、 [ACE] に基づく架空のACEセキュリティスキームを用いており、それは、この例では、`http://www.example.org/ace-security#`の名前空間で定義されている。このような追加のセキュリティスキームは、`SecurityScheme`というクラスのサブクラスでなければならないことに注意していただきたい。

例31

```
{
    @context: [
        "https://www.w3.org/2019/wot/td/v1",
        {
            "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"]
            }]
        }
    },
    "actions": {
        "configure": {
            ...
            "forms": [{
                "op": "invokeaction",
                "href": "coaps://rs.example.com/configure",
                "contentType": "application/cbor",
                "cov:methodName": "POST",
                "ace:scopes": ["special"]
            }]
        }
    },
    ...
}
```

§ 5.3.3 セキュリティ語彙の定義で定義しているすべてのセキュリティスキームは既にTDコンテキストの一部であり、TDコンテキスト拡張を介して含める必要はない。

8. 動作の言明 §

以下の言明は、TDの表現や情報モデルとは対照的に、WoTシステムの構成要素の動作に関連している。しかし、TDは記述的であり、特に既存のネットワークインターフェースを記述するために用いられる場合があることに注意していただきたい。これらの場合、そのような既存のインターフェースの動作を制限する言明を作成することはできない。代わりに、言明は、そのようなインターフェースを正確に表すためのTDの制約と解釈されなければならない。

8.1 セキュリティ構成情報 §

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

- Thingが相互作用のために特定のアクセスメカニズムを必要とする場合、Thing Descriptionのセキュリティ構成情報でそのメカニズムを指定しなければならない (*MUST*)。
- Thingが相互作用のために特定のアクセスメカニズムを必要としない場合、Thing Descriptionのセキュリティ構成情報でそのメカニズムを指定してはならない (*MUST NOT*)。

一部のセキュリティプロトコルでは、必要なエンコーディングや暗号化スキームなど、認証情報を動的に要求する場合がある。上記の結果の一つは、プロトコルが、Thing Descriptionで宣言されていないセキュリティ認証情報の形式、またはエンコーディングや暗号化スキームを要求する場合、Thing Descriptionは無効と見なされるということである。

8.2 データスキーマ §

TDで提供されるデータスキーマは、TDで指定されている相互作用で記述されているThingによって返され、受け入れられるデータペイロードを正確に表すべきである。一般的に、ConsumerはWoT Thing Descriptionで示されていないものは生成せず、データスキーマに厳密に従うべきだが、WoT Thing Descriptionで明示的に指定されていないThingからの追加データは受け入れるべきである。一般的に、ThingはWoT Thing Descriptionによって*記述*されるが、ConsumerはThingと相互作用するときにWoT Thing Descriptionに従うように*制約*される。

- WoT Thing Descriptionで記述されている別のターゲットのThingと相互作用するときにConsumerの役割を果たすThingは、対応する相互作用で指定されている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 (*MUST*)。
- WoT Thing Descriptionは、個々の相互作用によって返され、受け入れられるデータを正確に記述しなければならない*MUST*)。
- Thingは、相互作用から追加のデータを、そのようなデータがWoT Thing Descriptionで示されているデータスキーマに記述されていない場合でも返すことができる (*MAY*)。これは、返されるデータに追加のプロパティーやアイテムがある可能性がある`ObjectSchema`と`ArraySchema` (`items`が`DataSchema`の配列である場合) に適用される。これは、 [JSON-SCHEMA] で定義されている`"additionalProperties":true`や`"additionalItems":true`であるかのように動作する。
- 別のThingと相互作用するときにConsumerの役割を果たすThingは、ターゲットのThingのThing Descriptionで示されているデータスキーマに記述されていない追加データをエラーなしに受け入れなければならない (*MUST*)。これは、返されるデータに追加のプロパティーやアイテムがある可能性がある`ObjectSchema`と`ArraySchema` (`items`が`DataSchema`の配列である場合) に適用される。これは、 [JSON-SCHEMA] で定義されている`"additionalProperties":true`や`"additionalItems":true`であるかのように動作する。
- 別のThingと相互作用するときにConsumerの役割を果たすThingは、そのThingのThing Descriptionで示されている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 (*MUST NOT*)。
- 別のThingと相互作用するときにConsumerの役割を果たすThingは、ターゲットのThingのThing Descriptionで示されているURIテンプレート、基底URI、およびフォームのhrefパラメータに従ってURIを生成しなければならない (*MUST*)。
- WoT Thing DescriptionのURIテンプレート、基底URI、およびhrefメンバーは、Thing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 (*MUST*)。

8.3 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、相互作用の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なメッセージへのマッピングである。相互作用のアフォーダンスのプロトコルバインディングは、§ 6.3.9 `forms` (フォーム) で定義している`forms`としてシリアライズされる。

WoT Thing Descriptionのすべてのフォームには、`href`メンバーで示される送信ターゲットがなければならない。この送信ターゲットのURIスキームは、Thingがどのようなプロトコルバインディングを実装しているかを示す [WOT-ARCHITECTURE] 。例えば、ターゲットが`http`または`https`で始まっていれば、Consumerは、ThingがHTTPに基づくプロトコルバインディングを実装していることを推測でき、フォームのインスタンスにおいてHTTP固有の用語を予期すべきである (次の項、§ 8.3.1 HTTPに基づくプロトコルバインディングを参照)。

- WoT Thing Descriptionのすべてのフォームは、その`href`メンバーのURIスキームによって示される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 (*MUST*)。
- WoT Thing Descriptionのすべてのフォームは、相互作用でThingが受け入れるリクエスト (リクエストヘッダーが存在する場合は、それを含む) を正確に記述しなければならない (*MUST*)。

8.3.1 HTTPに基づくプロトコルバインディング §

デフォルトでは、Thing Descriptionは、*RDF 1.0のHTTP語彙* [HTTP-in-RDF10] からのHTTP RDF語彙の定義を含めることにより、HTTPに基づくプロトコルバインディングをサポートする。この語彙は、`http://www.w3.org/2011/http#`を指し示す接頭辞である`htv`を用いることで、TDのインスタンス内で直接使用できる。HTTPに基づくプロトコルバインディングの詳細は、 [WOT-BINDING-TEMPLATES] にある。

HTTPに基づくプロトコルバインディングを実装するThingと相互作用するために、Consumerは、フォームを送信するときにどのHTTPメソッドを用いるべきかを知る必要がある。一般的なケースでは、Thing Descriptionには、メソッドを示す用語、つまり`htv:methodName`を明示的に含めることができる。簡潔にするために、HTTPに基づくプロトコルバインディングは、下記の操作型のデフォルト値を定義し、これは、Thingが期待するメソッド (例えば、読み取りにGET、書き込みにPUT) の収束を目的としている。HTTPに基づくプロトコルバインディングを表している形式でメソッドが示されていない場合、次の表で示しているデフォルト値を想定しなければならない (*MUST*)。

語彙用語	*デフォルト値*	*コンテキスト*
`htv:methodName`	`GET`	操作型が`readproperty`、`readallproperties`、`readmultipleproperties`の`Form`
`htv:methodName`	`PUT`	操作型が`writeproperty`、`writeallpropertieswritemultipleproperties`の`Form`
`htv:methodName`	`POST`	操作型が`invokeaction`の`Form`

例えば、§ 1. はじめにの例1では、フォームに操作型とHTTPメソッドが含まれていない。例1のフォームでは、次のデフォルト値を想定すべきである。

例32

☑ *HTTPに基づくプロトコルバインディングで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": {
        "basic_sc": {
            "scheme": "basic",
            "in": "header"
        }
    },
    "security": [
        "basic_sc"
    ] ,
    "properties": {
        "status": {
            "type": "string",
            "forms": [
                {
                    "op": "readproperty",
                    "href": "https://mylamp.example.com/status",
                    "htv:methodName": "GET"
                },
                {
                    "op": "writeproperty",
                    "href": "https://mylamp.example.com/status",
                    "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",
                    "subprotocol": "longpoll"
                }
            ]
        }
    }
}
```

8.3.2 その他のプロトコルバインディング §

Thingが実装できるプロトコルバインディングの数には制限はない。その他のプロトコルバインディング (例えば、CoAP、MQTT、またはOPC UAなどの) は、*RDF 1.0のHTTP語彙* [HTTP-in-RDF10] に似たプロトコル語彙やデフォルト値の定義が含まれている仕様などの別の文書で標準化される予定である。このようなプロトコルは、TDコンテキスト拡張のメカニズムを用いることで、簡単にTDに統合できる (§ 7. TDコンテキスト拡張を参照)。

IoTプラットフォームとエコシステムを記述する方法については、 [WOT-BINDING-TEMPLATES] を参照していただきたい。

9. セキュリティとプライバシーに関する留意点 §

この項は非規範的である。

一般的に、WoTシステムを保護するために講じられるセキュリティ対策は、システムが直面する可能性のある脅威と攻撃者、および保護する必要がある資産の価値に依存する。さらに、プライバシーのリスクは、Thingと特定可能な人物との関連性、および直接的な情報とそのような関連性から入手可能な推測される情報の両方に依存する。様々な状況に適応できる脅威モデルを含む、Web of Thingsのセキュリティとプライバシーに関する留意点の詳細な議論は、参考情報のドキュメント [WOT-SECURITY-GUIDELINES] に記載されている。この項では、WoT Thing Descriptionに直接関連するセキュリティとプライバシーのリスクおよび可能な軽減策についてのみ論じる。

WoT Thing Description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。Thing Descriptionを既存のネットワークインターフェースに後付けする場合、ネットワークインターフェースのセキュリティステータスは変わらないと予期される。

WoT Thing Descriptionを用いると、次の項で示しているセキュリティとプライバシーのリスクが生じる。個々のリスクの後で、いくつかの可能な軽減策を提案する。

9.1 プライバシーのリスクをフェッチするコンテキスト §

JSON-LD [json-ld11] ドキュメントの`@context`メンバーで指定されている語彙ファイルをフェッチすると、プライバシーのリスクになりえる。WoTの場合、攻撃者はそのようなフェッチによって生成される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語彙が用いられている場合に、デバイスに関する情報を推測するために、宛先IPアドレスなどのフェッチのメタデータを使用できる。これは、接続が暗号化されていてもリスクであり、DNSプライバシーのリークに関連している。

軽減策:

語彙ファイルの実際のフェッチを回避する。語彙ファイルは可能な限りキャッシュすべきである。`@context`メンバーのURIが (既知の) 語彙の識別子としてのみ機能するようにして、をれを変更不可能にして、解釈デバイスに組み込み、完全にフェッチされなくするのが理想的である。そのためには、既存のURIで変更不可能なデータを参照できるように更新時に新しいURIを用いるべきであるため、厳密なバージョン管理を用いる必要がある。Thing Descriptionのメタデータを解釈するシステムがコンテキストファイルをローカルで利用できる可能性を高めるために、可能な限り有名な標準語彙ファイルを使用する。

9.2 不変な識別子に関するプライバシーのリスク §

識別子 (`id`) を含んでいるThing Descriptionは、特定可能な人物に関連付けられているThingを記述することができる。このような識別子は、追跡を含む様々なリスクをもたらす。しかし、識別子も変更不可能である場合、デバイスが別の人に販売または譲渡され、その人を追跡するために既知のIDが用いられるため、追跡のリスクが増幅される。

軽減策:

すべての識別子は変更可能であるべきであり、Thingの`id`を更新するメカニズムがあるべきである。具体的には、Thingの`id`はハードウェア内で固定されるべきではない。しかし、これは、識別子が固定URIであるというリンクトデータの理想と矛盾する。多くの場合、Thingが再初期化された場合にのみ識別子の更新を認めることは許容されるだろう。このケースでは、ソフトウェアエンティティーとしての古いThingは存在しなくなり、新しいThingが作成される。これは、例えば、デバイスが新しい所有者に販売されたときに追跡の連鎖を断ち切るのに十分でありえる。あるいは、デバイスの運用段階の間により頻繁な変更が必要な場合は、変更時に、認可されたユーザのみに識別子の変更を通知するメカニズムを導入できる。しかし、一部

のクラスのデバイス、例えば、医療デバイスでは、一部の法的管轄区域において法律によって変更不可能なIDが求められる場合がある。この場合、そのような変更不可能な識別子を含むThing Descriptionなどのファイルへのアクセスを保護するために特別な注意を払うべきである。このような場合、可能な限り、TDで「真の」変更不可能な識別子を共有しないことが望ましいかもしれない。

9.3 指紋に関するプライバシーのリスク §

上記のように、TDの`id`メンバーはプライバシーのリスクを引き起こす可能性がある。しかし、前述のように`id`を更新して追跡のリスクを軽減しても、TDを特定の物理デバイスに関連付け、そこから指紋採取により特定可能な人物に関連付けることができる場合がある。

指紋採取では特定のデバイスのインスタンスを特定できない場合でも、相互作用の集合など、TDの情報からデバイスの類型を推測し、この類型を用いて、病状などの特定可能な人物に関する個人情報を推測することができる。

軽減策:

Thingに対するThing Descriptionへのアクセスは、認可されたユーザのみに提供すべきであり、認可のレベルとユースケースに必要な量の情報のみを提供すべきである。例えば、認証が必要なディレクトリサービスなどの安全で機密性のあるチャネルを介して、TDが認可されたユーザにのみ配信されていれば、外部の認可されていない当事者がTDにアクセスして指紋を取得することはできない。このリスクをさらに軽減するためには、TDの特定のユースケースに不要な情報は可能な限り除外すべきである。例えば、ConsumerがThingに関する状態を保存しない場合のデバイスへのアドホックな接続では、`id`は除外できる。Consumerがユースケースで特定の相互作用を必要としない場合、それは除外できる。Consumerが特定の相互作用の使用を認可されていなければ、同様に除外できる。タイトルや内容記述などの人間が読める情報を表示する機能をConsumerが持っていなければ、それらを除外したり、長さがゼロの文字列に置き換えることができる。

9.4 グローバルに一意な識別子に関するプライバシーのリスク §

グローバルに一意な識別子は、その作成と配信に中央機関が必要な場合、第三者が識別子を知っているため、プライバシーのリスクをもたらす。

軽減策:

TDの`id`フィールドは、意図的にグローバルに一意であることを求めていない。中央での登録を必要としない分散方式で適切なIDを生成するために利用可能な暗号化のメカニズムがいくつかある。通常、これらのメカニズムで重複する識別子が生成される可能性は非常に低いためシステム設計ではこれを考慮する必要がある (例えば、必要に応じて重複を検出し、IDを再生成する)。IDの範囲もグローバルである必要はない。家内や工場内などの特定のコンテキストでのみThingを区別する識別子を用いることもできる。

9.5 TDの傍受と改ざんに関するセキュリティのリスク §

例えば、TD内のURLを書き換えて、データをキャプチャまたは操作できる悪意のある仲介者にアクセスをリダイレクトすることにより、TDの傍受と改ざんは、中間者攻撃を開始するために使用できる。

軽減策:

相互に認証されたチャネルを通じてのみThing Descriptionを取得する。これにより、Consumerとサーバーの両方が通信相手の身元を確実に把握できる。これは、認可されたユーザにのみTDを配信するためにも必要である。

9.6 コンテキストの傍受と改ざんに関するセキュリティのリスク §

コンテキストファイルの傍受および改ざんは、語彙の解釈を変更することにより攻撃を容易にするために使用できる。

軽減策:

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のが理想的であるが、逆参照された場合に傍受と変更に対して脆弱であるHTTP URLを用いて多くのコンテキストが示されることは注意に値する (そして残念である)。しかし、コンテキストファイルが変更不可能かつキャッシュされ、可能な限り逆参照が回避されれば、このリスクは低減できる。

9.7 個人識別可能情報の推測に関するプライバシーのリスク §

多くの場所では、ユーザのプライバシーを保護するために、個人識別可能情報、つまり特定の人物に関連付けることができる情報の取り扱いに法的要件がある。もちろん、そのような情報はIoTデバイスによって直接生成される可能性がある。しかし、IoTデバイスの存在とメタデータ (Thing Descriptionに保存される種類のデータ) は、個人識別可能情報を含めたり、推測するために利用されたりする可能性もある。この情報は、特定の人が特定の種類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る場合があるが、それがその人物に関する追加の推論につながる可能性がある。

軽減策:

個人のデバイスに関連付けられたThing Descriptionを、個人識別可能情報が含まれているかのように扱う。この原則の適用事例として、ユーザの同意を得る方法を検討する。Thingが生成する個人を特定できるデータの使用に対する同意は、Thingとデータを利用するシステムとを組み合わせる際にしばしば得られる。これは、デバイスにアクセスするために、Thing Descriptionがローカルディレクトリや Thing Descriptionを利用するシステムに登録される際にも多い。このケースでは、Thingのデータの利用に対する同意は、Thingの Thing Descriptionへのアクセスに関する同意と組み合わせることができる。2番目の例として、TDに個人識別可能情報を含むことを検討する場合には、TDを無期限に保持したり、同意を得られた以外の目的で使用したりすべきではない。

10. IANAに関する留意点 §

10.1 `application/td+json`メディアタイプの登録 §

タイプ名:
application

サブタイプ名:
td+json

必須パラメータ:
なし

任意のパラメータ:
なし

コード化に関する留意点:
RFC 6839の3.1項を参照。

セキュリティに関する留意点:
RFC 8259の12項を参照。

WoT Thing DescriptionはThingのメタデータの純粋なデータ交換形式であることを目指しているため、そのシリアライゼーションは、解析されるJavaScriptの`eval()`関数などのコード実行メカニズムを介して渡されるべきではない (*SHOULD NOT*)。 (不正な) ドキュメントには、実行時にシステムのセキュリティを危険にさらす予期しない副作用を引き起こす可能性のあるコードが含まれている場合がある。

WoT Thing DescriptionはJSON-LD 1.1プロセッサで評価できる。これは通常、遠隔のコンテキスト (つまり、TDコンテキスト拡張。W3C WoT Thing Descriptionの7項を参照) へのリンクを自動的にたどり、その結果、個々に対するConsumerの明示的なリクエストがなくてもファイルが転送される。第三者が遠隔のコンテキストを提供している場合、それによって彼らがプライバシー上の懸念につながる使用パターンや同様の情報を収集できる場合がある。リソースに制約のあるデバイスでの実装では、(JSON-LD処理とは対照的に) 未加工のJSONの処理を実行すると思われるが、一般的に実装は、サポートしているコンテキスト拡張の検査済みのバージョンを静的にキャッシュし、遠隔のコンテキストへのリンクをたどらないようにすべきである (*SHOULD*)。代わりに、サポートしているコンテキスト拡張は、安全なソフトウェア更新メカニズムにより管理できる。

HTTPなどの安全でない接続を通じてウェブから読み込まれたコンテキスト拡張 (W3C WoT Thing Descriptionの7項を参照) には、セキュリティを侵害しえる方法でTD情報モデルを変更するといった、攻撃者による変更が行われるリスクがある。このため、Consumerは、システムによる利用を許可する前に、遠隔のコンテキストを再検査してキャッシュすべきである (*SHOULD*)。

JSON-LD処理には通常、長いIRI [RFC3987] の短い用語への置換えが含まれることを考慮すると、JSON-LD 1.1プロセッサを用いて処理するとWoT Thing Descriptionが大幅に拡張され、最悪の場合、結果のデータによ

って受信者のリソースがすべて消費される可能性がある。Consumerは、TDメタデータを懐疑的に扱うべきである (*SHOULD*)。

互換性に関する留意点:
RFC 8259を参照。

適合と不適合の両方のコンテンツを処理するための規則は、この仕様で定義している。

公開済み仕様書:
https://w3c.github.io/wot-thing-description

このメディアタイプを使用するアプリケーション:
W3C Web of Thingsに関与しているすべてのエンティティー、つまり、Web of Things (WoT) アーキテクチャで定義されているThing、Consumer、およびIntermediary。

フラグメント識別子に関する留意点:
RFC 6839の3.1項を参照。

追加情報:

マジックナンバー:
該当しない

ファイル拡張子:
.jsontd

マッキントッシュファイルタイプコード:
TEXT

詳細情報に関する連絡先:
Matthias Kovatsch <w3c@kovatsch.net>

意図する使途:
汎用

使用上の制限:
なし

著者:
WoT Thing Descriptionの仕様は、Web of Thingsワーキンググループの成果である。

改版管理者:
W3C

10.2 CoAPコンテンツ形式の登録 §

IANAは、Constrained RESTful Environments (CoRE) パラメータのレジストリ [RFC7252] 内のCoAPコンテンツ形式サブレジストリのメディアタイプにコンパクトなCoAPコンテンツ形式IDを割り当てている。WoT Thing Descriptionのコンテンツ形式IDは、432である。

メディアタイプ:
application/td+json

コード化:
-

ID:
432

参考文献:
["Web of Things (WoT) Thing Description", 2019年5月]

A. Thing Descriptionのインスタンスの例 §

この項は非規範的である。

A.1 CoAPプロトコルバインディングを用いたMyLampThingの例 §

Thingの機能リスト

- タイトル: MyLampThing
- コンテキスト拡張: なし
- 提供されるアフォーダンス: 1 Property、1 Action、1 Event
- セキュリティ: PSKSecurityScheme
- プロトコルバインディング: CoAP [RFC7252] over TLS
- コメント: § 7.2 プロトコルバインディングの追加も参照。

例33: CoAPプロトコルバインディングを用いたMyLampThing

```
{
   "@context": [
      "https://www.w3.org/2019/wot/td/v1",
      {
      "cov": "http://www.example.org/coap-binding#"
      }
    ] ,
    "id": "urn:dev:ops:32473-WoTLamp-1234",
    "title": "MyLampThing",
    "description" : "MyLampThing uses JSON serialization",
    "securityDefinitions": {"psk_sc":{"scheme": "psk"}},
    "security": ["psk_sc"] ,
    "properties": {
        "status": {
            "description" : "Shows the current status of the lamp",
            "type": "string",
            "forms": [{
                "op": "readproperty",
                "href": "coaps://mylamp.example.com/status",
                "cov:methodName" : "GET"
            }]
        }
    },
    "actions": {
        "toggle": {
            "description" : "Turn on or off the lamp",
            "forms": [{
                "href": "coaps://mylamp.example.com/toggle",
                "cov:methodName" : "POST"
            }]
        }
    },
    "events": {
        "overheating": {
            "description" : "Lamp reaches a critical temperature (overheating)",
            "data": {"type": "string"},
            "forms": [{
                "href": "coaps://mylamp.example.com/oh",
                "cov:methodName" : "GET",
                "subprotocol" : "cov:observe"
            }]
        }
    }
}
```

A.2 MQTTプロトコルバインディングを用いたMyIlluminanceSensorの例 §

Thingの機能リスト:

- タイトル: MyIlluminanceSensor
- コンテキスト拡張: なし

- 提供されるアフォーダンス: 1 Event
- 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ブローカーにより、照度データ (数値はテキスト形式でシリアライズされている) を`/illuminance`というトピックに頻繁に公開する。

例34: MQTTプロトコルバインディングを用いたMyIlluminanceSensor

```
{
    "@context": "https://www.w3.org/2019/wot/td/v1",
    "title": "MyIlluminanceSensor",
    "id": "urn:dev:ops:32473-WoTIlluminanceSensor-1234",
    "securityDefinitions": {"nosec_sc": {"scheme": "nosec"}},
    "security": ["nosec_sc"] ,
    "events": {
        "illuminance": {
            "data":{"type": "integer"},
            "forms": [
                {
                    "href": "mqtt://192.168.1.187:1883/illuminance",
                    "contentType" : "text/plain",
                    "op" : "subscribeevent"
                }
            ]
        }
    }
}
```

A.3 Webhook Eventの例 §

Thingの機能リスト:

- タイトル: WebhookThing
- コンテキスト拡張: HTTPプロトコルバインディングの補足を使用 (htv接頭辞は既にTDコンテキストに含まれている)
- 提供されるアフォーダンス: 1 Event
- セキュリティ: なし
- プロトコルバインディング: HTTP
- コメント: *WebhookThing*は、Webhookメカニズムを用いてConsumerに定期的に最新の温度値をプッシュするEventアフォーダンス`temperature`を提供し、Thingは、Consumerが提供するコールバックURIにPOSTリクエストを送信する。これを記述するために、`subscription`メンバーは、書き込み専用パラメータ`callbackURL`を定義し、`subscribeevent`フォームを介して送信しなければならない。読み取り専用パラメータ`subscriptionID`が登録によって返される。*WebhookThing*は、`data`によって定義されたペイロードを用いて、このコールバックURIに定期的にPOSTを行う。登録解除をするには、ConsumerはURIテンプレートを用いる`unsubscribeevent`フォームを送信する必要がある。`uriVariables`メンバーは、Consumerに`subscriptionID`文字列を含めるように通知する。これは、TDコンテキスト拡張を用いて適切なセマンティックアノテーションを含めることにより、さらに自動化できる。または、`subscription`と同様に`cancellation`メンバーを用いて登録解除をすることを想像し、これを、登録解除をするペイロードを持つPOSTリクエストを記述する`unsubscribeevent`フォームと組み合わせることができる。

例35: 登録と中止を伴う温度Event

```
{
    "@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": {
            "description": "Provides periodic temperature value updates.",
            "subscription": {
                "type": "object",
                "properties": {
                    "callbackURL": {
                        "type": "string",
                        "format": "uri",
                        "description": "Callback URL provided by subscriber for Webhook n
                        "writeOnly": true
                    },
                    "subscriptionID": {
                        "type": "string",
                        "description": "Unique subscription ID for cancellation provided
                        "readOnly": true
                    }
                }
            },
            "data": {
                "type": "number",
                "description": "Latest temperature value that is sent to the callback URL
            },
            "cancellation": {
                "type": "object",
                "properties": {
                    "subscriptionID": {
                        "type": "integer",
                        "description": "Required subscription ID to cancel subscription."
                        "writeOnly": true
                    }
                }
            },
            "uriVariables": {
                "subscriptionID": { "type": "string" }
            },
            "forms": [
                {
                    "op": "subscribeevent",
                    "href": "http://192.168.0.124:8080/events/temp/subscribe",
                    "contentType": "application/json",
                    "htv:methodName": "POST"
                },
                {
                    "op": "unsubscribeevent",
                    "href": "http://192.168.0.124:8080/events/temp/{subscriptionID}",
                    "htv:methodName": "DELETE"
                }
            ]
        }
    }
}
```

B. TDのインスタンス検証用JSONスキーマ §

この項は非規範的である。

下記は、JSONベースの形式でシリアライズされたThing Descriptionのインスタンスを構文的に検証するための JSONスキーマ [JSON-SCHEMA] ドキュメントである。

> 注
>
> この文書で定義しているThing Descriptionでは、JSON-LD [json-ld11] で知られている`@context`メカニズムを用いて外部語彙を追加でき、これらの外部語彙の用語は、§ 5. TD情報モデルで定義している用語に加えて使用できる。そのため、下記のJSONスキーマはその点で意図的に厳密ではない。外部語彙が用いられていない場合には、より厳密な検証を行うために、異なる範囲/レベルで`additionalProperties`スキーマプロパティーの値である`true`を`false`に置き換えることができる。

> 注
>
> 一部のJSONスキーマ検証ツールは、`iri`の文字列形式をサポートしていないことに注意していただきたい。

TDのインスタンスを検証するための次のJSONスキーマでは、デフォルト値を持つ用語が存在している必要はない。したがって、デフォルト値を持つ用語はオプションである。 (§ 5.4 デフォルト値の定義も参照)

```
{
   "title": "WoT TD Schema - 16 October 2019",
   "description": "JSON Schema for validating TD instances against the TD model. TD ins
   "$schema ": "http://json-schema.org/draft-07/schema#",
   "definitions": {
       "anyUri": {
           "type": "string",
           "format": "iri-reference"
       },
       "description": {
           "type": "string"
       },
       "descriptions": {
           "type": "object",
           "additionalProperties": {
               "type": "string"
           }
       },
       "title": {
           "type": "string"
       },
       "titles": {
           "type": "object",
           "additionalProperties": {
               "type": "string"
           }
       },
       "security": {
           "oneOf": [{
                   "type": "array",
                   "items": {
                       "type": "string"
                   }
               },
               {
                   "type": "string"
               }
           ]
       },
       "scopes": {
           "oneOf": [{
                   "type": "array",
                   "items": {
                       "type": "string"
                   }
               },
               {
                   "type": "string"
               }
           ]
       },
       "subprotocol": {
           "type": "string",
           "enum": [
               "longpoll",
               "websub",
               "sse"
           ]
       },
       "thing-context-w3c-uri": {
           "type": "string",
           "enum": [
               "https://www.w3.org/2019/wot/td/v1"
           ]
       },
       "thing-context": {
```

```
            "oneOf": [{
                    "type": "array",
                    "items": [{
                        "$ref": "#/definitions/thing-context-w3c-uri"
                    }] ,
                    "additionalItems": {
                        "anyOf": [{
                                "$ref": "#/definitions/anyUri"
                            },
                            {
                                "type": "object"
                            }
                        ]
                    }
                },
                {
                    "$ref": "#/definitions/thing-context-w3c-uri"
                }
            ]
        },
        "type_declaration": {
            "oneOf": [{
                    "type": "string"
                },
                {
                    "type": "array",
                    "items": {
                        "type": "string"
                    }
                }
            ]
        },
        "dataSchema": {
            "type": "object",
            "properties": {
                "@type": {
                    "$ref": "#/definitions/type_declaration"
                },
                "description": {
                    "$ref": "#/definitions/description"
                },
                "title": {
                    "$ref": "#/definitions/title"
                },
                "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"
                    }
                }
            }
        },
        "form_element_property": {
            "type": "object",
            "properties": {
                "op": {
                    "oneOf": [{
                            "type": "string",
                            "enum": [
                                "readproperty",
                                "writeproperty",
```

```
                            "observeproperty",
                            "unobserveproperty"
                        ]
                    },
                    {
                        "type": "array",
                        "items": {
                            "type": "string",
                            "enum": [
                                "readproperty",
                                "writeproperty",
                                "observeproperty",
                                "unobserveproperty"
                            ]
                        }
                    }
                ]
            },
            "href": {
                "$ref": "#/definitions/anyUri"
            },
            "contentType": {
                "type": "string"
            },
            "contentCoding": {
                "type": "string"
            },
            "subprotocol": {
                "$ref": "#/definitions/subprotocol"
            },
            "security": {
                "$ref": "#/definitions/security"
            },
            "scopes": {
                "$ref": "#/definitions/scopes"
            },
            "response": {
                "type": "object",
                "properties": {
                    "contentType": {
                        "type": "string"
                    }
                }
            }
        },
        "required": [
            "href"
        ] ,
        "additionalProperties": true
    },
    "form_element_action": {
        "type": "object",
        "properties": {
            "op": {
                "oneOf": [{
                        "type": "string",
                        "enum": [
                            "invokeaction"
                        ]
                    },
                    {
                        "type": "array",
                        "items": {
                            "type": "string",
                            "enum": [
                                "invokeaction"
                            ]
```

```
                        }
                    }
                ]
            },
            "href": {
                "$ref": "#/definitions/anyUri"
            },
            "contentType": {
                "type": "string"
            },
            "contentCoding": {
                "type": "string"
            },
            "subprotocol": {
                "$ref": "#/definitions/subprotocol"
            },
            "security": {
                "$ref": "#/definitions/security"
            },
            "scopes": {
                "$ref": "#/definitions/scopes"
            },
            "response": {
                "type": "object",
                "properties": {
                    "contentType": {
                        "type": "string"
                    }
                }
            }
        },
        "required": [
            "href"
        ] ,
        "additionalProperties": true
    },
    "form_element_event": {
        "type": "object",
        "properties": {
            "op": {
                "oneOf": [{
                        "type": "string",
                        "enum": [
                            "subscribeevent",
                            "unsubscribeevent"
                        ]
                    },
                    {
                        "type": "array",
                        "items": {
                            "type": "string",
                            "enum": [
                                "subscribeevent",
                                "unsubscribeevent"
                            ]
                        }
                    }
                ]
            },
            "href": {
                "$ref": "#/definitions/anyUri"
            },
            "contentType": {
                "type": "string"
            },
            "contentCoding": {
                "type": "string"
```

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

```
                    "$ref": "#/definitions/scopes"
                },
                "response": {
                    "type": "object",
                    "properties": {
                        "contentType": {
                            "type": "string"
                        }
                    }
                }
            },
            "required": [
                "href"
            ] ,
            "additionalProperties": true
        },
        "property_element": {
            "type": "object",
            "properties": {
                "@type": {
                    "$ref": "#/definitions/type_declaration"
                },
                "description": {
                    "$ref": "#/definitions/description"
                },
                "descriptions": {
                    "$ref": "#/definitions/descriptions"
                },
                "title": {
                    "$ref": "#/definitions/title"
                },
                "titles": {
                    "$ref": "#/definitions/titles"
                },
                "forms": {
                    "type": "array",
                    "minItems": 1,
                    "items": {
                        "$ref": "#/definitions/form_element_property"
                    }
                },
                "uriVariables": {
                    "type": "object",
                    "additionalProperties": {
                        "$ref": "#/definitions/dataSchema"
                    }
                },
                "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": {
                    "$ref": "#/definitions/dataSchema"
                }
            },
            "required": {
                "type": "array",
                "items": {
                    "type": "string"
                }
            }
        },
        "required": [
            "forms"
        ] ,
        "additionalProperties": true
    },
    "action_element": {
        "type": "object",
        "properties": {
            "@type": {
                "$ref": "#/definitions/type_declaration"
```

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

```
                }
            },
            "uriVariables": {
                "type": "object",
                "additionalProperties": {
                    "$ref": "#/definitions/dataSchema"
                }
            },
            "subscription": {
                "$ref": "#/definitions/dataSchema"
            },
            "data": {
                "$ref": "#/definitions/dataSchema"
            },
            "cancellation": {
                "$ref": "#/definitions/dataSchema"
            }
        },
        "required": [
            "forms"
        ] ,
        "additionalProperties": true
    },
    "link_element": {
        "type": "object",
        "properties": {
            "href": {
                "$ref": "#/definitions/anyUri"
            },
            "type": {
                "type": "string"
            },
            "rel": {
                "type": "string"
            },
            "anchor": {
                "$ref": "#/definitions/anyUri"
            }
        },
        "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": [
                                "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": [
                            "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": [
                        "bearer"
                    ]
                },
                "authorization": {
                    "$ref": "#/definitions/anyUri"
                },
                "alg": {
                    "type": "string"
                },
                "format": {
                    "type": "string"
                },
                "in": {
                    "type": "string",
                    "enum": [
                        "header",
                        "query",
                        "body",
                        "cookie"
                    ]
                },
                "name": {
                    "type": "string"
                }
            },
            "required": [
                "scheme"
            ]
        },
        {
            "type": "object",
            "properties": {
                "@type": {
                    "$ref": "#/definitions/type_declaration"
                },
                "description": {
                    "$ref": "#/definitions/description"
                },
                "descriptions": {
                    "$ref": "#/definitions/descriptions"
                },
                "proxy": {
                    "$ref": "#/definitions/anyUri"
                },
                "scheme": {
                    "type": "string",
                    "enum": [
                        "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": [
                                "oauth2"
                            ]
                        },
                        "authorization": {
                            "$ref": "#/definitions/anyUri"
                        },
                        "token": {
                            "$ref": "#/definitions/anyUri"
                        },
                        "refresh": {
                            "$ref": "#/definitions/anyUri"
                        },
                        "scopes": {
                            "oneOf": [{
                                    "type": "array",
                                    "items": {
                                        "type": "string"
                                    }
                                },
                                {
                                    "type": "string"
                                }
                            ]
                        },
                        "flow": {
                            "type": "string",
                            "enum": [
                                "code"
                            ]
                        }
                    },
                    "required": [
                        "scheme"
                    ]
                }
            ]
        }
    },
    "type": "object",
    "properties": {
        "id": {
            "type": "string",
            "format": "uri"
        },
        "title": {
            "$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",
            "format": "date-time"
        },
        "modified": {
```

```
            "type": "string",
            "format": "date-time"
        },
        "security": {
            "oneOf": [{
                    "type": "string"
                },
                {
                    "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. Thing Descriptionテンプレート §

この項は非規範的である。

*Thing Description*テンプレートは、Thingの*クラス*の記述である。これは、Thingのグループ全体で共有されるプロパティー、アクション、イベント、および共通のメタデータを記述し、それにより、クラウドサーバーが、Thingごとに行うのは現実的ではない、何千ものデバイスの共通処理を行うことができる。Thing Descriptionテンプレートは、§ 5. TD情報モデルと同じコア語彙と情報モデルを用いる。

Thing Descriptionテンプレートにより、次のことが可能になる。

- クラウドサービスによる複数のThingの管理。
- 未開発のデバイス/Thingのシミュレーション。
- Thingの共通するモデルを共有する様々なメーカーのデバイスにまたがる共通のアプリケーション。
- 複数のモデルを一つのThingに結合すること。

Thing Descriptionテンプレートは、インターフェースとデバイスとの可能な相互作用 (プロパティー、アクション、イベント) の論理的な記述であるが、シリアル番号、GPS上の位置、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のデバイス固有の情報は含まれていない。

Thing Descriptionテンプレートは、特定のエンドポイントに対する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ないため、*フォーム*と*securityDefinitions* (セキュリティ定義) および*セキュリティ*のキーが存在してはならない。

同じThing Descriptionテンプレートを複数のベンダーのThingに実装できる。Thingは複数のThing Descriptionテンプレートを実装し、追加のメタデータ (ベンダー、場所、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できる。共通するThingに結合される様々なThing Descriptionテンプレートのプロパティー、アクション、およびイベント間の衝突を避けるために、これらすべての識別子はThingの中で一意でなければならない。

あるクラスのデバイスに共通するThing Descriptionテンプレートを用いると、ベンダーにまたがるアプリケーションを記述でき、アプリケーション開発者にとってより魅力的な市場が作られる。具体的なThing Descriptionは、複数のThing Descriptionテンプレートを実装できるため、機能ブロックを一つの結合デバイスに統合することができる。

クラウドベンダーのビジネスモデルは通常、何千もの同じデバイスの管理に基づいて構築されている。同じThing Descriptionテンプレートを持つすべてのデバイスは、クラウドアプリケーションが同じ方法で管理できる。インターフェースとインスタンスを別々に扱うと、多数のシミュレートされたデバイスを簡単に作成できる。

Thing Descriptionテンプレートは、一部のオプションと必須の語彙用語が存在しないThing Descriptionのサブセットであるため、Thing Descriptionと同じ方法、同じ形式でシリアライズできる。Thingのテンプレートのインスタンスは、必須の用語が欠けているため、Thing Descriptionインスタンスと同じ方法で検証することはできないことに注意していただきたい。

C.1 Thing Descriptionテンプレートの例 §

この項では、照明のThing DescriptionテンプレートとブザーのThing Descriptionテンプレートを示す。

C.1.1 Thing Descriptionテンプレート: 照明 §

例36: JSONでシリアライズされたMyLampThingTemplate

```
{
    "@context": ["https://www.w3.org/2019/wot/td/v1"] ,
    "@type" : "ThingTemplate",
    "title": "Lamp Thing Description Template",
    "description" : "Lamp Thing Description Template",
    "properties": {
        "status": {
            "description" : "current status of the lamp (on|off)",
            "type": "string",
            "readOnly": true
        }
    },
    "actions": {
        "toggle": {
            "description" : "Turn the lamp on or off"
        }
    },
    "events": {
        "overheating": {
            "description" : "Lamp reaches a critical temperature (overheating)",
            "data": {"type": "string"}
        }
    }
}
```

C.1.2 Thing Descriptionテンプレート: ブザー §

例37: JSONでシリアライズされたMyBuzzerThingTemplate

```
{
    "@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 secon
     "actions": {
        "buzz": {
            "description" : "buzz for 10 seconds"
        }
     }
}
```

D. JSON-LDコンテキストの使用法 §

この項は非規範的である。

現在の仕様では、TD情報モデルを様々な語彙に対する制約の集合、つまり語彙用語の集合として導入している。この項では、TDドキュメントの必須の`@context`を用いて、これらの制約の機械可読な定義をクライアントアプリケーションに統合する方法について簡単に説明する。

TDドキュメントからTD情報モデルへのアクセスは、二つのステップで実行される。最初に、クライアントはJSON文字列からIRIへのマッピングを取得する必要がある。このマッピングは、後で説明するとおり、JSON-LDコンテキストとして定義される。次に、クライアントは、このIRIを逆参照することにより、これらのIRIで定義されている制約にアクセスできる。制約は、RDF形式の論理公理として定義され、クライアントプログラムが容易に解釈できる。

§ 5. TD情報モデルで参照しているすべての語彙用語は、TDドキュメント内の (コンパクトな) JSON文字列としてシリアライズされている。しかし、これらの各用語は、最初のリンクトデータ原則 [LINKED-DATA] に従って、完全なIRIによって明確に識別される。JSONキーからIRIへのマッピングは、TDの`@context`値が指し示すものである。例えば、次のファイル

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

には、次 (抜粋) のマッピングが含まれている。

```
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ファイルは、JSON-LD 1.1構文 [JSON-LD11] に従う。多数のJSON-LDライブラリがTDの`@context`を自動的に処理し、それに含まれているすべてのJSON文字列を展開できる。

TDのすべての語彙用語がIRIに展開されると、次のステップは、このIRIを逆参照して、その語彙用語を参照するTD情報モデルのフラグメントを取得することである。例えば、次のIRIを逆参照すると、

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

`ObjectSchema`という用語はクラスであり、より正確には`DataSchema`のサブクラスであると述べるRDFドキュメントが作成される。このような論理公理は、様々な複雑さの形式を用いてRDFで表される。ここでは、サブクラス関係はRDFスキーマ公理 [RDF-SCHEMA] として表している。さらに、これらの公理は様々な形式でシリアライズできる。ここでは、それらはTurtle形式 [TURTLE] でシリアライズしている。

```
<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> .
```

デフォルトでは、ユーザエージェントが内容交渉を実行しない場合は、RDFドキュメントの代わりに人間が読めるHTMLドキュメントが返される。内容交渉を行うためには、クライアントはリクエストにHTTPヘッダー `Accept: text/turtle`を含めなければならない。

E. 最近の仕様変更 §

E.1 勧告案からの変更 §

- IANAによってCoAPコンテンツ形式ID 432が割り当てられた。 (§ 6.4 識別および§ 10.2 CoAPコンテンツ形式の登録を参照)。
- § 8.1 セキュリティ構成情報の項では、一部のセキュリティプロトコルによって求められる動的な認証情報と、Thing Descriptionで宣言されるセキュリティ構成情報の間の相互作用について明確化した。
- 付録§ B. TDのインスタンス検証用JSONスキーマの項のいくつかの誤植のインスタンスを修正した。

E.2 第2勧告候補からの変更 §

- リスクのある機能としていた、`CertSecurityScheme`、`PublicSecurityScheme`、`PoPSecurityScheme`、および`OAuth2SecurityScheme`の`implicit`、`password`、`client`のフローを削除した。
- § 5.3.2 データスキーマ語彙の定義の項で、データスキーマとコンテンツタイプの関係を明確化した。
- § 6.3.9 `forms` (フォーム) の項で、操作型である`writemultipleproperties`、`readmultipleproperties`、`readallproperties`に対するConsumerとThingの予期に関して明確化した。
- § 8.3.1 HTTPに基づくプロトコルバインディングの項で、デフォルト値の`GET`または`PUT`が語彙用語である`htv:methodName`に用いられるコンテキストを明確化した。
- 付録§ A.2 MQTTプロトコルバインディングを用いたMyIlluminanceSensorの例の項のThing Descriptionの例を改善し、MQTTを用いたより良い例を作成した。

E.3 最初の勧告候補からの変更 §

最初の勧告候補からの変更点は、2番目の勧告候補で記述している。

F. 謝辞 §

編集者は、貢献、助言、専門知識の提供に対し、Michael Koster、Michael Lagally、Kazuyuki Ashimura、Ege Korkan、Daniel Peintner、Toru Kawaguchi、Maria Poveda、Dave Raggett、Kunihiko Toumura、Takeshi Yamada、Ben Francis、Manu Sporny、Klaus Hartke、Addison Phillips、Jose M. Cantera、Tomoaki Mizushima、Soumya Kanti Datta、Benjamin Klotzに謝意を表す。

また、この文書の改善につながったサポート、技術情報、提案に対し、W3Cのスタッフ、およびW3C Web of Things利害団体 (WoT IG) とワーキンググループ (WoT WG) の現在および以前のすべての関係者にも感謝する。

最後に、WoT IGの創設から2年にわたってリードし、Thing Descriptionを含むWoT構成要素の概念にグループを導いてくれたJoerg Heuerに特に感謝する。

G. 参考文献 §

G.1 規範的な参考文献 §

[BCP47]
Tags for Identifying Languages. A. Phillips; M. Davis. IETF. September 2009. IETF Best Current Practice. URL: https://tools.ietf.org/html/bcp47

[html]

HTML Standard. Anne van Kesteren; Domenic Denicola; Ian Hickson; Philip Jagenstedt; Simon Pieters. WHATWG. Living Standard. URL: https://html.spec.whatwg.org/multipage/

[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]
Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119

[RFC3339]
Date and Time on the Internet: Timestamps. G. Klyne; C. Newman. IETF. July 2002. Proposed Standard. URL: https://tools.ietf.org/html/rfc3339

[RFC3629]
UTF-8, a transformation format of ISO 10646. F. Yergeau. IETF. November 2003. Internet Standard. URL: https://tools.ietf.org/html/rfc3629

[RFC3986]
Uniform Resource Identifier (URI) : Generic Syntax. T. Berners-Lee; R. Fielding; L. Masinter. IETF. January 2005. Internet Standard. 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 Template. J. Gregorio; R. Fielding; M. Hadley; M. Nottingham; D. Orchard. IETF. March 2012. Proposed Standard. URL: https://tools.ietf.org/html/rfc6570

[RFC6749]
The OAuth 2.0 Authorization Framework. D. Hardt, Ed.. IETF. October 2012. Proposed Standard. 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. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7252

[RFC7516]
JSON Web Encryption (JWE) . M. Jones; J. Hildebrand. IETF. May 2015. Proposed Standard. URL: https://tools.ietf.org/html/rfc7516

[RFC7519]
JSON Web Token (JWT) . M. Jones; J. Bradley; N. Sakimura. IETF. May 2015. Proposed Standard. 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. February 2016. Proposed Standard. URL: https://tools.ietf.org/html/rfc7797

[RFC8174]
Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. Leiba. IETF. May 2017. Best Current Practice. URL: https://tools.ietf.org/html/rfc8174

[RFC8252]
OAuth 2.0 for Native Apps. W. Denniss; J. Bradley. IETF. October 2017. Best Current Practice. URL: https://tools.ietf.org/html/rfc8252

[RFC8259]
The JavaScript Object Notation (JSON) Data Interchange Format. T. Bray, Ed.. IETF. December 2017. Internet Standard. URL: https://tools.ietf.org/html/rfc8259

[RFC8288]
Web Linking. M. Nottingham. IETF. October 2017. Proposed Standard. URL: https://httpwg.org/specs/rfc8288.html

[RFC8392]
CBOR Web Token (CWT). M. Jones; E. Wahlstroem; S. Erdtman; H. Tschofenig. IETF. May 2018. Proposed Standard. 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/2018/REC-websub-20180123/

[XMLSCHEMA11-2-20120405]
W3C XML Schema Definition Language (XSD) 1.1 Part 2: Datatypes. David Peterson; Sandy Gao; Ashok Malhotra; Michael Sperberg-McQueen; Henry Thompson; Paul V. Biron et al. W3C. 5 April 2012. W3C Recommendation. URL: https://www.w3.org/TR/2012/REC-xmlschema11-2-20120405/

G.2 参考情報の参考文献 §

[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. 2 February 2017. W3C Note. URL: https://www.w3.org/TR/2017/NOTE-HTTP-in-RDF10-20170202/

[IANA-MEDIA-TYPES]
Media Types. 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; Dave Longley. W3C. 16 March 2020. W3C Candidate Recommendation. URL: https://www.w3.org/TR/2020/CR-json-ld11-20200316/

[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). Mark Davis; CLDR Contributors. URL: https://unicode.org/reports/tr35/

[LINKED-DATA]
Linked Data Design Issues. Tim Berners-Lee. W3C. 27 July 2006. W3C-Internal Document. URL: https://www.w3.org/DesignIssues/LinkedData.html

[MQTT]
MQTT Version 3.1.1. Andrew Banks; Rahul Gupta. OASIS. 10 December 2015. OASIS Standard Incorporating Approved Errata 01. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html

[OPENAPI]
OpenAPI Specification: Version 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. 7 December 2017. 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

[RFC7231]

Hypertext Transfer Protocol (HTTP/1.1) : Semantics and Content. R. Fielding, Ed.; J. Reschke, Ed.. IETF. June 2014. Proposed Standard. URL: https://httpwg.org/specs/rfc7231.html

[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]
Semantic Versioning 2.0.0. Tom Preston-Werner. 26 December 2017. URL: https://semver.org/

[SMARTM2M]
ETSI TS 103 264 V2.1.1 (2017-03) : SmartM2M; Smart Appliances; Reference Ontology and oneM2M Mapping. ETSI. March 2017. Published. URL: http://www.etsi.org/deliver/etsi_ts/103200_103299/103264/02.01.01_60/ts_103264v020101p.pdf

[string-meta]
Strings on the Web: Language and Direction Metadata. Addison Phillips; Richard Ishida. W3C. 11 June 2019. W3C Working Draft. URL: https://www.w3.org/TR/2019/WD-string-meta-20190611/

[TURTLE]
RDF 1.1 Turtle. Eric Prud'hommeaux; Gavin Carothers. W3C. 25 February 2014. W3C Recommendation. URL: http://www.w3.org/TR/2014/REC-turtle-20140225/

[VOCAB-SSN]
Semantic Sensor Network Ontology. Armin Haller; Krzysztof Janowicz; Simon Cox; Danh Le Phuoc; Kerry Taylor; Maxime Lefrancois. W3C. 19 October 2017. W3C Recommendation. URL: https://www.w3.org/TR/2017/REC-vocab-ssn-20171019/

[WOT-ARCHITECTURE]
Web of Things (WoT) Architecture. Matthias Kovatsch; Ryuichi Matsukura; Michael Lagally; Toru Kawaguchi; Kunihiko Toumura; Kazuo Kajimoto. W3C. 9 April 2020. URL: https://www.w3.org/TR/2020/REC-wot-architecture-20200409/

[WOT-BINDING-TEMPLATES]
Web of Things (WoT) Binding Templates. Michael Koster; Ege Korkan. W3C. 30 January 2020. W3C Note. URL: https://www.w3.org/TR/2020/NOTE-wot-binding-templates-20200130/

[WOT-SECURITY-GUIDELINES]
Web of Things (WoT) Security and Privacy Guidelines. Elena Reshetova; Michael McCool. W3C. 6 November 2019. URL: https://www.w3.org/TR/2019/NOTE-wot-security-20191106/

[xml]
Extensible Markup Language (XML) 1.0 (Fifth Edition) . Tim Bray; Jean Paoli; Michael Sperberg-McQueen; Eve Maler; Francois Yergeau et al. W3C. 26 November 2008. W3C Recommendation. URL: http://www.w3.org/TR/2008/REC-xml-20081126/

↑