

JF-WoT-Architecture(仮)

Web of Things アーキテクチャ

Web of Things Architecture Specification

第0版

2020年x月xx日制定

ドラフト２月6日版

一般社団法人

情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

本書は、一般社団法人情報通信技術委員会が著作権を保有しています。

内容の一部又は全部を一般社団法人情報通信技術委員会の許諾を得ることなく複製、転載、改変、転用及びネットワーク上での送信、配布を行うことを禁止します。

目　次

[<参考> 6](#_Toc28968898)

[Web of Things (WoT) アーキテクチャ 7](#_Toc28968899)

[１　本標準の範囲 8](#_Toc28968900)

[２　規約 9](#_Toc28968901)

[３　用語 9](#_Toc28968902)

[４　ユースケース 12](#_Toc28968903)

[4.1 アプリケーション領域 12](#_Toc28968904)

[4.1.1 コンシューマ 12](#_Toc28968905)

[4.1.2 インダストリ 13](#_Toc28968906)

[4.1.2.1 スマートファクトリ 13](#_Toc28968907)

[4.1.3 運輸・物流 13](#_Toc28968908)

[4.1.4 電力 13](#_Toc28968909)

[4.1.5 石油・ガス 14](#_Toc28968910)

[4.1.6 保険 14](#_Toc28968911)

[4.1.7 建築 14](#_Toc28968912)

[4.1.8 農業 14](#_Toc28968913)

[4.1.9 ヘルスケア 14](#_Toc28968914)

[4.1.10 環境 14](#_Toc28968915)

[4.1.11 スマートシティ 14](#_Toc28968916)

[4.1.12 スマートビルディング 15](#_Toc28968917)

[4.1.13 コネクテッドカー 15](#_Toc28968918)

[4.1.13.1 コネクテッドカーの事例 15](#_Toc28968919)

[4.2 共通パターン 15](#_Toc28968920)

[4.2.1 デバイスコントローラ 16](#_Toc28968921)

[4.2.2 Thing対Thing 16](#_Toc28968922)

[4.2.3 リモートアクセス 16](#_Toc28968923)

[4.2.4 ホームゲートウェイ 17](#_Toc28968924)

[4.2.5 エッジデバイス 17](#_Toc28968925)

[4.2.6 デジタルツイン 17](#_Toc28968926)

[4.2.6.1 クラウド対応デバイス 18](#_Toc28968927)

[4.2.6.2 レガシーデバイス 18](#_Toc28968928)

[4.2.7 マルチクラウド 18](#_Toc28968929)

[4.2.8 クラスドメインコラボレーション 18](#_Toc28968930)

[4.3 まとめ 19](#_Toc28968931)

[５　要件 19](#_Toc28968932)

[5.1 機能要件 19](#_Toc28968933)

[5.1.1 共通原則 19](#_Toc28968934)

[5.1.2 Thingの機能 20](#_Toc28968935)

[5.1.3 検索・発見 20](#_Toc28968936)

[5.1.4 記述メカニズム 20](#_Toc28968937)

[5.1.5 属性の記述 20](#_Toc28968938)

[5.1.6 機能の記述 21](#_Toc28968939)

[5.1.7 ネットワーク 21](#_Toc28968940)

[5.1.8 デプロイメント(配置) 21](#_Toc28968941)

[5.1.9 アプリケーション 21](#_Toc28968942)

[5.1.10 レガシー適用 21](#_Toc28968943)

[5.2 技術要件 22](#_Toc28968944)

[5.2.1 WoTとWoTアーキクチャで構成されるコンポーネント 22](#_Toc28968945)

[5.2.2 デバイス 22](#_Toc28968946)

[5.2.3 アプリケーション 22](#_Toc28968947)

[5.2.4 デジタルツイン 23](#_Toc28968948)

[5.2.5 発見 23](#_Toc28968949)

[5.2.6 セキュリティ 23](#_Toc28968950)

[5.2.7 アクセシビリティ 23](#_Toc28968951)

[６　WoTアーキテクチャ 24](#_Toc28968952)

[6.1 概要 24](#_Toc28968953)

[6.2 アフォーダンス 26](#_Toc28968954)

[6.2 Web Thing 26](#_Toc28968955)

[6.4 対話モデル 27](#_Toc28968956)

[6.4.1 プロパティ 27](#_Toc28968957)

[6.4.2 アクション 27](#_Toc28968958)

[6.4.3 イベント 28](#_Toc28968959)

[6.5 ハイパーメディアコントロール 28](#_Toc28968960)

[6.5.1 リンク 28](#_Toc28968961)

[6.5.2 フォーム 29](#_Toc28968962)

[6.6 プロトコルバインディング 30](#_Toc28968963)

[6.6.1 ハイパーメディア主導（ドリブン） 31](#_Toc28968964)

[6.6.2 URIs 31](#_Toc28968965)

[6.6.3 メソッドの標準セット 31](#_Toc28968966)

[6.6.4 メディアタイプ 31](#_Toc28968967)

[6.7 WoTシステムコンポーネントと相互接続性 32](#_Toc28968968)

[6.7.1 直接通信 33](#_Toc28968969)

[6.7.2 間接通信 33](#_Toc28968970)

[７　WoTビルディングブロック 34](#_Toc28968971)

[7.1 WoT Thing Description 34](#_Toc28968972)

[7.2 WoTバインディングテンプレート 35](#_Toc28968973)

[7.3 WoT Scripting API 36](#_Toc28968974)

[7.4 WoTセキュリティとプライバシのガイドライン 37](#_Toc28968975)

[８　サービエントのインプリメンテーション 37](#_Toc28968976)

[8.1 ビヘイビアのインプリメンテーション 38](#_Toc28968977)

[8.2 WoTランタイム 38](#_Toc28968978)

[8.3 WoTスクリプティングAPI 39](#_Toc28968979)

[8.4 公開されたThingと消費されるもののアブストラクション 39](#_Toc28968980)

[8.5 プライベートセキュリティ構成 40](#_Toc28968981)

[8.6 プロトコルスタックのインプリメンテーション 40](#_Toc28968982)

[8.7 システムAPI 40](#_Toc28968983)

[8.8 代替サービエントおよびWoTインプリメンテーション 41](#_Toc28968984)

[8.8.1 ネイティブWoT API 41](#_Toc28968985)

[8.8.2 TDで記述される既存デバイス 41](#_Toc28968986)

[９　WoTの導入 42](#_Toc28968987)

[9.1 Thingのコンシューマの役割 42](#_Toc28968988)

[9.2 WoTシステムのトポロジと展開シナリオ 43](#_Toc28968989)

[9.2.1 同一ネットワーク上のコンシューマとThing 43](#_Toc28968990)

[9.2.2 仲介者を介して接続されたコンシューマとThing 43](#_Toc28968991)

[9.2.2.1 プロキシとしての仲介者 43](#_Toc28968992)

[9.2.2.2 デジタルツインとしての仲介者 44](#_Toc28968993)

[9.2.3 クラウドサービスから制御されるローカルネットワーク内のデバイス 44](#_Toc28968994)

[9.2.4 Thingディレクトリを使用した検出 45](#_Toc28968995)

[9.2.5 複数ドメイン間のサービス対サービス接続 46](#_Toc28968996)

[9.2.5.1 ディレクトリ同期による接続 46](#_Toc28968997)

[9.2.5.2 プロキシ同期による接続 46](#_Toc28968998)

[１０　セキュリティとプライバシに関する考慮事項 47](#_Toc28968999)

[10.1 WoT TDリスク 47](#_Toc28969000)

[10.1.1 TDのプライベートセキュリティデータリスク 47](#_Toc28969001)

[10.1.2 TDの個人情報リスク 48](#_Toc28969002)

[10.1.3 TD通信メタデータリスク 48](#_Toc28969003)

[10.2 WoTスクリプティングAPIのセキュリティとプライバシのリスク 48](#_Toc28969004)

[10.2.1 クロススクリプトセキュリティとプライバシーリスク 49](#_Toc28969005)

[10.2.2 物理デバイスダイレクトアクセスセキュリティとプライバシリスク 49](#_Toc28969006)

[10.3 WoTランタイムセキュリティとプライバシリスク 50](#_Toc28969007)

[10.3.1 セキュリティリスクのプロビジョニングと更新 50](#_Toc28969008)

[10.3.2 セキュリティ証明書ストレージのセキュリティとプライバシリスク 50](#_Toc28969009)

[A. 謝辞 51](#_Toc28969018)

[B. 参考文献 51](#_Toc28969019)

[B.1 Normative references 51](#_Toc28969020)

[B.2 Informative references 51](#_Toc28969021)

# <参考>

（W3C文書を翻訳・発行するうえで、記載すべき情報を記載する）

1. 国際勧告等の関連

本標準は、2020年x月に勧告が承認されたW3C Web of Things Architectureに準拠している。

2. 上記国際勧告等に対する追加項目等

2. 1 オプション選択項目

特になし

2. 2 ナショナルマター項目

特になし

2. 3 原標準に対する変更項目

　できるだけ原文に忠実に翻訳を行ったが、理解を助けるために内容を補っている部分がある。また、元文書にある付録A「仕様の修正履歴」については削除している。

2. 4 その他

特になし

2. 5 現勧告との章立て構成比較表

上記国際勧告との章立て構成の相違はない。

3. 改版の履歴

|  |  |  |
| --- | --- | --- |
| 版数 | 制定日 | 改版内容 |
| 第1.0版 | 2020年x月xx日 | 制定 |
|  |  |  |

4. 工業所有権

本標準に関わる「工業所有権等の実施の権利に係る確認書」の提出状況は、TTCホームページで御覧になれます。

5. 標準策定部門

次世代ホームネットワークシステム専門委員会（本標準の策定）

W3C Web of Things Working Group（W3C Web of Things Architectureの策定）

Web of Things (WoT) アーキテクチャ

W3Cのタイトルページの記載方法（芦村先生）

**This version:**

https://w3c.github.io/wot-architecture/

**Latest published version:**

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

**Latest editor's draft:**

https://w3c.github.io/wot-architecture/

**Editors:**

Matthias Kovatsch (Huawei)

Ryuichi Matsukura (Fujitsu Ltd.)

Michael Lagally (Oracle Corp.)

Toru Kawaguchi (Panasonic Corp.)

Kunihiko Toumura (Hitachi, Ltd.)

Kazuo Kajimoto (Former Editor, when at Panasonic)

**概要**

W3C Web of Things (WoT) は、IoT プラットフォーム間、適用領域間とそれぞれの間の相互接続性を実現することを目的とする。全体として、WoTの目標は、既存のIoT規格とソリューションをできるだけ活用し、不足する機能を補うことである。概してW3C WoTアーキテクチャは、何を実装するかを規定するものではなく、何が存在するかを説明するように設計された。

WoTアーキテクチャ仕様は、W3C Web of Thingsの抽象アーキテクチャを説明する。この抽象アーキテクチャは、複数の適用分野におけるユースケースとユースケースから導かられる要件に基づいた。このユースケースと要件は、両方ともこの文書に記載される。モジュラー化された構成要素は、詳細の仕様が他の文書に記載されている。本文書はこれらの構成要素がどのように関連し、連携して動くかを示した。WoT抽象アーキテクチャは、基本的な概念フレームワークを定義するものであり、具体的に展開されるシナリオに対応付けられる。この例をいくつか示す。ただし、本文書で記述される抽象アーキテクチャは、具体的なメカニズムを定義したり、具体的な実装を規定するものでない。

**文書の状態**

本項では、本文書が発行された時点での状態を記述する。このため、他の仕様がこの文書より新しい場合もある。現行のW3C出版物一覧および本テクニカルレポートの最新版は、https://www.w3.org/TR/のW3Cテクニカルレポートインデックスに記載されている。

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

# １　はじめに

Web of Things (WoT)の目標は、Internet of Things (IoT)の相互接続性および有用性を改善することである。長年にわたり多くのステークホルダを含めた関係者による共同作業を通じて、これらの課題に対処するために必要ないくつかのビルディングブロック（構成要素）が明らかになってきた。

本仕様はW3C WoTの標準規格の範囲を示すことに重点を置き、これらのビルディングブロックと、それらの関係を示す抽象アーキテクチャについて述べる。個々のビルディングブロックについては、別の仕様書によって定義され、その詳細が記述されている。一方、本アーキテクチャ文書では抽象アーキテクチャと、用語定義や概念的なフレームワークについての定義することに加え、WoTビルディングブロックを紹介し、それら相互の関係についても説明する。

* Web of Things (WoT) Thing Description [WOT-THING-DESCRIPTION]は、Thingのメタデータやネットワークインターフェースの記述について、機械可読データの形式を説明します。Thing Descriptionは、相互作用アフォーダンスのように、この文書で導入される基本的な概念に基づいています。
* Web of Things (WoT) Binding Template [WOT-BINDING-TEMPLATE]は、Protocol Bindingと呼ぶ、Thingのネットワークインターフェースを定義する方法に関するガイドラインを説明します。ネットワークインターフェースは、特定のプロトコルやIoTエコシステムと接続します。本文書では、いくつかの既存のIoTエコシステムを使ったネットワークインターフェース定義の例を示します。
* Web of Things (WoT) Scripting API [WOT-SCRIPTING-API]を使うと、WebブラウザのAPIと同様のJavaScript APIによって、Thingを扱うアプリケーションロジックを実現することができる。オプション仕様であるが、この仕様は、IoTアプリケーションの開発を簡単にし、異なるベンダーやデバイス間でのポータビリティを確保する。
* Web of Things (WoT) Security and Privacy Guidelines [WOT-SECURITY]は、他のすべての構成要素に横断的に関わるビルディングブロックである。この解説文書はThingのセキュリティに関する実装や設定に関するガイドラインを提供する。また、W3C WoTを備えるいかなるシステムにおいても考慮されるべき課題について議論する。ただし、セキュリティとプライバシは、WoTの抽象アーキテクチャでは十分に明確化されない、特定の実装における具体的なメカニズムをすべて把握した上でのみ充分な評価ができることは注意されるべきである。例えば、WoTが既存システムを記述する方法として使われる場合、特に注意が必要である。W3C WoTは、そのようなシステムを記述することはできますが、動作を制限することができないためです。この文書でも、10章Security and Privacy Considerationsにおいて、プライバシとセキュリティのリスクとその軽減方法について簡単に述べています。

また本文書は、WoTシステムの実装を支援するため、その展開方法について、アーキテクチャを参照しながら構成や条件などについても述べる。これらのガイドラインは展開シナリオの例として記述されますが、特定の具体的な実装方法を定義しません。

本文書は、W3C WoT仕様の全体概要を表し、W3C Web of Thingsの用語定義や基礎となる抽象アーキテクチャを定義します。要約すると、本仕様書の目的は次のような内容になります。

* 4章Use Casesでは、W3C WoTアーキテクチャにつながるユースケースを示す。
* 5章Requirementsでは、WoT仕様の設計に求められる要件を示す。
* 6章WoT Architectureでは、抽象アーキテクチャの定義を示す。
* 7章WoT Building Blocksでは、ビルディングブロックの概要とそれらの関係を示す。
* 8章Abstract Servient Architectureでは、抽象アーキテクチャを具体的な実装へマッピングするため解説的なガイドラインを示す。
* 9章Example WoT Deploymentでは、実現可能なシステム展開シナリオの解説的な例を示す。
* 10章Security and Privacy Considerationsでは、ハイレベルではあるが、W3C WoTアーキテクチャに基づくシステムを実装するうえで注意すべきセキュリティとプライバシに関する考察を示す。

追加する要件、ユースケース、概念的な特徴、新しいビルディングブロックは、この文書の将来の改版で記述される。

# ２　適合

参考情報と明示される章と同様に、本文書における全てのガイドライン、図、例、および注釈は規定でない。これ以外の本文書に記載されるものはすべて規定である。

本文書内のキーワード“**してもよい（MAY）**”、”**しなければならない（MUST）**“、及び”**すべきである（SHOULD）**”は、**太字**で表示されているときにのみBCP 14 [RFC2119] [RFC8174]の記述に従い解釈するものとする。

注：原文では、「すべて大文字で表現される場合はBCP14に従う」とある。

# ３　用語定義

本章は参考情報である。

本仕様では、ここで定義されている以下の用語を使用する。WoTを接頭辞に持つ用語は、Web of Thingsにおける概念のために明確に(再)定義されていることを示すために使用される。

**Action/アクション**

相互作用アフォーダンスの1つ。Thingの機能を起動するときに使われる。これは、状態を変更するとき（例えば、電灯のオン・オフを切り替える)、あるいはThingの処理を起動する(例えば、次第にランプを暗くする)ときに使われる相互作用アフォーダンス。

**Binding Templates/バインディングテンプレート**

各種IoTプラットフォームとのやり取りを実現するための、再利用可能な青写真のコレクション。これらの青写真は、WoT Thing Descriptionの中で相互作用アフォーダンスをプラットフォーム固有のメッセージに対応付ける相互作用アフォーダンスための情報を提供する。同時に、必要なプロトコルスタックまたは専用通信ドライバの実装で留意すべき情報を提供する。

**Consumed Thing**

ローカルのアプリケーションから利用される遠隔のThingの抽象表現。抽象表現はネイティブなWoT Runtimeで生成されるか、WoT Scripting APIを通じてオブジェクトとして生成される。

**Consuming a Thing（アーキ文書では使われていない）**

TDドキュメントの構文解析および処理を行い、ローカルのRuntime環境におけるアプリケーション向けインターフェースとして、Consumed Thingの抽象表現を作成すること。

**Consumer**

WoT Thing Description (JSON ベースの表現形式を含む) を処理し、Thingと相互作用する (Thingのを利用する)ことができるエンティティ。

**Data Schema**

Data schemaは、以下を記述する。情報モデル、ThingとConsumerとの相互作用で受け渡されるペイロードの構造とデータ項目を示す。

**Digital Twin/デジタルツイン**

デジタルツインは、クラウドまたはエッジノード上にある、デバイスまたは一群のデバイスの仮想的な表現である。これは、継続的にオンライン状態でない可能性のある物理的なデバイスを表すときに使用できる。または、新規のアプリケーションやサービスが配置される前にシミュレーションするケースにも使用することができる。

**Domain-specific Vocabulary/ドメイン固有の語彙**

WoT TDで使用することができるが、W3C WoTでは定義されていないLinked Dataの語彙。

**Edge Device/エッジデバイス**

エンタープライズまたはサービス提供者のコアネットワークへのエントリポイントを提供するデバイス。例えば、ゲートウェイ、ルータ、スイッチ、マルチプレクサ、および、様々なその他のアクセスデバイスである。

**Event/イベント**

イベントの発生元を示す、相互作用アフォーダンスの1つである。イベントの発生は、非同期でイベントデータをConsumuerにプッシュする(例：オーバーヒート警報)。

**Exposed Thing/公開されたThing**

遠隔のConsumerからネットワーク経由でアクセス可能なデバイスに対応する、Thingの側のソフトウェア抽象表現。この抽象表現は、ネイティブのWoT Runtimeで生成されるか、WoT Scripting APIを通じてオブジェクトとして生成される。

**Exposing a Thing/Thingを公開する**

Runtime環境上のソフトウェア抽象表現であるExposed Thingを生成すること。これによってThingの状態と振る舞いを実現するインターフェースを管理する。

**Hypermedia Control/ハイパーメディアコントロール**

プロトコルバインディングを、ハイパーメディアを用いて具体的な表現(シリアライゼーション)にしたものである。具体的には、関連情報につながるWebリンク[RFC8288]や、操作を実行するWebフォームのことを指す。Webフォームは、Thingにより提示されたリクエスト上のテンプレートであり、Consumerが内容を入力し、送信するものとみなすことができる。。

**Interaction Affordance/相互作用アフォーダンス**

Consumerにとって可能な選択肢を示す、Thingのメタデータである。これにより、ConsumerはThingとどのように相互作用すればよいかを理解することができる。潜在的には多くのタイプのアフォーダンスがありうるが、W3C WoTでは、3タイプの相互作用アフォーダンス、すなわち、プロパティ、アクション、イベントを定義している。第4の相互作用アフォーダンスは、関連情報をたどっていくことであるが、これはリンクという形でWebでは広く利用可能となっている。

**Interaction Model/相互作用モデル**

アプリケーションの意図から具体的なプロトコル操作へのマッピングを、絞り込んだ形で定式化する。W3C WoTでは、定義された一連の相互作用アフォーダンスが相互作用モデルを構成する。

**Intermediary/仲介者**

ConsumerとThingの間に存在するエンティティである。Thingの代理、拡張、あるいは複数Thingを兼備することなどを目的とし、元のThingの代わりにIntermediaryにあるWoTインターフェースを指し示すWoT Thing DescriptionをConsumerに対して再発行する。RESTの階層化システムに沿っているので、Consumerにとっては、Intermediaryは元のThingと区別できないかもしれない。

**IoT Platform/IoTプラットフォーム**

特定のIoTエコシステムのこと。例えば、OCF、oneM2M、MozillaのProject Thingsのように、アプリケーションインターフェース、データモデルと、プロトコルあるいはプロトコル構成に関する独自の仕様が規定されるもの。

**Metadata/メタデータ**

エンティティの抽象化された特徴の記述を提供するデータ。例えば、Thing DescriptionはThingのMetadataである。

**Personally Identifiable Information (PII)/個人識別可能情報**

特定の個人を識別するために使われる情報。その個人に関連する直接、あるいは間接的にその個人につながる情報を示す。[ISO-IEC-29100]に示される定義と同様に使う。

**Privacy/プライバシ**

個人に関するデータへの不当または違法な収集と使用による、個人の生活や活動への侵害から逃れられること。[ISO-IEC-2382]に示される定義と同様に使う。個人特定可能情報やセキュリティ、および[ISO-IEC-29100]に示される、その他の関連する用語の定義も参照のこと。

**Private Security Data**

Private Security Dataは、他のデバイスやユーザと共有されずに秘匿されるThingのセキュリティ設定情報の構成要素である。例としては、PKIシステムにおける秘密鍵（private key）がある。理想的には、このようなデータはアプリケーションにアクセスできない分離されたメモリ領域に置かれると同時に、それを使うアプリケーションにも明らかにされず、例えば署名のような抽象的な操作によってのみ利用される。

**Property/プロパティ**

Thingの状態を示す相互作用アフォーダンスの1つ。この状態は読み取り、書き込みすることができる。Thingは、プロパティを観察可能（observable）に選択することができ、そのプロパティの値が変更されると変更後の新しい状態を通知する。

**Protocol Binding/プロトコルバインディング**

相互作用アフォーダンスから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のこと。Consumerは、相互作用アフォーダンスの利用の仕方を、このマッピングから知ることができる。W3C WoTでは、プロトコルバインディングをハイパーメディアコントロールとしてシリアライズする。

**Public Security Metadata**

Public Security Metadataは、Thingにアクセスするときに必要となるセキュリティメカニズムとアクセス権限を記述するThingのセキュリティ設定情報の構成要素である。これには、いかなる秘密情報もあるいは公開鍵のような具体的なデータも含まれず、この情報だけでは、Thingへのアクセスはできない。例えば、ユーザがどのようにして自己を認証すべきかについての方法など、認可されたユーザThingへのアクセスを獲得するためのメカニズムが記述される。

**Security/セキュリティ**

情報の機密性（Confidentiality）、完全性（Integrity）、可用性（Availability）を保持すること。他に、真正性（Authenticity）、責任追跡性（Accountability）、否認防止（Non-Repudiation）、信頼性（Reliability）などの性質も含まれることがあります。この定義は、[ISO-IEC-27000]の情報セキュリティにある定義から適用されており、前に列挙した各語の定義に関してはさらに具体的な性質が追加定義として含まれています。他の関連する定義に関しても、この仕様書を参照してください。また、これらの性質は、通常と攻撃を受けている両方の場合で維持されることが望ましいことに注意してください。

**Security Configuration/セキュリティ設定情報**

Publish Security Metadata、Private Security Dataおよび、操作の際にThingのセキュリティメカニズムを設定するために必要な他の設定情報の組み合わせ。

**Servient**

WoTビルディングブロックを実現するソフトウェアスタックである。Servientは、Thingを公開する機能と、Thingを利用するConsumerの機能を両方、もしくは片方を提供することができる。Servientは、様々なIoTプラットフォームと相互作用できるようにするために複数のプロトコルバインディングをサポートすることができる。

**Subprotocol/サブプロトコル**

転送プロトコル（Transfer Protocol）への拡張機能である。これは、Thingの機能に転送プロトコルを経由してアクセスするときに必須の追加情報がある場合に記載する。例としては、HTTPでのlong pollingがある。

**TD**

WoT Thing Description の短縮形。

**TD Vocabulary/TD語彙**

WoT Thing Descriptionのインスタンス内でThingのメタデータを記述するのに利用される、W3C WoTの仕様により直接定義されたリンクデータ語彙。WoTバインディングテンプレートの通信メタデータなども、これに含まれる。

**ThingまたはWeb Thing**

メタデータおよびインターフェースがWoT TDによって記述されている物理あるいは仮想エンティティの抽象表現である。仮想エンティティは1つまたは複数のThingで構成される。

**Thing Directory**

Thing Descriptionのディレクトリサービス。このサービスはTDを登録し、Thing Descriptionを検索するためのWebインターフェースを提供する。機能的に関連する技術として、[CoRE-RD]で定義された登録と検索の機能、およびSPARQLクエリを利用する検索の機能がある。

**Transfer Protocol/転送プロトコル**

オプションあるいはサブプロトコルメカニズムに関するアプリ固有の要件または制約を持たない基礎となる標準アプリケーション層プロトコル。例は、HTTP、CoAP、またはMQTT。

**Virtual Thing/ヴァーチャルThing**

別のシステムコンポーネントに存在するThingを表すThingのインスタンス。

**WoT Interface/WoTインターフェース**

WoT Thing Descriptionによって記述されるThingのネットワーク側に提供されるインターフェース。

**WoT Runtime/WoTランタイム**

アプリケーションの実行環境を保有するRuntimeシステムのこと。また、このRuntimeシステムは、Thingを公開または利用し、WoT Thing Descriptionを処理し、セキュリティ設定情報を保有し、プロトコルバインディングの実装とのインターフェース機能を持つ。WoT Runtimeが提供するAPIは、独自のAPIでも良いし、オプション仕様のWoT Scripting APIを選択することもできる。

**WoT Scripting API/スクリプティングAPI**

WoT Servientがアプリケーションに提供するインターフェース。WoT Runtimeランタイム内で実行されるServientの動作やアプリケーションの実装を容易にする。WebブラウザAPIに相当する。WoT Scripting APIは、W3C WoTビルディングブロックの1つであるが、オプション仕様である。

**WoT Servient**

Servientのこと。

**WoT Thing DescriptionまたはThing Description**

Thingを記述する構造化されたデータ。WoT Thing Descriptionは、一般メタデータ、領域固有メタデータ、相互作用アフォーダンス(サポートされるプロトコルバインディングを含む)、および関連するThingへのリンクで構成されている。WoT Thing Descriptionの記述形式は、W3C WoTの中心的なビルディングブロックである。

# ４　ユースケース

本章は参考情報である。

本章では、W3C WoTの対象であるアプリケーション領域とユースケースを提示する。また、ここで示すユースケースは、7章WoT Building Blockで考察される抽象アーキテクチャを導出するために使用される。

WoTアーキテクチャは、ユースケースやアプリケーション領域に関していかなる制限も置いていない。抽象アーキテクチャが満足しなければならない様々なアプリケーション領域において、共通パターンを見い出すために議論されてきた。

以下に記述する内容は網羅的ではない。むしろ、接続されたThingが付加的な価値を提供する、あるいは、新たなシナリオを想起させることができる内容になっている。

## 4.1 アプリケーション領域

### 4.1.1 コンシューマ

コンシューマ領域では、接続されることでメリットが得られる複数の設備がある。部屋に人がいるかどうかで照明およびエアコンをオフにすることができ、窓のブラインドは気象条件や予報に基づいて自動的に閉じられる。エネルギーおよび他のリソース消費は使用パターンや予測に基づいて最適化される。

この節で述べるコンシューマのユースケースとして、スマートホームが含まれている。

図1は、スマートホームの例である。この例では、KNX、ECHONET、ZigBee、DECT ULE、Wi−SUNなどの通信プロトコルを利用して、センサ、カメラ、家電などのエッジデバイスがゲートウェイに接続されている。1軒の家に複数のゲートウェイを設置することが可能であり、それぞれのゲートウェイは複数の通信プロトコルをサポートすることができる。

ゲートウェイは、インターネット経由でクラウドに接続することができ、複数の家電をクラウドに接続することができる。クラウド提供されるサービスは、エッジデバイスからデータを収集し、そのデータを分析して、家電のようなエッジデバイスおよびUXデバイス（スマートフォン）を介してユーザに価値を提供する。

図1. スマートホーム

スマートホームは利用者に対して、遠隔アクセスや制御、音声制御、ホームオートメーションなどのサービスを提供する。また、メーカによるデバイスの遠隔監視や保守も可能にする。さらに付加サービスとして、エネルギー管理やセキュリティ監視なども実現することができる。

### 4.1.2 産業適用→NOTE:以下の節は、4.1.2.1のみがサブ節で良いか確認

本節のインダストリのユースケースは、様々な垂直統合サービスに適用可能である。

アプリケーションシナリオは構造が似ているため、様々な垂直統合サービスが同様のユースケースとなっている。

#### 4.1.2.1 スマートファクトリ

図2は、スマートファクトリの例である。このユースケースは、フィールド、セル、ラインのコントローラが、PROFINET、Modbus、OPC UA TSN、EtherCAT、CANなどの産業用通信プロトコルを利用して、様々な工場設備を自動化している。工場のエッジデバイスは様々なコントローラからデータを収集し、デバイスがクラウドサービスから利用可能となる。例えば、ダッシュボードを利用した遠隔モニタリングや、故障予防のための分析サービスを提供する。

図2. スマートファクトリ

スマートファクトリは、接続された製造装置や製造された製品の高度な監視を必要とする。製造装置の故障やメンテナンスによる生産停止を避けるために、装置の故障予測や早期の異常検出することにメリットがある。

さらに、生産工場における製造装置や、有害ガス、過剰ノイズ、熱などの環境のモニタリングは、作業者の安全性を高め、故障や事故のリスクを低減することになる。

製造装置のリアルタイム監視やKPI計算は、生産性の問題を検出し、サプライチェーンを最適化することができる。

### 4.1.3 運輸・物流

車両、燃料コスト、保守時期の監視は、車両全体のメンテナンスを最適化するのに役立つ。

輸送される荷物の品質や状態を保証するために、荷物はルート追跡することができる。これは、倉庫から輸送する冷蔵トラックまでの温度管理ができていることを証明する際に有用である。

倉庫やヤードの在庫の集中監視・管理は、在庫切れや過剰在庫を防ぐことができる。

### 4.1.4 電力

住居用、商業ビルや工場の電力メータの自動検針と課金は、リソースの消費とそのときの潜在的なボトルネックに関する継続した分析を提供する。

分散された再生可能エネルギー電源の状態や出力を監視することで、分散電源の最適化が可能になる。

配電装置の監視や遠隔制御は、配電網の自動化に役立つ。

発電と配電網を継続的に監視することで、現場作業員の安全性を向上させている。

### 4.1.5 石油・ガス

海上の油田監視、パイプラインの漏れ検出および予測、ならびに、タンクや貯蔵施設の水位監視と制御は、労働と環境の安全性を高めるのに役立つ。

様々な貯蔵タンクや、パイプラインやトラック輸送による分散した貯蔵方法を決定する自動計算は、計画の精度向上や資源量最適化を可能にする。

### 4.1.6 保険

繋ぎ合わされた建造物、フリート車両などの高価値な資産のプロアクティブな資産監視では、インシデントの予測や早期検出により深刻な損害や高額な修理コストのリスクを回避する。

利用ベース保険は、利用状況をトラッキングすることと、保険条件をカスタマイズすることで提供可能となる。

天気予報をモニタし、屋根のある車庫への車両の誘導により、雹や樹木による車両の損傷を抑えることができる。

### 4.1.7 建築

作業現場への立ち入りをモニタリングすることで、セキュリティ上の危険源によるリスクを軽減する。また、建設現場の工作機械をモニタすることで、機械故障による損害や作業遅延等の損失を防ぐことができる。

### 4.1.8 農業

土壌中の水分量や養分量のモニタリングや最適化の計画作成は、農産物の生育状態監視と同様に、農産物の品質と収穫量を最大化できる。

### 4.1.9 ヘルスケア

臨床試験のデータ収集と分析は、新領域への知見を得るのに役立つ。

病院内外で利用可能な遠隔での患者モニタリングは、高齢者や入院後の患者にとってこれまで見過ごされた深刻な状況に陥るリスクを回避できる。

### 4.1.10 環境モニタリング

環境モニタリングは、通常、多数の分散されたセンサにより実現される。そして、その測定データは、共通のゲートウェイ、エッジデバイス、およびクラウドサービスに送信する。

環境における深刻な状況を検出するために、大気汚染や水質汚染、その他の環境リスク要因として大気中の微粒子、オゾン、揮発性有機化合物、放射性物質の濃度、温度、湿度などデータをモニタすることは、回復不能な健康や環境被害を防ぐことができる。

### 4.1.11 スマートシティ

橋梁、ダム、堤防、運河における材質、劣化、振動をモニタすることで保守修理の必要なタイミングを検出し、重大な事故を防ぐ。また、高速道路の車の流れをモニタし、適切な案内掲示をすることにより、交通の流れをスムーズにできる。

スマートパーキングは、パーキングスペースの利用や空き状況を追跡し、最適化すると同時に、課金や予約を自動化する。

人の存在検出、気象予測などに基づく街路灯のスマート制御はコストを削減する。

地域に設置されるごみ収集コンテナは廃棄物管理や回収ルートを効率化・最適化するためにモニタすることができる。

### 4.1.12 スマートビルディング

ビル全体のエネルギー使用量のモニタリングは、資源消費の最適化と無駄の削減に役立つ。

暖房換気空調（HVAC）、エレベータなどのビル内の機器モニタリングと問題の早期改善が、入居者の満足度を改善する。

### 4.1.13 コネクテッドカー

運行状況のモニタリング、サービスニーズの変化予測は、保守必要性とコストとのバランスを最適化する。運転者の安全性は、道路や交通状況の事前警報システムからの通知によって高められる。

#### 4.1.13.1 コネクテッドカーの事例→NOTE:このタイトルは不要ではないか？勧告前に削除。

図3は、コネクテッドカーの構成例である。ここでは、ゲートウェイがCANを介して自動車のコンポーネントに、また、車載ネットワークのインターフェースを介してカーナビに接続されている。クラウドサービスは、自動車のコンポーネントから送られるデータを収集し、複数の自動車からのデータをまとめて分析して運行パターンを決定する。ゲートウェイは、また、クラウドサービスを利用して運行データを取得し、カーナビを通じて運転者に知らせる。

図3. コネクテッドカー

運行状況のモニタリング、サービスニーズの変化予測は、保守必要性とコストとのバランスを最適化する。運転者の安全性は、道路や交通状況の事前警報システムの通知によって高められる。→NOTE：同じ記述がパラグラフ最初にあるため削除

## 4.2 共通パターン

本節では、デバイス/Thingがコントローラ、他のデバイス、エージェント、およびサーバとどのように相互作用するかを示す共通的なユースケースパターンを紹介する。クライアント役という用語をトランスポートプロトコルのイニシエータとして使用し、サーバ役という用語をトランスポートプロトコルの受動コンポーネントとして使用する。しかし、個々のシステム構成要素に対して特定の役割が決められているということではない。一つのデバイスがクライアントおよびサーバの役を同時に果たすこともできる。

この二重の役割の一例としては、クラウドサービスに登録し、定期的にクラウドにセンサ値を送信するセンサである。応答メッセージの中で、クラウドは、センサのメッセージの送信速度を調整するか、または将来のメッセージで送信される特定のセンサ属性を選択することができる。センサはそれ自体をクラウドに登録・接続を開始するので、「クライアント」役である。しかし、リクエストにも反応して応答メッセージを送信するので、「サーバ」としての役割も果たす。

以下の節では、より複雑な役割、タスク、およびユースケースのパターンについて説明する。それらは、網羅的ではなく、本仕様の後続の項で定義されるWoTアーキテクチャ、およびそのビルディングブロックへの興味付けのために提示されている。

### 4.2.1 デバイスコントローラ

第1のユースケースは、図4のようなユーザが操作するリモートコントローラによって制御されるローカルデバイスである。リモートコントローラは、ローカルホームネットワークを通じて直接電子機器にアクセスできる。この場合、このリモートコントローラは、ブラウザまたはネイティブアプリによって実装することができる。

このパターンでは、少なくとも1つの電子機器のようなデバイスは、他のデバイスからのリクエストを受け、それらに応答し、機械的動作をも開始することもあるサーバ役を担う。リモートコントローラのような他のデバイスは、センサ値を読み取る、またはデバイスをオンにするなどといったリクエストの入ったメッセージを送信することができるクライアント役を担う。さらに、デバイスの現況またはイベント通知を送るために、デバイスは、クライアント役として、サーバ役の別のデバイスにメッセージを送信することができる。

図4. デバイスコントロール

### 4.2.2 Thing対Thing

図5は、直接的なThing対Thingの相互作用の例である。ここでのシナリオは、センサが、例えば、温度がしきい値を超えるなど部屋の状態の変化を検出し、電子機器に「オンにする」などの制御メッセージを送るというものである。このセンサユニットは、他のデバイスにいくつかのトリガーメッセージを出すことができる。

この場合、サーバの役割を有する2つのデバイスが接続されているとき、少なくとも1つのデバイスは、作動または通知するために他方にメッセージを発行するクライアントの役割も有していなければならない。

図5. コントロールエージェント

### 4.2.3 リモートアクセス

このユースケースでは、図6に示されるような(例：スマートフォン上の)モバイルリモートコントローラが使われている。このリモートコントローラは、例えば、セルラネットワークとWi-FiとBluetoothのようなプロトコルを使ったホームネットワークのような異なるネットワーク接続とプロトコルを切り替えることができる。コントローラがホームネットワーク内にある場合、コントローラは信頼できるデバイスであり、追加のセキュリティまたはアクセス制御は必要ない。信頼できるネットワーク内にない場合、信頼できる関係を保証するために、追加のアクセス制御およびセキュリティメカニズムを適用しなければならない。このシナリオでは、ネットワーク接続性は、異なるネットワークアクセスポイント間の切り替え、あるいは、セルラー基地局間の切り替えによって変化することがあることに留意されたい。

このパターンでは、リモートコントローラおよび電子機器は図4の関連するシナリオ同様、クライアントおよびサーバの役割を担う。

図6. リモートアクセス

### 4.2.4 スマートホーム・ゲートウェイ→NOTE:Smart Home Gateway, Smart home gatewayなど大文字小文字の揺れがある。また、smart home gatewayとhome gatewayの記述があり、どちらかに統一すべき

図7は、スマートホーム・ゲートウェイを使用するユースケースである。スマートホーム・ゲートウェイは、ホームネットワークとインターネットとの間に配置される。ゲートウェイは、家庭内の電子機器を管理し、インターネットを介してリモートコントローラから、例えば、前出のユースケースのようにスマートフォンからコマンドを受信することができる。これは、デバイスの仮想表現でもある。スマートホーム・ゲートウェイは、通常、プロキシおよびファイアウォール機能を提供する。

このパターンでは、ホームゲートウェイは、クライアント役とサーバ役の両方を担う。リモートコントローラが電子機器を作動させると、クライアント役の電子機器とサーバ役のリモートコントローラとに接続することができる。電子機器がリモートコントローラにメッセージを発信すると、ゲートウェイは、電子機器のサーバ役として動作し、リモートコントローラのクライアント役として動作する。

図7. スマートホーム・ゲートウェイ

### 4.2.5 エッジデバイス

エッジデバイスあるいはエッジゲートウェイは、スマートホーム・ゲートウェイに似ている。この用語は、エッジゲートウェイによって実行される追加のタスクを指すために使用される。図8のホームゲートウェイは、本来、公衆ネットワークと信頼できるネットワーク間を単にブリッジするだけであるが、エッジデバイスは、ローカルでの計算能力を有し、通常、異なるプロトコルをブリッジする。エッジデバイスは、基本的に、産業ソリューションで使用され、接続されたデバイスとセンサによって提供されるデータの前処理、フィルタリング、および集約を行うことができる。

図8. エッジデバイス

### 4.2.6 デジタルツイン

デジタルツインは、仮想表現、すなわち、クラウドサーバまたはエッジデバイス上に存在する一個のデバイスまたは一群のデバイスのモデルである。これは、継続的にオンライン状態にない可能性がある現実世界のデバイスを表すために、または、現実のデバイスに配備される前に、新しいアプリおよびサービスのシミュレーションを実行するために使用することができる。

図9. デジタルツイン

デジタルツインは、単一のデバイスをモデル化することができ、または、組み合わされたデバイスの単一の仮想表現として複数のデバイスを集約することができる。

図10. 複数デバイスのためのデジタルツイン

デジタルツインは、デバイスがすでにクラウドに接続されているかどうか、または、クラウドに接続されているゲートウェイに接続されているかどうかに応じて、様々な方法で実現できる。

#### 4.2.6.1 クラウド対応デバイス

図11は、電子機器がクラウドに直接接続される例である。クラウドは、機器を再現し、デジタルツインとして動作して、リモートコントローラ(例：スマートフォン)からコマンドを受信することができる。許可されたコントローラは、デジタルツインへグローバルに到達可能であるため、どこにでも配置することができる。

図11. クラウド対応デバイス用機器ツイン

#### 4.2.6.2 レガシーデバイス→NOTE:脚注をつける　非WoTデバイスのこと

図12は、レガシー電子機器がクラウドに直接接続できない例である。接続を中継するためにゲートウェイが必要となる。ゲートウェイは、以下のように動作する。

* 物理的論理的観点で多様なレガシー通信プロトコルの統合
* インターネット用ファイアウォール
* 実画像および/または音声を置換し、ローカルでデータを記録するプライバシーフィルタ
* ネットワーク接続が中断された場合のローカルエージェント
* 火事警報などが発生したときにローカルで実行する緊急サービス

クラウドは、接続されているすべての機器を含めゲートウェイを再現し、ゲートウェイと連携してクラウド内で機器を管理するデジタルツインとして機能する。さらに、クラウドは、任意の場所に配置することができるリモートコントローラ(例：スマートフォン)からのコマンドを受信することができる。

図12. レガシーデバイス用デジタルツイン

### 4.2.7 マルチクラウド

典型的なIoT展開は複数(数千)のデバイスで構成される。標準メカニズムなしでは、特定のクラウドのファームウェア更新の管理は多くの労力を必要とし、より大きな規模でのIoT採用を妨げる。

デバイスおよびデバイスタイプを記述するための標準メカニズムの主な利点は、デバイスソフトウェア/ファームウェアレベルでカスタマイズを行う必要がなく、すなわち、クラウド固有のコードをデバイスにインストールする必要なしに、デバイスを異なるクラウド環境に展開できるということである。これは、このソリューションが、複数のIoTクラウド環境においてデバイスの導入と利用が可能となるようにデバイスを記述できる柔軟性を持っていることを意味する。

これにより、1つのクラウドから他のクラウドへの既存のデバイスのマイグレーションが可能となるほか既存システムの中で新しいデバイスが容易に使用できるため、WoTデバイスの採用が推進される。

### 4.2.8ドメイン間協調

図13は、ドメイン間協調の例である。各システムは、スマートシティとスマートファクトリ、スマートホームとスマートシティというように、他のドメイン内の他のシステムを巻き込んでいる。この種のシステムは、[IEC-FOTF]に示されているように、「共生」エコシステムと呼ばれる。直接的な協調と間接的な協調の2つの協調モデルがあり、直接コラボレーションモデルでは、システムは、ピアツーピア方式で直接情報を交換する。間接的な協調では、システムは何らかの協調プラットフォームを介して情報を交換する。この協調を維持・継続するために、各システムは、その能力およびインターフェースのメタデータを提供するとともに、システムはそのメタデータを用いて他のシステムに順応する。

図13. ドメイン間協調

## 4.3 まとめ

前項では、様々なアーキテクチャパターンについて説明した。これらのパターンでは、レガシーデバイス、コントローラ、ゲートウェイ、クラウドサーバなどのデバイス等いくつかの機能エンティティは、屋内、屋外、データセンタなどに物理的に配置される。図14は、これらエンティティの組み合わせと通信パスを表す概観図である。

トランスポートプロトコル層では、各エンティティは、通信のためにふさわしい役割を任意に選択する。例えば、デバイスが不定数のアプリにサービスを提供する場合、デバイスはサーバとして動作することになる。一方、デバイスのネットワーク接続性が限られている、または、断続的な場合、デバイスはクライアントとして動作し、ネットワークが利用可能なときにアプリにメッセージを能動的に送信する。このような複雑さとは関係なく、アプリケーション層においては、アプリケーションは、デバイスが相互作用するための抽象インターフェースを提供し、アプリがその抽象インターフェースを使用してデバイスと相互作用することができるというようにみえる。

図14. ユースケースの俯瞰図

# ５　要件

本章は規定である。

## 5.1 機能要件

本項は、抽象Web of Things (WoT)アーキテクチャで必要となる性質を定義する。

### 5.1.1 共通原則

* WoTアーキテクチャは、Web技術を用いて異なるエコシステムの連携を可能にすべきである。
* WoTアーキテクチャは、RESTful APIを使用したWebアーキテクチャに基づくべきである。
* WoTアーキテクチャは、Webで一般的に使用される複数のペイロードフォーマットの使用が可能であるべきである。
* WoTアーキテクチャは、異なるデバイスアーキテクチャを使用可能にしなければならず、また、システムコンポーネントのクライアントあるいはサーバインプリメンテーションを強制してはならない。
* 柔軟性

WoTの実現には物理デバイス構成の多様なバリエーションが存在する。WoTの抽象アーキテクチャは、すべてのバリエーションにマッピングされることができ、対応できなければならない。

* 互換性

多くのビジネス分野において多くのIoTソリューションがあり、また、IoT標準化が進められている。WoTは、これらの既存または開発中のIoTソリューションと、WoTコンセプトに基づくWeb技術とをブリッジすべきである。WoTは、既存のIoTソリューションと現在の標準規格に対して、上位互換でなければならない。

* スケーラビリティ

WoTは、数千から数百万のデバイスを組み込んだIoTソリューションに対応できなければならない。これらのデバイスは、たとえ異なるメーカによって製造されたものに対しても機能を提供する。

* 相互接続性

WoTは、デバイスとクラウドの事業者を相互に接続できなければならない。WoT対応デバイスを使い、異なる事業者のクラウドサービスに接続できなければならない。

### 5.1.2 Thingの機能

WoTアーキテクチャは、Thingに下記の機能を与えること

* Thingのステータス情報を読み取る
* 動作を引き起こしうるThingのステータス情報の更新
* Thingのステータス情報への変更通知への購読、受信、購読停止
* 特定の動作または計算をトリガーする入出力パラメータを有する機能を呼び出す
* 単なるステータス情報への変更通知よりもより全般的なイベント通知の受け取り、受信、配信停止

### 5.1.3 検索・発見

* WoTアーキテクチャは、クライアントが、Thing自体にアクセスする前に、Thingの属性、機能、およびそれらのアクセスポイントを知ることができるようにするべきである。
* WoTアーキテクチャは、クライアントがその属性および機能によってThingを検索できるようにするべきである。
* WoTアーキテクチャは、機能の名称がいかなるものであっても、統一された語彙に基づいて必要な機能を提供するThingのセマンティック検索を可能にすべきである。

### 記述メカニズム

* WoTアーキテクチャは、Thing及びそれらの機能の記述を可能にする、共通の記述メカニズムをサポートすべきである。
* このような記述は、人間による読み取りが可能であるだけでなく、機械による読み取りが可能であるべきである。
* このような記述は、その構造および記述された内容のセマンティックなアノテーションを可能にすべきである。
* このような記述は、Webで一般に使用される複数のフォーマットを使用して交換できるべきである。

### 5.1.5 属性の記述

* WoTアーキテクチャは、下記のようなThingの属性を記述することができるべきである。
* 名称
* 説明
* スペック、フォーマット、記述自体のバージョン
* その他の、関連するThingとメタデータへのリンク
* このような記述は、国際化をサポートすべきである。

### 5.1.6 機能の記述

* WoTアーキテクチャは、「5.1.2 Thingの機能」に掲載されているThingの機能の記述を可能にすべきである。

### 5.1.7 ネットワーク

* WoTアーキテクチャは、一般に使用される複数のWebプロトコルをサポートすべきである。
* そのようなプロトコルには、下記が含まれる。

１． インターネット上で一般的に使用されているプロトコル

２． ローカルエリアネットワーク上で一般的に使用されているプロトコル

* WoTアーキテクチャは、同じ機能にアクセスするために複数のWebプロトコル使用を可能にすべきである。
* WoTアーキテクチャは、同じThingの機能に対して複数のプロトコルの組合せ(例：HTTPとWebSocket)の使用を可能にすべきである。

### 5.1.8 デプロイメント(配置)

* WoTアーキテクチャは、同じモデルに基づいて、リソース制限のあるエッジデバイスやクラウド上のVirtual Thingなど、多様なThingのもつ機能をサポートすべきである。
* WoTアーキテクチャは、ゲートウェイおよびプロキシなどの仲介者エンティティを有する複数レベルのThingの階層をサポートすべきである。
* WoTアーキテクチャは、ネットワークアドレス変換を考慮して、ローカルネットワーク外部(インターネットあるいはもう1つのローカルネットワーク)からローカルネットワーク内のThingへのアクセスをサポートすべきである。

### 5.1.9 アプリケーション

* WoTアーキテクチャは、同じモデルに基づくWeb標準技術を使用して、エッジデバイス、ゲートウェイ、クラウド、およびUI/UXデバイスなどの多様なThingのためのアプリの記述を可能にすべきである。

### 5.1.10 レガシー機器適用

* WoTアーキテクチャは、レガシープロトコルが終端され変換される場合、様々なトポロジをサポートしながら、レガシーのIPおよび非IPプロトコルからWebプロトコルへのマッピングを可能にすべきである。
* WoTアーキテクチャは、RESTfulアーキテクチャに従う既存のIPプロトコルを変換することなく透過的な使用を可能にすべきである。
* WoT アーキテクチャは、デバイスおよびサービスに対してクライアントまたはサーバ役を強制してはならない。IoTデバイスは、システムアーキテクチャに従い、クライアントまたはサーバあるいはその両方になることができ、これは、エッジサービスおよびクラウドサービスについても同様である。

## 5.2 技術要件

「4.2 共通パターン」では、様々なユースケースを提示し、アーキテクチャのコンポーネントの組み合わせパターンを列挙することで、WoTの抽象アーキテクチャを定義した。本節では、抽象アーキテクチャから導かれる技術要件について説明する。

### 5.2.1 Web of Thingsの構成要素とWeb of Thingsアーキクチャ

ユースケースでは基本的なコンポーネントとして、デバイス、デバイスにアクセスし制御するアプリケーション、デバイス間に位置するプロキシ(すなわち、ゲートウェイやエッジデバイス)を確認できた。また、いくつかのユースケースの中で有用な追加のコンポーネントは、デバイスの発見を助けるディレクトリである。

これらのコンポーネントは、インターネットあるいはオフィス、工場、その他施設内のフィールドネットワークに接続される。全てのコンポーネントは１つのネットワークに接続されることもあるが、通常は、複数のネットワークにまたがって接続できることにも留意すべき。

### 5.2.2 デバイス

デバイスへのアクセスは、それらの機能およびインターフェースの記述を使用して行われる。この記述をThing Description（TD)と呼ぶ。TDには、デバイスに関する一般的なメタデータ、機能を表す情報モデル、情報モデルを操作するためのトランスポートプロトコル記述、セキュリティの記述からなる。

一般的なメタデータは、デバイス識別子(URI)、シリアル番号、製造日、設置場所等の情報の他、利用者に必要なデバイス情報が含まれる。

情報モデルは、デバイス属性を定義し、デバイスの内部設定、制御機能、通知機能を表現する。同じ機能を有するデバイスは、使用するトランスポートプロトコルが異なっていても、同じ情報モデルを有する。

WoTアーキテクチャに基づくシステムの多くはドメインをまたがるシステムであるので、情報モデルで使用される語彙やメタデータ(例えば、オントロジ)は、関係者が共通に理解できるものであるべきである。RESTアーキテクチャで使用されるトランスポートに加えて、PubSubもサポートされる。

セキュリティ情報には、認証、認可、セキュア通信に関する記述が含まれている。デバイスは、TDをデバイスの内部または外部のいずれかに置き、他のコンポーネントがTDの検索を行い、アクセスできるようにすることが必要である。

### 5.2.3 アプリケーション

アプリケーションは、メタデータ記述に基づいてネットワークインターフェースおよびプログラムインターフェースを生成し、使用できなければならない。

アプリケーションは、ネットワークを通じてこれらのメタデータ記述を取得できなければならない。そのために、ネットワークを通じて検索を行い、必要なメタデータ記述を取得することができなければならない。

### 5.2.4 デジタルツイン

デジタルツインは、メタデータ記述に基づいて内部的に生成したプログラムインターフェースにより仮想デバイスを表現する必要がある。デジタルツインは仮想デバイスのメタデータ記述を生成し、それが外部から利用できるようにしなければならない。

仮想デバイスの識別子は、新たに割り当てる必要があるため、元のデバイス識別子とは異なる。これにより、仮想デバイスと元デバイスが別のエンティティとして明確に区別されることが保証される。仮想デバイスのトランスポートやセキュリティメカニズムおよび設定は、必要に応じて元デバイスとは異なるものであってもよい。仮想デバイスは自身のメタデータ記述を、デジタルツインにより直接、または外部から利用可能にする必要がある。いずれの場合も、他のコンポーネントがメタデータに紐付けられるデバイスを検出して使用できるように、メタデータ記述を利用可能にする必要がある。

### 5.2.5 ディスカバリ

デバイスと仮想デバイスのTDは、デバイス、アプリケーション、デジタルツインから参照可能となるように、TDを共有する仕組みが必要となる。ディレクトリサービスは、デバイスやデジタルツインが自身のメタデータ記述を、自動的に、あるいはユーザによる手動で登録する手段を持つ必要がある。

デバイスおよび仮想デバイスのメタデータ記述は、外部のエンティティから検索可能である必要がある。ディレクトリは、デバイス記述や情報モデルの一般的な記述に含まれるキーワードのような検索キーを使用して検索できなければならない。

### 5.2.6 セキュリティ

デバイスおよび仮想デバイスに関係するセキュリティ情報は、デバイス記述に記述されることが必要である。これには、認証、認可、ペイロード暗号化等の情報が含まれる。

WoTアーキテクチャは、Basic認証、Digest認証、Bearer認証、OAuth2.0認証のような、Webで一般的に使用されるセキュリティメカニズムを複数サポートするべきである。

### 5.2.7 アクセシビリティ

WoTは、主に、マシン対マシンの通信を対象とする。この通信に関わるユーザは、通常、Thingをアプリケーションで制御するソフトウェア開発者である。エンドユーザは、アプリケーションのフロントエンド、またはデバイス自体の物理的なユーザインターフェースを使用する。両方とも、W3C WoT仕様の範囲外である。本仕様はユーザではなくIoTに焦点を当てているため、アクセシビリティは直接的な要求事項ではなく、したがって本仕様では取り上げられない。

しかしながら、アクセシビリティには興味深い側面がある。上記の要件を満たすことで、マシンはデバイスのネットワークインターフェースを理解することが可能となる。異なるモダリティのユーザインターフェースを提供し、それによって、物理デバイスおよびIoT関連アプリを使用することに対する障壁を除去するために、アクセシビリティツールがこれを利用できる。

# ６　WoTアーキテクチャ

本章は規定である。

4章のユースケースに対応し、5章の要件を満たすために、Web of Things(WoT)は、いわゆるConsumerが使用できるWeb Things (以下ではThings)の概念の上に作られる。本章では、W3C WoTアーキテクチャを定義するための背景と規定アサーションについて説明する。WoTは、異なるドメインからのステークホルダ対応するため、Web技術、特にハイパーメディアの概念について詳細に説明する。

## 6.1 概要

Thingは、物理エンティティまたは仮想エンティティ(例えば、デバイスまたは部屋)の抽象であり、規定されるメタデータで記述される。W3C WoTでは、メタデータは、WoT Thing Description (TD) [WOT-THING-DESCRIPTION]で記述されなければならない。Consumerは、JSON [RFC8259]に基づくTDの記述フォーマットを解釈し、処理できなければならない。ベースとなる情報モデルはグラフベースであり、JSON-LD 1.1 [json-ld-syntax] と互換があるシリアライゼーションであるため、この記述フォーマットは従来のJSON ライブラリまたはJSON-LD モジュールのいずれかで処理できる。JSON-LDを使用するのは、追加としてRDFへの変換を含むセマンティック処理を実現するためである。Consumerが自律的に処理を進められるように、オントロジーに基づいてセマンテックインタフェースや実行すべき処理を付加する。TDは、インスタンス毎に定義(すなわち、Thingの型ではなく個々を記述する)され、外部のテキストによる（Web）表現である。HTMLによるユーザインターフェースや、単純に物理エンティティの画像であるとか、クローズなシステムのWebに基づかない表現でThingが表現してもよい。

しかし、Thingであるためには、TD表現が少なくとも1つ以上有効でなければならない。WoT Thing Descriptionは標準化された機械可読な表現形式であり、この表現はConsumerがThingと相互作用するときに、Thingのもつ機能を(セマンティックなアノテーションを介して)発見・解釈し、異なるインプリメンテーション(例えば、異なるプロトコルまたはデータ構造)であっても適用可能である。この機能によって、異なるIoTプラットフォーム、すなわち、異なるエコシステムや異なる標準でも相互接続を可能にする。

図15. ConsumerとThingとの相互作用

Thingは、仮想的なエンティティの抽象となることもできる。仮想的なエンティティは、1つまたは複数のThing(例えば、複数のセンサとアクチュエータのある部屋)で構成される。選択可能な構成の1つは、仮想エンティティののもつ機能を内包する、単一の統合されたWoT Thing Descriptionを提供することである。統合されたTDがかなり複雑な場合には、そのTDは階層的に下位に存在するThingにリンクしてもよい。上位のTDはエントリポイントととなり、一般的なメタデータや潜在的に包括する機能のみを記述する。これは、さらに複雑なThingの特定の様相をグループ化することができる。

リンクは、階層的なThingに適用されるだけでなく、Thingと他のリソースとの関係にも適用される。リンク関係のタイプは、複数のThingがどのように関係するかを表す。例えば、ライトを制御するスイッチや、運動センサで監視される部屋のような関係を示す。Thingに関連する他のリソースは、マニュアル、予備部品のカタログ、CADファイル、GUI、またはWeb上のドキュメントである。概して、Thing間のWebリンクは、人間と機械の両方にとってWeb of Thingsを誘導する。これは、利用可能なThingのTD表現を通常はキャッシュし、Thingのカタログを管理するThingのディレクトリを提供することでさらに促進される。つまり、WoT Thing Descriptionは、Web of Thingsを実現するためにWeb上の他のThingやリソースにリンクしてよい。

図16. リンクされたThing

Thingは、ネットワークインターフェース、すなわちThingのWoTインターフェースを介して、相互作用を実現するソフトウェアスタックをもつネットワーク化されたシステムコンポーネント上でホストしなければならない。一例としては、Thingの抽象表現が示す物理的なエンティティとしてつながる、センサやアクチュエータを中に持つデバイスで動作するHTTPサーバである。ただし、W3C WoTは、Thingがホストする場所を問わない。IoTデバイスが直接ホストすることもできるし、ゲートウェイなどのエッジデバイスやクラウドでホストすることもできる。

よくある配置上の課題となるシナリオは、IPv4ネットワークアドレス変換(NAT)またはファイアウォールデバイスがあるために、ローカルネットワークにあるThingにインターネットから接続できないことある。この状況を解決するために、W3C WoTは、ThingとConsumerとの間の仲介者を置くことができる。

仲介者はThingのプロキシとしてふるまい、元のThingと似たWoT Thing Descriptionと持つが、仲介者が提供するWoTインターフェースにつながるようにする。また、仲介者は、既存のThingに機能を追加したり、複数の利用可能なThingから新しいThingを合成し、それを新しい仮想エンティティとすることもできる。Consumerにとって、仲介者は、WoT Thing Descriptionを持ちWoTインターフェースを提供するのでThingのように見えるので、Web [REST]のような階層化されたシステムアーキテクチャ上ではThingと区別できないことがある。WoT Thing Descriptionに記述される識別子は、元のThingまたは物理的なエンティティを示す複数のTDとの関係を考慮しなければならない。

図17. 仲介者

制約のあるローカルネットワークに対する別の課題に対する解決は、WoTインターフェースが別のプロトコルに紐づけることである。このプロトコルは、ローカルネットワークに接続されるThingにつながるはずのConsumerへの接続を確立するプロトコルである。

Thing対Thingの相互作用を可能にするために、ThingはConsumerと一体化してもよい。通常、Consumerのふるまいは、Thingのふるまいを実行もするソフトウェアコンポーネントに埋め込まれる。Consumerのふるまいに関する情報は、「Thing」を介して公開してもよい。

W3C WoTのコンセプトは、IoTアプリケーションに関連するすべてのレベル、すなわち、デバイスレベル、エッジレベル、およびクラウドレベルにおいても適用可能である。これは、異なるレベルにわたって共通インターフェースやAPIを提供し、IoTアプリケーションのために、Thing対Thing、Thing対ゲートウェイ、Thing対クラウド、ゲートウェイ隊クラウド、さに、2つ以上のサービスプロバイダの相互接続されたクラウドコンピューティング環境のようなクラウド連携、といった多様な統合パターンを可能にする。図18は、上記で紹介されたWoTコンセプトが、どのように適用され、組み合わされて、4.3要約で示されたユースケースに対応することができるのか、その概要を表したものである。

図18. WoTの抽象アーキテクチャ

## 6.2 アフォーダンス

W3C WoTにおける中心的な側面は、マシン理解可能なメタデータ(すなわち、WoT TDの提供である。理想的には、そのようなメタデータは、コンシューマは、Thingがどのような機能を提供するのか、提供された機能をどのように使用するのかを判断することができるように自己記述的である。この自己記述性の鍵は、アフォーダンスの概念にある。

アフォーダンスという用語は生態学的心理学に由来するが、Donald Normanによる定義に基づいた人と機械との相互作用（Human-Computer Interaction [HCI]）の分野で採用された。「アフォーダンス（Affordance）」は、Thingが知覚される実際の特性を意味し、主に、そのThingがどのように使用され得るかを決定する基本的な特性である。【NORMAN】

例としては、ハンドルのついたドアである。ドアハンドルはアフォーダンスであり、ドアを開けることができることを示唆している。人間にとって、ドアハンドルは、通常、ドアをどのように開くことができるかを示唆する。アメリカのノブは回すように、ヨーロッパのレバーハンドルは押し下げるということを示唆する。

RESTアーキテクチャスタイル[REST]の中核基盤の1つであるハイパーメディア原理は、情報のコンシューマがWebのナビゲート方法とWebアプリ制御方法について系統だった知識を入手できるように、Web上で入手可能な情報を他の情報にリンクすることを要求する。ここで、情報と制御の同時表示(ハイパーリンク形式で提供される)は、Webアプリを駆動する手段をWebクライアントに提供する機構である。この意味では、アフォーダンスは、WebクライアントがWebアプリを駆動する手段を持てるようにするメカニズムである。どのようにナビゲートするか、おそらくはリンクされたリソースに対してどのようなアクションを取るかを示すハイパーリンク(例えば、リンク関係タイプとリンクターゲット属性を介する)の記述である。従って、リンクはナビゲーションアフォーダンスを提供している。

このハイパーメディア原理からわかるように、WoTは、相互作用アフォーダンス相互作用アフォーダンスをコンシューマに選択肢を示し説明するThingのメタデータとして定義しており、それによって、コンシューマがThingとどのように相互作用することができるかを提案する。一般的な相互作用アフォーダンス相互作用アフォーダンスはナビゲーションであり、これは以下のリンクをたどることによってアクティブ化され、それによって、コンシューマはThingのWebをブラウズすることをできる。6.4 相互作用モデルでは、W3C WoT用の更に３つの相互作用アフォーダンス相互作用アフォーダンスタイプ、すなわち、プロパティ、アクション、およびイベントを定義する。

全体として、このW3C WoT定義は、一般にWebサービスに取り組んでいるRESTおよびマイクロサービスコミュニティは、もちろん物理的Thingを作成するHCIや相互作用の設計者と整合が取れている。

## 6.3 Web Thing

Web Thingには、図19に示すように、ビヘイビア、相互作用アフォーダンス相互作用アフォーダンス、セキュリティ構成、およびプロトコルバインディングという4つのアーキテクチャ上の重要な側面がある。Thingのビヘイビア側面には相互作用アフォーダンス相互作用アフォーダンスのための自律的ビヘイビアとハンドラ両方が含まれる。相互作用アフォーダンス相互作用アフォーダンスは、特定のネットワークプロトコルまたはデータ符号化を参照せずに、コンシューマが抽象操作を行うことによってThingとどのように相互作用することができるかのモデルを提供する。プロトコルバインディングは、各相互作用アフォーダンス相互作用アフォーダンスを特定のプロトコルの具象メッセージにマッピングするのに必要な詳細を追加する。通常、単一のThingの中であっても、相互作用アフォーダンス相互作用アフォーダンスの異なるサブセットをサポートするために、異なる具象プロトコルを使用することができる。Thingのセキュリティ構成側面は、相互作用アフォーダンス相互作用アフォーダンスへ、および、関連するパブリックメタデータおよびプライベートメタデータの管理へのアクセスを制御するために使用されるメカニズムである。

図19. Thingのアーキテクチャ側面

## 6.4 相互作用モデル

元来、Webリソースは、Webクライアントが簡単に取り出すことができるWorld Wide Web上のドキュメントを意味していた。Webサービスの導入により、リソースは、どのような種類の振る舞いも実施することができるより一般的な相互作用エンティティとなった。この非常に高いレベルの抽象化は、その多数の相互作用の可能性のために、アプリケーションとリソース間に疎結合な仕組みを提供することを困難にしている。その結果、書込み時には、代表的なAPIの記述では、アプリケーションの意図するリソースのアドレス、メソッド、要求ペイロード、応答ペイロード、予想されるエラーを含む内容が、静的マッピングされて記述される。このため、WebクライアントとWebサービス間に緊密な結合が求められる。

このW3C WoTの相互作用モデルは、アプリケーションの目的から具体的なプロトコル操作のための手法を形式化するために中間処理を説明し、相互作用アフォーダンス相互作用アフォーダンスをモデル化するための可能性もまた説明する。

Webリンクというようなあて先を示すアフォーダンスに加えて、Thingは、本仕様で定義される3種類の相互作用アフォーダンス、すなわち、プロパティ、アクションを提供してもよい。ConsumerとThingとをナローウエスト（narrow waist）モデルで分離するが、これらの4種類の相互作用アフォーダンスでIoTデバイスとサービスにある可能な全ての相互作用をモデル化することができる。

### 6.4.1 プロパティ→NOTE：retrieve=取得、update=更新、observe=観察、push=通知

プロパティとは、Thingの状態を公開する相互作用アフォーダンスである。プロパティによって公開される状態は、取得可能でなければならない。プロパティによって公開される状態は更新可能(書き込み可能)であってもよい。Thingは、変更後に新しい状態を通知して、プロパティを観察可能にしてもよい。 (リソースの観察 [RFC7641]参照)。書込専用の状態は、アクションを介して更新すべきである。

使用されるプロトコルバインディングによって(例えば、メディアタイプを介して)、データの型が十分に明治されていない場合、プロパティは公開する状態のために1つのデータスキーマを有してもよい。

プロパティの例としては、センサ値(読み取り専用)、内部状態を持つアクチュエータ(読み書き可能)、設定パラメータ(読み書き可能)、Thingの状態(読み取り専用、または読み書き可能)、または、計算結果(読み取り専用)が挙げられる。

### 6.4.2 アクション

アクションは、Thingの関数を呼び出すことを可能にする相互作用アフォーダンスである。アクションは、直接公開されていない状態（“プロパティ”参照)の操作、一度に複数のプロパティの操作、内部ロジック(例：トグル)に基づいたプロパティの操作を行うこともできる。アクションを呼び出しにより、時間の経過とともに状態(アクチュエータによる物理的状態を含む)を操作するThingの上でプロセスをトリガーすることもできる。

データが、使用されるプロトコルバインディングによって(例えば、メディアタイプを介して)十分に指定されていない場合、アクションは、オプションの入力パラメータおよび出力結果のためのデータスキーマを持っても差し支えない。

アクションの例としては、複数のプロパティを同時に変更、光の明るさを落とす(薄暗くする)など時間とともに、また、独自の制御ループアルゴリズムなど開示されないプロセスでのプロパティ変更、または文書印刷など長期にわたるプロセス呼び出しが挙げられる。

### 6.4.3 イベント

イベント相互作用アフォーダンスは、Thingからコンシューマへ非同期にデータをプッシュするイベントソースを記述する。ここでは、状態ではなく、状態遷移(すなわち、イベント)が通信される。イベントは、プロパティとして公開されていない条件によってトリガーされ得る。

使用されるプロトコルバインディングによって(例えば、メディアタイプを介して) データが十分に指定されていない場合、イベントは、イベントデータおよび可能なサブスクリプション制御メッセージ(例えば、WebhookコールバックURIでサブスクライブする)のためのデータスキーマを含むことができる。

イベントの例としては、定期的にプッシュされるアラーム又は時系列のサンプルのような不連続なイベントがある。

## 6.5 ハイパーメディアコントロール

Web上では、アフォーダンスは、情報と制御の同時提示であり、その結果、ユーザがその情報を使って選択肢を取得するアフォーダンスになる。人間にとって、その情報は、通常、ハイパーリンクを記述または装飾するテキストあるいは画像である。制御はWebリンクであり、そこには少なくともターゲット・リソースのURIが入っており、Webブラウザによって参照解除することができる(すなわち、リンクをたどることができる)。しかし、マシンも、リレーションタイプおよびターゲット属性によってWebリンクがさらに記述されていれば、意味のある方法でリンクをたどることもできる。ハイパーメディア制御は、アフォーダンスをどのようにアクティブ化するかに関するマシン理解可能な記述である。ハイパーメディア制御は、通常、Webサーバから出されており、Webクライアントがサーバと相互作用している間に帯域内で検出される。このように、Webサーバは、クライアントの現在の状態および認可など他のファクターを考慮して、クライアントをダイナミックにWebアプリ上で活動させることができる。これは、クライアント(例えば、RPC、WS−\*、Webサービス、固定URIメソッド応答定義を持つHTTPサービス)にプリインストールまたはハードコードする必要がある帯域外インターフェース記述とは対照的である。

W3C WoTでは、Webをナビゲートするための確立した制御であるWebリンク[RFC8288]とあらゆる種類の操作を可能にするためのより強力な制御としてのWebフォームという2種類のハイパーメディアコントロールを利用している。リンクは、CoREリンクフォーマット[RFC6690]、OMA LWM2M [LWM2M]、OCF [OCF]など他のIoT規格およびIoTプラットフォームで既に使用されている。フォームは、W3C WoTに加えて、IETFによって定義されたConstrained RESTful Application Language (CoRAL) [CoRAL]も導入している新しい概念である。

### 6.5.1 リンク

リンクは、コンシューマ(または広義のWebクライアント)による、コンテキストとリンクターゲットとの関係に応じて、現在のコンテキスト(Webブラウザで現在レンダリングされているリソース表現参照。)　変更、あるいは、現在のコンテキストへのリソース追加を可能にする。コンシューマは、ターゲットURIを参照解除することによって、すなわち、リンクをたどりリソース表現をフェッチすることによって行うことができる。

W3C WoTは、Web Linking [RFC8288]の定義に従う。ここで、リンクは以下のものでから構成される。

* リンクコンテキスト、
* リレーションタイプ、
* リンクターゲット、
* 選択可能なものとして、ターゲット属性

リンク関係タイプは、ABNF [RFC5234] LOALPHA \* (LOALPHA / DIGIT / "." / "-") (例: stylesheet) 準拠のIANA [IANA-RELATIONS] に登録されているあらかじめ定義されているトークンか、URI [RFC3986] の形式の拡張タイプである。拡張関係型は、大文字小文字を区別することなく、文字列として比較されなければならない。(それらが異なるフォーマットでシリアル化されている場合は、URIに変換する。)　しかし、すべて小文字のURIが拡張リレーションタイプ用に使用されるべきである。[RFC8288]

WoTでは、リンクは、検出のため、また、ThingとThingの関係(例えば、階層または機能)と、Web上の他の文書との関係(例えば、マニュアル、CADモデルなどの代替表現)を表現するために使用される。

### 6.5.2 フォーム

フォームは、コンシューマ(または広義のWebクライアント)が、URIの参照解除より上位の動作を実行する(例えば、Thingの状態を操作する)ことを可能にする。コンシューマは、フォームに記入し、提出ターゲットに提出することによって上記の操作を行う。これには、通常、(要求)メッセージの内容について、リンクが提供できる以上の詳細な情報(例えば、メソッド、ヘッダフィールド、または他のプロトコルオプション)が要求される。

W3C WoTは、フォームを新しいハイパーメディア制御として定義する。CoRALにおける定義は、実質的に同一であり、したがって互換性がある[CoRAL]ということに留意されたい。フォームは、以下のもので構成される。

* フォームコンテキスト
* 操作タイプ
* サブミット・ターゲット
* 要求メソッド、および
* 任意で、フォーフィールド

フォームは、オプションのフォームフィールドが求められる要求をさらに記述する場合に、フォームコンテキスト上のオペレーションタイプの操作を実行する、サブミッションターゲットに要求メソッド要求を発行する」というステートメントとして表示することができる。

フォームコンテキストとサブミッションターゲットは、両方とも国際化リソース識別子(IRI)[RFC3987]でなければならない。しかし、多くのプロトコル(HTTPなど)はIRIをサポートしていないため、一般的には、URI[RFC3986]でもある。

フォームコンテキストおよびサブミッションターゲットは、同じリソースまたは異なるリソースを指す可能性があり、サブミッションターゲットリソースは、コンテキストの操作を実行する。

オペレーションタイプは、オペレーションのセマンティクスを識別する。操作タイプはリンク関係タイプと同様に表示される。

* 周知の操作タイプは、ABNF ALLOPHA \* (LOALPHA / DIGIT / "." / "-")に従わなければならない。周知の演算タイプは、大文字小文字を区別しないで比較しなければならない。本仕様で定義されているWoTの周知の操作タイプを表1のとおり。
* あらかじめ定義された操作タイプは、アプリによって選択された拡張操作タイプによって増補することができる。拡張操作タイプは、そのタイプのみを識別するURI[RFC3986]でなければならない。拡張操作タイプは、大文字小文字を区別せずに文字列として比較しなければならない。しかしながら、すべて小文字のURIが拡張操作タイプ用として使われるべきである。

要求メソッドは、サブミッションターゲットURIスキームによって識別される標準プロトコルの1つのメソッドを識別しなければならない。

フォームフィールドはオプションであり、さらに、与えられた操作のために期待されるリクエストメッセージを指定してもよい。これは、ペイロードに限定されず、プロトコルヘッダにも影響し得るということに留意されたい。フォームフィールドは、URIスキームで指定されているように、サブミッションターゲットに使用されているプロトコルに依存してもよい。例としては、HTTPヘッダフィールド、CoAPオプション、要求ペイロードのためのパラメータ(すなわち、フルコンテンツタイプ)を含むプロトコル非依存メディアタイプ、または予想される応答に関する情報などである。

表1. Web of Thingsの周知操作タイプ

（表は別途作成）

Editor’s note

本仕様のように、周知の操作タイプは、WoT相互作用モデルから生じる固定セットである。他の仕様は、それぞれのドキュメントフォーマットまたはフォームシリアライゼーションに有効な周知のオペレーションタイプを更に定義するかもしれない。本仕様の後のバージョンまたは別の仕様は、WoT仕様を超えて適用され得る拡張およびより包括的なWebフォームモデルを可能にするために、将来、IANAレジストリを設定することがあるかもしれない。

## 6.6 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象メッセージまでのマッピングである。これは、ネットワークインタ

ーフェースを介して、相互作用アフォーダンスをどのようにアクティブ化するかという情報をコンシューマ与えるものである。プロトコルバインディングは、相互接続性をサポートするために、REST [REST]の統一インターフェース制約を遵守している。したがって、すべての通信プロトコルがW3C WoTのためのプロトコルバインディングを実行する資格を有するわけではなく、本要件は以下のアサーションにより提示されている。

第6.2項アフォーダンスでドアを使った例では、ノブ対レバーのレベルでドアハンドルとプロトコルバインディングを対応させ、ドアはどのように開けられるのかを示唆している。

### 6.6.1 ハイパーメディア主導（ドリブン）

相互作用アフォーダンスは、1つあるいは複数のプロトコルバインディングを持っていなければならない。プロトコルバインディングは、相互作用アフォーダンスの起動方法に関し自己記述的であるために、ハイパーメディア制御としてシリアル化されなければならない(第6.5項ハイパーメディア制御参照)。ハイパーメディア制御は、対応する相互作用アフォーダンスを提供しているThingを管理するオーソリティのものでなければならない。オーソリティは、実行時にTDドキュメントを生成する(その現在の状態に基づいて、そのIPアドレスなどのネットワークパラメータを含む)か、または現在のネットワークパラメータのみが挿入された状態でメモリからそれを供給する、Thing自体とすることができる。オーソリティはまた、そのネットワークパラメータおよび内部構造(例えば、ソフトウェアスタック)

を含むThingの完全かつ最新の知識を有する外部エンティティとすることができる。これは、Thingとコンシューマとの間の緩い結合を可能にし、独立したライフサイクルおよび進化を可能にする。ハイパーメディアコントロールは、Thingの外部にキャッシュもでき、最新性を決定するためにキャッシュメタデータが利用可能である場合、オフライン処理のために使用もできる。

### 6.6.2 URIs

W3C WoTの適格なプロトコルは、IANA [RFC4395]に登録された関連URIスキームを持たなければならない。ハイパーメディアコントロールは、リンクおよびサブミッションターゲットを識別するためにURIに依存する。これにより、URIスキーム(「:」までの第1のコンポーネント)は、Thingとの相互作用アフォーダンスのために使用される通信プロトコルを識別する。W3C WoTは、これらのプロトコルを転送プロトコルと呼ぶ。

### 6.6.3 メソッドの標準セット

W3C WoTの適格なプロトコルは、演繹的に知られている標準的なメソッドセットに基づいていなければならない。メソッドの標準セットは、メッセージを自己記述的にして、例えばプロキシによる相互作用アフォーダンスの中間処理を可能にし、またはプロトコルバインディング[REST]間の変換を可能にする。さらに、コンシューマは、HTTP、CoAP、またはMQTTなどの共通転送プロトコルの再使用可能なプロトコルスタックを持つことができ、コンシューマのための特定のコードまたはプラグインを避ける。

### 6.6.4 メディアタイプ

相互作用アフォーダンスをアクティブ化する際に交換されるすべてのデータ(別名コンテンツ)は、プロトコルバインディング中のメディアタイプ [RFC6838]によって識別されなければならない。メディアタイプは、表現フォーマットを識別するためのラベルであり、例えば、JSON [RFC8259]の場合はapplication/json、CBOR [RFC7049]の場合はapplication/cborである。それらはIANAによって管理される。

いくつかのメディアタイプは、使用される表現フォーマットを完全に指定するために追加のパラメータを必要とする場合がある。例として、text/plain; charset=utf-8 またはapplication/ld+json; profile="http://www.w3.org/ns/json-ld#compacted"がある。これは、特に、Thingに送信されるデータを記述するときに考慮される必要がある。また、コンテンツ符号化[RFC7231]など、データ上に標準化された変換が存在する可能性がある。プロトコルバインディングはメディアタイプ単独というよりも詳細な表現形式を指定する追加情報を持っている場合もある。

多くのメディアタイプは、その要素(例えば、XML、JSON、CBOR)のためのさらなるセマンティクスを提供しない一般的なシリアライゼーションフォーマットを識別するだけであることに留意されたい。したがって、対応する相互作用アフォーダンスは、交換されるデータのより詳細な構文メタデータを提供するために、データスキーマを宣言すべきである。

## 6.7 WoTシステムコンポーネントと相互接続性

第6.1項概要は、Thing、コンシューマ、および仲介者などの抽象WoTアーキテクチャコンポーネントに関してWoTアーキテクチャを説明した。これらの抽象WoTアーキテクチャコンポーネントが、WoTアーキテクチャにおいて特定の役割を果たすソフトウェアスタックとして実装される場合、そのようなソフトウェアスタックは、サービアントと呼ばれる。WoTアーキテクチャのシステムは、システムの目標を達成するために互いに通信するサービアントを持っている。

本項では、システム構成図を使用して、WoTアーキテクチャのシステムを構築するためにサービアントがどのように協働するのかを説明する。

Thingは、サービアントによって実施することができる。Thingの中で、サービアントソフトウェアスタックは、公開されたThingと呼ばれるThingの描出を含み、そのWoTインターフェースをThingのコンシューマが利用できるようにする。この公開されたThingは、サービアント上の他のソフトウェアコンポーネント(例えば、アプリ)が使用でき、Thingのビヘイビアを実装することができる。

図20. Thingとしてのサービエント

一方、コンシューマは、TDフォーマットを処理できなければならず、TDに含まれるプロトコルバインディング情報を介して構成できるプロトコルスタックを持っていなければならないため、常にサービアント側によって実装される。

コンシューマ側では、サービアントソフトウェアスタックは、消費されるThingと呼ばれるThingの描出を提供し、それを、Thingと相互作用するためにTDを処理する必要があるサービアント上で実行されるアプリが利用できるようにする。

図21. コンシューマとしてのサービエント

サービアントソフトウェアスタック内の消費されるThingインスタンスは、プロトコルレベルの複雑さをアプリから分離する役割を果たし、アプリに代わって公開Thingと通信している。

同様に、仲介者は、サービアントによって実装されるもう一つのWoTアーキテクチャコンポーネントである。仲介者は、Thingとそのコンシューマとの間に配置され、コンシューマ(Thingに対し)とThing(コンシューマに対し)の両方の役割を果たす。仲介者側では、サービアントソフトウェアスタックは、コンシューマ(消費されるものThing)およびThing(公開されたThing)の両方となる。

図22. 仲介者としての役割

### 6.7.1 直接通信

図23は、TDを介して相互作用アフォーダンスを公開しているThingと、相互作用アフォーダンスによってThingを使用するコンシューマとの直接通信を示している。直接通信は、両方のサービアントが同じネットワークプロトコルを使用し、互いにアクセス可能であるときに適用される。

図23. ConsumerとThingのハイレベルアーキテクチャ

公開されたThingは、Thingによって提供される相互作用アフォーダンスのWoTインターフェースであり、Thingのアブストラクションのソフトウェア表現である。

消費されるThingは、コンシューマによって消費されているリモートのThingのソフトウェア表現であり、アプリのためのリモートのThingへのインターフェースとなる。コンシューマは、TDドキュメントを構文解析および処理することによって、消費されるThingのインスタンスを生成することができる。コンシューマとThingとの相互作用は、コンシューマと公開されたThingの直接的なネットワーク接続を上でメッセージを交換するコンシューマと公開されたThingが行う。

### 6.7.2 間接通信

図24において、コンシューマとThingは、仲介者を介して互いに接続している。サービアントが異なるプロトコルを使用する場合、あるいは、認証を要求し、アクセス制御を提供する異なるネットワーク上にある場合(ファイアウォールなど)は、仲介者が必要となる。

図24. 仲介者を使用した高レベルアーキテクチャ

仲介者は、公開されたThingと消費されるThingの機能を組み合わせる。仲介者の機能は、コンシューマとThingとの間の相互作用アフォーダンスのためのメッセージを中継することと、任意でThingのデータをより速い応答のためにキャッシュすることとThingの機能を仲介者が拡張するるときに通信を変換することなどである。仲介者側では、消費されるThingがThingの公開されたThingの代理オブジェクトを作成し、コンシューマは、それ自体の消費されるThingを介して代理オブジェクト(すなわち、仲介の公開されたThing)にアクセスすることができる。

コンシューマと仲介者は、仲介者とThing間とは異なるプロトコルで通信することができる。例えば、仲介者は、CoAPを使用するThingと、HTTPを使用するコンスーマのアプリとの間のブリッジを提供することができる。

仲介者とThing間で複数の異なるプロトコルが使用される場合であっても、コンシューマは、仲介者を介して単一のプロトコルを使用して、Thingと間接的に通信することができる。認証についても同様である。コンシューマの消費されるThingは、単一のセキュリティメカニズムを使用し、仲介者の公開されたThingとの認証だけが必要となる。一方、仲介者は、異なるThingで認証するために複数のセキュリティメカニズムを必要とする。

通常、仲介者は、元のThingのTDに基づいてその代理オブジェクトのTDを生成する。ユースケースの要件に応じて、代理オブジェクトのTDは、元のThingのTDと同じ識別子を使用するか、または、新しい識別子を割り当てられる。必要ならば、仲介者によって生成されたTDは、他の通信プロトコルのためのインターフェースを含むこともできる。

# ７　WoTビルディングブロック

本章は規定である。

WoTビルディングブロックは、抽象WoTアーキテクチャに準拠するシステムのインプリメンテーションを可能にする。これらのビルディングブロックの詳細は、別の仕様で定義され、本項では、概要および要約を述べる。

WoTビルディングブロックは、第6.3項で論じられ、第19図に描かれたThingのアーキテクチャ的側面の各要素をサポートする。個々のビルディングブロックは、第25図の抽象的なThingという形で表される。これは、抽象的な図であり、特定のインプルメンテーシを表すものではなく、ビルディングブロックとThingの主要なアーキテクチャ的側面との関係を示している。本図では、WoTビルディングブロックは黒い輪郭で強調表示されている。分野横断的な項目であるセキュリティは、パブリックおよび保護されたプライベートコンポーネントに分離されている。WoTスクリプティングAPIはオプションであり、バインディングテンプレートは情報を提供する。

図25. WoTビルディングブロックとThingのアーキテクチャ的側面の関係

以下の項では、各WoTビルディングブロックに関する追加情報、すなわち、WoT TD、WoTバインディングテンプレート、およびWoT スクリプティングAPIを説明する。セキュリティは、分野横断的な項目であるが、第4のビルディングブロックとみなすことができる。

## 7.1 WoT Thing Description

WoTTD仕様[WOT-THING-DESCRIPTION]は、セマンティックな語彙に基づく情報モデルとJSONに基づく直列表現を定義する。TDは、人間も読むことができ、機械も理解できる方法でThingの豊富なメタデータを提供する。TDの情報モデルと表現フォーマットは両方とも、リンクされたデータ[LINKED-DATA]と整列され、その結果、未加工のJSON処理に加えて、インプリメンテーションは、メタデータの強力なセマンティック処理を可能にするためにJSON-LD [JSON-LD11]およびグラフデータベースを利用することを選択することができる。

TDは、名前、ID、説明などの一般的なメタデータを有するThingのインスタンスを記述し、関係するThingまたは他のドキュメントへのリンクを介して関係メタデータを提供することもできる。TDは、また、パブリックセキュリティ構成メタデータとなる6.4相互作用モデルで定義された相互作用モデルに基づく相互作用アフォーダンスメタデータと、プロトコルバインディングを定義する通信メタデータとを含む。TDは、ハイパーメディアコントロールを使用して記述・提供されるサービスおよび関連するリソースを知るためのエントリポイントを提供するので、Thingのindex.htmlとみなすことができる。

理想的には、TDは、Thing自体によって作成および/あるいは提供され、検出時に取り出される。

しかし、Thingにリソース制限(例えば、制限されたメモリ空間、制限された電力)がある場合、または既存のデバイスがWoTの一部になるように改修されている場合には、外部的に提供されることも可能である。(例えば、制約されたデバイスのために)検出を改善し、デバイス管理を容易にするための一般的なパターンは、TDをディレクトリに登録することである。コンシューマは、通知メカニズムと組み合わされたTDキャッシングメカニズムを使用することが推奨され、その通知メカニズムは、Thingが更新された場合に、TDの新しいバージョンをフェッチが必要のときに通知をしてくれる。

セマンティックな相互接続性のために、TDは、明らかな拡張ポイントが提供されるドメイン固有の語彙を利用することができる。しかしながら、特定のドメイン固有語彙の開発は、現在、W3C WoT標準化活動の範囲外である。

潜在的に有用な外部IoT語彙の3つの例は、SAREF [SAREF]　iot.schema.org[iot-schema-org]、W3C Semantic Sensor Network ontology [vocabssn]となっている。TDにおけるこのような外部語彙の使用は任意である。将来、追加のドメイン特有の語彙が開発され、TDと共に使用される可能性がある。

全体として、WoTTDビルディングブロックは、2つの方法で相互接続性を促進する:第1に、TDは、WoTにおけるマシン対マシンの通信を可能にする。第2には、TDは、開発者が、IoTデバイスにアクセスし、そのデータを利用することができるアプリを作成するために必要なすべての詳細を文書化し、取り出すための共通の統一フォーマットとなることができる。

## 7.2 WoTバインディングテンプレート

本章は標準規格に含まれない。

すべてのコンテキストに適合するプロトコルがないため、IoTは、デバイスにアクセスするために様々なプロトコルを使用する。したがって、WoTにとっての中心的な課題は、特定の標準に従わないが、適切なネットワークプロトコルを介して適切なインターフェースを提供する過剰な異なるIoTプラットフォーム(たとえば、OCF、oneM2M、OMA LWM2M、OPC UA)およびデバイスとの相互作用を可能にすることである。WoTは、いくつかの制約を満たさなければならないプロトコルバインディングを通じて、この多様性に取り組んでいる(第6.6項プロトコルバインディング参照)。

標準規格に含まれないWoTバインディングテンプレート仕様[WOT-BINDING-TEMPLATES]は、異なるIoTプラットフォームとの相互作用方法に関するガイダンスを提供する通信メタデータブループリント集を提供している。特定のIoTデバイスまたはサービスを記述する場合、対応するIoTプラットフォームのバインディングテンプレートを使用して、そのプラットフォームをサポートするためにTDで提供されなければならない通信メタデータを検索することができる。

図26. バインディングテンプレートからプロトコルバインディングへ

図26は、バインディングテンプレートがどのように適用されるかを示している。WoTバインディングテンプレートは、各IoTプラットフォームに対して一度だけ作成され、そのプラットフォームのデバイスに対する全てのTDにおいて再使用することができる。TDを処理しているコンシューマは、対応するプロトコルスタックを含め、TDで与えられた情報に従ってスタック(またはそのメッセージ)を構成することによって、必要なプロトコルバインディングを実装しなければならない。

プロトコルバインディングの通信メタデータには5つ要素がある。

* IoT プラットフォーム:

IoTプラットフォームは、しばしば、プラットフォーム固有のHTTPヘッダフィールドまたはCoAPオプションなどの独自仕様の変更をアプリケーション層で導入する。フォーム(第6.5.2項フォーム参照)は、使用されるアプリケーション層プロトコルのために定義された追加フォームフィールドにこれらの微調整を適用するために必要な情報を含むことができる。

* メディアタイプ:

IoTプラットフォームは、しばしば、データを交換するために使用される表現フォーマット(シリアル化としても知られる)が異なる。メディアタイプ[RFC6838]は、これらのフォーマットを識別するが、パラメータは、それらをさらに指定することができる。フォームは、HTTPからわかるコンテンツタイプフィールドなどの追加フォームフィールドにメディアタイプおよびオプションのパラメータを含むことができ、HTTPは、メディアタイプおよび他のオプションのパラメータ(たとえば、text/plain; charset=utf−8)を組み合わせる。

* 転送プロトコル:

WoTは、アプリ固有のオプションまたはサブプロトコルメカニズムを持たない基礎となる標準化されたアプリケーション層プロトコルのための転送プロトコルという用語を使用する。フォーム(サブミッション)ターゲットのURIスキームには、転送プロトコルを識別するのに必要な情報、例えば、HTTP、CoAP、またはWebSocketが含まれる。

* サブプロトコル:

転送プロトコルは、うまく相互作用することがわかっている拡張メカニズムをもているかもしれない。そのようなサブプロトコルは、URIスキームから識別することができず、明示的に宣言されなければならない。例としては、ロングポーリング[RFC6202]やServer-SentEvent[EVENTSOURCE]などのHTTPのプッシュ通知回避策がある。フォームは、追加のフォームフィールド内のサブプロトコルを識別するために必要な情報を含んでいることがある。

* セキュリティ:

セキュリティメカニズムは、通信スタックの様々な層に適用することができ、しばしば相互補完するために合わせて使用されることがある。例は、(D)TLS [RFC8446]/[RFC6347]、IPSec [RFC4301]、OAuth [RFC6749]、およびACE [RFC7744]である。セキュリティの分野横断的性質故に、適切なメカニズムを適用するために必要な情報をThingの一般的なメタデータ内で入れることもできる。

## 7.3 WoT Scripting API

本章は標準規格に含まれない。

WoTスクリプティングAPIは、WebブラウザAPIと同様のECMAScriptベースのAPI [ECMAScript]を提供することによってIoTアプリ開発を容易にするW3C WoTのオプションの“簡便性”ビルディングブロックである。スクリプティングランタイムシステムをWoTランタイムに結合することによって、WoTスクリプティングAPIは、Thing、コンシューマ、および仲介者のビヘイビアを定義するポータブルアプリスクリプトの使用が可能となる。

従来、IoTデバイスロジックは、ファームウェアで実装され、その結果、比較的複雑な更新プロセスなどの組み込み開発と同様の生産性制約が生じる。対比サポートにおけるWoTスクリプティングAPIは、Webブラウザと同じIoTアプリのためのランタイムシステムで実行される再利用可能なスクリプトによるデバイスロジックインプリメンテーションをサポートし、生産性の向上と結合コストの削減を目指す。さらに、標準化されたAPIは、アプリモジュールの移植性を可能にし、例えば、計算専用ロジックをデバイスからローカルゲートウェイに移動させる、あるいは、タイムクリティカルロジックをクラウドからゲートウェイまたはエッジノードに移動させることができる。

標準規格に含まれないWoTスクリプティングAPI仕様「WOT-SCRIPTING-API]は、スクリプトがWoTTDを検出、フェッチ、消費、生成、公開することを可能にするプログラミングインターフェースの構造とアルゴリズムを定義する。WoTスクリプティングAPIのランタイムシステムは、他のThingおよびそれらの相互作用アフォーダンス(プロパティ、アクション、およびイベント)へのインターフェースとして作用するローカルオブジェクトのインスタンスを生成する。また、スクリプトがThingを公開すること、すなわち、相互作用アフォーダンスを定義・実装し、Thingの記述を公開することも可能にする。

## 7.4 WoTセキュリティとプライバシのガイドライン

本節は標準規格に含まれない。

セキュリティは分野横断的な項目であり、システム設計のあらゆる側面において考慮されるべきである。WoTアーキテクチャでは、セキュリティは、TD内のパブリックセキュリティメタデータのサポートのような明示的な特性、および、WoTスクリプティングAPIの設計の中で問題を分離することによってサポートされている。各ビルディングブロックの仕様には、また、そのビルディングブロック特有のセキュリティおよびプライバシ考慮事項考察が含まれている。別の標準規格でない仕様であるWoT Security and Privacy Guideline [WOT-SECURITY]は、さらなる分野横断的なセキュリティおよびプライバシーガイダンスを提供している。

# ８　抽象Servientアーキテクチャ

本章は参考情報である。

6.7節 WoTシステムコンポーネント及びそれらの相互接続性において定義されるように、8サービアントとは、前項において提示されたWoTビルディングブロックを実装するソフトウェアスタックである。サービアントは、Thingを提供・公開することができ、かつ/あるいは、Thingを消費することができる(すなわち、コンシューマを提供する)。プロトコルバインディングに応じて、サービアントはサーバとクライアントの両方の役割を演じることができ、両開きカバン的な命名となっている。

前項では、WoTビルディングブロックが概念的に互いにどのように関係し合うのか、また、それらが抽象WoTアーキテクチャにどのように対応するのかを説明している(6章 WoT Architecture参照のこと)。これらの概念を実装する場合、特定の技術的側面を考慮に入れたより詳細な概説が必要となる。本項では、サービアント実装の詳細なアーキテクチャについて説明する。

図27は、(オプションの) WoTスクリプティングAPIビルディングブロックを使用しているサービアントインプリメンテーションである。ここで、WoTランタイムは、WoT固有の側面を管理することに加えて、アプリスクリプトを解釈・実行するスクリプティングランタイムシステムでもある。WoT スクリプティングAPI をサポートするサービアントは、通常、高度なデバイス、エッジノード、または、クラウド上で実行される。WoTアーキテクチャは、WoTランタイムのアプリAPIをJavaScript/ECMAScriptに限定しない。また、他のランタイムシステムを使用して、サービアントを実装することも可能である。

第8.8.1項ネイティブWoT APIは、WoTスクリプティングAPIビルディングブロックなしで、代替のサービアントインプリメンテーションを提示している。WoTランタイムは、そのアプリAPIのために任意のプログラミング言語を使用することができる。通常、これは、サービアントソフトウェアスタックのネイティブ言語であり、例えば、組み込まれたサービアントの場合はC/C++、クラウドベースのサービアントの場合はJavaである。また、アプリスクリプトの利点を低リソース消費と組み合わせるには、Luaなどの代替スクリプト言語であってもよい。

図27. スクリプティングAPIを使用したサービエントのインプリメンテーション

図27の各モジュールの役割および機能は、以下の項で説明される。

## 8.1 ビヘイビアのインプリメンテーション

ビヘイビアは、Thingの全体的なアプリケーションロジックを定義する。これにはいくつかの側面がある。

これは、Thingの自律的ビヘイビア(例えば、アクチュエータのセンサまたは制御ループのサンプリング)、相互作用アフォーダンスのハンドラ(すなわち、アフォーダンスがアクティブ化されたときに行われる具体的なアクション)、コンシューマビヘイビア(例えば、Thingを制御する、または、マッシュアップを実現する)、および仲介者ビヘイビア(例えば、単にThingを代理する、または、仮想エンティティを構成する)などである。サービアント内のビヘイビアインプリメンテーションは、どのThing、コンシューマ、および仲介者がこのコンポーネント上で提供されるかを定義する。

図27は、JavaScript [ECMAScript]で書かれたポータブルアプリスクリプトがビヘイビアを定義するオプションのWoTスクリプティングAPIビルディングブロックを実装しているサービアントである。これらは、WoTランタイムの一部であるスクリプティングランタイムシステムによって実行される(WoTスクリプティングAPIまたは任意の他のスクリプトベースのAPIを提供する場合)。これらは、共通のWoTスクリプトAPI定義に対応するように書かれているので移植可能であり、したがって、このビルディングブロックを持つすべてのサービアントが実行することができる。これにより、例えば、コンシューマをクラウドからエッジノードに移動してネットワーキング要件を満たすなど、システムコンポーネント間でアプリロジックを移動させること、または、増大するリソース需要を満たすために仲介物をクラウドに移動させることが可能になる。ポータブルアプリは、サービアントの配備後に追加のビヘイビアを「インストール」することを可能にする。

原則として、相互作用アフォーダンスがWoTインターフェースを介して外部にある限り、あらゆるプログラミング言語およびAPIを使用して、Thingのビヘイビアを定義することができる。アプリAPIとプロトコルスタック間の適応は、WoTランタイムによって処理される。WoT スクリプティングAPI ビルディングブロックを使用しないビヘイビアインプリメンテーションについては、第8.8.1項ネイティブWoT API参照。

## 8.2 WoTランタイム

基本的に、Thingの抽象化およびその相互作用モデルは、ランタイムシステムにおいて実装される。このWoTランタイムは、ビヘイビアインプリメンテーションの実行環境を維持し、Thingを公開および/または消費することができる。したがって、WoTTDをフェッチし、処理し、直列化し、供給することができなければならない。

あらゆるWoTランタイムは、ビヘイビアインプリメンテーションのためのアプリインターフェース(すなわち、API)を持っている。図27のオプションWoTスクリプティングAPIビルディングブロックは、Thingのアブストラクションに従い、アプリスクリプトを介したランタイム中のビヘイビアインプリメンテーションの展開を可能にする、そのようなアプリインターフェースを定義している。コンパイル時間中にのみ使用することができる代替APIに関しては、第8.8.1項ネイティブWoT API参照。一般的に、アプリロジックは、WoTランタイムの管理側面、特にプライベートセキュリティ構成への無許可アクセスを防止するために、分離された実行環境で実行されるべきである。マルチテナントサービアントでは、実行環境分離がテナントごとに更に要求される。

WoTランタイムは、Thingのライフサイクル、より正確には、それらのソフトウェアブストラクションおよび記述を管理するために、特定の動作を提供する必要がある。ライフサイクル管理(LCM)システムは、これらのライフサイクル動作をサービアント内にカプセル化し、内部インターフェースを使用してライフサイクル管理を実現することができる。そのような動作の詳細は、インプリメンテーションにより様々である。WoTスクリプティングAPIは、LCM機能を持ち、したがって、そのようなシステムのインプリメンテーションの可能例となっている。

WoTランタイムは、ビヘイビアインプリメンテーションをプロトコルバインディングの詳細から切り離すので、サービアントのプロトコルスタックインプリメンテーションとインターフェースしなければならない。WoTランタイムは、また、通常、例えば、取り付けられたセンサおよびアクチュエータなどのローカルハードウェアにアクセスするために、あるいは、ストレージなどのシステムサービスにアクセスするために、基本システムとインターフェースする。両方のインターフェースは、インプリメンテーション固有であるが、WoTランタイムは、実装されたThingのアブストラクションに必要となる適応性を提供しなければならない。

## 8.3 WoTスクリプティングAPI

WoT スクリプティングAPIビルディングブロックは、WoT TD仕様「WOT-THING-DESCRIPTION]を遵守するECMAScript APIを定義する。これは、ビヘイビアインプリメンテーションとスクリプトベースのWoTランタイム間のインターフェースを定義するものである。他のより単純なAPIは、例えば、WebブラウザAPIのjQueryと同様に、重ねて実装することができる。

詳細は「WOT-SCRIPTING-API]参照。

## 8.4 公開されたThingと消費されるもののアブストラクション

WoTランタイムは、TDに基づいてThingのソフトウェア表現のインスタンスを生成する。これらのソフトウェア表現は、ビヘイビアインプリメンテーションに対するインターフェースとなる。

公開されたThingのアブストラクションは、ローカルに提供され、サービアントのプロトコルスタックインプリメンテーションによって、外部からアクセス可能なThingを表す。ビヘイビアインプリメンテーションにより、そのメタデータおよび相互作用アフォーダンスを定義し、その自律的ビヘイビアを提供することによって、公開されたThingは完全制御可能となる。

消費されるThingのアブストラクションは、通信プロトコルを使用してアクセスする必要があるコンシューマのためのリモートで提供されるThingを表す。消費されるThingは代理オブジェクトまたはスタブである。ビヘイビアインプリメンテーションは、対応するTDに記載されている通りにそのメタデータを読み取り、その相互作用アフォーダンスをアクティブ化することに限定されている。また、消費されるThingは、独自仕様またはレガシー通信プロトコルの背後にあるローカルハードウェアまたはデバイスなどのシステム機能を表すこともできる。この場合、WoTランタイムは、システムAPIと消費されるThingの間の必要な適応性を提供しなければならない。さらに、それは、対応するTDを提供し、例えば、WoTランタイムがどのような検出メカニズムを提供しているのかにかかわらず、アプリAPI (例えば、WoTスクリプティングAPI 「WOT-SCRIPTING-API]で定義されるdiscover()メソッド)を使って拡張することによって、ビヘイビアインプリメンテーションがそれらを利用できるようにしなければならない。

WoTスクリプティングAPI を使用する場合、公開されたThing および消費されるThingはJavaScript オブジェクトであり、アプリスクリプトによって生成、操作、および破棄することができる。しかしながら、アクセスは、例えば、マルチテナントサービエントにおいて、セキュリティメカニズムにより制限され得る。

## 8.5 プライベートセキュリティデータ

また、プライベートセキュリティメタデータは、概念的にはWoTランタイムによって管理されるが、意図的にアプリに直接アクセス可能とはされない。実際、最も安全なハードウェアインプリメンテーションでは、そのようなプライベートセキュリティデータは、別個の隔離されたメモリ(例えば、安全なエレメントまたはTPM上)に格納され、攻撃対象領域を限定し、このデータの外部開示を防止する(場合によっては、隔離されたプロセッサおよびソフトウェアスタックによって実装されていることさえある)抽象的な動作セットのみが提供される。プライベートセキュリティデータは、プロトコルバインディングによって透過的に使用され、インタラクションの整合性および機密性を許可および保護する。

## 8.6 プロトコルスタックのインプリメンテーション

サービエントのプロトコルスタックは、公開されたThingのWoTインターフェースを実装し、(消費されるThingを介して)遠隔のThingのWoTインターフェースにアクセスするためにコンシューマが使用する。これは、ネットワーク上で相互作用するための具体的なプロトコルメッセージを生成する。サービエントは、複数のプロトコルを実装し、したがって、様々なIoTプラットフォームとの相互作用を可能にするために複数のプロトコルバインディングをサポートすることができる。

標準プロトコルが使用される場合の多くは、プラットフォーム固有のメッセージ(例えば、HTTP(S)方言用、CoAP(S)方言用、MQTTソリューション用など)を生成するために、汎用プロトコルスタックを使用することができる。この場合、TDからの通信メタデータが、適切なスタック(例えば、適切なヘッダフィールドを有するHTTPまたは適切なオプションを有するCoAP)を選択・構成するために使用される。メディアタイプ[RFC2046]によって識別される期待されるペイロード表現フォーマット(JSON、CBOR、XMLなど)のパーサおよびシリアライザも、これらの汎用プロトコルスタックで共用することができる。

詳細は[WOT-BINDING-TEMPLATES] 参照。

## 8.7 システムAPI

WoTランタイムが実装されると、あたかも通信プロトコルを介してアクセス可能であるかのように、Thingのアブストラクションを介してビヘイビアインプリメンテーションにローカルハードウェアまたはシステムサービスを提供することができる。この場合、WoTランタイムは、ビヘイビアインプリメンテーションが、プロトコルスタックの代わりにシステムと内部的にインターフェースする消費されるThingのインスタンスを生成できるようにしなければならない。これは、アプリWoTランタイムAPIが提供する検出メカニズムにより、ローカルWoTランタイムでのみ利用可能であるシステムのThingをリストにすることによって行うことができる。

デバイスは、また、物理的にサービアントの外部にあってもよいが、独自のプロトコル、または、WoTインターフェースとして不適格なプロトコルを介して接続もできる(プロトコルバインディングについては、第6.6項参照)。この場合、WoTランタイムは、独自のAPIを介してそのようなプロトコル(たとえば、ECHONET Lite、BACnet、X10、I2C、SPIなど)を有するレガシーデバイスにアクセスすることができるが、Thingのアブストラクションを介してビヘイビアインプリメンテーションにThingを公開することを選択することができる。そうすると、サービアントは、レガシーデバイスへのゲートウェイとして作用することができる。これは、レガシーデバイスがWoT TDを使用して直接記述できない場合にのみ行われるべきである。

ビヘイビアインプリメンテーションは、また、独自仕様のAPIまたは他の手段を介して、ローカルハードウェアまたはシステムサービス(例えば、ストレージ)にアクセスすることもできる。しかしながら、これは、移植性を妨げるため、W3C WoT標準化の範囲外である。

## 8.8 代替サービエントおよびWoTインプリメンテーション

WoTスクリプティングAPI ビルディングブロックはオプションである。WoTランタイムが、様々なプログラミング言語で書かれることもあるアプリロジックのための代替APIを提供する場合、代替的なサービアントインプリメンテーションが可能である。

さらに、W3C WoTを認識しないデバイスまたはサービスは、そのために適切に形成されたWoT TDを提供することが可能である場合も、消費されることが可能である。この場合、TDは、ブラックボックスインプリメンテーションを有するThingのWoTインターフェースを記述する。

### 8.8.1 ネイティブWoT API

開発者がWoTスクリプティングAPIを使用せずにサービアントを実装することを選択する理由は様々である。メモリまたはコンピューティングリソースが不十分で、開発者は必要なソフトウェアスタックまたはフル機能のスクリプトエンジンを使用することができないということに起因する可能性もある。あるいは、ユースケース(例えば、独自の通信プロトコル)をサポートするために、開発者は、特定のプログラミング環境または言語を介してのみ利用可能な特定の機能またはライブラリを使用しなければならない場合がある。

この場合、WoTランタイムは依然として使用することができるが、WoTスクリプティングAPIの代わりに、代替のアプリインターフェースを使用して公開される同等のアブストラクションおよび機能性が必要となる。その点を除き、第８項サービアントインプリメンテーションのブロック記述はすべて図28でも有効である。

図28. ネイティブAPIを使用したサービエントのインプリメンテーション

### 8.8.2 TDで記述される既存デバイス

既存のIoTデバイスまたはサービスをThingのW3CWebに結合し、これらのデバイスまたはサービスのTDを作成することによって、それらをThingとして使用することも可能である。このようなTDは、手動で、または、ツールまたはサービスを使って作成することができる。例えば、TDは、別のエコシステム依存の機械読み取り可能フォーマットによって提供されるメタデータの自動翻訳を提供するサービスによって生成することができる。ただし、これは、ターゲットデバイスがプロトコルバインディングを使用して記述できるプロトコルを使用している場合にのみ行うことができる。その要件は、第6.6項プロトコルバインディングで述べられている。前述の考察の多くが、Thingそれ自体のTDを提供することを意味する。これは有用なパターンであるが、必須ではない。特に、そのTDを直接提供するために既存のデバイスを改修することはできないであろう。この場合、TDは、ディレクトリまたは他の外部的な別個の配布メカニズムなどのサービスを使用して別途提供されなければならない。

図29. 既存IoTデバイスのW3C WoTへの結合

# ９　WoTの導入

本章は標準規格に含まれない。

本項では、Thingおよびコンシューマを実装するデバイスおよびサービスが様々なトポロジおよび展開シナリオで接続されるときに、WoTが全体としてどのように機能するのかを考察する。

特定のトポロジを論じる前に、まず、WoTネットワークにおいてThingとコンシューマが果たすことができる役割、および、公開されたThingと消費されるThingのソフトウェアブストラクションとの関係を検討する。公開されたThingと消費されるThingは、サービアントのビヘイビアインプリメンテーションがそれぞれ、Thingおよびコンシューマ役として内部的に利用できる。

## 9.1 Thingのコンシューマの役割

Thing役のサービアントは、TDに基づいて公開されたThingを作成する。TDは、公開され、コンシューマまたは仲介者役の他のサービス提供者が利用できる。TDは、Thingのディレクトリサービスなどの管理システムに登録することができ、または、TDに対する要求を受信すると、Thingは要求側にTDを提供する。特定のアプリシナリオでは、TDをThingにスタティックに関連付けることさえ可能である。

コンシューマ役のサービアントは、検出メカニズムを使用してThingのTDを取得し、取得したTDに基づいて消費されるThingを作成する。具体的な検出メカニズムは、個々の展開シナリオによるが、スタティックな割り当てなどでThingのディレクトリ、検出プロトコルなどの管理システムにより提供することができる。設置されたセンサおよびアクチュエータとの相互作用など、デバイスの内部システム機能も、任意選択で、消費されるThingのアブストラクションとして表すことができる。

消費されるThingによってサポートされる機能は、プログラミング言語インターフェースを介してコンシューマのビヘイビアインプリメンテーションに提供される。WoTスクリプティングAPIでは、消費されるThingはオブジェクトによって表わされる。Thingの中で実行されるビヘイビアインプリメンテーション(すなわち、アプリロジック)は、公開されたThingによって提供されるプログラミング言語インターフェースを使用することによって、相互作用アフォーダンスを介してコンシューマと連携することができる。

Thingは、必ずしも物理的デバイスを表すということではない。Thingは、ゲートウェイまたはクラウド上で実行されるデバイスまたは仮想サービスの集合を表すこともできる。同様に、コンシューマは、ゲートウェイまたはクラウド上で実行されるアプリまたはサービスを代理することができる。コンシューマをエッジデバイス上に実装することもできる。仲介者側では、単一のサービアントが、単一のWoTランタイムを共用しているThingとコンシューマ両方の役割を同時に実行する。

## 9.2 WoTシステムのトポロジと展開シナリオ

WoTシステムの様々なトポロジおよび展開シナリオを本項で考察する。これらはパターン例にすぎず、他の相互接続トポロジも可能である。ここで説明するトポロジは、WoTのユースケース(4章ユースケース)と、そこから抽出された技術的要件(第５項要件)から導き出されるものである。

### 9.2.1 同一ネットワーク上のコンシューマとThing

図30に示す最も単純な相互接続トポロジでは、コンシューマとThingは同じネットワーク上にあり、仲介者なしで互いに直接通信することができる。このトポロジが生まれる1つのユースケースは、コンシューマが、ゲートウェイ上で実行されるオーケストレーションサービスまたは何らかの他のIoTアプリであり、Thingがセンサまたはアクチュエータにインターフェースするデバイスである場合である。しかしながら、クライアント/サーバ関係は、容易に逆転することができ、クライアントが、ゲートウェイ上でまたはクラウド上でThingとして、実行されるサービスにアクセスするコンシューマ役のデバイスであることも可能である。

図30. 同一ネットワーク上にあるコンシューマとThing

Thingがクラウド上にあり、コンシューマがローカルネットワーク上にある場合(Smart Homeのユースケースについては図1参照)、実際のネットワークトポロジは、例えば、NATトラバーサルを必要とし、特定の検出形態を許可しないなど、より複雑になる可能性がある。このような場合、後述のより複雑なトポロジのうちのいずれかがより適切であろう。

### 9.2.2 仲介者を介して接続されたコンシューマとThing

仲介者は、ネットワーク上でThingとコンシューマの両方の役割を果たし、そのWoTランタイム内で公開されたThingと消費されるThing両方のソフトウェアブストラクションをサポートする。仲介者は、デバイスとネットワーク間でプロキシ、またはデジタルツインのために使用することができる。

#### 9.2.2.1 プロキシとして動作する仲介者

仲介者の簡単なアプリの1つのは、Thingのためのプロキシである。仲介者がプロキシとして動作する場合、仲介者は、2つの別個のネットワークまたはプロトコルとのインターフェースを有する。これは、TLSエンドポイントを提供するなど追加のセキュリティメカニズムのインプリメンテーションを含むことができる。通常、プロキシは、相互作用セットを変更することはない。したがって、仲介者によって公開されたTDは、消費されるTDと同じ相互作用となるが、接続メタデータは変更される。

このプロキシパターンを実装するために、仲介者はThingのTDを取得し、消費されるThingを生成する。これは、同じ相互作用アフォーダンスを有するソフトウェアインプリメンテーションとしてThingのプロキシオブジェクトを生成する。次いで、新しい識別子を有し、場合によっては新しい通信メタデータ(プロトコルバインディング)および/あるいは新しいパブリックセキュリティ構成メタデータを有するプロキシオブジェクトのためのTDを生成する。最後に、公開されたThingがこのTDに基づいて生成され、仲介者は適切な公開メカニズムを介してTDの他のコンシューマまたは仲介者に通知する。

図31. プロキシとして仲介者を介したコンシューマとThingの接続

#### 9.2.2.2 デジタルツインとして動作する仲介者

より複雑な仲介者は、デジタルツインと呼ばれる。デジタルツインは、プロトコル変更あるいはネットワーク間での変換をすることもしないこともあるが、状態キャッシング、延期された更新、またはターゲットデバイスのビヘイビアの予測シミュレーションなど追加のサービスを提供する。例えば、IoTデバイスの電力が制限されている場合、IoTデバイスは、比較的スリープ解除回数を少なくし、デジタルツインと同期し、直ちに再びスリープに入ることを選択するかもしれない。この場合、通常、デジタルツインは、(クラウド内またはゲートウェイ上などの)より電力の制約の少ないデバイス上で動作し、制約のあるデバイスに代わって相互作用に応答することができる。プロパティの現在の状態に関する要求は、キャッシュされた状態を使用してデジタルツインが行なうかもしれない。ターゲットIoTデバイスのスリープ状態時に到着する要求は、待ち行列に入れられ、スリープ解除時に、ターゲットIoTデバイスに送信される。このパターンを実施するために、仲介者、すなわちデジタルツインは、デバイスがいつスリープ解除状態であるのかを知る必要がある。Thingとしてのデバイスインプリメンテーションは、そのための通知メカニズムを持つ必要があるであろう。これは、別個の一対のコンシューマ/Thingを使用して、あるいは、この目的のためにイベントインタラクションを使用することによって実施することができるであろう。

### 9.2.3 クラウドサービスから制御されるローカルネットワーク内のデバイス

スマートホームのユースケースでは、ホームネットワークに接続されたデバイス(センサおよび家電)は、監視される場合が多く、場合によっては、クラウドサービスによっても制御されている。通常、デバイスが接続されるホームネットワークとクラウドとの間にはNATデバイスが置かれる。NATデバイスは、接続を選択的にブロックするファイアウォールサービスを提供することも多く、また、IPアドレスも変換する。ローカルデバイスとクラウドサービスは、通信がゲートウェイをうまくトラバースすることができる場合にのみ、互いに通信することができる。

典型的なアーキテクチャは、ITU-T勧告Y.4409/Y.2070[Y.4409-Y.2070]に定義されるように、図32のとおりである。本構造には、ローカルな仲介者とリモートの仲介者が存在する。ローカル仲介者は、複数のThingから送られる相互作用アフォーダンスを、一つの共通のプロトコルにマッピングできる(1組の)公開されたThing(たとえば、共通のベースサーバを持ち、単一のポートを使用する単一のURLネームスペースにすべての相互作用がマッピングされたHTTP)に集める。これにより、ローカル仲介者がNATデバイスをトラバースすることができるプロトコルを使っており、このサービスをインターネット(STUN、TURN、DyDNSなど)に公開する何らかの方法を持っているという想定のもとで、NATデバイスの背後にあるすべてのThingにアクセスする簡単な方法がリモート仲介者に提供される。加えて、ローカル仲介者は、Thingのプロキシとして機能することができ、したがって、接続されたThingがそれぞれ異なるプロトコル(HTTP、MQTT、CoAPなど)および/あるいは異なるエコシステム規約を使用する場合であっても、公開されたThingは、それらを単一のプロトコルに収束させることができ、したがって、コンシューマは、Thingが使用する様々なプロトコルを認識する必要がない。

図32では、2つのクライアントがリモート仲介者に接続されており、この仲介者は、NAT境界外に存在するサービスを集約し、追加のプロトコル変換またはセキュリティサービスを提供することができる。特に、ローカル仲介者は、限られた容量のネットワーク上にあって、そのサービスを全てのユーザに直接利用可能にすることは実現できないかもしれない。この場合、ローカル仲介者へのアクセスは、リモート仲介者のみに提供される。その場合、リモート仲介者は、より一般的なアクセス制御機構を実装し、コンシューマを過剰なトラフィックから保護するためにキャッシングまたはスロットリングを実行することもできる。また、これらのコンシューマは、仲介者と通信するためにオープンインターネット(例えば、HTTPS)に適した単一のプロトコルを使用するが、これによりクライアント開発が非常に簡単になる。

このトポロジでは、コンシューマとThingの間にNATおよびファイアウォール機能があるが、ローカルおよびリモートの仲介者は、ファイアウォールを介してすべての通信をトンネリングするために協働する。そのため、コンシューマおよびThingはファイアウォールについて何も知る必要がない。ペアリングされた仲介者は、また、アクセス制御およびトラフィック管理を提供することによって、ホームデバイスを保護する。

図32. ペアとなった仲介者を介してThingとして実装されたローカルデバイスに接続されたコンシューマとして実装されたクラウドアプリケーション

より複雑な場合では、NATおよびファイアウォールトラバーサルは、図示のように正確に機能しないことがある。特に、ISPは、公的にアクセス可能なアドレスをサポートしないかもしれず、または、STUN/TURNおよび/あるいはDyDNSはサポートされないか、または、利用可能でないかもしれない。この場合、仲介者は、(クラウド内でそのリモート仲介者に最初に接続しているローカル仲介者との)初期接続を確立するために仲介者間のクライアント/サーバの役割を逆にすることができ、そして、そのペアとなった仲介者 (例えば、接続を保護するためにTLSを使用するSecureWebSocketを使用して) はトンネルを確立することができる。次いで、このトンネルは、カスタムプロトコルを使用して仲介者間のすべての通信を符号化するために使用することができる。この場合、初期接続も、標準ポートを使用するHTTPSを介して、通常のブラウザ/Webサーバ対話と同様に、ローカル仲介者からリモート仲介者に対して確立可能である。この初期接続は、ほとんどのホームファイアウォールをトラバースすることができるべきで、接続は出力であるため、ネットワークアドレス変換は何ら問題を引き起こさない。しかし、カスタムのトンネリングプロトコルが必要であるにしても、リモート仲介者は、このカスタムプロトコルを標準外部プロトコルに再変換することができる。接続されたコンシューマおよびThingは、それについて知る必要はない。この例をThingとコンシューマ両方がNAT境界の一方で接続できるユースケースに拡張することも可能である。しかしながら、これには、また、2つの仲介者間に双方向トンネルが確立されること要求される。

### 9.2.4 Thingディレクトリを使用した検出

クラウド上のサービスによってローカルデバイス(および場合によってはサービス)を監視または制御することができるようになると、様々な追加サービスを上に構築することができる。例えば、クラウドアプリは収集されたデータの分析に基づいてデバイスの動作条件を変更することもできる。

しかしながら、リモート仲介者が、クライアントアプリ用クラウドプラットフォームの一部である場合、クライアントは、例えば、接続されたデバイスのディレクトリにアクセスすることによって、デバイス情報を見つけることができる必要が出てくる。以下の図では単純化のために、すべてのローカルデバイスがThingとして実装され、すべてのクラウドアプリがコンシューマとして実装されていると想定した。Thingとして実装されたローカルデバイスのメタデータをクラウドアプリが利用できるようにするために、それらのメタデータはThingディレクトリサービスに登録することができる。このメタデータは、具体的には、リモート仲介者によって提供されるセキュリティ構成および通信メタデータを反映するように修正されたローカルデバイスのTDである。クライアントアプリは、Thingディレクトリに問い合わせることによってその機能を達成するためにローカルデバイスと通信するのに必要なメタデータを取得することができる。

図33. Thingディレクトリ を使用したクラウドサービス

図には示されていないが、より複雑な状況では、Thingとして働くクラウドサービスも存在し得る。これらは、Thingディレクトリにそれ自身を登録することもできる。ThingディレクトリはWebサービスであるため、NATまたはファイアウォールデバイスを介してローカルデバイスが見ることができなければならず、そのインターフェースには、それ自体のTDを設けることさえできる。コンシューマとして働くローカルデバイスは、Thingディレクトリを介してクラウド内のThingを検出し、直接に、または、例えば、プロトコル変換が必要な場合、ローカル仲介者を介してThingに接続することができる。

### 9.2.5 複数ドメイン間のサービス対サービス接続

それぞれ異なるIoTプラットフォームに基づく複数のクラウドエコシステムが連携して、より大規模なシステムズオブシステムエコシステムを構築することができる。下図は、前述したクラウドアプリエコシステムの構造を踏まえ、システムズオブシステムを構築するために相互接続された2つのエコシステムである。1つのエコシステム(すなわち、下記のコンシューマA)のクライアントが別のエコシステム(すなわち、下記のThingB)でサーバを使用する必要があるケースを考えてみてください。このクロスエコシステムアプリデバイス結合を達成するためにメカニズムが複数存在します。以下では、これがどのように達成できるかを説明するために、2つのメカニズムをそれぞれ図を用いて説明する。

#### 9.2.5.1 ディレクトリ同期による接続

図34では、2つのTDが情報を同期することで、コンシューマAはThingディレクトリAを介してThingBの情報を取得することができます。前項で説明したように、リモート仲介者BはThingBのシャドウインプリメンテーションを維持する。BがこのシャドウデバイスのTDを取得することで、コンシューマAは、リモート仲介者Bを介してThingBを使用することができる。

図34. ディレクトリ同期による複数のクラウド接続

#### 9.2.5.2 プロキシ同期による接続

図35では、2つのリモート仲介者がデバイス情報を同期している。ThingBのシャドウがリモート仲介者Bで生成されると、 このシャドウのTDは、リモート仲介者A内に同期される。同期されると、リモート仲介者Aは、自分のThing Bシャドウを生成し、TDをThingディレトリAに登録する。このメカニズムでは、Thingディレトリ間の同期は不要である。

図35. 仲介者同期による複数のクラウド接続

# １０　セキュリティとプライバシに関する考慮事項

本章は標準規格に含まれない。

セキュリティは、すべてのWoTビルディングブロックおよびWoTインプリメンテーションにおいて考慮される必要がある分野横断的な問題である。本章では、WoTインプリメンテーションのセキュリティとプライバシを保護するのに役立ついくつかの一般的な問題とガイドラインを要約する。しかし、それらは一般的なガイドラインであって、この文書に記載されているような抽象アーキテクチャ自体がセキュリティとプライバシを保証することはできない。セキュリティとプライバシの問題のより詳細で完全な分析については、WoTセキュリティとプライバシーポリシー・ガイドライン仕様[WOT-SECURITY] 参照。

全体として、WoTの目標は、セキュリティを含みIoTデバイスおよびサービスの既存のアクセスメカニズムおよびプロパティを記述することである。一般に、W3C WoTは、何を実装すべきかを規定するのではなく、何が存在するかを記述するように設計されている。既存システムの記述は、たとえそのシステムがセキュリティに関して理想的なものでなくても正確にそのシステムを記述できるべきである。システムのセキュリティ脆弱性を明確に理解することで脅威の軽減ができるが、そのような情報は悪意を持って利用しようとする者にまで配布する必要はない。

しかしながら、特に新規に構築するシステムに対して、WoTアーキテクチャは、セキュリティおよびプライバシにおけるベストプラクティスの使用を可能にすべきである。一般に、WoTセキュリティアーキテクチャは、それが接続するIoTプロトコルおよびシステムの目標およびメカニズムをサポートしなければならない。これらシステムは、そのセキュリティ要件およびリスク許容度が異なり、したがって、セキュリティメカニズムも、これらの要因に基づいて変化する。

IoTデバイスは、自律的に動作する必要があり、多くの場合、個人データにアクセスすることができ、かつ/または、安全上重要なシステムを制御することができるので、セキュリティおよびプライバシは、IoTドメインにおいて特に重要である。IoTデバイスは、ITシステムとは異なり、場合によっては、より高いリスクを避けられない。IoTシステムが他のコンピュータシステムへの攻撃を開始するために使用されないようにするためにIoTシステムを保護することも重要である。

一般に、セキュリティおよびプライバシを保証することはできない。WoTが安全でないシステムを安全なシステムに変えることは不可能である。しかしながら、WoTアーキテクチャは、損害を与えないようにする必要がある。WoTアーキテクチャは、記述対象のシステムがサポートするの同様に、少なくともセキュリティおよびプライバシをサポートすべきである。

## 10.1 WoT TDリスク

WoT TDに含まれるメタデータは、潜在的にセンシティブである。ベストプラクティスとして、TDは、整合性保護メカニズムおよびアクセス制御ポリシーとともに使用されるべきであり、許可されたユーザにのみ提供されるべきである。

さらなる詳細および考察については、WoT TD仕様のセキュリティおよびプライバシ考慮事項の項を参照。

### 10.1.1 TDのプライベートセキュリティデータリスク

TDはパブリックセキュリティメタデータのみを持つように設計されている。TDの制作者は、プライベートセキュリティデータがTDに含まれていないことを保証しなければならない。パブリックセキュリティメタデータとプライベートセキュリティデータは厳密に分離されるべきである。TDは、コンシューマが認可されている場合に限り、システムにアクセスするために必要となるパブィックセキュリティメタデータのみを持つべきである。認可は、別個に管理されたプライベート情報をベースとして実施されるべきである。

TD仕様で定義されている組み込みTDセキュリティスキームは、プライベートセキュリティデータの符号化を意図的にサポートしていない。しかしながら、この情報を符号化するために、人間が読み取り可能な記述などの他のフィールドが(不適切に)使用されるリスクや、そのような情報を符号化する拡張メカニズムを介して新しいセキュリティスキームが定義され、展開されるリスクがある。

軽減対策:

TDおよびTDでの使用を意図した拡張の作成者は、パブリックセキュリティメタデータのみTDに格納されることを保証しなければならない。

### 10.1.2 TDの個人識別可能情報リスク

TDは、潜在的に、様々なタイプの個人識別可能情報を含むことができる。明示的でない場合であっても、TDと識別可能な人物との関連付けは、その人物に関する情報を推論するために使用可能である。例えば、位置を判別できるモバイルデバイスがフィンガープリンティング可能なTDを公開することは、追跡リスクとなりうる。たとえ個別のデバイスが特定されないとしても、TDに示されたデバイスの型とこじんが結び付けれられば、個人情報となりうる。例えば、医療デバイスによってその利用者の健康状態を推論することができる。

一般に、TD内の個人識別可能情報は可能な限り制限されるべきである。ただし、回避できない場合もある。TD内に直接的あるいは推論可能なPIIが存在する可能性があることは、TDが他の形式のPIIと同様に扱われるべきであることを意味する。それらは、安全な方法で格納・送信されるべきであり、認可された利用者のみに提供されるべきであり、限られた時間だけキャッシュされるべきであり、要求に応じて削除されるべきであり、それらがユーザの同意を与えられた目的のためにのみ使用されるべきであり、そうでなければ、PIIの使用のためのすべての要件(すべての法的要件を含む)を満たすべきである。

軽減対策:

TDへのPIIの保存は、可能な限り最小限にすべきである。TD内に明示的なPIIがなくても、追跡および識別プライバシリスクが存在する可能性がある。このリスクを最小限に抑えるために、TDは、一般的に、あたかもPIIを持つと捉え、他のPIIと同じ管理ポリシーを遵守すべきである。TDは、認可されたコンシューマにのみ提供されるべきである。特定のユースケースにおいて必要のない情報は、可能な限りTD内に公開しないようにすべきである。例えば、明示的な型や個体を特定できる情報は、そのユースケースで必要とされていなければ含まないようにすべきである。ユースケースで必要とされる場合であっても、追跡のリスクを最小限にするため、グローバルで一意な識別子ではなく分散型でスコープの限られた識別子を使うべきである。人間が読み込み可能な記述などの他の形式の情報も、フィンガープリンティングのリスクを削減するためにユースケースによっては省略すべきである。

### 10.1.3 TD通信メタデータリスク

WoTバインディングテンプレートは、そのプラットフォームがWoTでの使用に適格であると考えられるようにするために、基礎となるIoTプラットフォームが使用するセキュリティメカニズムを正しくサポートしなければならない。大規模にIoTを展開するのに必要となるネットワーク上での相互作用の自動化のために、オペレータは、Thingが、そのセキュリティポリシーに準拠する形で公開・利用されることを保証する必要がある。

軽減対策:

可能な限り、TD作成者は、WoTバインディングテンプレートに提供されている審査済みの通信メタデータを使用すべきである。WoTバインディングテンプレートでカバーされていないIoTエコシステムのためのTDを生成するときは、そのIoTプラットフォームのセキュリティ要件がすべて満たされることを保証すること。

## 10.2 WoTスクリプティングAPIのセキュリティとプライバシのリスク

WoT Runtime インプリメンテーションおよびWoT スクリプティングAPI は、システムへの悪意のあるアクセスを防ぎ、マルチテナントサービエント間でスクリプトを隔離するメカニズムを持つべきである。より具体的には、WoTスクリプティングAPIと共に使用される場合のWoTランタイムインプリメンテーションは、以下のセキュリティおよびプライバシリスクを考慮に入れ、推奨される軽減対策を実装すべきである。

### 10.2.1 クロススクリプトセキュリティとプライバシリスク

基本的なWoTセットアップでは、WoTランタイム内で実行されるすべてのスクリプトは、信頼できるるものであるとみなされ、製造者によって配布される。したがって、各実行中のスクリプトインスタンス間での厳密な隔離は必ずしも必要ではない。ただし、デバイスの機能、展開ユースケースシナリオ、リスクレベルによっては、そうすることが望ましい場合がある。例えば、1つのスクリプトが、機密のプライバシ関連のPIIデータを処理し、十分に監査されている場合、同じシステム内の他のスクリプトが実行時に危険にさらされる場合に、データ露出リスクを最小限に抑えるために、スクリプトインスタンスを残りのスクリプトインスタンスから隔離することが望ましい場合がある。別の例としては、単一のWoTデバイス上の異なるテナントの相互共存がある。この場合、各WoTランタイムインスタンスは異なるテナントを提供することになり、それらの間の隔離が必要とされる。

軽減対策:

WoTランタイムは、スクリプトがプライバシ関連又はその他重要セキュリティデータを取り

扱う場合に、スクリプトインスタンス及びそれらのデータの間の隔離を実行しなければならない。同様に、WoTデバイスが複数のテナントを有する場合、WoTランタイムインプリメンテーションは、WoTランタイムインスタンスおよびそれらのデータの隔離を実行しなければならない。そのような隔離は、デバイス上で利用可能なプラットフォームセキュリティメカニズムを使用して、WoTランタイム内で実行することができる。詳細については、WoTセキュリティおよびプライバシー ガイドライン仕様[wotsecurity] 項の「WoT サービエント単一テナント」および「WoT サービエント複数テナント」参照。

### 10.2.2 物理デバイスダイレクトアクセスセキュリティとプライバシリスク

スクリプトが危険にさらされるか、または誤動作する場合、基礎となる物理デバイス(および潜在的な周囲環境)は、スクリプトが直接公開されたネイティブデバイスインターフェースを使用できる場合、被害をこうむる可能性がある。そのようなインターフェースが、その入力に関する安全性チェックが十分でない場合、そのインターフェースは、基礎をなす物理デバイス(または環境)を安全でない状態に晒す可能性がある。

軽減対策:

WoTランタイムは、ネイティブデバイスインターフェースをスクリプト開発者に直接公開することを避けるべきである。その代わりに、WoTランタイムインプリメンテーションは、ネイティブデバイスインターフェースにアクセスするためのハードウェア抽象レイヤを提供すべきである。そのようなハードウェア抽象レイヤは、デバイス(または環境)を安全でない状態に晒す可能性があるコマンドの実行を拒否するべきである。さらに、スクリプトを攻撃された場合に物理WoTデバイスへの損害を低減するために、その機能に基づいてそのスクリプトに公開されるかまたはアクセス可能なインターフェースの数を最小限に抑えることが重要である。

## 10.3 WoTランタイムセキュリティとプライバシリスク

### 10.3.1プロビジョニングと更新におけるセキュリティリスク

WoTランタイムインプリメンテーションが、それ自体、スクリプト、または関連データ(セキュリティ資格情報を含む)の製造後のプロビジョニングまたは更新をサポートする場合、それは、主要な攻撃ベクトルとなりうる。攻撃者は、更新またはプロビジョニングプロセス中に上記の要素を変更しようと試みるか、または単に攻撃者のコードおよびデータを直接プロビジョニングすることができる。

軽減対策:

WoTランタイム自体、または関連データのスクリプト製作後のプロビジョニングまたは更新は安全な方法で行われるべきである。安全な更新と製作後のプロビジョニングに関する推奨事項は、WoTセキュリティとプライバシガイドライン仕様 [WOT-SECURITY] に掲載されている。

### 10.3.2 セキュリティ資格情報ストレージのセキュリティとプライバシリスク

通常、WoTランタイムは、ネットワーク内で動作するためにWoTデバイスにプロビジョニングされるセキュリティ資格情報を格納する必要がある。攻撃者が、これらの資格情報の機密性または整合性を危険にさらすことができる場合、攻撃者は資産へのアクセスを取得し、他のWoTThing、デバイス、またはサービスになりすます、サービス妨害(DoS)攻撃を開始することができる。

軽減対策:

WoTランタイムは、プロビジョニングされたセキュリティ資格情報を安全に格納し、それらの完

全性および機密性を保証しなければならない。1つのWoT対応機器に複数のテナントが存在する場合、WoTランタイムインプリメンテーションは、各テナントのプロビジョニングされたセキュリティ資格情報の隔離を保証するべきである。さらに、プロビジョニングされたセキュリティ資格情報が危険にさらされるリスクを最小限に抑えるために、WoTランタイムインプリメンテーションは、プロビジョニングされたセキュリティ証明書を照会するためのスクリプトのAPIを公開すべきではない。そのような資格情報(あるいは、それらを使用するがそれらを公開しない抽象オペレーション)は、それらを使用するプロトコルバインディングインプリメンテーションにのみアクセス可能であるべきである。

# 謝辞

本文書へ御寄稿くださったMichael McCool、Takuki Kamiya、Kazuyuki Ashimura、Sebastian Kabisch、Zoltan Kis、Elena Reshetova、Ari Keranen、Kazuaki Nimura、Klaus Hartke、Philippe Le Hegaretに対し特に謝意を表します。

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

WoT WGは、[wot-pioneers-1] [wot-pioneers-2] [wot-pioneers-3] [wot-pioneers-4]などの出版物として、また、2010年に開始された年次のWoTに関する国際ワークショップを学術的なイニシアチブとしてスタートした「Web of Things」のコンセプトに関する先駆者的な取り組みに感謝いたします。

# 参考文献

## B.1 Normative references

**[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

**[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

**[RFC5234]**

Augmented BNF for Syntax Specifications: ABNF. D. Crocker, Ed.; P. Overell. IETF. January 2008. Internet Standard. URL: https://tools.ietf.org/html/rfc5234

**[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

**[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

## B.2 Informative references

**[CoRAL]**

The Constrained RESTful Application Language (CoRAL). Klaus Hartke. IETF. March 2019. Internet-Draft. URL: https://tools.ietf.org/html/draft-hartke-t2trg-coral

**[CoRE-RD]**

CoRE Resource Directory. M. Koster; C. Bormann; P. van der Stok; C. Amsuess. IETF. 13 June 2019. Internet-Draft. URL: https://tools.ietf.org/html/draft-ietf-core-resource-directory-21

**[ECMAScript]**

ECMAScript Language Specification. Ecma International. URL: https://tc39.es/ecma262/

**[EVENTSOURCE]**

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

**[HCI]**

The Encyclopedia of Human-Computer Interaction, 2nd Ed. Interaction Design Foundation. 2013. URL: https://www.interaction-design.org/literature/book/the-encyclopedia-of-human-computer-interaction-2nd-ed

**[IANA-RELATIONS]**

Link Relations. IANA. URL: https://www.iana.org/assignments/link-relations/

**[IANA-URI-SCHEMES]**

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

**[IEC-FOTF]**

Factory of the future. IEC. October 2015. URL: https://www.iec.ch/whitepaper/pdf/iecWP-futurefactory-LR-en.pdf

**[IOT-SCHEMA-ORG]**

Schema Extensions for IoT Community Group. URL: https://www.w3.org/community/iotschema/

**[ISO-IEC-2382]**

Information technology — Vocabulary. ISO. 2015. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en

**[ISO-IEC-27000]**

Information technology — Security techniques — Information security management systems — Overview and vocabulary. ISO. 2018. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en

**[ISO-IEC-29100]**

Information technology — Security techniques — Privacy framework. ISO. 2011. URL: https://www.iso.org/obp/ui/#iso:std:iso-iec:29100:ed-1:v1:en

**[JSON-LD11]**

JSON-LD 1.1. Gregg Kellogg; Pierre-Antoine Champin. W3C. 9 September 2019. W3C Working Draft. URL: https://www.w3.org/TR/json-ld11/

**[LINKED-DATA]**

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

**[LWM2M]**

Lightweight Machine to Machine Technical Specification: Core. OMA SpecWorks. August 2018. Approved Version: 1.1. URL: http://openmobilealliance.org/release/LightweightM2M/V1\_1-20180710-A/OMA-TS-LightweightM2M\_Core-V1\_1-20180710-A.pdf

**[MQTT]**

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

**[NORMAN]**

The Psychology of Everyday Things. Donald A. Norman. Basic Books. 1988.

**[OCF]**

OCF Core Specification. Open Connectivity Foundation. April 2019. Version 2.0.2. URL: https://openconnectivity.org/developer/specifications

**[REST]**

REST: Architectural Styles and the Design of Network-based Software Architectures. Roy Thomas Fielding. University of California, Irvine. 2000. PhD thesis. URL: https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding\_dissertation.pdf

**[RFC4301]**

Security Architecture for the Internet Protocol. S. Kent; K. Seo. IETF. December 2005. Proposed Standard. URL: https://tools.ietf.org/html/rfc4301

**[RFC6202]**

Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. S. Loreto; P. Saint-Andre; S. Salsano; G. Wilkins. IETF. April 2011. Informational. URL: https://tools.ietf.org/html/rfc6202

**[RFC6347]**

Datagram Transport Layer Security Version 1.2. E. Rescorla; N. Modadugu. IETF. January 2012. Proposed Standard. URL: https://tools.ietf.org/html/rfc6347

**[RFC6690]**

Constrained RESTful Environments (CoRE) Link Format. Z. Shelby. IETF. August 2012. Proposed Standard. URL: https://tools.ietf.org/html/rfc6690

**[RFC6749]**

The OAuth 2.0 Authorization Framework. D. Hardt, Ed.. IETF. October 2012. Proposed Standard. URL: https://tools.ietf.org/html/rfc6749

**[RFC7049]**

Concise Binary Object Representation (CBOR). C. Bormann; P. Hoffman. IETF. October 2013. Proposed Standard. URL: https://tools.ietf.org/html/rfc7049

**[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

**[RFC7252]**

The Constrained Application Protocol (CoAP). Z. Shelby; K. Hartke; C. Bormann. IETF. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7252

**[RFC7641]**

Observing Resources in the Constrained Application Protocol (CoAP). K. Hartke. IETF. September 2015. Proposed Standard. URL: https://tools.ietf.org/html/rfc7641

**[RFC7744]**

Use Cases for Authentication and Authorization in Constrained Environments. L. Seitz, Ed.; S. Gerdes, Ed.; G. Selander; M. Mani; S. Kumar. IETF. January 2016. Informational. URL: https://tools.ietf.org/html/rfc7744

**[RFC8446]**

The Transport Layer Security (TLS) Protocol Version 1.3. E. Rescorla. IETF. August 2018. Proposed Standard. URL: https://tools.ietf.org/html/rfc8446

**[SAREF]**

Smart Appliances REFerence (SAREF) ontology. ETSI. November 2015. URL: https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology

**[VOCAB-SSN]**

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

**[WOT-BINDING-TEMPLATES]**

Web of Things (WoT) Protocol Binding Templates. Michael Koster. W3C. 5 April 2018. W3C Note. URL: https://www.w3.org/TR/wot-binding-templates/

**[WOT-PIONEERS-1]**

Mobile Service Interaction with the Web of Things. E. Rukzio, M. Paolucci; M. Wagner, H. Berndt; J. Hamard; A. Schmidt. Proceedings of 13th International Conference on Telecommunications (ICT 2006), Funchal, Madeira island, Portugal. May 2006. URL: https://pdfs.semanticscholar.org/3ee3/a2e8ce93fbf9ba14ad54e12adaeb1f3ca392.pdf

**[WOT-PIONEERS-2]**

Putting Things to REST. Erik Wilde. UCB iSchool Report 2007-015, UC Berkeley, Berkeley, CA, USA. November 2007. URL: http://dret.net/netdret/docs/wilde-irep07-015-restful-things.pdf

**[WOT-PIONEERS-3]**

Poster Abstract: Dyser – Towards a Real-Time Search Engine for the Web of Things. Benedikt Ostermaier; B. Maryam Elahi; Kay Römer; Michael Fahrmair; Wolfgang Kellerer. Proceedings of ACM SenSys 2008, Raleigh, NC, USA. November 2008. URL: https://www.vs.inf.ethz.ch/publ/papers/ostermai-poster-2008.pdf

**[WOT-PIONEERS-4]**

A Resource Oriented Architecture for the Web of Things. Dominique Guinard; Vlad Trifa; Erik Wilde. Proceedings of Internet of Things 2010 International Conference (IoT 2010). Tokyo, Japan. November 2010. URL: https://ieeexplore.ieee.org/abstract/document/5678452

**[WOT-SCRIPTING-API]**

Web of Things (WoT) Scripting API. Zoltan Kis; Kazuaki Nimura; Daniel Peintner; Johannes Hund. W3C. 29 November 2018. W3C Working Draft. URL: https://www.w3.org/TR/wot-scripting-api/

**[WOT-SECURITY]**

Web of Things (WoT) Security and Privacy Considerations. Elena Reshetova; Michael McCool. W3C. 3 December 2018. W3C Note. URL: https://www.w3.org/TR/wot-security/

**[WOT-THING-DESCRIPTION]**

Web of Things (WoT) Thing Description. Sebastian Käbisch; Takuki Kamiya; Michael McCool; Victor Charpenay. W3C. 16 May 2019. W3C Candidate Recommendation. URL: https://www.w3.org/TR/wot-thing-description/

**[Y.4409-Y.2070]**

ITU-T Rec. Y.4409/Y.2070 (01/2015) Requirements and architecture of the home energy management system and home network services . ITU-T. January 2015. Recommendation. URL: https://www.itu.int/rec/T-REC-Y.2070-201501-I