[【注意】 この⽂書は、W3CのWeb of Things (WoT) Architecture W3C Recommendation 9 April 2020の和訳である。](https://www.w3.org/TR/2020/REC-wot-architecture-20200409/)

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](https://github.com/wot-jp-community/wot-architecture/issues)までお願いしたい。

First Update: 2020年05⽉25⽇ | Last Update: 2021年5⽉6⽇

✧  ✧  ✧

Web of Things (WoT) アーキテクチャ

# W3C勧告 2020年4⽉9⽇

本バージョン:

<https://www.w3.org/TR/2020/REC-wot-architecture-20200409/>

最新公開バージョン:

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

最新編集者草案:

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

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-architecture-20200130/>

編集者:

Matthias Kovatsch ([Huawei)](https://www.huawei.com/)

Ryuichi Matsukura ([Fujitsu Ltd.)](https://www.fujitsu.com/)

Michael Lagally ([Oracle Corp.](https://www.oracle.com/))

Toru Kawaguchi ([Panasonic Corp.](https://www.panasonic.com/))

Kunihiko Toumura ([Hitachi, Ltd.)](https://www.hitachi.com/)

Kazuo Kajimoto (Former Editor, when at Panasonic)

参加可能:

[GitHub w3c/wot-architecture](https://github.com/w3c/wot-architecture/)

[File a bug](https://github.com/w3c/wot-architecture/issues/)

[Commit history](https://github.com/w3c/wot-architecture/commits/master)

[Pull requests](https://github.com/w3c/wot-architecture/pulls/)

貢献者:

[In the GitHub repository](https://github.com/w3c/wot-architecture/graphs/contributors) 公開以後に報告されたエラーや問題がないか[正誤表を](https://w3c.github.io/wot-architecture/errata.html)確認のこと。

[翻訳版も](http://www.w3.org/2003/03/Translations/byTechnology?technology=wot-architecture)参照のこと。

[Copyright © 2017-2020 W3C® (MIT, ERCIM, Keio, Beihang). W3C liability, trademark and permissive document license rules apply.](https://www.w3.org/Consortium/Legal/2015/copyright-software-and-document)

# 概要

W3Cの Web of Things (WoT) は、IoTプラットフォームとアプリケーション領域にまたがる相互運⽤性を可能にすることを⽬的としている。全体として、WoTの⽬標は、既存のIoT標準とソリューションを維持し補完することである。⼀般的に、W3C WoTアーキテクチャ

は、何を実装するのかを規定するのではなく、何が存在するのかを記述することを⽬指している。

このWoTアーキテクチャの仕様では、W3C Web of Things の抽象アーキテクチャを記述している。この抽象アーキテクチャは、複数のアプリケーション領域のユースケースから導かれた⼀連の要件に基づいており、ユースケースおよび要件の両⽅とも、この⽂書で⽰されている。他の⽂書で詳細な仕様が⽰されているモジュール構成要素についても確認を⾏っている。この⽂書では、これらの構成要素がどのように関連付けられ、連携するかを記述している。WoT抽象アーキテクチャは、様々な具体的な展開シナリオにマッピングできる基本的な概念フレームワークを定義したものであり、そのいくつかの例を⽰している。しかし、この仕様で記述している抽象アーキテクチャ⾃体は、具体的なメカニズムを定義したり、具体的な実装を規定したりするものではない。

# この⽂書のステータス

この節は、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](https://www.w3.org/TR/)ある。

この⽂書は、抽象的なアーキテクチャの設計について記述している。しかし、関連するWoT

Thing Descriptionの仕様に基づいて⼀連の具体的な実装を記述してい[る実装報告書](https://w3c.github.io/wot-thing-description/testing/report.html)がある。これらは、W3C Web of Things のアーキテクチャに準拠した実装である。

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、他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、[Web of Thingsワーキンググループ](https://www.w3.org/WoT/WG/)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](https://github.com/w3c/wot-architecture/issues/)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)](https://lists.w3.org/Archives/Public/public-wot-wg/) に送信いただきたい。

この⽂書は、[W3C特許⽅針](https://www.w3.org/Consortium/Patent-Policy/)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](https://www.w3.org/2004/01/pp-impl/95969/status)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](https://www.w3.org/Consortium/Patent-Policy/#def-essential) (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](https://www.w3.org/Consortium/Patent-Policy/" \l "sec-Disclosure)[⽅針の6項](https://www.w3.org/Consortium/Patent-Policy/#sec-Disclosure)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](https://www.w3.org/2019/Process-20190301/)に準拠している。

# ⽬次

1. はじめに
2. 適合性
3. ⽤語
4. ユースケース
   1. アプリケーション領域
      1. 消費者
      2. 産業
         1. 例: スマートファクトリー
      3. 輸送と物流
      4. 公益事業
      5. オイルとガス
      6. 保険
      7. ⼟⽊⼯事と建設
      8. 農業
      9. 医療
      10. 環境モニタリング
      11. スマートシティ
      12. スマートビルディング
      13. コネクテッドカー
          1. コネクテッドカーの例
   2. 共通パターン
      1. デバイスコントローラー
      2. ThingとThing
      3. リモートアクセス
      4. スマートホームゲートウェイ
      5. エッジデバイス
      6. デジタルツイン
         1. クラウド対応デバイス
         2. 旧式デバイス
      7. マルチクラウド
      8. 領域横断型連携
   3. 要約
5. 要件
   1. 機能要件
      1. ⼀般的な原則
      2. Thingの機能
      3. 検索と発⾒
      4. 記述⽅法
      5. 属性の記述
      6. 機能の記述
      7. ネットワーク
      8. デプロイメント
      9. アプリケーション
      10. 旧式技術への適合
   2. 技術要件
      1. Web of Things および Web of Thingsアーキテクチャの構成要素
      2. デバイス
      3. アプリケーション
      4. デジタルツイン
      5. ディスカバリ
      6. セキュリティ
      7. アクセシビリティ
6. WoTアーキテクチャ
   1. 概要
   2. アフォーダンス
   3. Web Thing
   4. 相互作⽤モデル
      1. Property
      2. Action
      3. Event
   5. ハイパーメディア制御
      1. リンク
      2. フォーム
   6. プロトコルバインディング
      1. ハイパーメディア駆動
      2. URI
      3. メソッドの標準的な集合
      4. メディアタイプ
   7. WoTシステム構成要素とその相互接続性
      1. 直接通信
      2. 間接通信
7. WoT構成要素
   1. WoT Thing Description
   2. WoTバインディングテンプレート
   3. WoTスクリプトAPI
   4. WoTセキュリティとプライバシーに関するガイドライン
8. 抽象的なServientのアーキテクチャ
   1. 動作の実装
   2. WoTランタイム
   3. WoTスクリプトAPI
   4. 公開されたThingと利⽤されるThingの抽象化
   5. Private Security Data
   6. プロトコルスタックの実装
   7. システムAPI
   8. 代替のServientとWoT実装
      1. ネイティブなWoT API
      2. 既存デバイスのThing Description
9. WoTのデプロイメント例
   1. ThingとConsumerの役割
   2. WoTシステムのトポロジーとデプロイメントシナリオ
      1. 同じネットワーク上の利⽤者とモノ
      2. Intermediaryを介して接続されたConsumerとThing
         1. プロキシとして機能するIntermediary
         2. デジタルツインとして機能するIntermediary
      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
      4. Thing Directoryを⽤いた発⾒
      5. 複数の領域にまたがるサービス間の接続
         1. Thing Directoryの同期を介した接続
         2. プロキシの同期を介した接続
10. セキュリティとプライバシーへの配慮
    1. WoT Thing Descriptionに関するリスク
       1. Thing DescriptionのPrivate Security Dataに関するリスク
       2. Thing Descriptionの個⼈識別可能情報に関するリスク
       3. Thing Descriptionのコミュニケーションメタデータに関するリスク
    2. WoTスクリプトAPIのセキュリティとプライバシーに関するリスク
       1. クロススクリプトのセキュリティとプライバシーに関するリスク
       2. 物理デバイス直接アクセスのセキュリティとプライバシーに関するリスク
    3. WoTランタイムのセキュリティとプライバシーに関するリスク
       1. プロビジョニングと更新のセキュリティリスク
       2. セキュリティ証明書保管のセキュリティとプライバシーに関するリスク
11. 最近の仕様変更
12. 謝辞
13. 参考⽂献
    1. 規範的な参考⽂献
    2. 参考情報の参考⽂献

# 1. はじめに §

Web of Things (WoT) の⽬標は、Internet of Things (IoT) の相互運⽤性と使いやすさを改善することである。多くの利害関係者が関わった⻑年にわたるコラボレーションを通じて、これらの課題の対処に役⽴ついくつかの構成要素が特定されている。

この仕様は、W3C WoTの標準化の範囲に焦点を当てており、その構成要素と、その関係を定義する抽象アーキテクチャとに分類できる。構成要素は、別の仕様で詳細に定義され説明されている。しかし、抽象アーキテクチャとその⽤語や概念フレームワークの定義に加え、この仕様は、WoT構成要素に対する⼊⾨としての機能も果たし、それらの連携について説明している。

Web of Things (WoT) Thing Description [WOT-THING-DESCRIPTION] は、Thing のメタデータとネットワーク向けインターフェースを記述するための機械可読データ形式を規定的に提供する。これは、相互作⽤のアフォーダンスなどの、この⽂書で紹介している基本概念に基づいている。

Web of Things (WoT) バインディングテンプレート [WOT-BINDING-TEMPLATES]

は、特定のプロトコルとIoTエコシステムに対する、Thingのネットワーク向けインターフェースを定義する⽅法 (プロトコルバインディングと呼ぶ) に関する参考情報のガイドライン を提供している。また、この⽂書では、多くの既存のIoTエコシステムと標準の例も提供している。

Web of Things (WoT) スクリプティングAPI [WOT-SCRIPTING-API] は、オプション

であり、これを⽤いると、ウェブブラウザAPIのような⼀般的なJavaScript APIでThing のアプリケーションロジックを実装できるようになる。これによって、IoTアプリケーションの開発が簡素化され、ベンダーやデバイスにまたがる移植が可能となる。

Web of Things (WoT) のセキュリティとプライバシーに関するガイドライン [WOT-

SECURITY] は、分野横断的な構成要素を表す。この参考情報である⽂書は、Thingの安全な実装と設定に関するガイドラインを提供し、W3C WoTを実装するシステムで検討すべき課題について論じている。しかし、セキュリティとプライバシーは、特定の実装に関する⼀揃いの具体的なメカニズムのコンテキストでのみ完全に評価されうるものであり、WoTの抽象アーキテクチャでは完全には規定していないことを強調しておくべきである。これは特に、WoTアーキテクチャが既存のシステムに記述的に⽤いられる場合に当てはまる。なぜならば、W3C WoTはそのようなシステムの動作を制約することはできず、それを記述することしかできないからである。この⽂書では、§ 10. セキュリティとプライバシーへの配慮の項で、プライバシーとセキュリティに関するリスクとその軽減策について概観する。

この仕様は、WoTシステムの展開に関する⾮規定的なアーキテクチャの側⾯と条件もカバーしている。この仕様は特定の具体的な実装を規定的に定義するものではないが、このガイドラインは、展開シナリオの例に照らして記述されている。

この仕様は、W3C WoT仕様を包括する機能を有しており、⽤語やW3C Web of Thingsの基本となる抽象アーキテクチャなどの基礎を定義している。要約すると、この仕様の⽬的は次を提供することである。

§ 4. ユースケースでは、W3C WoTアーキテクチャへとつながったユースケース、

§ 5. 要件では、WoT実装の要件、

§ 6. WoTアーキテクチャでは、抽象アーキテクチャの定義、

§ 7. WoT構成要素では、WoT構成要素とその相互作⽤の概要、

§ 8. 抽象的なServientのアーキテクチャでは、抽象アーキテクチャを存在しうる具体的な実装にマッピングする⽅法に関する参考情報のガイドライン、

§ 9. WoTのデプロイメント例では、存在しうるデプロイメントシナリオに関する参考情報の例、

および§ 10. セキュリティとプライバシーへの配慮では、W3C WoTアーキテクチャに基づくシステムを実装する際に注意すべきセキュリティとプライバシーに関する留意点の⾼いレベルでの議論。

追加の要件、ユースケース、概念的な機能、および新しい構成要素に関しては、この⽂書の将来の改訂で取り組む。

# 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)」、「すべきである∕する必要がある (SHOULD)」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](https://tools.ietf.org/html/bcp14) [RFC2119] [RFC8174] で記述されているように解釈されるべきである。

# 3. ⽤語 §

この章は参考情報である。

この仕様では、ここで定義しているとおりに次の⽤語を⽤いる。WoTの接頭辞は、モノのウェブの概念のために特別に (再) 定義されている⽤語の曖昧さを回避するために⽤いる。

アクション (Action)

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。バインディングテンプレート (Binding Templates)

様々なIoTプラットフォームとの通信のための再利⽤可能な⻘写真の集合。この⻘写真

は、WoT Thing Descriptionと、必要なプロトコルスタックや専⽤通信ドライバーに関する実装注記によって、相互作⽤の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マッピングするための情報を提供する。

利⽤されるThing (Consumed Thing)

ローカルなアプリケーションが⽤いるリモートのThingを表すソフトウェア抽象化。この抽象化は、ネイティブなWoTランタイムが作成したり、WoTスクリプティングAPIがオブジェクトとしてインスタンス化する可能性がある。

Thingを利⽤する (Consuming a Thing)

TDドキュメントを解析して処理し、それから、ローカルなランタイム環境にあるアプリケーションに対するインターフェースとして、利⽤されるThingのソフトウェア抽象化を作成すること。

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 (つまり、Thingを利⽤する) ことができるエンティティー。

Data Schema

Data Schemaは、情報モデルおよび関連するペイロード構造と、相互作⽤中にThingと

Consumerの間で受け渡される対応するデータ項⽬を記述する。

デジタルツイン (Digital Twin)

デジタルツインは、クラウドやエッジノード上に存在するデバイスやデバイス群の仮想表現である。オンライン上に継続的に存在していない可能性のある実在するデバイスを表したり、実際のデバイスに展開する前に新しいアプリケーションやサービスをシミュレーションしたりするために使⽤できる。

領域固有の語彙 (Domain-specific Vocabulary)

WoT Thing Descriptionで使⽤できるリンクトデータの語彙だが、W3C WoTでは定義していない。

エッジデバイス (Edge Device)

企業やサービス提供者の基幹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。例には、ゲートウェイ、ルーター、スイッチ、マルチプレクサ、およびその他の様々な接続デバイスが含まれる。

イベント (Event) イベントの情報源を記述している相互作⽤のアフォーダンスで、イベントデータを利⽤者に⾮同期でプッシュする (例えば、オーバーヒートの警報)。

公開されたThing (Exposed Thing)

リモートの利⽤者がネットワーク経由でアクセスできる、ローカルで提供されている

Thingを表すソフトウェア抽象化。この抽象化は、ネイティブなWoTランタイムが作成したり、WoTスクリプティングAPIがオブジェクトとしてインスタンス化する可能性がある。

Thingを公開する (Exposing a Thing)

Thingの状態を管理し、動作の実装と連動するために、ローカルなランタイム環境にある公開対象のThingのソフトウェア抽象化を作成すること。

ハイパーメディア制御 (Hypermedia Control)

ハイパーメディアにおけるプロトコルバインディングのシリアライゼーション、つま

り、ナビゲーション⽤のウェブリンク [RFC8288] や、他の操作を実⾏するためのウェブフォーム。フォームは、利⽤者が項⽬に記⼊して送信するための、Thingが提供するリクエストテンプレートと考えることができる。

相互作⽤のアフォーダンス (Interaction Affordance)

可能な選択肢を利⽤者に提⽰し説明するThingのメタデータで、これにより、利⽤者がどのようにThingと相互作⽤できるかを提案する。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、プロパティー、アクション、イベントという3種類の相互作⽤のアフォーダンスを定義している。四つ⽬の相互作⽤のアフォーダンスはナビゲーションで、これは、ウェブでは既にリンク付けという⽅法で利⽤できる。

相互作⽤モデル (Interaction Model) アプリケーションの意図 (application intent) から具体的なプロトコル操作へのマッピングを形式化し、限定する中間の抽象化。W3C WoTでは、定義済みの相互作⽤のアフォーダンスの集合が相互作⽤モデルを構成する。

Intermediary

Thingを代理、拡張、または構成する、ConsumerとThingの間のエンティティーで、元のThingではなく仲介のWoTインターフェースを指し⽰すWoT Thing Descriptionを再公開できる。RESTの階層化システムの制約により、利⽤者には、Intermediaryは

Thingと⾒分けがつかないかもしれない。

IoTプラットフォーム (IoT Platform)

OCF、oneM2M、MozillaプロジェクトのThingなどの特定のIoTエコシステムで、アプリケーション向けのAPI、データモデル、プロトコルまたはプロトコル設定に関して独

⾃の仕様を備えている。

メタデータ (Metadata)

エンティティーの抽象的な特性に関する記述を提供するデータ。例えば、Thing

DescriptionはThingのメタデータである。

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

ある情報と関係性がある⾃然⼈を特定するために使⽤できる情報、または⾃然⼈に直接または間接的にリンクされている、またはその可能性がある情報。 [ISO-IEC-29100] と同じ定義を⽤いる。プライバシー (Privacy)

私⽣活や個⼈的な出来事への侵害 (その個⼈に関するデータを不当または違法に収集し

使⽤した結果、侵害が⽣じた場合) がないこと。 [ISO-IEC-2382] と同じ定義を⽤いる。個⼈を特定できる情報、セキュリティ、および [ISO-IEC-29100] のその他の関連定義も参照のこと。

Private Security Data

Private Security Dataは、Thingのセキュリティ設定の構成要素であり、秘密に保た

れ、他のデバイスやユーザと共有されない。ひとつの例は、PKIシステムの秘密鍵である。理想的には、そのようなデータは、アプリケーションがアクセスできない別のメモリに保存され、それを利⽤するアプリケーションにも秘密情報を漏らさない、署名などの抽象的な操作を介してしか⽤いられない。

プロパティー (Property)

Thingの状態を公開する相互作⽤のアフォーダンス。この状態は、後で取得 (読み取り) し、オプションで更新 (書き込み) 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、プロパティーを監視可能にすることも選択できる。

プロトコルバインディング (Protocol Binding)

相互作⽤の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。これにより、利⽤者に相互作⽤のアフォーダンスを作動させる⽅法を通知する。

W3C WoTは、プロトコルバインディングをハイパーメディア制御としてシリアライズする。

Public Security Metadata

Public Security Metadataは、Thingにアクセスするために必要なセキュリティメのメカニズムとアクセス権を記述した、Thingのセキュリティ設定の構成要素である。これには秘密情報や具体的なデータ (公開キーを含む) は含まれておらず、単独でThingへのアクセスを提供することはない。代わりに、ユーザがどのように認証を得なければならないかを含め、承認されたユーザがアクセスを取得する⽅法が記述されている。

セキュリティ (Security)

情報の機密性、完全性、および可⽤性の保持。真正性、責任追跡性、否認防⽌、信頼性などのプロパティーも関係する場合がある。この定義は、 [ISO-IEC-27000] の情報セキュリティの定義の焼き直しで、これには、⾔及されている個々のより具体的なプロパティーに関する追加定義も含まれている。その他の関連する定義については、この⽂書を参照いただきたい。さらに、このプロパティーは、通常動作時とシステムが攻撃にさらされている時の両⽅の場合において保持されることが望ましいことに注意のこと。

セキュリティ構成情報 (Security Configuration)

Public Security Metadata、Private Security Data、およびThingのセキュリティメカ

ニズムを運⽤上構成するために必要なその他の構成情報 (公開キーなど) の組み合わせ。

Servient

WoT構成要素を実装するソフトウェアスタック。Servientは、Thingを提供し公開することができ、かつ (または) Thingを利⽤するConsumerの役割を務めることができる。

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、複数のプロトコルバインディングをサポートできる。

サブプロトコル (Subprotocol) 相互作⽤をうまく⾏うために知っていなければならない転送プロトコルに対する拡張メカニズム。例は、HTTPに対するロングポーリング (long polling) である。

TD

WoT Thing Descriptionの略。

TD語彙 (TD Vocabulary)

WoTバインディングテンプレートの通信メタデータを含む、WoT Thing Descriptionに

おいてThingのメタデータにタグ付けを⾏うためのW3C WoTによるリンクトデータの統制語彙。

Thing またはWeb 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。それに対し、仮想エンティティーは⼀つ以上のThingの合成物である。

Thing Directory

([CoRE-RD] のように) TDを登録して検索する (例えば、SPARQLクエリやCoRE RDル

ックアップインターフェース [CoRE-RD] を⽤いて) ためのウェブインターフェースを提供する，TDのディレクトリサービス。

転送プロトコル (Transfer Protocol)

オプションやサブプロトコルメカニズムに対してアプリケーション固有の要件や制約のない、基礎となる標準的なアプリケーション層のプロトコル。例は、HTTP、CoAP、またはMQTTである。ヴァーチャルThing

別のシステム構成要素に置かれているThingを表す、Thingのインスタンス。

WoTインターフェース (WoT Interface)

WoT Thing Descriptionで記述されている、htingのネットワーク向けインターフェース。

WoTランタイム (WoT Runtime)

アプリケーションの実⾏環境を維持し、Thingを公開および (または) 利⽤し、WoT

Thing Descriptionを処理し、セキュリティ設定を維持し、プロトコルバインディング実装と連動できるランタイムシステム。WoTランタイムは、特注のAPIを持つか、オプションのWoTスクリプトAPIを⽤いることができる。

WoTスクリプティングAPI (WoT Scripting API)

WoTランタイムで実⾏される動作やアプリケーションの実装を容易にするために、

Servientが提供するアプリケーション向けプログラミングインターフェース。ウェブブラウザAPIに相当する。WoTスクリプティングAPIは、W3C WoTのオプションの構成要素である。

WoT Servient

Servientの同義語。

WoT Thing DescriptionまたはThing Description

Thingを記述した構造化データ。WoT Thing Descriptionは、⼀般的なメタデータ、領域固有のメタデータ、相互作⽤のアフォーダンス (サポートされるプロトコルバインディングを含む)、および関係するThingへのリンクで構成される。WoT Thing

Descriptionは、W3C WoTの中⼼となる構成要素である。

# 4. ユースケース §

この章は参考情報である。

この項では、W3C WoTの対象である、§ 7. WoT構成要素で論じている抽象アーキテクチャを導き出すために⽤いたアプリケーション領域とユースケースを⽰す。

Web of Thingsのアーキテクチャは、ユースケースとアプリケーション領域に制限を設けていない。抽象アーキテクチャが満たさなければならない⼀般的なパターンを収集するために、様々なアプリケーション領域について検討が⾏われた。

以下の項は網羅的ではない。むしろ、それらは例⽰としての機能を果たすものであり、接続されたモノからさらに恩恵を得たり、新しいシナリオが可能となったりする可能性がある。

## 4.1アプリケーション領域 §

4.1.1 消費者 §

消費者空間には、接続によって恩恵を得られる複数の資産がある。部屋の利⽤状況に基づいて照明とエアコンをオフにできる。気象条件と⼈の存在に基づいてブラインドを⾃動的に閉じることができる。エネルギーやその他の資源の利⽤は、使⽤パターンと予測に基づいて最適化できる。

この項の消費者のユースケースには、スマートホームのユースケースが含まれる。

図1は、スマートホームの例である。このケースでは、ゲートウェイは、対応するKNX、

ECHONET、ZigBee、DECT ULE、Wi-SUNなどのローカルな通信プロトコルにより、センサー、カメラ、家電などのエッジデバイスに接続されている。⼀つの家に複数のゲートウェイが存在可能で、各ゲートウェイは複数のローカルなプロトコルをサポートできる。

ゲートウェイはインターネット経由でクラウドに接続できるが、クラウドに直接接続できる機器もある。クラウドで実⾏されているサービスは、エッジデバイスからデータ収集を⾏い、そのデータを分析し、その後でエッジデバイスやその他のUXデバイスを介してユーザに価値を提供する。

ZigBee

/DECT ULE

/Wi

-

SUN

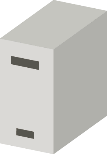
KNX

/ECHONET

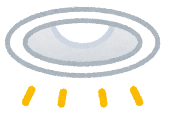
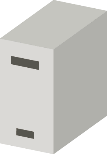


リモート

アクセス



サービス



遠

隔

監

視

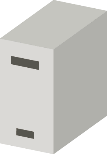
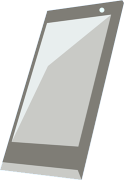


図1 スマートホーム

スマートホームは、リモートのアクセスと制御、⾳声制御、ホームオートメーションなどのメリットを消費者に提供する。スマートホームにより、デバイスの製造者がリモートでデバイスを監視しメンテナンスを⾏うことも可能となる。スマートホームは、エネルギー管理やセキュリティ監視などの付加価値サービスを実現できる。

4.1.2 産業 §

この項の産業に関するユースケースは、様々な産業部⾨に適⽤できる。

アプリケーションシナリオには重複する性質があるため、異なる業種に似たようなユースケースが存在する。

4.1.2.1 例: スマートファクトリー§

図2は、スマートファクトリーの例である。このケースでは、現場レベル、セル、⼯場ラインの制御装置が、PROFINET、Modbus、OPC UA TSN、EtherCAT、CANなどの産業⽤通信プロトコルに基づいて、様々な⼯場設備をオートメーション化している。産業⽤のエッジデバイスは、様々な制御装置からデータを選択収集して、例えば、ダッシュボードを⽤いた遠隔監視などのクラウドのバックエンドサービスでデータを利⽤できるようにしたり、予防保全のためにデータの分析を⾏う。



監

視

、

予

測



サービス



図2 スマートファクトリー

スマートファクトリーでは、接続された製造設備と製品の⾼度な監視が必要である。機械の故障を予測し、異常を早期に発⾒して、コストのかかるダウンタイムとメンテナンスの労⼒を防ぐことで恩恵を受けられる。

さらに、有毒ガス、過度な騒⾳や熱の存在に関して、接続された製造設備と⽣産設備の環境を監視することにより、労働者の安全性が向上し、事件や事故のリスクが減少する。

⽣産設備のリアルタイム監視と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のに役⽴つ。

4.1.3 輸送と物流 §

⾞両、燃料費、メンテナンスの必要性、割り当ての監視は、業務⽤⾞両を最⼤限活⽤できるように最適化するのに役⽴つ。

輸送品の⼀貫した品質と状態を確保するために、積み荷が配送中であることを追跡できる。これは、倉庫から冷蔵トラック、配達までのコールドチェーンの完全性を⾔明するのに特に役⽴つ。

倉庫とヤードでの⼀元的な在庫の監視と管理により、在庫切れや過剰在庫の状況を防ぐことができる。

4.1.4 公益事業 §

住宅や商⼯業 (C&I) のメーターの⾃動読み取りと請求により、資源の消費と潜在的なボトルネックに関する継続的な洞察が得られる。

分散型再⽣可能エネルギー⽣成装置の状態と出⼒を監視することにより、分散型エネルギー資源の最適化が可能になる。配電設備の監視と遠隔制御は、配電プロセスの⾃動化に役⽴つ。

発電と配電のインフラストラクチャの継続的な監視により、公益事業の現場担当者の安全性が向上している。

4.1.5 オイルとガス §

タンクや貯蔵庫における貯蔵量を監視および制御することに加えて、海洋プラットフォームの監視や、パイプラインの漏れを検知および予測することは、環境のために加えて，従業員にとっての労働安全性改善に役⽴つ。

様々な貯蔵タンクと配送パイプ/トラックを通じた分散型在庫の⾃動計算により、計画の改善と資源の最適化が可能となる。

4.1.6 保険 §

連結構造物、業務⽤⾞両などの⾼い価値を持つ資産の予防的資産監視により、事件の予測と早期発⾒を通して、深刻な損害と⾼い損失のリスクを軽減する。

利⽤状況の追跡とカスタマイズされた保険契約を⽤いて、利⽤ベースの保険を提供できる。

予測的な気象監視を⾏い、業務⽤⾞両を屋根付き⾞庫にルート変更させると、ひょう害、樹

⽊の被害による損失を抑えることができる。

4.1.7 ⼟⽊⼯事と建設 §

産業上の安全性を監視することにより、セキュリティ上の危険性のリスクが軽減される。建設現場の資産を監視することで、被害や損失を防⽌できる。

4.1.8 農業 §

⼟壌の状態監視と、散⽔、施肥の最適な計画の作成、農産物の状態監視により、農産物の品質と⽣産量が最適化される。

4.1.9 医療 §

臨床試験データのデータ収集と分析は、新しい領域に関する洞察を得るのに役⽴つ。

遠隔患者モニタリングは、⾼齢者や⼊院後の患者の重篤な状況を⾒逃すリスクを軽減する。

4.1.10 環境モニタリング §

環境モニタリングは通常、測定データを共通のゲートウェイ、エッジデバイス、クラウドサービスに送信する多くの分散型センサーに依存している。

クリティカルな環境状態を検出するために、⼤気汚染や⽔質汚染、そして、微粉塵、オゾン、揮発性有機化合物、放射能、温度、湿度などのその他の環境リスク要因を監視することにより、回復不能な健康や環境の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、運河の資材の状態、劣化、振動の監視により、保守修理作業を発⾒し、重⼤な被害を防⽌することができる。幹線道路の監視と適切な標識の提供により、最適な交通の流れが確保される。

スマートパーキングにより、駐⾞スペースの利⽤と可⽤性の最適化と追跡が⾏われ、課⾦/予約が⾃動化される。

存在検知、天気予報などに基づく街灯のスマート制御により、コストが削減される。

ゴミ容器の監視により、廃棄物管理やゴミ収集ルートを最適化できる。

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監視は、資源消費の最適化と無駄の削減に役⽴つ。

冷暖房空調設備 (HVAC)、エレベーターなどのビル内の設備を監視し、早期に問題を解決することで、居住者の満⾜度が向上する。

4.1.13 コネクテッドカー §

稼働状況の監視、サービスニーズの予測により、メンテナンスの必要性とコストが最適化される。危機的な道路・交通状況に関する早期警告システムの通知により、ドライバーの安全性が強化される。

### 翻訳者のメモ

「稼働状況の監視、サービスニーズの予測により、メンテナンスの必要性とコストが最適化される。」という部分は、「適切なコストで適切なメンテナンスがなされる」という趣旨である。』

4.1.13.1 コネクテッドカーの例§

図3は、コネクテッドカーの例である。このケースでは、ゲートウェイはCANを介して⾞の部品に接続され、独⾃のインターフェースを介してカーナビゲーションシステムに接続される。クラウドで実⾏されているサービスは、交通のパターンを判断するために、⾞の部品からプッシュ配信されるデータを収集し、複数の⾞からのデータを分析する。このケースでは、ゲートウェイはクラウドサービスを利⽤して交通データを取得し、カーナビゲーションシステムを通じてドライバーに表⽰することもできる。

CAN



CAN



DCU

ゲートウェイ

ECU

ECU

ECU

サービス

図3 コネクテッドカー

## 4.2 共通パターン §

この項では、デバイス/Thingが、コントローラー、他のデバイス、エージェント、およびサーバー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のパターンを紹介する。この項では、トランスポートプロトコルの発動要素としてクライアントの役割 (client role) という⽤語を⽤い、トランスポートプロトコルの受動要素としてサーバーの役割 (server role) という⽤語を⽤いる。これは、システム構成要素に特定の役割を定めることを意味するものではない。⼀つのデバイスは、クライアントとサーバーの役割を同時に持つことができる。

### 翻訳者のメモ

英語原本中では「client role」が<em>要素によって強調されている⼀⽅で，「server

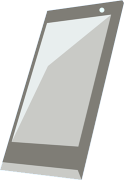
role」は強調されていないが，「server role」も同様に<em>要素により強調するべきと考えられる．

この⼆重の役割の例の⼀つはセンサーであり、クラウドサービスに⾃⾝を登録するとともに、センサーの測定値を定期的にクラウドに送信する。応答メッセージでは、クラウドは、センサーのメッセージ送信速度を調整したり、特定のセンサー属性を選択したりすることができ、これらは今後送信されることになるメッセージを対象としたものである。センサーは⾃⾝をクラウドで登録して接続を開始するため、これは「クライアント」の役割である。しかし、応答メッセージで送信されるリクエストにも反応するため、「サーバー」の役割も果たす。以下の項では、複雑さが増しつつある役割、タスク、ユースケースのパターンについて説明する。これらは網羅的なものではなく、この仕様の後半の項で定義しているWoTアーキテクチャと構成要素を動機付けるために提⽰されるものである。

4.2.1 デバイスコントローラー §

図4で⽰しているように、最初のユースケースは、ユーザが操作するリモートコントローラーで制御されるローカルなデバイスである。リモートコントローラーは、ローカルなホームネットワークを介して電⼦機器に直接アクセスできる。このケースでは、リモートコントローラーはブラウザやネイティブなアプリケーションで実装できる。

このパターンでは、電⼦機器などの少なくとも⼀つのデバイスには、他のデバイスからのリクエストを受け⼊れて応答できるサーバーの役割があり、時として機械的なアクションを開始する。リモートコントローラーのような他のデバイスには、センサー値の読み取りやデバイスの電源投⼊などの、リクエストに応じてメッセージを送信できるクライアントの役割がある。さらに、デバイスの現在の状態やイベントの通知を送信するために、デバイスは、サーバーの役割を持つ別のデバイスに対してメッセージを送信できるクライアントの役割を持つことができる。



リモート

コントローラー

電

⼦

機

器

信頼できる環境

図4 デバイス制御

4.2.2 ThingとThing §

図5は、直接的なThingとThing (Thing-to-Thing) の相互作⽤の例を⽰している。シナリオは次のとおりである。センサーが部屋の状況変化、例えば、温度が基準値を超えていることを検出し、電⼦機器に対して「オンにする」などの制御メッセージを発信する。センサーユニットは、他のデバイスにトリガーメッセージを発信できる。

このケースでは、サーバーの役割を持つ⼆つのデバイスが接続されている場合、少なくとも⼀⽅のデバイスには、他⽅に対して作動させたり通知するためにメッセージを発信するクライアントの役割もなければならない。



制

御

エージェント

電

⼦

機

器



信頼できる環境

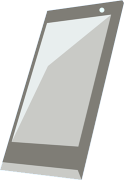
図5 制御エージェント

4.2.3 リモートアクセス §

図6で⽰しているように、このユースケースには、モバイルリモートコントローラー (例えば、スマートフォン) が含まれる。リモートコントローラーは、セルラーネットワークや、

Wi-FiやBluetoothなどのプロトコルを⽤いたホームネットワークなどの様々な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。コントローラーがホームネットワークにある場合、それは信頼できるデバイスであり、セキュリティやアクセス制御を追加する必要はない。コントローラが信頼できるネットワークの外にある場合は、アクセス制御やセキュリティのメカニズムを追加適⽤して、信頼関係を確保しなければならない。このシナリオでは、様々なネットワークアクセスポイントやセルラー基地局間の切り替えにより、ネットワークの接続性が変わりえることに注意すること。

このパターンでは、図4の関連するシナリオと同様に、リモートコントローラーと電⼦機器は、クライアントとサーバーの役割を持っている。



リモート

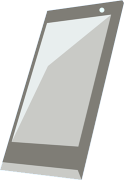
コントローラー

電

⼦

機

器



リモート

コントローラー

信

頼

でき

る

環

境

図6 複数のネットワークインターフェース

4.2.4 スマートホームゲートウェイ §

図7は、スマートホームゲートウェイを⽤いたユースケースを⽰している。このゲートウェイは、ホームネットワークとインターネットの間に位置づけられる。これは、屋内の電⼦機器を管理し、前述のユースケースのようにスマートフォンからなど、インターネット経由でリモートコントローラーからの命令を受信できるようにする。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、プロキシとファイアウォールの機能を提供する。

### 翻訳者のメモ

英語原本中、「It is also is a virtual representation of a device.」とあるが、⼀⽂中に

「is」が⼆つあり，⼆つ⽬の「is」 (=「is a virtual representation」の「is」) は不要。

このパターンでは、ホームゲートウェイは、クライアントとサーバーの両⽅の役割を持っている。リモートコントローラーが電⼦機器を作動させた時に、クライアントの役割の電⼦機器とサーバーの役割のリモートコントローラーとの接続を可能とする。電⼦機器がリモートコントローラーにメッセージを送信する際は、ゲートウェイは、電⼦機器に対するサーバーの役割を果たし、リモートコントローラーに対するクライアントの役割を果たす。

### 翻訳者のメモ

英語原本中、「the gateway act as server roles ... and it act as client roles」とある

が、「gateway」が単数形であることから、「act」は、いずれも「acts」が正しいと考えられる。 また、「act server roles」および「act client roles」が正しいと考えられる。



電

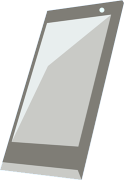
⼦

機

器



ゲートウェイ



リモート

コントローラー

信

頼

でき

る

環

境

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 §

エッジデバイスまた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類似している。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⽰すために⽤いる。図8のホームゲートウェイは主に公開されているネットワークと信頼できるネットワークの間を橋渡しするだけだが、エッジデバイスにはローカルな計算性能があり、通常は、異なるプロトコル間の橋渡しを⾏う。エッジデバイスは通常、産業界のソリューションで⽤いられ、その場合には、接続されたデバイスとセンサーが提供するデータの前処理、フィルタリング、集約を⾏うことができる。



電

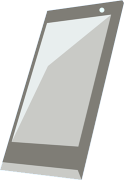
⼦

機

器



エッジデバイス



リモート

コントローラー

信

頼

でき

る

環

境



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である。つまり、クラウドサーバーやエッジデバイスに存在するデバイスまたはデバイス群のモデルである。これは、継続的にオンライン上に存在するとは限らないデバイスを表すため、または、新しいアプリケーションやサービスを実際のデバイスに展開する前にシミュレーションを実⾏するために使⽤される。

接

続

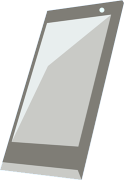
さ

れ

た

デバイス

デジタルツイン



デバイス

クライアント

信

頼

でき

る

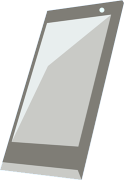
環

境

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もでき、結合されたデバイスの仮想表現に複数のデバイスを集約することもできる。

デジタルツイン



デバイス

クライアント

接

続

さ

れ

た

デバイス

接

続

さ

れ

た

デバイス

接

続

さ

れ

た

デバイス

図10 複数のデバイスのデジタルツイン

デジタルツインは、デバイスが既にクラウドに接続されているか、それとも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ている例を⽰す。クラウドは機器をミラーリングし、デジタルツインとして機能し、リモートコントローラー (例えば、スマートフォン) から命令を受信する。デジタルツインはグローバルに到達可能であるため、承認されたコントローラーはどこにでも設置できる。



電

⼦

機

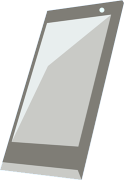
器



機

器

ツイン



リモート

コントローラー

信

頼

でき

る

環

境

図11 クラウド対応デバイスの機器ツイン

4.2.6.2 旧式デバイス§

図12は、旧式の電⼦機器をクラウドに直接接続できない例を⽰す。ここでは、接続を中継するためにゲートウェイが必要である。ゲートウェイは次のような機能を果たす。

物理的と論理的の両⽅の観点での様々な旧式の通信プロトコルのインテグレータインターネットに対するファイアウォール

実際の画像や⾳声を置き換え、データをローカルで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なエージェント

⽕災警報や同様のイベントが発⽣したときにローカルで実⾏される緊急サービス

クラウドは、接続されたすべての機器とゲートウェイをミラーリングし、それらをゲートウェイと連携してクラウド内で管理するデジタルツインとして機能する。さらに、クラウドはどこにでも設置できるリモートコントローラー (例えば、スマートフォン) からの命令を受信することができる。



電

⼦

機

器



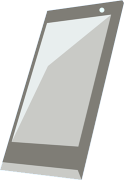
ゲートウェイ



機

器

ツイン



リモート

コントローラー

信

頼

でき

る

環

境

図12 旧式デバイスのデジタルツイン

4.2.7 マルチクラウド §

典型的なIoTのデプロイメントは、複数 (数千) のデバイスで構成される。標準的なメカニズムがなければ、特定のクラウドのためのファームウェア更新の管理には、多⼤な労⼒が必要であり、IoTの広範な採⽤の妨げとなる。

デバイスおよびデバイス型を記述するための標準的なメカニズムの主な利点は、デバイスに対してソフトウェアやファームウェアのレベルでカスタマイズを⾏う必要なく、つまり、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く、デバイスを異なるクラウド環境にデプロイできることである。これは、このソリューションは、複数のIoTクラウド環境

のデバイスに対して新規に接続し (on-boarding) 使⽤することを可能とするような形でデバイスの記述ができるくらいに柔軟なものであることを意味する。これにより、既存のデバイスをクラウドからクラウドへ移⾏することが可能になるとともに、既存のデプロイメントにおける新しいデバイスの利⽤が簡単になるため、Web of

Thingsデバイスの採⽤が促進される。

4.2.8 領域横断型連携 §

図13は、領域横断型連携の例を⽰している。この場合、各システムは、スマートファクトリーとスマートシティ、スマートシティとスマートホームのように、他の領域に含まれる他のシステムと関わり合いを持っている。 [IEC-FOTF] で⽰す通り、この種のシステムは「共

⽣」 (symbiotic) エコシステムと呼ばれる。その中には、直接的連携と間接的連携の⼆つのモデルがある。直接的連携モデルでは、システムはピアツーピア (peer-to-peer) ⽅式で互いに情報を直接交換する。間接的連携では、システムは何らかの連携プラットフォームを介して情報を交換する。この連携を維持・継続するために、各システムは⾃⾝の能⼒やインタフェースに関するメタデータを提供し、他のシステムに適応させる。

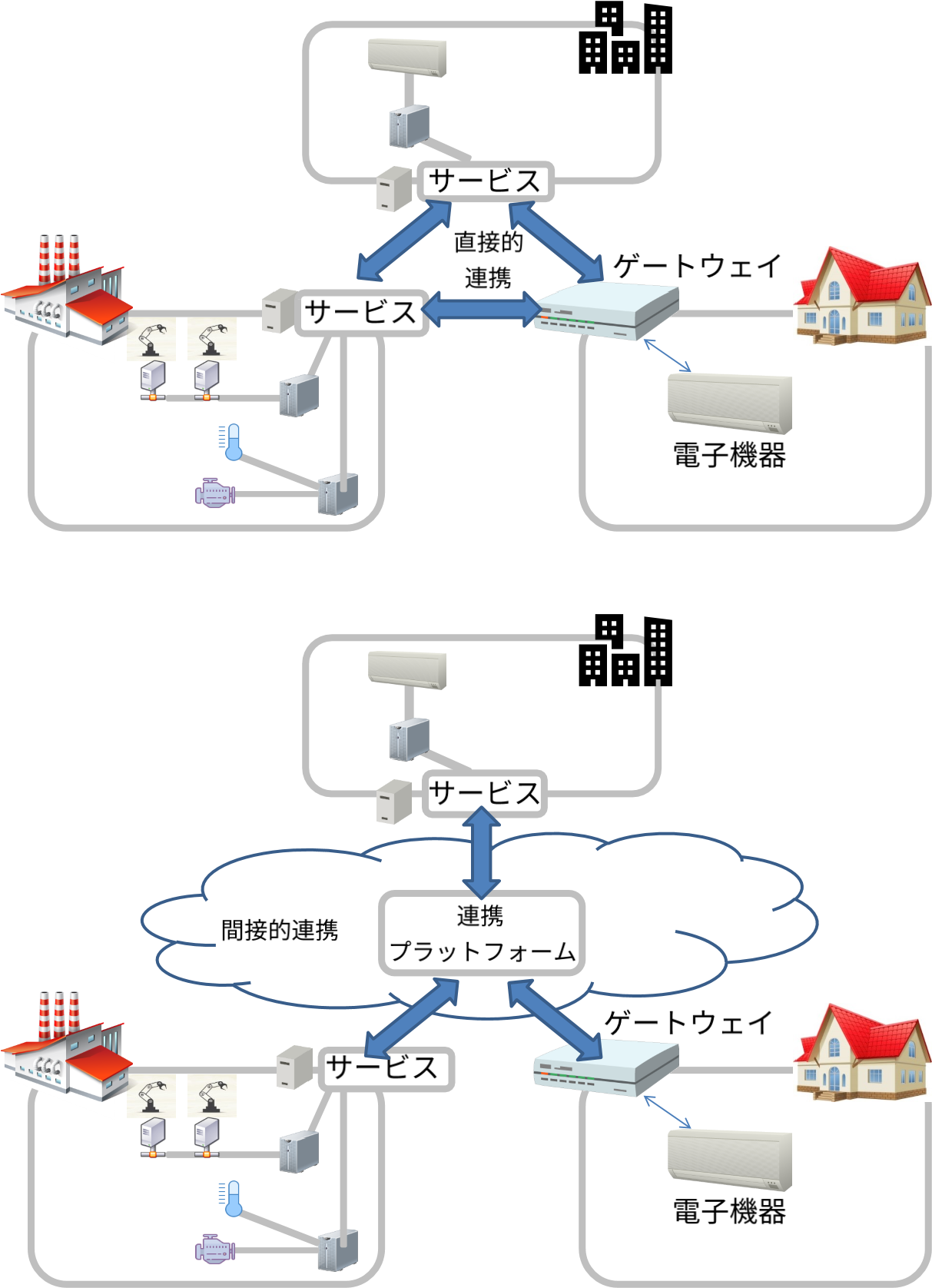


図13 領域横断型連携

## 4.3 要約 §

前項では、様々なアーキテクチャのパターンについて説明した。これらのパターンでは、旧式デバイス、コントローラー、ゲートウェイ、クラウドサーバーを含むデバイスなどの⼀部の機能のエンティティーは、建物の内外やデータセンターなどの物理的な場所に置かれる。

図14は、これらのエンティティーの組み合わせと通信経路を⽰した概要である。

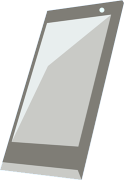
トランスポートプロトコル層では、各エンティティーが通信に適した役割を任意に選択する。例えば、デバイスが不特定多数のアプリケーションにサービスを提供する場合、そのデバイスはサーバーとして機能する。⼀⽅で、デバイスのネットワーク接続が制限されていたり断続的であったりする場合は、デバイスはクライアントとして機能し、ネットワークが利⽤できるときにアプリケーションにメッセージを活発に送信することもできる。これに関係なく、アプリケーション層では、アプリケーションは、デバイスが相互作⽤を⾏うための抽象的なインターフェースを提供していることを理解し、その抽象的なインターフェースを⽤いてデバイスと相互作⽤を⾏うことができる。

### 翻訳者のメモ

英語原本中で「On the other hand, if a device has limited or intermittent network

connectivity, they may act as a client...」とあるが、「they may act as...」の

「they」は、「device」に対する代名詞であることから「it」が正しいと考えられる。その他、いくつかの誤字について、W3C側に指摘ずみ。



リモート

コントローラー

電

⼦

機

器

信

頼

でき

る

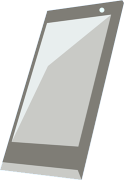
環

境

制

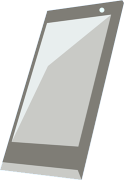
御

エージェント



ゲートウェイ

エッジデバイス



接

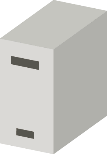
続

さ

れ

た

デバイス



サービス



領

域

横

断

型

連

携

サービス

デジタルツイン

図14 ユースケースの概要

# 5. 要件 §

この章は規定である。

## 5.1 機能要件 §

この節では、Web of Things (WoT) の抽象アーキテクチャに必要な特性を定義する。

5.1.1 ⼀般的な原則 §

WoTアーキテクチャは、ウェブ技術を⽤いて異なるエコシステムの相互運⽤を可能にすべきである。

WoTアーキテクチャは、RESTful APIを⽤いたウェブアーキテクチャに基づくべきである。

WoTアーキテクチャでは、ウェブで⼀般的に⽤いられている複数のペイロード形式を使

⽤できるべきである。

WoTアーキテクチャでは、様々なデバイスアーキテクチャが可能でなければならず、システム構成要素として特定のクライアントやサーバーの実装を強制してはならない。

柔軟性

WoT実装には、様々な物理デバイスの構成がある。WoTの抽象アーキテクチャは、そのすべてのバリエーションに対してマッピングができ、それらをカバーできるべきである。

互換性

多くの業界には、すでに多くの既存のIoTソリューションと進⾏中のIoT標準化活動がある。WoTは、これらの既存および開発中のIoTソリューションとウェブ技術との間の橋渡しをWoTの概念に基づいて提供すべきである。WoTは、既存のIoTソリューションおよび現在の標準の上位互換であるべきである。

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む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、製造者が異なっていても同じ機能を提供する可能性がある。

相互運⽤性

WoTは、デバイスとクラウドの製造者にまたがる相互運⽤性を提供しなければならない。WoT対応デバイスをデバイスと異なる製造者のクラウドサービスに追加設定なしで接続できなければならない。

5.1.2 Thingの機能 §

WoTアーキテクチャでは、Thingが次のような機能を持てるようにすべきである。

Thingのステータス情報を読む。

動作を発⽣させるThingのステータス情報を更新する。

Thingのステータス情報の変更通知の登録、受信、登録解除を⾏う。⼊出⼒パラメータにより、特定の動作または計算を発⽣させる機能を呼び出す。単なる状態遷移の報告だけでなく、⼀般的なイベントの通知の登録、受信、登録解除を⾏う。

5.1.3 検索と発⾒ §

WoTアーキテクチャでは、クライアントは、Thing⾃体にアクセスする前に、Thingの属性、機能、およびアクセスポイントを知ることができるべきでである。

WoTアーキテクチャでは、クライアントはその属性と機能によってThingを検索できるべきである。

WoTアーキテクチャでは、機能の名前に関係なく、統⼀された語彙に基づいて必要な機能を提供するThingのセマンティックな検索ができるべきである。

5.1.4 記述⽅法 §

WoTアーキテクチャは、Thingとその機能の記述を可能にする⼀般的な記述⽅法をサポートすべきである。

このような記述は、⼈が読めるだけでなく、機械処理が可能であるべきである。

このような記述により、その構造と記述内容のセマンティックアノテーションが可能となるべきである。

このような記述は、ウェブで⼀般的に⽤いられている複数の形式を⽤いてやり取りできるべきである。

5.1.5 属性の記述 §

WoTアーキテクチャでは、次のようなThingの属性を記述できるべきである。

名前説明

仕様のバージョン、形式、記述内容⾃体

他の関係するThingやメタデータ情報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アーキテクチャでは、§ 5.1.2Thingの機能で⽰しているThingの機能を記述できるべきである。

5.1.7 ネットワーク §

WoTアーキテクチャは、⼀般的に⽤いられている複数のウェブプロトコルをサポートすべきである。

そのプロトコルには、次のようなものが含まれる。

1. インターネットで⼀般的に⽤いられているプロトコル
2. ローカルエリアネットワークで⼀般的に⽤いられているプロトコル

WoTアーキテクチャでは、複数のウェブプロトコルを⽤いて同じ機能にアクセスできるべきである。

WoTアーキテクチャでは、同じThingの機能に対して複数のプロトコルを組み合わせて使⽤できるべきである (例えば、HTTPとWebSocket)。

5.1.8 デプロイメント §

WoTアーキテクチャは、同じモデルに基づいて、資源制限のあるエッジデバイスやクラウド上の仮想的なモノなど、様々なモノの性能をサポートすべきでである。

WoTアーキテクチャは、ゲートウェイやプロキシなどの中間エンティティーを⽤いて、複数レベルのモノ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の変換を考慮して、ローカルネットワークの外部 (インターネットや別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モノへのアクセスをサポートすべきでである。

5.1.9 アプリケーション §

WoTアーキテクチャでは、同じモデルに基づくウェブ標準技術を⽤いて、エッジデバイス、ゲートウェイ、クラウド、UI/UXデバイスなどの、様々なモノのためのアプリケーションを記述できるべきである。

5.1.10 旧式技術への適合 §

### 翻訳者のメモ

5.1.10項の英語原⽂タイトルは「Legacy Adoption」だが、その内容より「旧式技術への適合」について記述したものであると考えられるため、むしろ英語原⽂タイトルは

「Legacy Adaptation」であるべきと考えられる。

WoTアーキテクチャは、旧式のIPプロトコルや⾮IPプロトコルをウェブプロトコルにマッピングできるようにし、そのような旧式のプロトコルが終了および変換された場合の様々なトポロジーをサポートすべきである。

WoTアーキテクチャは、既存のIPプロトコルがRESTfulアーキテクチャに準拠している場合、そのプロトコルを変換せずに透過的な利⽤を可能とするべきである。

WoTアーキテクチャは、デバイスやサービスに対してクライアントやサーバーの役割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応じて、クライアントまたはサーバー、あるいはその両⽅になりえる。エッジサービスとクラウドサービスにおいても同様である。

## 5.2 技術要件 §

§ 4.2 共通パターンは、様々なユースケースを⽰し、アーキテクチャの構成要素を組み合わるためのパターンを列挙することにより、Web of Thingsの抽象アーキテクチャを定義したものである。この項では、抽象アーキテクチャから導かれた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アーキテクチャ §

### 翻訳者のメモ

本5.2.1項の英語原⽂タイトルは「Components in the Web of Things and the Web of

Things Architecture」であり、⽇本語訳は「Web of Thingsの構成要素とWeb of

Thingsアーキテクチャ」となるが、その記述内容は単に「⼀部のユースケースにおいてディレクトリが構成要素として⽤いられること」や「構成要素は⼀つの構成要素に接続される場合と、複数のネットワークに展開されることがある」ということであり、「構

成要素とWeb of Thingsアーキテクチャ」という項タイトルには違和感があるため、⽂書構成を⾒直すべきと考えられる。

ユースケースは、デバイスやアプリケーションにアクセスしたり、制御したりするデバイスやアプリケーション、デバイス間にあるプロキシ (例えば、ゲートウェイやエッジデバイス) などの基本的な構成要素を識別するのに役⽴つ。⼀部のユースケースで有⽤な追加的な構成要素は、発⾒を⽀援するディレクトリである。

これらの構成要素は、インターネットや、オフィス、⼯場、その他の施設のフィールドネットワークに接続される。関係するすべての構成要素が⼀つのネットワークに接続されている場合もあるが、⼀般的に、構成要素は複数のネットワークに展開される。

5.2.2 デバイス §

デバイスへのアクセスは、デバイスの機能とインターフェースの記述を⽤いて⾏われる。この記述をThing Description (TD) と呼ぶ。Thing Description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、情報モデルで稼働するためのトランスポートプロトコルの記述、およびセキュリティ情報が含まれる。⼀般的なメタデータには、デバイスの識別⼦ (URI)、シリアル番号、製造⽇、場所などのデバイス情報、その他の⼈間が読める情報が含まれる。

情報モデルは、デバイスの属性を定義し、デバイスの内部設定、制御機能、通知機能を表す。同じ機能を持つデバイスは、⽤いるトランスポートプロトコルに関係なく、同じ情報モデルを有する。

### 翻訳者のメモ

英語原本中では「Information models defines device attributes」となっているが、主語が「Information models」と複数形であることを考慮すると、「define」が正しいと思われる。

WoTアーキテクチャに基づく多くのシステムは、システム領域を越えて利⽤されるため、関係者は、情報モデルで⽤いられる語彙とメタデータ (オントロジーなど) に共通の理解を持っているべきである。RESTトランスポートに加えて、PubSubトランスポートもサポートされている。

### 翻訳者のメモ

上記⽇本語訳中で「システム領域を越えて」としている箇所は、英語原本中で

「crossing system Domains」となっているが、「Domains」の頭⽂字が⼤⽂字である理由が不明であり、単に「domains」とすべきであると思われる。

セキュリティ情報には、認証、認可、安全な通信に関する記述が含まれる。デバイスは、TD をデバイスの内部か外部のいずれかに置き、TDをアクセス可能にして、他の構成要素がその

TDを⾒つけてアクセスできるようにする必要がある。

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とプログラムのインターフェースを⽣成し使⽤できる必要がある。

アプリケーションはネットワークを通じてこの記述を取得できなければならないため、ネットワーク上で検索を⾏って必要な記述を取得できる必要がある。

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) に基づいて内部でプログラムのインターフェースを⽣成し、そのプログラムのインターフェースを⽤いて仮想デバイスを表す必要がある。ツインは、仮想デバイス⽤の記述を作成し、それを外部で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、元のデバイスとは異なる。これにより、仮想デバイスと元のデバイスが明確に別のエンティティーとして認識されるようになる。トランスポートとセキュリティのメカニズムと仮想デバイスの設定は、必要に応じて元のデバイスと異なる可能性がある。仮想デバイスは、ツインから直接提供される記述を持っているか、外部で利⽤できる記述を持っている必要がある。いずれの場合も、他の構成要素がデジタルツインに関係付けられるデバイスを⾒つけて利⽤できるように、記述を提供する必要がある。

5.2.5 ディスカバリ §

デバイス、アプリケーション、およびツインからデバイスと仮想デバイスのTDにアクセスするためには、TDを共有する共通の⽅法が必要である。ディレクトリは、デバイスとツイン⾃

⾝が⾃動で、またはユーザが⼿動で記述を登録できる機能を提供することで、この要件を満たすことができる。

デバイスと仮想デバイスの記述は、外部エンティティーにより検索できる必要がある。ディレクトリは、デバイスの記述や情報モデルの⼀般的な記述に含まれているキーワードなどの検索キーを⽤いて検索を処理できる必要がある。

5.2.6 セキュリティ §

デバイスと仮想デバイスに関するセキュリティ情報は、デバイスの記述に記載する必要がある。これには、認証・認可およびペイロード暗号化に関する情報が含まれる。

WoTアーキテクチャは、Basic、Digest、Bearer、OAuth2.0などの、ウェブで⼀般的に⽤いられている複数のセキュリティメカニズムをサポートすべきである。

5.2.7 アクセシビリティ §

Web of Thingsは、主に機械同⼠の通信を対象としている。関係のある⼈間は通常、Thing をアプリケーションに統合する開発者である。エンドユーザは、アプリケーションのフロントエンド、またはデバイス⾃⾝が提供する物理的なユーザインターフェースと対⾯する。どちらもW3C WoT仕様の範囲外である。ユーザではなくIoTに焦点を当てていることから、アクセシビリティは直接的な要件ではないため、この仕様では扱っていない。

しかし、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、機械はデバイスのネットワーク向けAPIを処理できる。アクセシビリティツールは、これを利⽤して、異なるモダリティのユーザインターフェースを提供できるため、物理的デバイスやIoT 関連のアプリケーションを⽤いる際の障壁が取り除かれる。

# 6. WoTアーキテクチャ §

この章は規定である。

4章のユースケースに対処し、5章の要件を満たすために、Web of Things (WoT) は、いわゆるConsumerが使⽤できるウェブのThing — 通常は単にThingと呼ばれる — の概念の上に構築される。この項では、W3C Web of Thingsの全体アーキテクチャを定義するための背景と規定的な⾔明を提供する。Web of Thingsは、様々な領域の利害関係者に対応するため、ウェブ技術の特定の側⾯、特にハイパーメディアの概念について詳しく説明する。

## 6.1 概要 §

Thingは、物理的または仮想的なエンティティー (例えば、デバイスや部屋) の抽象化であ

り、標準化されたメタデータで記述される。W3C WoTでは、記述メタデータはWoT Thing

Description (TD) [WOT-THING-DESCRIPTION] でなければならない (MUST)。

Consumerは、JSON [RFC8259] に基づくTD表現形式を解析し処理できなければならない

(MUST)。基礎となる情報モデルはグラフベースであり、そのシリアライゼーションは

JSON-LD 1.1 [JSON-LD11] と互換性があるため、この形式は従来のJSONライブラリか、 JSON-LDプロセッサのいずれかで処理できる。TDの処理にJSON-LDプロセッサを⽤いる

と、RDFトリプルへの変換、セマンティックな推論、オントロジー⽤語に基づいて与えられたタスクの実⾏などの、セマンティックな処理がさらに可能になり、Consumerがより⾃律的に⾏動できるようになる。TDはインスタンス固有であり (つまり、Thingの種類ではなく個々のThingを記述する)、デフォルトでは、外付けかつテキスト形式のThingの (ウェブ) 表現である。HTMLベースのユーザインターフェース、物理エンティティーの単なる画像、さ

らに、閉じたシステム内のウェブ以外の表現など、その他のThingの表現が存在しえる

(MAY)。

しかし、Thingであるためには、少なくとも⼀つのTD表現が利⽤できなければならない

(MUST)。WoT Thing Descriptionは、ConsumerがThingの機能を発⾒して解釈し (セマンティックなアノテーションを介して)、Thingとの相互作⽤時に様々な実装 (例えば、様々なプロトコルやデータ構造) に適応できるようにする、機械が理解できる標準化された表現形式であり、それにより、様々なIoTプラットフォーム、つまり、様々なエコシステムや標準にまたがる相互運⽤が実現される。

**Thing**

**Consumer**

記

述

処

理

相

互

作

⽤

**WoT Thing**

**Description**

図15 ConsumerとThingの相互作⽤

Thingは、仮想エンティティーの抽象化にもなりえる。仮想エンティティーは、⼀つ以上の

Thingの合成物である (例えば、いくつかのセンサーとアクチュエーターで構成される部

屋)。合成物の⼀つのオプションは、仮想エンティティーに関する機能のスーパーセットを含んだ⼀つの統合されたWoT Thing Descriptionを提供することである。合成物がかなり複雑な場合は、そのTDは合成物内の下位階層にあるThingにリンクすることができる。中⼼的な

TDはエントリーポイントとして機能し、⼀般的なメタデータのみ、そして、場合によっては包括的な機能を含む。これにより、より複雑なThingの特定の側⾯をグループ化できる。

リンクは、階層的なThingに対してのみでなく、Thingとその他の資源との⼀般的な関係にも適⽤される。リンク関係型は、例えば、照明を制御するスイッチや、モーションセンサーで監視している部屋など、Thing間の関連性を表現する。Thingに関連付けられるその他の資源には、マニュアル、予備の部品のカタログ、CADファイル、GUIや、その他のウェブ上の⽂書がある。全体として、Thingの間のウェブリンクにより、⼈間と機械の両⽅がWeb of

Thingsをたどっていけるようになる。これは、利⽤可能なThingのカタログを管理する

Thing Directoryを提供する (通常は、TD表現をキャッシュすることによる) ことでさらに容易になる。要約すると、モノのウェブを形成するために、WoT Thing Descriptionを他の

Thingやウェブ上の他の資源にリンクしてもよい (MAY)。

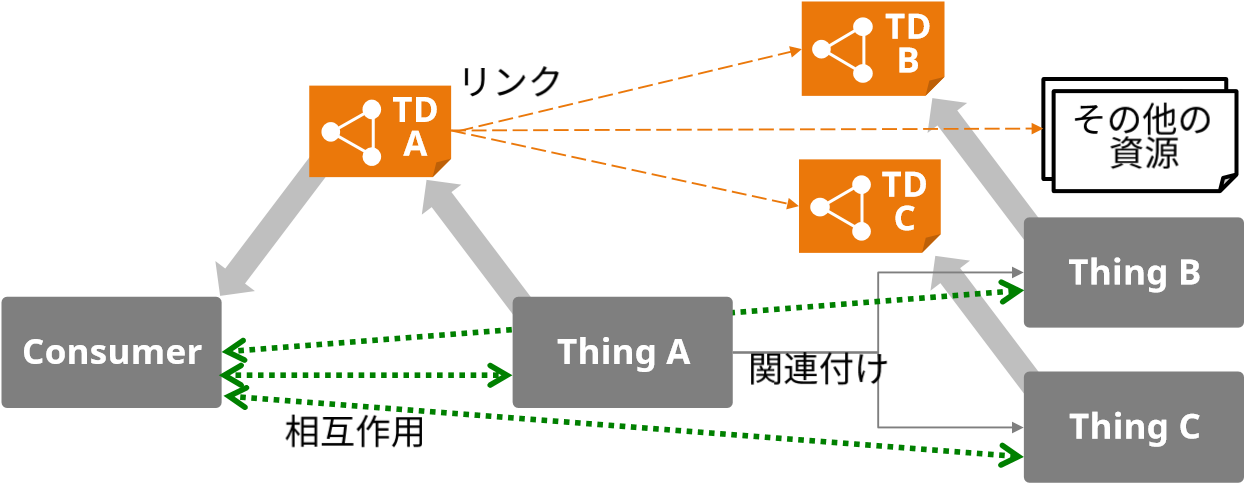


図16 リンクされたThing

ThingのWoTインターフェースであるネットワーク向けインターフェースを介した相互作⽤を実現するためには、ソフトウェアスタックを備えたネットワーク化されたシステム構成要素上でThingをホストしなければならない。この例の⼀つは、Thingの抽象化の背後にある物理エンティティーのインターフェースとなっている、センサーとアクチュエーターを備えた組み込みデバイスで実⾏されているHTTPサーバーである。しかし、W3C WoTは、Thing を提供する場所を強制しておらず、IoTデバイスに直接置いたり、ゲートウェイなどのエッジデバイスや、クラウドに置くことができる。

典型的なデプロイメントの課題は、通常、IPv4ネットワークアドレス変換 (NAT) やファイアウォールデバイスによりインターネットからローカルネットワークに到達できないというシナリオである。この状況を改善するために、W3C WoTはThingとConsumerの間に

Intermediaryを認めている。

IntermediaryはThingのプロキシとして機能できる。その場合、Intermediaryは、元の Thingと似たWoT Thing Descriptionを持っているが、それはIntermediaryが提供する

WoTインターフェースを指し⽰す。Intermediary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を構成し、それにより仮想エンティティーを形成することもできる。Intermediaryは、WoT Thing Descriptionを持ち、WoTインターフェースを提供するため、ウェブ [REST] のような階層化したシステムアーキテクチャのThingと

⾒分けがつかない場合があり、ConsumerにはThingと同じに⾒える。WoT Thing

Descriptionの識別⼦は、同じ元のThingまたは最終的に⼀意の物理エンティティーを表す複数のTDを関連づけることができなくてはならない (MUST)。

**Thing A**

**Thing B**

**TD**

**A**

**Consumer**

**Intermediary**

相

互

作

⽤

**TD**

**B**

**TD**

**C**

図17 Intermediary

制限のあるローカルネットワークの別の改善策は、WoTインターフェースを、ローカルネットワーク内のThingから公開されている到達可能なConsumerへの接続を確⽴するプロトコルにバインドすることである。

ThingをConsumerと束ねて、ThingとThingで相互作⽤することができる (MAY)。通常、 Consumerの振る舞いはソフトウェアの構成要素に埋め込まれており、それはThingの振る舞いも実装している。Consumerの振る舞いの設定は、Thingを通じて公開することができる (MAY)。

W3C WoTの概念は、デバイスレベル、エッジレベル、クラウドレベルという、IoTアプリケーションに関連するすべてのレベルに適⽤できる。これにより、様々なレベルで共通のインターフェースとAPIが促進され、ThingとThing、Thingとゲートウェイ、Thingとクラウ

ド、ゲートウェイとクラウド、さらにはクラウドフェデレーション、つまり、IoTアプリケーションに関する⼆つ以上のサービス提供者のクラウドコンピューティング環境の相互接続などの、様々な統合パターンが可能になる。図18は、§ 4.3 要約でまとめたユースケースに対処するために、上記で紹介したWoTの概念を適⽤して組み合わせる⽅法の概要を⽰している。

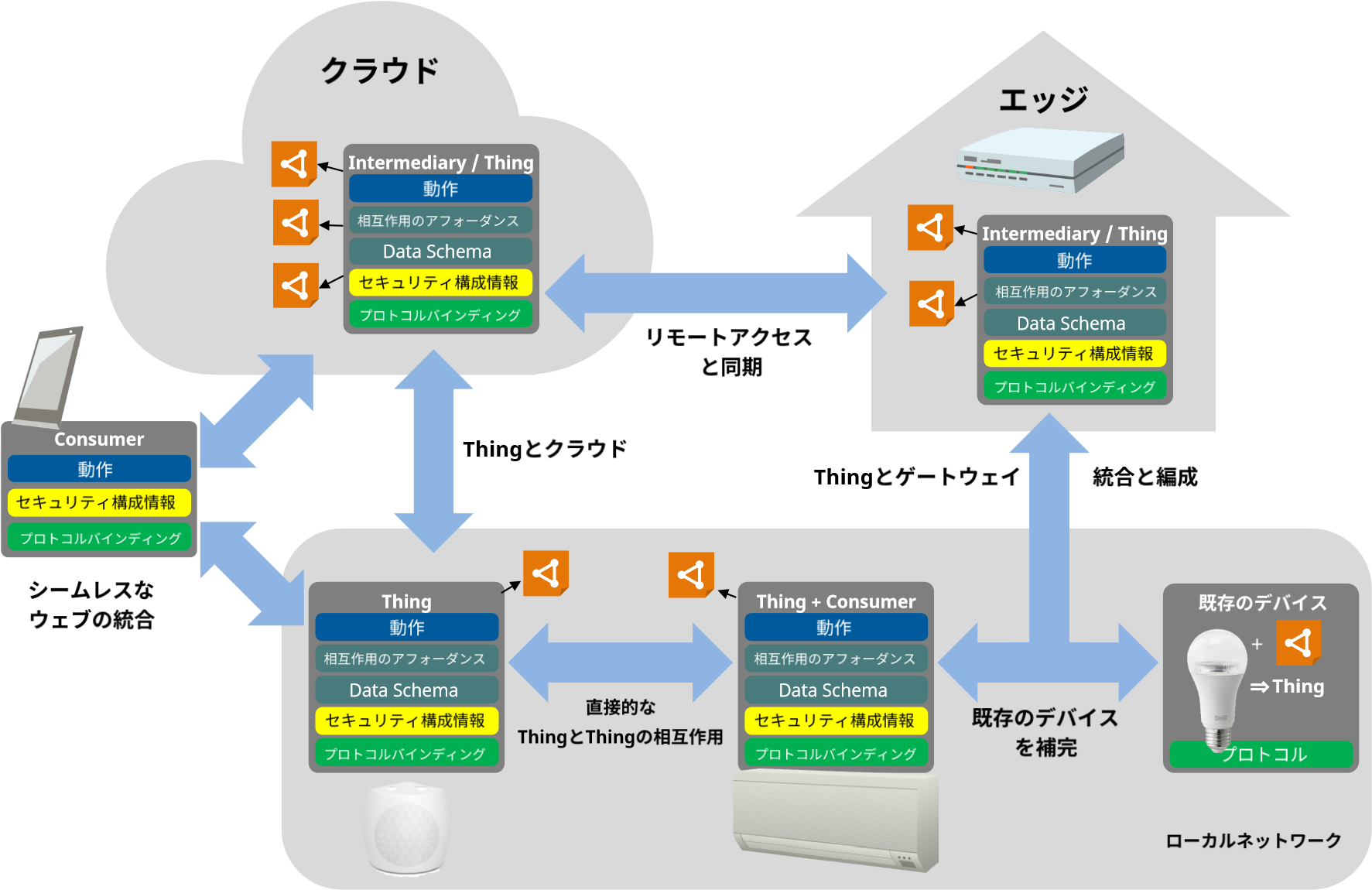


図18 W3C WoTの抽象アーキテクチャ

## 6.2 アフォーダンス §

W3C WoTの中⼼を成すのは、機械が理解できるメタデータ (つまり、) の提供である。理想的には、このようなメタデータは⾃⼰記述的であり、Thingがどのような機能を提供するかと、その提供された機能をどのように⽤いるかをConsumerが識別できる。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、⽣態⼼理学に由来するが、「『アフォーダンス』とは、物の知覚された実際の特性、主に、物がどのように利⽤されうるかを決定する基本的な特性を指す」という Donald Norman の定義に基づくヒューマンコンピューターインタラクション

[HCI] の分野で採⽤された。 [NORMAN]

これに関する例は、取っ⼿のあるドアである。ドアの取っ⼿はアフォーダンスであり、ドアを開くことができることを⽰唆する。⼈間にとって、ドアの取っ⼿は通常、どのようにドアが開くかも⽰す。アメリカのノブは、回転することを⽰唆し、ヨーロッパのレバーハンドルは押し下げることを⽰唆する。

RESTアーキテクチャスタイル [REST] の中核となる基盤の⼀つであるハイパーメディアの原則は、ウェブをナビゲートし、ウェブアプリケーションを制御する⽅法に関する明確な知識を情報の利⽤者が得られるように、ウェブで利⽤できる情報を他の情報にリンクすることを求めている。ここでは、情報と制御の同時表⽰ (ハイパーリンクの形式で提供) は、ウェブクライアントにウェブアプリケーションを動作させる⼿段を提供する (afford) メカニズムである。このコンテキストでは、アフォーダンスはハイパーリンクの記述であり (例えば、リンク関係型とリンクターゲット属性を介した)、ナビゲート⽅法を提案したり、リンクされている資源における動作⽅法をウェブクライアントに提案する可能性がある。したがって、リンクはナビゲーションアフォーダンスを提供する。

このハイパーメディアの原則から導き出されたWeb of Thingsでは、相互作⽤のアフォーダンスを、可能な選択肢をConsumerに⽰して記述したThingのメタデータと定義しており、それによってConsumerがThingと相互作⽤を⾏うことができる⽅法を提案する。⼀般的な相互作⽤のアフォーダンスはナビゲーションであり、これはリンクをたどることで作動し、それによってConsumerがWeb of Thingsを閲覧できるようになる。§ 6.4 相互作⽤モデルは、3種類のW3C WoTの相互作⽤のアフォーダンスであるProperty、Action、Eventを定義している。

全体として、このW3C WoTの定義は、物理的なモノを作成するHCIとインタラクションデザイナー、およびウェブサービス全般に取り組んでいるRESTとマイクロサービスコミュニティと連携を図っている。

## 6.3 Web Thing §

Web Thingには、図19に⽰しているように、その動作、その相互作⽤のアフォーダンス、そのセキュリティ設定、そのプロトコルバインディングという四つのアーキテクチャの側⾯がある。Thingの動作の側⾯には、⾃律的な動作と相互作⽤のアフォーダンスのハンドラーの両⽅が含まれる。相互作⽤のアフォーダンスは、特定のネットワークプロトコルやデータエンコーディングを参照せずに、抽象操作を通じてどのようにConsumerがThingと相互作⽤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、個々の相互作⽤のアフォーダンスを特定のプロトコルの具体的なメッセージにマッピングするために必要な詳細を追加する。⼀般的に、様々な具体的なプロトコルを⽤いて、⼀つのThingの中であっても、相互作⽤のアフォーダンスの様々なサブセットをサポートできる。Thingのセキュリティ設定の側⾯は、相互作⽤のアフォーダンスへのアクセスと、関連するとPrivate Security Dataの管理を制御するために⽤いられるメカニズムを表す。

**Thing**

動

作

相

互

作

⽤

の

アフォーダンス

Data Schema

WoT Thing

Description

プロトコルバインディング

セキュリティ

構

成

情

報

図19 Thingのアーキテクチャの側⾯

## 6.4 相互作⽤モデル §

当初、ウェブ資源とは通常、ウェブクライアントが容易に取得できるウェブ上の⽂書を表していた。ウェブサービスの導⼊により、資源は、あらゆる種類の動作を実装できる、より汎⽤的な相互作⽤のエンティティーとなった。この⾮常に⾼度な抽象化により、多種多様な相互作⽤の可能性があるため、アプリケーションと資源との間を疎結合することが難しくなっ

ている。その結果、執筆時点で、⼀般的なAPI記述は、アプリケーションの意図

(application intent) から資源のアドレス、メソッド、リクエストペイロード構造、応答ペイロード構造、および予期されるエラーへの静的マッピングで構成されている。これにより、ウェブクライアントとウェブサービスが密結合されることとなる。

W3C WoTの相互作⽤モデルは、アプリケーションの意図 (application intent) から具体的なプロトコル操作へのマッピングを形式化する仲介的な抽象化を導⼊し、相互作⽤のアフォーダンスをどのようにモデル化するかの可能性を絞り込む。

ナビゲーションのアフォーダンス (つまり、ウェブリンク) に加えて、Thingは、この仕様で定義しているProperty、Action、Eventという他の3種類の相互作⽤のアフォーダンスを提供できる (MAY)。このナローウエスト (narrow waist) は、ConsumerとThingの分離を可能とする⼀⽅で、これら4種類の相互作⽤のアフォーダンスにより、IoTのデバイスおよびサービスで⾒られるほぼすべての相互作⽤の可能性をモデル化することができる。

6.4.1 Property §

Propertyは、Thingの状態を公開する相互作⽤のアフォーダンスである。Propertyによって公開される状態は、検索可能 (読み取り可能) でなければならない (MUST)。オプションで

Propertyによって公開される状態は、更新可能 (書き込み可能) である (MAY)。Thingは、

変更後に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を選択できる (MAY) (資源の監視 [RFC7641] を参照)。書き込み専⽤の状態は、Actionを介して更新すべきである。

プロトコルバインディングを⽤いているが、データが完全には指定されていない場合 (例えば、メディアタイプにより)、Propertyには公開の状態に関するデータスキーマを⼀つ含めることができる (MAY)。

Propertyの例は、センサー値 (読み取り専⽤)、ステートフルなアクチュエーター (読み書

き)、設定パラメータ (読み書き)、モノの状態 (読み取り専⽤または読み書き)、または計算結果 (読み取り専⽤) である。

6.4.2 Action §

Actionは、Thingの機能を呼び出すことができる相互作⽤のアフォーダンスである。Action は、直接公開されていない状態を操作したり (Propertyを参照)、⼀度に複数のPropertyを操作したり、内部ロジックに基づいてPropertyを操作したりすることができる (MAY) (例えば、トグル)。Actionの呼び出しは、経時的に状態 (アクチュエータを介した物理的な状態を含む) を操作するThingのプロセスを始動させることもできる (MAY)。

⽤いるプロトコルバインディングでデータが完全に指定されていない場合 (例えば、メディアタイプにより)、Actionにはオプションの⼊⼒パラメータと出⼒結果のdata schemaを含めることができる (MAY)。

Actionの例は、複数のPropertyを同時に変更すること、照明の明るさを暗くしたり (減光)、独⾃の制御ループアルゴリズムなどの⾮公開なプロセスを⽤いるなどして経時的にProperty を変更すること、⽂書の印刷などの⻑時間にわたるプロセスを呼び出すことである。

6.4.3 Event §

Eventは、Thingから利⽤者にデータを⾮同期的にプッシュする出来事の発⽣源を記述する相互作⽤のアフォーダンスである。ここでは状態ではなく、状態の遷移 (つまり、イベント) が伝達される。Eventは、Propertyとして公開されていない条件によって始動させることができる (MAY)。

使⽤されるプロトコルバインディングによる (例えば、メディアタイプを通じての) データの指定が完全には⾏われていない場合、Eventには、その出来事に関するデータのためのデータスキーマ、および (例えば、WebhookコールバックURIにより登録するために) 可能な登録制御メッセージを含めることができる (MAY)。

Eventの例は、アラームや時系列のサンプルなどの、定期的にプッシュされる離散的な出来事である。

## 6.5 ハイパーメディア制御 §

ウェブでは、アフォーダンスは情報と制御の同時表⽰であり、その情報はユーザが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しているテキストや画像である。制御は、少なくともターゲット資源のURIが含まれているウェブリンクで、ウェブブラウザで逆参照できる (つまり、リンクをたどることができる)。しかし、さらにウェブリンクが関係型とターゲット属性で記述されている場合、機械は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、どのようにアフォーダンスを作動させるかに関する機械が理解可能な記述である。ハイパーメディア制御は通常、ウェブサーバーから発信され、ウェブクライアントがそのウェブサーバーと相互作⽤を⾏っている間にインバンド (in-band) で⾒つけられる。このようにして、現在の状態や認証などの他の要因を考慮に⼊れることにより、ウェブサーバーはウェブアプリケーションを通じてクライアントを動的に駆動できる。これは、クライアントにプレインストール、またはハードコーディングする必要があるアウトオブバンド (out-of-band) のインターフェースの記述とは対照的である (例えば、RPC、WS-\* ウェブサービス、固定のURIメソッド応答定義を持つHTTPサービス)。

W3C WoTは、ウェブをナビゲートするための確⽴した制御であるウェブリンク [RFC8288] と、あらゆる種類の操作を可能にするより強⼒な制御としてのウェブフォームという2種類のハイパーメディア制御を⽤いる。リンクは既に、CoREリンク形式 [RFC6690] 、OMA

LWM2M [LWM2M] 、OCF [OCF] などの他のIoT標準やIoTプラットフォームで⽤いられている。フォームは、W3C WoT以外に、IETFで定義されている制約付きRESTfulアプリケーション⾔語 (CoRAL) [CoRAL] でも導⼊されている新しい概念である。

6.5.1 リンク §

リンクにより、Consumer (または広義ではウェブクライアント) は、コンテキストとリンクターゲットの関係に応じて、現在のコンテキスト (ウェブブラウザで現在表⽰されている資源の表現など) を変更したり、追加の資源を現在のコンテキストに含めたりすることができる。Consumerは、ターゲットURIを逆参照することで、つまりリンクをたどって資源の表現を取得することでこれを⾏う。

### 翻訳者のメモ

英語原本中で「cf. the set of resource representations currently rendered in the

Web browser」となっている箇所は、直前の「current context」の例⽰であると思われるため、「cf.」というよりはむしろ「e.g.」 (例えば) という形で訳した。

W3C WoTは、Web Linking [RFC8288] の定義に従っており、リンクは次のもので構成される。

リンクのコンテキスト関係型

リンクターゲット

オプションで、ターゲット属性

リンク関係型は、ABNF [RFC5234] LOALPHA \* ( LOALPHA / DIGIT / "." / "-" ) (例え

ば、stylesheet) に準拠していなければならないIANA [IANA-RELATIONS] に登録されている定義済みトークンか、URIの形式の拡張型 [RFC3986] のいずれかである。拡張関係型は、⼤⽂字と⼩⽂字を区別しない⽐較⽅法により、⽂字列として⽐較しなければならない

(MUST)。 (異なる形式でシリアル化されている場合は、URIに変換すべき)。それにも関わらず、拡張関係型には、すべて⼩⽂字のURIを⽤いるべきである (SHOULD)。 [RFC8288]

Web of Thingsでは、リンクは発⾒に⽤いたり、Thing間の関係 (例えば、階層的か、機能的か) やウェブ上の他の⽂書 (例えば、マニュアルや、CADモデルなどの代替表現) との関係を表したりするために⽤いる。

6.5.2 フォーム §

フォームにより、Consumer (または広義のウェブクライアント) は、URIの解決を超える操作を実⾏できる (例えば、Thingの状態を操作するなど)。Consumerは、フォームに記⼊して、その送信ターゲットに送信することでこれを⾏う。これには通常、リンクが提供できるよりも詳細な (リクエスト) メッセージの内容に関する情報が必要である (例えば、メソッ

ド、ヘッダフィールド、またはその他のプロトコルのオプション)。フォームはリクエストテンプレートと考えることができ、提供者は、独⾃のインターフェースと状態に従って情報の⼀部を事前に⼊⼒し、Consumer (または、⼀般的にはウェブクライアント) が⼊⼒する部分を空⽩のままにする。

W3C WoTは、フォームを新しいハイパーメディア制御と定義している。CoRALの定義は実質的に同じであり、したがって互換性があることに注意すること [CoRAL] 。フォームは次のもので構成される。

フォームのコンテキスト操作型

送信ターゲット

リクエストメソッド

オプションでフォームフィールド

フォームは、「あるフォームのコンテキストに関する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発信せよ」という表明と考えることができ、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さらに記述できる。

フォームのコンテキストと送信ターゲットは、両⽅とも国際化資源識別⼦

(IRI;Internationalized Resource Identifier) [RFC3987] でなければならない (MUST)。しかし、⼀般的なケースでは、多くのプロトコル (HTTPなど) がIRIをサポートしていないため、それらはURI [RFC3986] になることもある。

フォームのコンテキストと送信ターゲットは、同じ資源または異なる資源を指し⽰すことができ (MAY)、その場合、送信ターゲットの資源はコンテキストの操作を実装する。

操作型は、操作のセマンティクスを⽰す。操作型は、リンク関係型と同様の形で⽰される。

よく知られた (well-known) 操作型は、ABNF

LOALPHA \* ( LOALPHA / DIGIT / "." / "-" ) に従わなければならない (MUST)。よく知られた操作型は、⼤⽂字と⼩⽂字を区別しない⽐較により⽐較されなければならない (MUST)。この仕様で定義しているWeb of Thingsのよく知られた操作型を表1に⽰す。

事前に定義されている操作型は、アプリケーションが選択した拡張操作型によって拡張できる (MAY)。拡張操作型は、型を⼀意に識別するURI [RFC3986] でなければならない (MUST)。拡張操作型は、⼤⽂字と⼩⽂字を区別しない⽐較により、⽂字列として⽐較されなければならない (MUST)。それにも関わらず、拡張操作型には、すべて⼩⽂字のURIを⽤いるべきである (SHOULD)。リクエストメソッドでは、送信ターゲットのURIスキームで識別されるプロトコルでの標準的なメソッド集合のうちの⼀つを特定しなければならない (MUST)。

フォームフィールドはオプションであり、特定の操作に期待されるリクエストメッセージをさらに指定できる (MAY)。これはペイロードに限定されず、プロトコルのヘッダにも影響する可能性があることに注意すること。フォームフィールドは、URIスキームで指定されている送信ターゲットに⽤いられるプロトコルに依存することもある (MAY)。例えば、HTTPヘッダフィールド、CoAPオプション、リクエストペイロードのパラメータ (つまり、完全なコンテントタイプ) などのプロトコルに依存しないメディアタイプ [RFC2046] 、または期待される応答に関する情報などである。

表1 Web of Thingsのよく知られた操作型

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 操作型 説明   |  |  | | --- | --- | | readproperty | 対応するデータを検索するための、Propertyのアフォーダンスに関する読み取り操作 | | writeproperty | 対応するデータを更新するための、Propertyのアフォーダンスに関する書き込み操作 | | observeproperty | Propertyが更新されたときに新しいデータにより通知を受けるための、Propertyのアフォーダンスに関する監視操作 | | unobserveproperty | 対応する通知を停⽌するための、Propertyのアフォーダンスに関する監視解除操作 | | invokeaction | 対応するActionを実⾏するための、Actionのアフォーダンスに関する呼び出し操作 | | subscribeevent | Eventが発⽣したときにThingによって通知を受けるための、Eventのアフォーダンスに関する登録操作 | | unsubscribeevent | 対応する通知を停⽌するために、Eventのアフォーダンスに関する登録解除操作 | | readallproperties | すべてのPropertyのデータを1回の相互作⽤で検索するための、Thingに関するreadallproperties (すべてのPropertyの読み込み) 操作 | |
| 操作型 説明   |  |  | | --- | --- | | writeallproperties | すべての書き込み可能なPropertyのデータを1回の  相互作⽤で更新するための、Thingに関する  writeallproperties (すべてのPropertyの書き込み) 操作 | | readmultipleproperties | 選択したプロパティーのデータを1回の相互作⽤で検索するための、Thingに関する  readmultipleproperties (複数のPropertyの読み込  み) 操作 |   writemultipleproperties 選択した書き込み可能なPropertyのデータを1回の相互作⽤で更新するための、Thingに関する  writemultipleproperties (複数のPropertyの書き込み) 操作 |

### 編集者のメモ

本仕様の執筆時点では、(上記の) よく知られた操作型は、WoT相互作⽤モデルに由来する⼀定の集合である。他の仕様では、それぞれの⽂書形式やフォームのシリアライゼーションにとって有効な、よく知られた操作型を追加定義してもよい。本仕様の今後のバージョンや別の仕様では、将来、WoT関連仕様の範囲を越えて適⽤される可能性のある拡張やより汎⽤的なWebフォームモデルを可能にするためにIANAレジストリを設定してもよい。

## 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、相互作⽤のアフォーダンスから、HTTP [RFC7231] 、CoAP

[RFC7252] 、MQTT [MQTT] などの特定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。これは、ネットワークに接続するインタフェースを介して相互作⽤のアフォーダンスをどのように作動させるかをConsumerに⽰す。プロトコルバインディングは、相互運⽤性をサポートするためにREST [REST] の統⼀インターフェース (Uniform Interface) 制約に従う。したがって、すべての通信プロトコルがW3C WoTのプロトコルバインディングの実装のために適格なわけではない。なお、要件は以下の⾔明で⽰す通り。

§ 6.2 アフォーダンスで⽰したドアの例では、プロトコルバインディングは、ノブかレバーかのレベルでドアの取っ⼿に対応しており、それは、ドアがどのように開くかを⽰している。

6.6.1 ハイパーメディア駆動 § 相互作⽤のアフォーダンスには、⼀つ以上のプロトコルバインディングが含まれていなければならない (MUST)。相互作⽤のアフォーダンスを作動させる⽅法を⾃⼰記述的にするために、プロトコルバインディングをハイパーメディア制御 (§ 6.5 ハイパーメディア制御を参照) としてシリアル化しなければならない (MUST)。ハイパーメディア制御は、対応する相互作⽤のアフォーダンスを提供しているThingを管理する発信元から発信されなければならない (MUST)。その発信元は、Thing⾃体である場合もあり、実⾏時に (現在の状態に基づいて、IPアドレスなどのネットワークパラメータを含めて) TDドキュメントを⽣成するか、最新のネットワークパラメータのみが挿⼊されたメモリ上の情報からTDドキュメントを提供する。その発信元は、ネットワークパラメータや内部構造 (例えば、ソフトウェアスタック) を含む、Thingの完全で最新の知識を持つ外部エンティティーにもなりえる。これにより、

ThingとConsumerの間の疎結合が可能になり、独⽴したライフサイクルと進化が可能になる。ハイパーメディア制御は、Thingの外部でキャッシュされ、鮮度を判断するためにメタデータのキャッシュが利⽤できる場合にはオフライン処理に使⽤される (MAY)。

|  |
| --- |
| 翻訳者のメモ  上記で⾔う「発信元」に関して、§ 6.5 ハイパーメディア制御で以下の通り触れられていることから、原⽂における The hypermedia controls MUST originate from the  authority managing the Thing that is providing the corresponding Interaction  Affordance. 中の authority は「ハイパーメディア制御の発信元」を意味しており、通常、ウェブサーバーである。  ハイパーメディア制御は通常、ウェブサーバーから発信され、ウェブクライアントがサーバーと相互作⽤を⾏っている間にインバンド (in-band) で発⾒される。このようにして、ウェブサーバーは、現在の状態や承認などの他の要因を考慮に⼊れることにより、ウェブアプリケーションを通じてクライアントを動的に駆動できる。 |

6.6.2 URI §

W3C WoTの条件を満たしているプロトコルは、IANAに登録されている関連するURIスキーム [RFC3986] を持たなければならない (MUST) ([IANA-URI-SCHEMES] を参照)。ハイパーメディア制御は、URI [RFC3986] に依拠してリンクと送信ターゲットを識別する。これにより、URIスキーム (「:」までの最初の構成要素) は、Thingとの相互作⽤のアフォーダンスに

⽤い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 メソッドの標準的な集合 §

W3C WoTの条件を満たしているプロトコルは、先験的に知られている標準的なメソッドの集合に基づいていなければならない (MUST)。標準的なメソッドの集合は、例えばプロキシにより相互作⽤のアフォーダンスの中間処理を可能にしたり、プロトコルバインディング

[REST] 間での変換を⾏うために、メッセージを⾃⼰記述的なものにする。さらに、これにより、Consumerは、当該Consumer向けのThing固有のコードやプラグインを回避しながら、HTTP、CoAP、MQTTなどの⼀般的な転送プロトコルの、再利⽤可能なプロトコルスタックを持てるようになる。

6.6.4 メディアタイプ §

相互作⽤のアフォーダンスを作動させたとき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のメディアタイプ [RFC2046] により識別されなければならない (MUST)。メディアタイプは、例えば、JSONのapplication/json [RFC8259] また

はCBORのapplication/cbor [RFC7049] などの、表現形式を識別するためのラベルである。メディアタイプはIANAによって管理されている。

⼀部のメディアタイプでは、⽤いる表現形式を完全に指定するために追加のパラメータが必要になる場合がある。例は、text/plain; charset=utf-8やapplication/ld+json;

profile="http://www.w3.org/ns/json-ld#compacted"である。これは、Thingに送信さ

れるデータを記述する際に特に考慮する必要がある。HTTPのcontent coding [RFC7231] などのデータに関する標準的な変換も存在するかもしれない。プロトコルバインディングは、メディアタイプのみよりも詳細に表現形式を指定する追加情報を持つことができる (MAY)。

多くのメディアタイプは、その要素に追加的なセマンティクスを提供しない汎⽤的なシリアライゼーション形式のみを識別することに注意すること (例えば、XML、JSON、CBOR)。したがって、対応する相互作⽤のアフォーダンスは、交換されるデータにより詳細な構⽂メタデータを提供するdata schemaを宣⾔すべきである (SHOULD)。

## 6.7 WoTシステム構成要素とその相互接続性 §

§ 6.1 概要の節では、Thing、Consumer、Intermediaryなどの抽象的なWoTアーキテクチャの構成要素の観点からWoTアーキテクチャについて説明した。これらの抽象的なWoTアーキテクチャの構成要素がソフトウェアスタックとして実装され、WoTアーキテクチャで特定の役割を担う場合、そのようなソフトウェアスタックをServientと呼ぶ。WoTアーキテクチャに基づくシステムにはServientが含まれ、システムの⽬標を達成するために相互に通信を

⾏う。

この節では、システム構成図を⽤いて、複数のServientが連携してWoTアーキテクチャに基づくシステムを構築する⽅法を説明する。

Thingは、Servientによって実装できる。Thingの中には、Servientのソフトウェアスタック内に公開されたThingと呼ばれるThingの表現が含まれており、そのWoTインターフェースをThingのConsumerが利⽤できるようにする。この公開されたThingは、モノの動作を実装するために、Servient上のその他のソフトウェア構成要素 (例えば、アプリケーション) により使⽤することができる。

Thing

**Servient**

公

開

さ

れ

た

Thing

Consumer

図20 ThingとしてのServient

⼀⽅、Consumerは常にServientによって実装される。これは、(Consumerは) Thing

Description (TD) 形式を処理できる必要があるとともに、TDに含まれるプロトコルバインディング情報を通じて設定できるプロトコルスタックを持つ必要があるためである。

Consumerの中では、Servientのソフトウェアスタックが、利⽤されるThingと呼ばれる

Thingの表現を提供し、Thingと相互作⽤するためにTDを処理する必要がある、Servient上で実⾏されるアプリケーションから利⽤できるようにする。

Thing

**Servient**

利

⽤

さ

れる

Thing

Consumer

図21 ConsumerとしてのServient

Servientのソフトウェアスタック内の利⽤されるThingのインスタンスは、アプリケーションからプロトコルレベルの複雑性を分離するのに役⽴つ。これは、アプリケーションに代わって、公開されるThingと通信を⾏う。

同様に、Intermediaryはさらに、Servientによって実装される別のWoTアーキテクチャの構成要素である。Intermediaryは、ThingとそのConsumerの間に位置し、(Thingに対する) Consumerと (Consumerに対する) Thingの両⽅の役割を果たす。Intermediaryの中で

は、Servientのソフトウェアスタックに、Consumer (利⽤されるThing) とThing (公開されるThing) の両⽅の表現が含まれている。

Intermediary

**Servient**

公

開

さ

れ

た

Thing

利

⽤

さ

れる

Thing

ConsumerThing 図22 IntermediaryとしてのServient

6.7.1 直接通信 §

図23は、Thing Descriptionを介して相互作⽤のアフォーダンスを公開しているThingと、相互作⽤のアフォーダンスによりを⽤いるConsumerとの間の直接通信を⽰している。両⽅のServientが同じネットワークプロトコルを⽤い、相互にアクセスできる場合に、直接通信が適⽤される。

**Servient**

**Servient**

公

開

さ

れ

た

Thing

利

⽤

さ

れる

Thing

Thing

Consumer

**WoT Thing**

**Description**

図23 ConsumerとThingの⾼レベルなアーキテクチャ

公開されたThingは、Thingの抽象化のソフトウェア表現であり、Thingが提供する相互作

⽤のアフォーダンスのWoTインターフェースを提供する。

利⽤されるThingは、Consumerによって利⽤されているリモートのThingのソフトウェア表現であり、アプリケーションに対するリモートのモノへのインターフェースとして機能する。Consumerは、TDドキュメントを解析して処理することにより、利⽤されるThingのインスタンスを⽣成できる。ConsumerとThingとの相互作⽤は、利⽤されるThingと公開されたThingがそれらの間の直接ネットワーク接続を介してメッセージを交換することによって実⾏される。

6.7.2 間接通信 §

図24では、ConsumerとThingがIntermediaryを介して互いに接続されている。

Intermediaryは、Servient同⼠が異なるプロトコルを⽤いている場合や、認証が必要でアクセス制御 (例えば、ファイアウォール) を提供している異なるネットワーク上にある場合に必要である。

Intermediary

**Servient**

公

開

さ

れ

た

Thing

利

⽤

さ

れる

Thing

利

⽤

さ

れる

Thing

**Servient**

公

開

さ

れ

た

Thing

Thing

Consumer

**WoT Thing**

**Description**

**WoT Thing**

**Description**

**Servient**

図24 Intermediaryを⽤いた⾼レベルなアーキテクチャ

Intermediaryは、公開されたThingと利⽤されるThingの機能を組み合わせる。

Intermediaryの機能には、ConsumerとThingとの間で相互作⽤のアフォーダンスのメッセージを中継すること、オプションで、応答を⾼速化するためにThingのデータをキャッシュすること、IntermediaryによってThingの機能が拡張されたときに通信を変換することが含まれる。Intermediaryの中では、利⽤されるThingが、Thingの中の公開されたThingのプロキシオブジェクトを作成しており、Consumerは、⾃⾝の利⽤されるThingを通じて、そのプロキシオブジェクト (つまり、Intermediaryの中の、公開されたThing) にアクセスできる。

ConsumerとIntermediaryは、IntermediaryとThingの間で⽤いられるプロトコルとは異なるプロトコルで通信できる。例えば、Intermediaryは、CoAPを⽤いるThingと、HTTP を⽤いるConsumerのアプリケーションとの間の橋渡しを提供できる。

IntermediaryとThingの間で、複数の異なるプロトコルが⽤いられている場合でも、

Consumerは、Intermediaryを通じ、ある⼀つのプロトコルを⽤いてそれらのThingと間接的に通信できる。認証についても同じことが⾔える。Consumerの、利⽤されるThingは、ある⼀つのセキュリティメカニズムを⽤いて、Intermediaryの、公開されたThingと認証を⾏う必要があるが、Intermediaryが、複数の異なるThingによる認証を⾏うためには、複数のセキュリティメカニズムが必要となる場合がある。

通常、Intermediaryは、発信元であるThingの、Thing Descriptionに基づいて、そのプロ

キシオブジェクト⽤のThing Descriptionを⽣成する。各ユースケースごとの要件に応じて、プロキシオブジェクト⽤のTDは、元のThingのTDと同じ識別⼦を⽤いるか、新たな識別⼦が割り当てられる。必要に応じて、Intermediaryによって⽣成されたTDには、他の通信プロトコルのインターフェースを含むことができる (MAY)。

# 7. WoT構成要素 §

この章は規定である。

Web of Things (WoT) の構成要素により、WoTの抽象アーキテクチャに準拠したシステムを実装できる。この構成要素の詳細は、別の仕様で定義しており、この章では、概要と要約を⽰す。

WoT構成要素は、§ 6.3 Web Thingで論じ、図19で⽰しているThingの各アーキテクチャの側⾯をサポートする。個々の構成要素を図25の抽象的なThingのコンテキストで⽰してい

る。これは抽象的な図であり、特定の実装を表しているものではない。代わりに、構成要素とThingの主要なアーキテクチャの側⾯との関係を⽰している。この図では、WoT構成要素を黒の輪郭線で強調表⽰している。分野横断的な関⼼事であるセキュリティは、公開されている構成要素と保護されている⾮公開の構成要素とに分けている。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参考情報である。

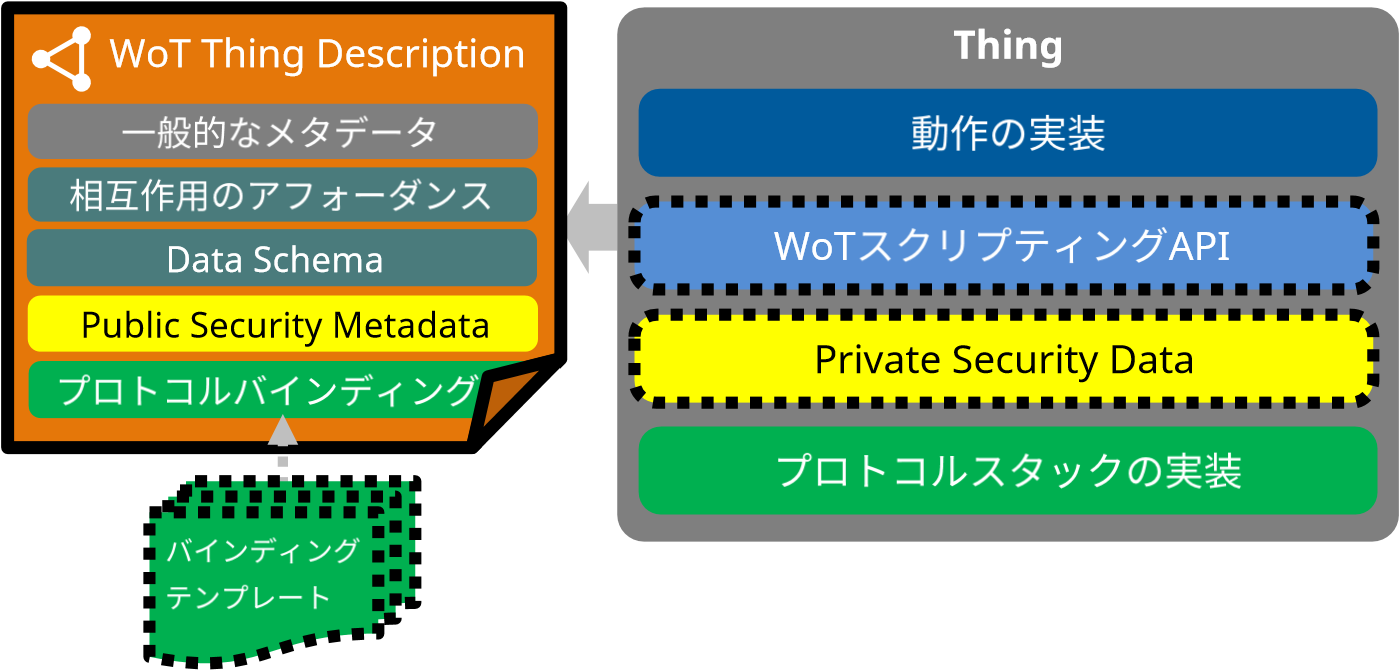


図25 Thingのアーキテクチャの側⾯に対するWoT構成要素の関係

以下の節では、WoT Thing Description、WoTバインディングテンプレート、WoTスクリプティングAPIという個々のWoT構成要素に関する追加情報を提供する。セキュリティは、分野横断的な関⼼事であるが、4番⽬の構成要素と考えることができる。

## 7.1 WoT Thing Description §

WoT Thing Description (TD) 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に基づくシリアル化表現を定義している。TDは、⼈間が読めて機械が理解できる⽅法でThingに豊かなメタデータを提供する。⽣のJSONとしての処理

に加えて、実装でJSON-LD [JSON-LD11] とグラフデータベースを使⽤してメタデータの強

⼒なセマンティック処理を⾏えるように、TDの情報モデルと表現形式は、どちらもリンクトデータ [LINKED-DATA] に準じている。

Thing Description (TD) は、名前、ID、内容記述などの⼀般的なメタデータでThingのインスタンスを記述し、関係するThingやその他の⽂書へのリンクを介して関連メタデータを提供することもできる。TDには、§ 6.4 相互作⽤モデルで定義している相互作⽤モデルに基づく相互作⽤のアフォーダンスのメタデータ、Public Security Metadataと、プロトコルバインディングを定義している通信メタデータも含まれている。TDは、提供されているサービスと関連資源 (どちらもハイパーメディアコントロールを使⽤して記述される) について知るためのエントリポイントを提供するため、Thingのindex.htmlと考えることができる。

理想的には、TDは、Thing⾃体によって作成および (または) 提供され、ディスカバリ時に取得される。しかし、Thingに資源の制限がある場合 (例えば、メモリ空間や電⼒が限られている場合) や、Web of Thingsの⼀部となるように既存のデバイスが改造されている場合には、外部で提供することもできる。ディスカバリを改善し (例えば、制約のあるデバイスに対する) 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。Consumerは、TDのキャッシングメカニズムを、Thingが更新されて新しいバージョンのTDを取得する必要がある場合に通知してくれる通知メカニズムと組み合わせて⽤いることが推奨される。セマンティックな相互運⽤性のために、TDは領域固有の語彙を利⽤でき、その語彙には明⽰的な拡張ポイントが提供される。しかし、特定の領域固有の語彙の開発は現在、W3C WoT 標準化活動の範囲外である。

有⽤でありえる外部IoT語彙の三つの例は、SAREF [SAREF] 、Schema Extensions for IoT

[IOT-SCHEMA-ORG] 、およびW3C Semantic Sensor Network Ontology [VOCAB-SSN]

である。TDにおけるこのような外部語彙の使⽤はオプションである。将来的に、追加の領域固有の語彙が開発され、TDで⽤いられるかもしれない。

### 翻訳者のメモ

英語原本において、「W3C Semantic Sensor Network ontology」となっている箇所は、上記の通り「W3C Semantic Sensor Network Ontology」が正しいと思われる。

全体として、WoT Thing Descriptionの構成要素は、⼆つの⽅法で相互運⽤性を促進する。まず、TDは、Web of Thingsにおける機械間通信を可能にする。次に、TDは、IoTデバイスにアクセスしてそのデータを利⽤できるアプリケーションを作成するために必要なすべての詳細情報を開発者が⽂書化し、取得するための共通の統⼀形式として機能する。

## 7.2 WoTバインディングテンプレート §

この節は参考情報である。

すべてのコンテキストに適した単⼀のプロトコルはないため、IoTはデバイスへのアクセスに様々なプロトコルを⽤いる。したがって、Web of Thingsの中⼼的な課題は、多くの様々なIoTプラットフォーム (例えば、OCF、oneM2M、OMA LWM2M、OPC UA) と、特定の標準には準拠していないけれども適切なネットワークプロトコルを介して適格なインターフェースを提供するデバイスとの相互作⽤を可能にすることである。WoTは、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おり、これは様々な制約を満たさなければならない (§ 6.6 プロトコルバインディングを参照)。

⾮規定的なWoTバインディングテンプレート仕様 [WOT-BINDING-TEMPLATES] は、様々なIoTプラットフォームと相互作⽤を⾏う⽅法に関するガイダンスを提供する通信メタデータの⻘写真のコレクションを提供する。特定のIoTデバイスやサービスを記述する場合に、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⽤いて、そのプラットフォームをサポートするためにThing Descriptionで提供しなければならない通信メタデータを検索できる。

**WoT Thing Description**

“

I

oT

プラットフォーム

”

×

“

メディアタイプ

”

×

“

転

送

プロトコル

”

×

“

サブプロトコル

”

×

“

セキュリティ

”

OCF

と

CBOR

で

CoAP

上

で

DTLS

を

使

⽤

oneM2M

と

JSON

で

MQTT

上

で

TLS

を

使

⽤

LWM2M

と

SenML

で

CoAP

上

で

DTLS

を

使

⽤

ウェブ

と

CBOR

で

CoAP

上

で

COSE+CWT

を

使

⽤

ウェブ

と

JSON

で

HTTP

上

で

OAuth

を

使

⽤

…

バインディング

テンプレート

プロトコル

バインディング

実

装

プロトコル

スタック

MQTT

HTTP

CoAP

CBOR

JSON

OAuth

TLS

DTLS

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の適⽤⽅法を⽰す。WoTバインディングテンプレートは、IoTプラットフォームごとに⼀度だけ作成され、そのプラットフォームのデバイスのすべてのTDで再利⽤できる。TDを処理しているConsumerは、対応するプロトコルスタックを組み込み、TDで提供される情報に従ってスタック (またはそのメッセージ) を設定することにより、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は、次の五つの次元にまたがっている。

IoTプラットフォーム:

IoTプラットフォームは、プラットフォーム固有のHTTPヘッダフィールドやCoAPオプ

ションなど、アプリケーション層で独⾃の変更を導⼊することがよくある。フォーム

(§ 6.5.2 フォームを参照) に必要な情報を含め、使⽤されるアプリケーション層プロトコル⽤に定義した追加のフォームフィールドにこれらの微調整を適⽤することができる。

メディアタイプ:

IoTプラットフォームは、多くの場合、データの交換に⽤いられる表現形式 (別名、シリアライゼーション) が多様である。メディアタイプ [RFC2046] は、これらのフォーマットを識別するが、メディアタイプのパラメータでそれ以上の指定が可能である。フォームには、HTTPの既知のコンテントタイプフィールドなどの追加のフォームフィールドにメディアタイプとオプションのパラメータを含めることができ、これによって、メディアタイプとその潜在的なパラメータが組み合わされる (例えば、text/plain; charset=utf-8)。

転送プロトコル:

Web of Thingsでは、アプリケーション固有のオプションやサブプロトコルのメカニズムのない、基礎となる標準的なアプリケーション層プロトコルに転送プロトコルという

⽤語を⽤いる。フォーム送信ターゲットのURIスキームには、例えば、HTTP、 CoAPS、WebSocket (それぞれhttp:、coaps:、ws:を介する) などの転送プロトコルを識別するために必要な情報が含まれている。

サブプロトコル:

転送プロトコルには、うまく相互作⽤を⾏うために知っていなければならない拡張メカニズムがある。このようなサブプロトコルは、URIスキームだけでは識別できず、明⽰的に宣⾔しなければならない。例は、ロングポーリング [RFC6202] やサーバー送信イベント [HTML] などのHTTPのプッシュ通知回避策である。フォームでは、サブプロトコルを識別するために必要な情報を、追加のフォームフィールドに含めることができる。

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でき、多くの場合、互いに補完するために⼀緒に使⽤できる。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な性質により、正しいメカニズムを適⽤するために必要な情報は、Thingの⼀般的なメタデータ内で提供するか、相互作⽤のアフォーダンスやフォームごとに特殊化することができる。

## 7.3 WoTスクリプトAPI §

この節は参考情報である。

WoTスクリプティングAPIは、ウェブブラウザAPIに似たECMAScriptベースのAPI

[ECMAScript] を提供することにより、IoTアプリケーション開発を容易にするW3C WoTのオプションの「便利な」構成要素である。スクリプティングランタイムシステムをWoTランタイムに統合することにより、WoTスクリプティングAPIは、Thing、Consumer、

Intermediaryの動作を定義するポータブルなアプリケーションのスクリプトの使⽤を可能にする。

従来、IoTデバイスのロジックはファームウェアに実装されており、その結果、⽐較的複雑な更新プロセスを含む、組み込み開発と同様の⽣産性の制約が⽣じる。対照的に、WoTスクリプティングAPIは、ウェブブラウザとあまり違いのないIoTアプリケーション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り、デバイスのロジックの実装を可能にし、⽣産性の向上と統合コストの削減を⽬指している。さらに、標準的なAPIにより、アプリケーションモジュールの移植が可能になる。例えば、計算集約型の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たり、タイムクリティカルなロジックをクラウドからゲートウェイやエッジノードに移動させたりできる。

### 翻訳者のメモ

英語原本中で「compute-intense」となっているのは、「compute-intensive」の間違いと思われる。

⾮規定的なWoTスクリプティングAPI仕様 [WOT-SCRIPTING-API] は、スクリプトがWoT Thing Descriptionを発⾒、取得、利⽤、作成、公開できるよう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とその相互作⽤のアフォーダンス (Property、Action、Event) に対するインターフェースとして機能するローカルオブジェクトをインスタンス化する。これにより、スクリプトがThingを公開すること、つまり相互作⽤のアフォーダンスを定義および実装し、Thing Descriptionを公開することもできるようになる。

## 7.4 WoTセキュリティとプライバシーに関するガイドライン §

この節は参考情報である。

セキュリティは分野横断的な関⼼事であり、システム設計のすべての側⾯において考慮すべきである。WoTアーキテクチャでは、TDのPublic Security Metadataのサポートなどの特定の明⽰的な機能、およびWoTスクリプティングAPIの設計における懸念の分離によって、セキュリティがサポートされる。各構成要素の仕様には、その構成要素の特定のセキュリティとプライバシーに関する留意点の議論も含まれている。別の参考情報仕様であるWoTセキュリティとプライバシーに関するガイドライン [WOT-SECURITY] は、分野横断的なセキュリティとプライバシーに関するガイダンスを追加提供する。

# 8. 抽象的なServientのアーキテクチャ §

この章は参考情報である。

§ 6.7 WoTシステム構成要素とその相互接続性で定義しているように、Servientは、前の項で⽰したWoT構成要素を実装するソフトウェアスタックである。Servientは、Thingを提供および公開したり、および (または) Thingを利⽤することができる (つまり、Consumerを

提供することができる)。プロトコルバインディングに応じて、サーバーとクライアントの両

⽅の役割を実⾏できることから、Servientという混成語が命名された。

前章では、WoT構成要素が概念的にどのように相互に関連し、それらが抽象WoTアーキテク

チャにどのように対応するかについて説明している (§ 6. WoTアーキテクチャを参照)。これらの概念を実装する場合、特定の技術的側⾯を考慮した、より詳細な観点が必要となる。

この賞では、Servientの実装に関するアーキテクチャの詳細について説明する。

図27は、(オプションの) WoTスクリプティングAPIという構成要素を⽤いたServientの実装を⽰している。ここでは、WoTランタイムは、WoT固有の側⾯の管理に加えて、アプリケーションスクリプトの解釈と実⾏も⾏う、スクリプティングランタイムシステムである。WoT スクリプティングAPIをサポートするServientは、通常、強⼒なデバイスやエッジノード、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ケーション向けAPIをJavaScript/ECMAScriptに制限しない。他のランタイムシステムを⽤いて

Servientを実装することもできる。

§ 8.8.1 ネイティブなWoT APIの項では、WoTスクリプティングAPIという構成要素を⽤いない、代替のServient実装を⽰している。WoTランタイムは、アプリケーション向けAPIに任意のプログラミング⾔語を使⽤できる。通常、それは、Servientのソフトウェアスタックのネイティブ⾔語である。例えば、組み込みServientではC/C++、クラウドベースの

ServientではJavaが挙げられる。それは、アプリケーションスクリプトの利点を、低い資源利⽤と組み合わせるLuaなどの代替スクリプト⾔語であってもよい。

**Servient**

スクリプティング

WoT

ランタイム

公

開

さ

れ

た

Thing

利

⽤

さ

れる

Thing

利

⽤

さ

れる

Thing

動

作

の

実

装

アプリケーション

の

スクリプト

プロトコルスタック

の

実

装

WoT

スクリプティング

API

Private Security

Data

公

開

利

⽤

WoT Thing

Description

WoT Thing

Description

システム

WoT

Thing Description

Public Security Metadata

相

互

作

⽤

の

アフォーダンス

⼀

般

的

な

メタデータ

Data Schema

プロトコルバインディング

バインディング

テンプレート

図27 WoTスクリプティングAPIを⽤いたServientの実装

図27で⽰した各モジュールの役割と機能については、以下の節で説明する。

## 8.1 動作の実装 §

動作は、Thingの全体的なアプリケーションロジックを定義し、いくつかの側⾯を持っている。

動作には、Thingの⾃律的な動作 (例えば、センサーのサンプリングまたはアクチュエータの制御ループ)、相互作⽤のアフォーダンスのハンドラー (つまり、アフォーダンスが作動したときに実⾏される具体的なアクション)、Consumerの動作 (例えば、Thingの制御またはマッシュアップの実現)、およびIntermediaryの動作 (例えば、単にThingを代理する、また

は仮想エンティティーを編成する) が含まれる。Servient内の動作の実装は、どのような

Thing、Consumer、Intermediaryがこの構成要素で提供されるかを定義する。

図27は、オプションのWoTスクリプティングAPIという構成要素を実装しているServientを⽰しており、JavaScript [ECMAScript] で記述されている移植可能なアプリケーションスクリプトが動作を定義する。アプリケーションスクリプトは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またはその他のスクリプトベースのAPIを提供する場合)。アプリケーションスクリプトは、共通の WoTスクリプティングAPIの定義に対して記述されているため、移植可能であり、そのため、WoTスクリプティングAPIという構成要素を備えた任意のServientによって実⾏できる。これにより、例えば、ネットワーク要件を満たすためにConsumerをクラウドからエッ

ジノードに移動させたり、増⼤する資源の需要を満たすためにIntermediaryをクラウドに移動させたりするなど、システム構成要素間でアプリケーションロジックを移動することができる。移植可能なアプリケーションにより、Servientのデプロイ後に追加の動作を「インストール」できる。

原則として、相互作⽤のアフォーダンスがWoTインターフェースを介して外部に⽰されている限り、Thingの動作を定義するために任意のプログラミング⾔語とAPIを使⽤できる。アプリケーション向けAPIとプロトコルスタックの間の適応は、WoTランタイムにより処理される。WoTスクリプティングAPIという構成要素を⽤いない動作の実装については、§ 8.8.1 ネイティブなWoT APIを参照のこと。

## 8.2 WoTランタイム §

技術的には、Thingの抽象化とその相互作⽤モデルはランタイムシステムに実装される。このWoTランタイムは、動作の実装に関する実⾏環境を整備し、Thingを公開および (または) 利⽤できるため、WoT Thing Descriptionを取得し、処理し、シリアライズし、提供することができなければならない。

すべてのWoTランタイムには、動作実装⽤のアプリケーション向けインターフェース (つまり、API) がある。図27に⽰される、オプション機能であるWoTスクリプティングAPIという構成要素は、Thingの抽象化に従ったアプリケーション向けインターフェースを定義し、実⾏中にアプリケーションのスクリプトを介して動作の実装がデプロイできるようにする。これも、コンパイル中にのみ使⽤できます。コンパイル時にのみ使⽤できる、WoTスクリプティングAPI以外のAPIに関しては、§ 8.8.1 ネイティブなWoT APIを参照のこと。⼀般的に、アプリケーションのロジックは、WoTランタイムの管理⾯、特にPrivate Security Dataに対する不正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。マルチテナント⽅式のServientでは、異なる利⽤者に対して実⾏環境の分離を追加する必要がある。

WoTランタイムは、Thingのライフサイクル、より正確にはそのソフトウェアの抽象化と記述を管理するための特定の操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操作をServient内にカプセル化し、内部インターフェースを⽤いてライフサイクル管理を実現してもよい。このような操作の詳細は、実装によって異なる。WoTスクリプティングAPIにはLCM機能が含まれているため、このようなシステムの⼀つの可能な実装となる。

WoTランタイムは、動作の実装をプロトコルバインディングの詳細から切り離すため、

Servientのプロトコルスタック実装とインタフェースしなければならない。また、WoTランタイムは通常、例えば、接続されているセンサーやアクチュエーターなどのローカルなハードウェアにアクセスしたり、ストレージなどのシステムサービスにアクセスしたりするために、根底にあるシステムともインタフェースする。これらのインターフェースは両⽅とも実装に固有のものだが、WoTランタイムは実装されたThingの抽象化に必要な適応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の構成要素は、WoT Thingの記述仕様 [WOT-THING-

DESCRIPTION] に厳密に従ったECMAScript APIを定義する。これは、動作の実装とスク

リプトを使ったWoTランタイムとの間のインターフェースを定義する。さらに、例えば、ウェブブラウザAPI向けのjQueryと同様に、スクリプティングAPIにもとづいて、よりシンプルなAPIを実装してもよい。

詳細に関しては、 [WOT-SCRIPTING-API] を参照すること。

## 8.4 公開されたThingと利⽤されるThingの抽象化 §

WoTランタイムは、TDに基づいてThingのソフトウェア表現をインスタンス化する。このソフトウェア表現は、動作の実装のためのインターフェースを提供する。

公開されたThingの抽象化は、ローカルで提供され、Servientのプロトコルスタックの実装を介して外部からアクセス可能なThingを表す。動作の実装は、メタデータと相互作⽤のアフォーダンスの定義と、⾃律的な動作の提供により、公開されたThingを完全に制御できる。

利⽤されるThingの抽象化は、通信プロトコルを⽤いてアクセスする必要のある、リモートで提供されるConsumer⽤のThingを表す。利⽤されるThingは、プロキシオブジェクトかスタブである。動作の実装は、メタデータの読み取りと、対応するTDに記述されているとおりの相互作⽤のアフォーダンスの作動に限定される。利⽤されるThingは、独⾃または旧式の通信プロトコルの背後にあるローカルなハードウェアやデバイスなどのシステムの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と利⽤されるThingとの間に必要な適合処理を⾏わなければならない。さらに、対応するTDを提供し、例えば、アプリ

ケーション向けAPIを介してWoTランタイムにより提供される (WoTスクリプティングAPI [WOT-SCRIPTING-API] で定義されているdiscover () メソッドなどの) 何らかの発⾒メ

カニズムを拡張することにより、そのTDを動作の実装において利⽤できるようにしなければならない。

WoTスクリプティグAPIを⽤いる場合、公開されたThingと利⽤されるThingは、JavaScript のオブジェクトであり、アプリケーションのスクリプトによって作成、操作、破棄できる。しかし、アクセスは、例えば、マルチテナント⽅式のServientなど、セキュリティのメカニズムにより制限される場合がある。

## 8.5 Private Security Data §

Thingと相互作⽤するための秘密鍵などのPrivate Security DataもWoTランタイムによって概念的に管理されるが、意図的にアプリケーションが直接アクセスできないようにされている。実際に、最も安全なハードウェアの実装では、このようなPrivate Security Dataは個別の分離されたメモリ (例えば、安全な処理要素またはTPM) に格納され、操作の抽象的な集合のみ (分離されたプロセッサとソフトウェアスタックによって実装される可能性さえある) が提供され、攻撃対象領域を制限してこのデータの外部漏洩を防⽌する。Private Security

Dataは、承認を⾏い、相互作⽤の完全性と機密性を保護するために、プロトコルバインディングによって透過的に⽤いられる。

### 翻訳者のメモ

⽂頭の「Private security data」も、「3. ⽤語」で定義された「Private Security

Data」として扱うべきと考えられる。

## 8.6 プロトコルスタックの実装 §

Servientのプロトコルスタックには公開されたThingのWoTインターフェースが実装され、それはConsumerが (利⽤されるThingを介して) リモートのThingのWoTインターフェースにアクセスするために⽤いられる。これにより、ネットワークを介して相互作⽤するための具体的なプロトコルのメッセージが作成される。Servientには複数のプロトコルを実装できるため、複数のプロトコルバインディングをサポートすれば、様々なIoTプラットフォームとの相互作⽤が可能となる。

多くの場合、標準プロトコルを⽤いている場合には、汎⽤プロトコルスタックを⽤いて、プラットフォーム固有のメッセージ (例えば、HTTP (S) 固有のもの、CoAP (S) 固有のもの、

MQTTソリューションのものなど) を作成できる。このケースでは、Thing Descriptionの通信メタデータを利⽤して、適切なスタック (例えば、正当なヘッダフィールドを備えたHTTP や、正当なオプションを備えたCoAP) を選択し設定する。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形式 (JSON、CBOR、XMLなど) のパーサーとシリアライザーも、これらの汎⽤プロトコルスタックで共有できる。

詳細に関しては、 [WOT-BINDING-TEMPLATES] を参照のこと。

## 8.7 システムAPI §

WoTランタイムの実装は、Thingの抽象化により、通信プロトコルを介してアクセスできているかのように、ローカルなハードウェアやシステムサービスを動作の実装に提供できる。このケースでは、WoTランタイムは、動作の実装が、プロトコルスタックの代わりにシステムと内部的に連動する利⽤されるThingをインスタンス化できるようにすべきでである。これは、アプリケーション向けのWoTランタイムAPIによって提供される発⾒メカニズムの結果に、そのようなシステムのThing (ローカルのWoTランタイムでのみ⼊⼿可能) を列挙することで⾏える。デバイスは、物理的にServientの外部に置くこともできるが、独⾃のプロトコルやWoTイン

ターフェースとしての条件を満たしていないプロトコルを介して接続することもできる

(§ 6.6 プロトコルバインディングを参照)。このケースでは、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 (例えば、ECHONET Lite、BACnet、X10、I2C、SPI など) で旧式デバイスにアクセスできるが、Thingの抽象化によって、それを動作の実装に公開することも選択できる。その後で、Servientは、旧式デバイスへのゲートウェイとして機能することができる。これは、WoT Thing Descriptionを⽤いて旧式デバイスを直接記述できない場合にのみ⾏うべきである。

動作の実装は、独⾃のAPIやその他の⼿段を介してローカルなハードウェアやシステムサービス (例えば、ストレージ) にアクセスすることもできる。しかし、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 代替のServientとWoT実装 §

WoTスクリプティングAPIの構成要素はオプションである。代替のServientを実装することができ、その場合、WoTランタイムは、アプリケーションロジックのための代替APIを提供するが、これは、任意のプログラミング⾔語で記述できる。

さらに、W3C WoTを意識していないデバイスやサービスであっても、それらに整形式の

WoT Thing Descriptionを提供できる場合は、利⽤できる。このケースでは、TDは、ブラックボックス的な実装を持つThingのWoTインターフェースを記述する。

8.8.1 ネイティブなWoT API §

開発者がWoTスクリプティングAPIを⽤いずに、Servientの実装を選択しうる理由は様々である。メモリやコンピュータ資源が不⾜しているために、開発者が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できないことが原因である場合がある。または、開発者が独⾃のユースケース (例えば、独⾃の通信プロトコル) をサポートするために、特定のプログラミング環境や⾔語でしか利⽤できない特定の機能やライブラリを⽤いなければならない場合もありえる。

このケースでは、WoTスクリプティングAPIの代わりに別のアプリケーション向けインターフェースを⽤いて同等の抽象化と機能を公開しつつ、依然としてWoTランタイムを使⽤できる。後者の場合を除き、§ 8. 抽象的なServientのアーキテクチャのすべての構成要素の説明は図28でも有効である。

**Servient**

ネイティブ

な

WoT

ランタイム



優先

⾔

語

での

動

作

の

実

装

ネイティブ

な

API

Private Security Data

プロトコルスタック

の

実

装

システム

利

⽤

WoT Thing

Description

WoT Thing

Description

公

開

WoT Thing Description

Public Security Metadata

相

互

作

⽤

の

アフォーダンス

⼀

般

的

な

メタデータ

Data Schema

プロトコルバインディング

バインディング

テンプレート

公

開

さ

れ

た

Thing

利

⽤

さ

れる

Thing

利

⽤

さ

れる

Thi

~~ng~~

図28 ネイティブなWoT APIを⽤いたServientの実装

8.8.2 既存デバイスのThing Description §

既存のIoTデバイスやサービスをW3C Web of Thingsに統合し、これらのデバイスやサービスに関するThing Descriptionを作成することにより、それらをThingとして⽤いることもできる。このようなTDは、⼿動でもツールやサービスを⽤いても作成できる。例えば、TD は、別のエコシステムに依存している機械可読形式で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り⽣成できる。しかし、これは、対象とするデバイスがプロトコルバインディングで記述できるプロトコルを⽤いている場合にのみ⾏えることである。これに関する要件は§ 6.6 プロトコルバインディングで⽰される。これまでの議論の多くは、Thingが⾃⾝のThing Descriptionを提供することも暗⽰している。これは便利なパターンだが、必須ではない。特に、既存のデバイスを変更して⾃⾝のThing Descriptionを直接提供することはできない場合がある。このケースでは、ディレクトリやその他の外部の別の配信メカニズムなどのサービスを⽤いてThing Descriptionを別途提供する必要がある。

既

存

の

デバイス

または

サービス

ブラックボックス

的

な

動

作

の

実

装

プロトコルスタック

の

実

装

Private Security Data

⼿

動

で

作

成

または

ツール

**Consumer**

相

互

作

⽤

処

理

記

述

WoT

Thing Description

Public Security Metadata

相

互

作

⽤

の

アフォーダンス

⼀

般

的

な

メタデータ

Data Schema

プロトコルバインディング

バインディング

テンプレート

図29 既存のIoTデバイスのW3C WoTへの統合

# 9. WoTのデプロイメント例 §

この章は参考情報である。

この章では、ThingとConsumerの役割を実装しているデバイスとサービスを様々な具体的なトポロジーとデプロイメントシナリオで相互接続する際に、Web of Things (WoT) の抽象アーキテクチャをインスタンス化する様々な⽅法の例を⽰す。これらのトポロジーとシナリオは規定的ではないが、WoT抽象アーキテクチャによって認められ、サポートされている。

特定のトポロジーについて論じる前に、まずThingとConsumerがWoTネットワークで担うことができる役割と、それらが持っている公開されたThingと利⽤されるThingのソフトウェア抽象化との関係について振り返る。公開されたThingと利⽤されるThingは、それぞれ

ThingとConsumerの役割をするServientの動作の実装に内部的に利⽤できる。

## 9.1 ThingとConsumerの役割 §

Thingの役割を持つServientは、Thing Description (TD) に基づいて公開されたThingを作り出す。TDは公開され、ConsumerまたはIntermediaryの役割を持つ他のServientから利

⽤できるようになります。TDは様々な⽅法で公開することができる。TDは、Thing

Directoryサービスなどの管理システムに登録されている場合もあるし、ThingがTDへのリクエストを受信するとリクエスト元にTDを提供するという場合もある。アプリケーションのシナリオによっては、TDをThingに静的に関連付けることすら可能である。

Consumerの役割を持つServientは、なにがしかの発⾒メカニズムを⽤いてThingのTDを取得し、その取得したTDに基づいて利⽤されるThingを作成する。具体的な発⾒メカニズムは、個々のデプロイメントシナリオに依存している。それは、Thing Directoryなどの管理システムや、発⾒プロトコル、あるいは静的な割り当て等により提供される可能性がある。

しかし、特定可能な⼈物に関連付けられているデバイスが記述されているTDは、プライバシー情報の推測に⽤いられる可能性があることに注意する必要がある。したがって、そのようなTDの配信に関する制約を具体的なTDの発⾒メカニズムに組み込まなければならない。可能であれば、特定のユースケースで絶対に必要な場合を除いて、IDや⼈間が読み取れる情報をフィルターで除外するなど、TDにより公開されている情報を制限する措置も講じなければならない。プライバシーに関する課題は、§ 10. セキュリティとプライバシーへの配慮でおおまかに論じられており、より詳細な議論が [WOT-THING-DESCRIPTION] 仕様で提供されている。

接続されたセンサーとアクチュエーターの相互作⽤などの、デバイスの内部システム機能もまた、オプションとして、利⽤されるThingの抽象化として表現することができる。

利⽤されるThingでサポートされる機能は、プログラミング⾔語のインターフェースを介してConsumerの動作の実装に提供される。WoTスクリプティングAPIでは、利⽤される

Thingはオブジェクトにより表現される。Thingで実⾏されている動作の実装 (つまり、アプリケーションのロジック) は、公開されたThingが提供するプログラミング⾔語インターフェースを⽤いることにより、相互作⽤のアフォーダンスを介してConsumerと連動することができる。

Thingは必ずしも物理デバイスを表現するとは限らない。Thingは、複数デバイスの集合

体、またはゲートウェイやクラウドで実⾏される仮想サービスを表現する可能性もある。同様に、Consumerは、ゲートウェイやクラウドで実⾏されているアプリケーションやサービ

スを表現する可能性がある。また、Consumerは、エッジデバイスに実装することもでき

る。Intermediaryにおいて、⼀つのServientが、⼀つのWoTランタイムを共有しながら、

ThingとConsumerの両⽅の役割を同時に果たしている。

## 9.2 WoTシステムのトポロジーとデプロイメントシナリオ §

この節では、WoTシステムの様々なトポロジーとデプロイメントシナリオについて論じる。これらはパターンの例にすぎず、他の相互接続トポロジーも可能である。ここで説明するトポロジーは、Web of Thingsのユースケース (§ 4. ユースケース) と、そこから抽出された技術要件 (§ 5. 要件) に基づくものである。

9.2.1 同じネットワーク上のConsumerとThing §

図30で⽰している最もシンプルな相互接続のトポロジーでは、ConsumerとThingは同じネットワーク上にあり、仲介なしに互いに直接通信を⾏える。このトポロジーになるユースケースの⼀つは、Consumerがゲートウェイで実⾏されているオーケストレーションサービス

(orchestration service) やその他のIoTアプリケーションで、Thingがセンサーやアクチュ

エータに接続しているデバイスである場合である。しかし、クライアント/サーバーの関係は簡単に逆転可能で、クライアントは、ゲートウェイやクラウド上でThingとして実⾏されているサービスにアクセスするConsumerの役割を持つデバイスになりえる。

**Thing**

**Consumer**

利

⽤

さ

れる

Thing

公

開

さ

れ

た

Thing

図30 同じネットワーク上のConsumerとThing

Thingがクラウド内にあり、Consumerがローカルネットワーク上にある場合 (スマートホームのユースケースの例については図1を参照) は、実際のネットワークトポロジーはより複雑になる。例えば、NATトラバーサルが必要で、ある種の発⾒は認められていない場合がある。このようなケースでは、後に論じるより複雑なトポロジーの⼀つの⽅が適切である。

9.2.2 Intermediaryを介して接続されたConsumerとThing § Intermediaryは、ネットワーク上でThingとConsumerの両⽅の役割を担い、WoTランタイム内で公開されたThingと利⽤されるThingの両⽅のソフトウェア抽象化をサポートす

る。Intermediaryは、デバイスとネットワークとの間のプロキシやデジタルツインに使⽤できる。

9.2.2.1 プロキシとして機能するIntermediary§

Intermediaryのシンプルな応⽤の⼀つは、Thingに対するプロキシである。Intermediary がプロキシとして機能する場合、⼆つの別々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持っている。これには、TLSエンドポイントの提供など、付加的なセキュリティのメカニズムの実装が伴う場合がある。⼀般的に、プロキシによって相互作⽤が変わることはないため、Intermediaryによって公開されるTDの相互作⽤は、利⽤されるTDの相互作⽤と同じであるが、接続メタデータは変更されている。

このプロキシのパターンを実装するために、Intermediaryは、ThingのTDを取得して利⽤されるThingを作成する。これにより、同じ相互作⽤のアフォーダンスを持つソフトウェアの実装としてThingのプロキシオブジェクトが作成される。次に、新しい識別⼦と、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 (または) 新しいPublic Security Metadataを備えたプロキシオブジェクトのTDが作成される。最後に、このTDに

基づいて公開されたThingが作成され、Intermediaryは、適切な公開メカニズムを介して、他のConsumerやIntermediaryにTDを通知する。

**Consumer Intermediary Thing**

利⽤されるThing 公開Thingされた 利⽤Thingされる 公開されたThing

図31 プロキシとして機能するIntermediaryを介したConsumerとThingの接続

9.2.2.2 デジタルツインとして機能するIntermediary§

より複雑なIntermediaryは、デジタルツインとして知られている。デジタルツインは、プロトコルの変更や、ネットワーク間の通訳を⾏う場合も⾏わない場合もあるが、状態のキャッシング、更新の先延ばし、対象デバイスの動作の予測シミュレーションなどの追加サービスを提供する。例えば、IoTデバイスの電⼒が限られていれば、あまり頻繁には起動せず、デジタルツインと同期した直後にスリープ状態に戻ることを選択できる。このケースでは通常、デジタルツインは電⼒の制約が少ないデバイス上 (クラウドやゲートウェイなど) で実⾏され、制約のあるデバイスの代わりに相互作⽤に応答することができる。現在のプロパティーの状態に対するリクエストを、デジタルツインがキャッシュされた状態を⽤いて満たす場合もある。対象としているIoTデバイスがスリープ状態にあるときに到着したリクエストは

キューに⼊れられ、起動時に送信される。このパターンを実装するためには、

Intermediary、つまりデジタルツインは、いつデバイスが起動しているかを知る必要がある。Thingとしてのデバイスの実装には、そのための通知メカニズムを含める必要がある。これは、別のConsumer/Thingのペアを⽤いて、またはこの⽬的のためのEventの相互作⽤を⽤いて実装できる。

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ているデバイス (センサーと家電) は監視されることが多く、場合によってはクラウドサービスによる制御も⾏われている。通常、デバイスが接続されているホームネットワークとクラウドの間にはNATデバイスが存在している。NATデバイスは、IPアドレスを変換するだけでなく、接続を選択的にブロックするファイアウォールのサービスを提供していることが多い。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通過できた場合にのみ相互に通信を⾏える。

ITU-T勧告Y.4409/Y.2070 [Y.4409-Y.2070] で採⽤されている典型的な構造を図32で⽰している。この構造には、ローカルとリモートの両⽅のIntermediaryがある。ローカルの

Intermediaryは、複数のThingの相互作⽤のアフォーダンスを⼀つの公開されたThing (の集合) へと集約させ、そのすべてを (すべての相互作⽤が、共通のベースとなるサーバが持つ

⼀つのURL名前空間にマッピングされていて、⼀つのポートを⽤いているHTTPなどの) 共通のプロトコルにマッピングすることができる。これにより、ローカルのIntermediaryが

NATデバイスをトラバースできる集約型プロトコルを⽤いていて、そのサービスをインターネットで公開する (STUN、TURN、DyDNSなどの) 何らかの⽅法を持っていると仮定すると、NATデバイスの背後にあるすべてのThingにアクセスするためのシンプルな⽅法がリモートのIntermediaryに提供される。さらに、ローカルのIntermediaryは、Thingのプロキシの機能を果たすことができるため、接続されたThingで、それぞれ (HTTP、MQTT、

CoAPなどの) 異なるプロトコルや様々なエコシステムの規定が⽤いられていたとしても、

Thingで⽤いられている様々なプロトコルをConsumerが意識しなくてもよいように、公開されたThingはそれらを⼀つのプロトコルに集約することができる。

図32にはリモートのIntermediaryに接続された⼆つのクライアントがあり、これは、NAT の境界の背後にあるサービスを集約しており、付加的なプロトコル変換やセキュリティのサービスを提供できる。特に、ローカルのIntermediaryは容量が限られたネットワーク上にある可能性があり、そのサービスをすべてのユーザが直接利⽤できるようにすることは現実的ではない。このケースでは、ローカルのIntermediaryへのアクセスはリモートの

Intermediaryにのみ提供される。次に、リモートのIntermediaryは、より⼀般的なアクセス制御メカニズムを実装し、キャッシングやスロットリングを実⾏して、過剰なトラフィックから消費者を保護することもできる。また、これらの消費者は、(HTTPSなどの) オープン

なインターネットに適した⼀つのプロトコルを⽤いてIntermediaryと通信すると考えられ、それによってクライアントの開発はよりシンプルになる。

このトポロジーでは、ConsumerとThingの間にNATとファイアウォールの機能があるが、ローカルとリモートのIntermediaryが連携してファイアウォールをトンネリングさせ、すべての通信を通すため、ConsumerとThingはファイアウォールについて何も知っている必要がない。ペアになったIntermediaryは、アクセス制御とトラフィック管理を提供することにより、ホームデバイスの保護も⾏う。

### 翻訳者のメモ

英語原⽂中で「between the consumers and things」は、本来、⼤⽂字の

「Consumers and Things」とすべきものと考えられる。

また、「so the consumers and things need to know」においても、「so the

Consumers and Things need to know」とすべきものと考えられる。

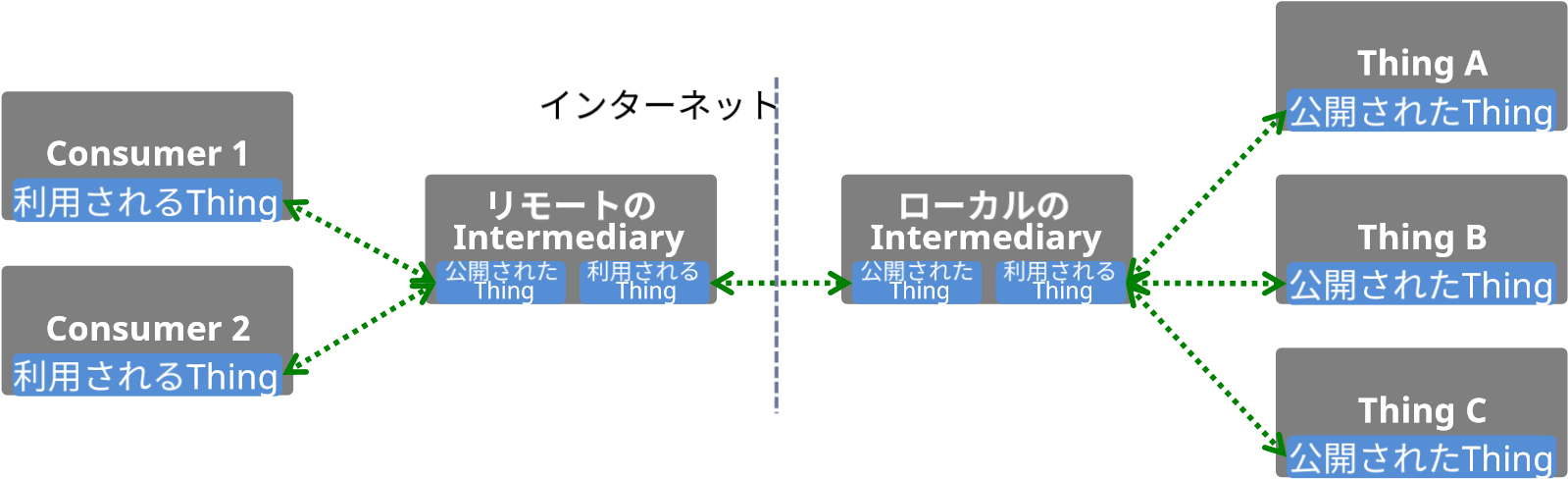


図32 ペアになったIntermediaryを介してThingとして実装されたローカルデバイスに接続された

Consumerとして実装されたクラウドアプリケーション

より困難なケースでは、NATとファイアウォールトラバーサルが⽰されているとおりに機能しない場合がある。特に、ISPがパブリックにアクセス可能なアドレスをサポートしていなかったり、STUN/TURNおよび (または) DyDNSがサポートされていなかったり利⽤できない場合がある。このケースでは、Intermediaryは、クライアント/サーバーの役割を逆にし

て初期接続を設定し (ローカルのIntermediaryが最初にクラウド上のリモートの

Intermediaryに接続して)、次に、ペアとなったIntermediaryが (例えば、TLSを⽤いて接続を保護した安全なWebSocketを⽤いて) トンネルを確⽴する。このトンネルは、特注のプロトコルを⽤いてIntermediaryの間のすべての通信をエンコードするために使うこともできる。このケースでは、通常のブラウザ/ウェブサーバーの相互作⽤と同様に、ローカルの

InermediaryからリモートのIntermediaryへと、標準ポートを⽤いてHTTPSで初期接続を⾏うことができる。これにより、ほとんどのホームファイアウォールを通過でき、接続が外向きであるため、ネットワークアドレス変換による問題が発⽣しない。しかし、特注のトンネリングプロトコルが必要な場合でも、リモートのIntermediaryはこの特注のプロトコルを標準の外部プロトコルに変換することができる。接続されたConsumerとThingは、この変換について知っている必要はない。この例を、ThingとConsumerの両⽅がNATの境界の

両側で接続できるユースケースに拡張することもできる。しかし、そのためには、⼆つの

Intermediaryの間に双⽅向のトンネルを確⽴する必要もある。

9.2.4 Thing Directoryを⽤いた発⾒ §

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) を監視または制御できるようになると、その上で様々な付加的なサービスを構築できる。例えば、クラウドアプリケーションは、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できる。

しかし、リモートのIntermediaryがクライアントアプリケーションにサービスを提供している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、接続されているデバイスのディレクトリにアクセスすることにより、デバイス情報を⾒つけることができる必要がある。下の図では、シンプルにするために、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ケーションがConsumerとして実装されていると想定している。Thingとして実装されているローカルデバイスのメタデータをクラウドアプリケーションで利⽤できるようにするために、そのメタデータをThing Directory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、リモートのIntermediaryによっ

て提供されるPublic Security Metadataと通信メタデータ (プロトコルバインディング) を反映するように変更されたローカルデバイスのTDである。その後で、Thing Directoryに照会することにより、クライアントアプリケーションは、ローカルデバイスと通信するために必要なメタデータを取得し、その機能を実現できるようになる。

リモート

の

**Intermediary**

公

開

さ

れ

た

Thing

**Consumer**

利

⽤

さ

れる

Thing

利

⽤

さ

れる

Thing

**Thing**

公

開

さ

れ

た

Thing

ローカル

の

**Intermediary**

公

開

さ

れ

た

Thing

利

⽤

さ

れる

Thing



インターネット

**Thing**

**Directory**

図33 Thing Directoryを⽤いたクラウドサービス

上図では⽰していない、より複雑な状況では、Thingとして機能するクラウドサービスもあ

りえる。この場合も、Thing Directoryに⾃⾝を登録することができる。Thing Directoryはウェブサービスであるため、NATやファイアウォールデバイスを介してローカルデバイスから⾒ることができるべきであり、そのインターフェースを⾃⾝のTDにより提供することさえ可能である。そして、Consumerとして動作するローカルデバイスは、Thing Directoryを介してクラウド内のThingを発⾒し、直接、Thingに接続することができる。あるいは、例えば、プロトコル変換が必要な場合、ローカルにあるIntermediaryを介して、Thingに接続することができる。

9.2.5 複数の領域にまたがるサービス間の接続 §

それぞれが異なるIoTプラットフォームに基づいている複数のクラウドのエコシステムは、連携してより⼤きな「システムのシステム」というエコシステムを構築できる。下の図は、前述のクラウドアプリケーションのエコシステムの構造に基づいて、「システムのシステム」を作成するために相互接続された⼆つのエコシステムを⽰している。あるエコシステムのクライアント (つまり、下記のConsumer A) が別のエコシステムのサーバー (つまり、下記のThing B) を利⽤する必要がある場合を考えてみよう。このエコシステムを横断するアプリケーションとデバイスの統合を実現するメカニズムは複数存在する。以下では、これを実現する⽅法を⽰すために、⼆つの⽅法について、それぞれ図を⽤いて説明する。

9.2.5.1 Thing Directoryの同期を介した接続§

図34では、⼆つのThing Directoryが情報を同期させており、それによって、Consumer A はThing Directory Aを介してThing Bの情報を取得できる。既に説明したように、リモートのIntermediary BはThing Bの影 (shadow) の実装を維持する。この影のデバイスのTD

を取得することにより、Consumer Aは、リモートのIntermediary Bを介してThing Bを使

⽤できる。

### 翻訳者のメモ

上記説明で⽤いられている「影 (shadow)」については、「デジタルツイン」とも呼ばれる、デバイスの仮想表現であることを明⽰的に説明するべき．

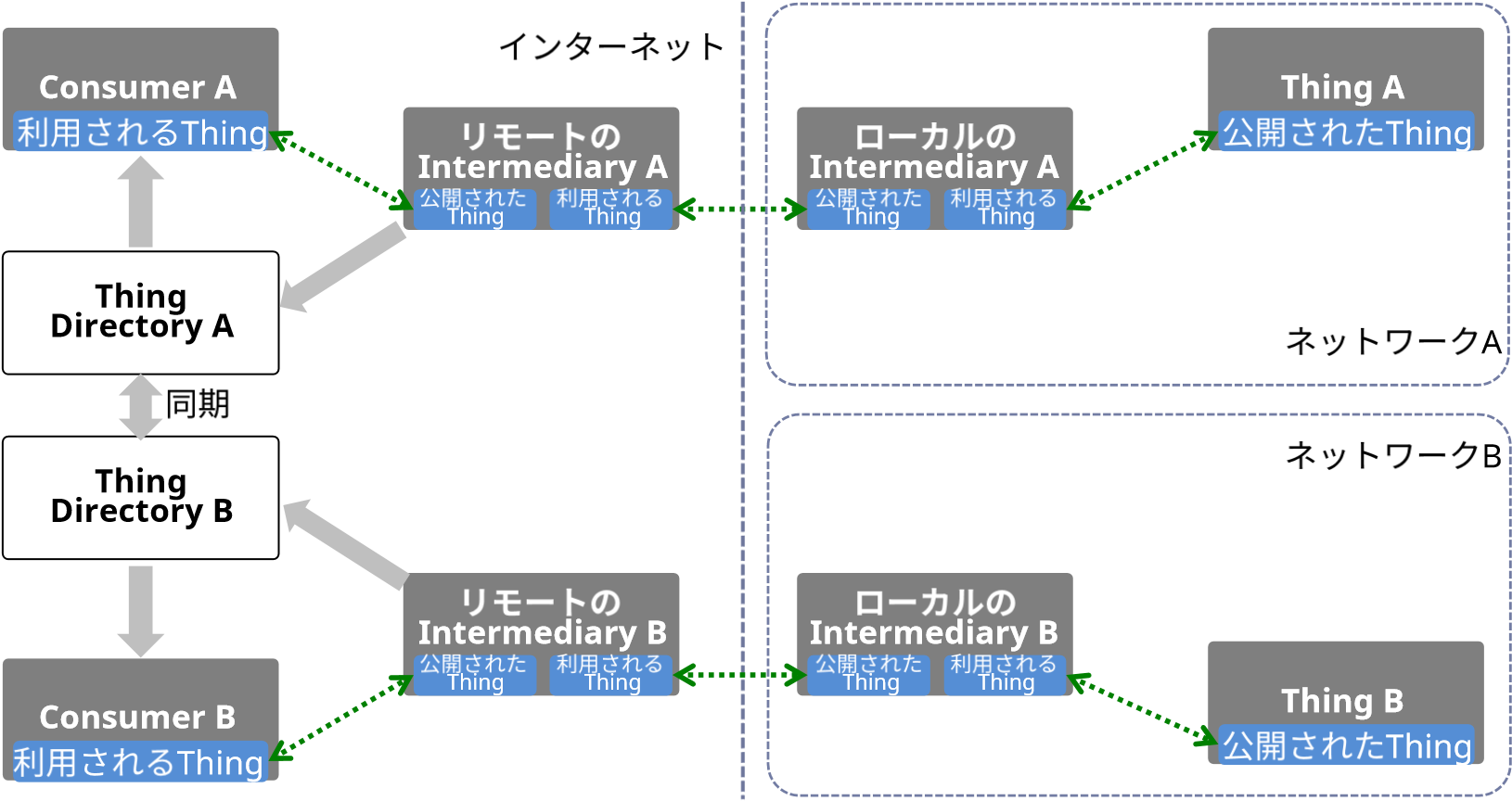


図34 Thing Directoryの同期を介した複数クラウドの接続

9.2.5.2 プロキシの同期を介した接続§

図35では、⼆つのリモートのIntermediaryがデバイス情報を同期させている。Thing Bの影がリモートのIntermediary Bで作成されると同時に、影のTDがリモートのIntermediary A に同期される。次に、リモートのIntermediary Aは、⾃分⾃⾝の中にThing Bの影を作成

し、TDをThing Directory Aに登録する。この⽅法であれば、Thing Directory間の同期は必要ない。

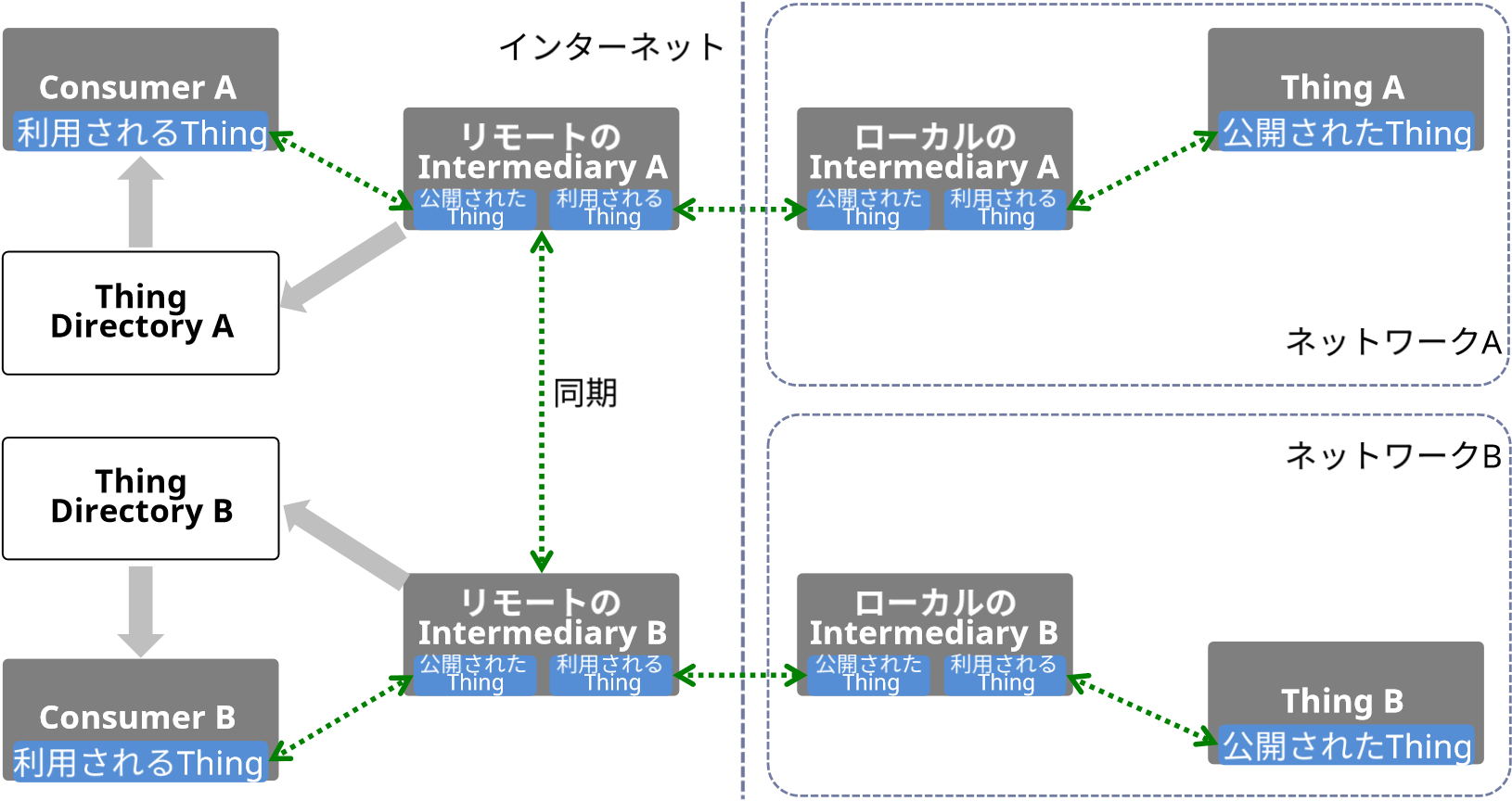


図35 Intermediaryの同期を介した複数クラウドの接続

# 10. セキュリティとプライバシーへの配慮 §

この章は参考情報である。

セキュリティとプライバシーは分野横断的な課題であり、すべてのWoT構成要素とWoT実装において考慮する必要がある。この章では、具体的なWoT実装のセキュリティとプライバシーの保護に役⽴つように、⼀般的な課題とガイドラインをまとめる。しかし、これらは⼀般的なガイドラインに過ぎず、この⽂書で記述し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ーを保証することはできない。その代わりに、具体的な実装の詳細について熟考する必要がある。セキュリティとプライバシーに関する課題の、より詳細で完全な分析は、WoTセキュリティとプライバシーに関するガイドライン仕様 [WOT-SECURITY] を参照のこと。

全体として、WoTの⽬標は、セキュリティを含めて、IoTデバイスとIoTサービスの既存のアクセス⽅法とプロパティーを記述することである。⼀般的に、W3C WoTは、何を実装するのかを規定することではなく、何が存在するのかを記述することを⽬指している。既存システムの記述は、仮に理想的なセキュリティの動作を備えていなかったとしても、そのシステムを正確に描写すべきである。システムのセキュリティの脆弱性を明確に理解することは、セキュリティ脅威の軽減に役⽴つ — しかし、もちろん、そのようなデータを、悪⽤する可能性のある⼈に配信する必要はない。

しかし、特に新しいシステムの場合は、WoTアーキテクチャにより、セキュリティとプライバシーのベストプラクティスを利⽤できるようになるべきである。⼀般的に、WoTのセキュリティアーキテクチャは、接続するIoTプロトコルとシステムの、⽬的とメカニズムをサポートしなければならない。このようなシステムでは、セキュリティ要件とリスクの許容度が多岐に渡るため、これらの要因に基づいてセキュリティメカニズムも多岐に渡る。

IoTデバイスは⾃律的に動作する必要があり、多くの場合、個⼈データへのアクセスを有しており、加えて (または) 安全を重視するシステムの制御下にありえるため、セキュリティとプライバシーは、IoTの領域において特に重要である。IoTデバイスは、ITシステムとは異なるリスク、場合によってはより⾼いリスクにさらされやすい。IoTシステムを保護し、他のコンピューターシステムへの攻撃を仕掛けるために⽤いられることのないようにすることも重要である。

⼀般的に、セキュリティとプライバシーを保証することはできない。安全でないシステムを WoTによって安全なシステムに変えることはできないが、WoTアーキテクチャが害を及ぼさないようにする必要があり、少なくとも、WoTアーキテクチャが記述しサポートする対象システムと同程度には、セキュリティとプライバシーをサポートすべきである。

## 10.1 WoT Thing Descriptionに関するリスク §

WoT Thing Description (TD) に含まれているメタデータには、潜在的に機密性がある。ベ

ストプラクティスとして、TDは完全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

⽤いるべきであり、権限を持つユーザにのみ提供を⾏うべきである。

これらの点に関する詳細や議論については、WoT Thing Description仕様のセキュリティとプライバシーに関する考察の項を参照すること。

10.1.1 Thing DescriptionのPrivate Security Dataに関するリスク §

TDは、Public Security Metadataのみの利⽤を⽬指している。TDの作成者は、TDに

Private Security DataがTDに含まれないようにしなければならない。Public Security

MetadataとPrivate Security Dataは厳密に分離されるべきである。TDにはPublic Security

Metadataのみが含まれているべきであり、Consumerが権限を持つ場合にのみ、システムとしてアクセスするために何をする必要があるのかを知らせる。そして、権限付与は別途管理されている⾮公開情報に基づくべきである。

TDの仕様で定義されているTDセキュリティスキームは、Private Security Dataのエンコーディングを意図的にサポートしていない。しかし、⼈間が読める記述などのその他のフィールドが誤⽤されてこの情報のエンコードが⾏われたり、そのような情報をエンコードする拡張メカニズムにより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。

軽減策:

TDとTDで⽤いられる拡張機能の作成者は、TDにはPublic Security Metadataのみが保存されるようにしなければならない。

10.1.2 Thing Descriptionの個⼈識別可能情報に関するリスク §

Thing Descriptionには、様々な種類の個⼈識別可能情報が含まれている可能性がある。明

⽰的ではない場合でも、TDと、それにより特定可能な⼈物とを関連付けると、その⼈物に関する情報を推測できるようになる。例えば、携帯デバイスによって公開された、フィンガープリンティング可能なTDとの関連付けによる位置情報の特定は、追跡のリスクになりえる。特定のデバイスのインスタンスを識別できない場合でも、TDで表現されるデバイスの種類が

⼈物に関連付けられていれば、個⼈情報となる可能性がある。例えば、ユーザに病状があると推測するために、医療機器が利⽤されるかもしれない。

⼀般的に、TD内の個⼈識別可能情報はできる限り制限すべきである。しかし、回避できない場合もありえる。TD中に直接的なPIIと推論可能なPIIの両⽅が存在している可能性があるということは、TDを別形式のPIIのように扱うべきであることを意味する。それは、安全な⽅法で保存され送信されるべきであり、承認されたユーザにのみ提供されるべきであり、限られた回数だけキャッシュされるべきであり、リクエストに応じて削除されるべきであり、ユーザの同意を得て提供された⽬的にのみ⽤いられるべきであり、そうでない場合は、PIIの使

⽤に関するすべての要件 (法的要件を含む) を満たすべきである。

軽減策:

TDでのPIIの保存はできる限り最⼩限に抑えるべきである。TDに明⽰的なPIIがなくても、追跡のリスクや特定のプライバシーリスクが存在する場合がある。このリスクを最⼩限に抑えるために、⼀般的にPIIが含まれているかのようにTDを扱い、他のPIIと同じ管理⽅針に従うべきである。それは、承認された利⽤者にのみ提供すべきである。特定のユースケースに不要な情報は、できる限りTDで公開すべきではない。例えば、TD内の情報を識別する明⽰的な型とインスタンスも、ユースケースで必要でない場合には含めるべきではない。ユースケースで必要である場合でも、追跡のリスクを最⼩限に抑えるために、できる限り、グローバルに⼀意な識別⼦ではなく、分散型の限定的な範囲の識別⼦を⽤いるべきである。⼀部のユースケースでは、フィンガープリンティングのリスクを減らすために、⼈間が読める記述などのその他の形式の情報も削除可能である。

10.1.3 Thing Descriptionのコミュニケーションメタデータに関するリスク §

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での使⽤の条件を満たしていると⾒なされるように、基盤となるIoTプラットフォームで採⽤されてい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。IoTを⼤規模に展開するために必要なネットワークの相互作⽤が⾃動化されているため、オペレーターは、セキュリティ⽅針に準拠した⽅法でThingが公開され利⽤されることを保証する必要がある。

軽減策:

TDの作成者は、できる限り、WoTバインディングテンプレートで提供されている、⼊念なチェック済みの通信メタデータを⽤いるべきである。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エコシステム⽤にTDを⽣成する場合、IoTプラットフォームのすべてのセキュリティ要件が満たされていることを保証する必要がある。

## 10.2 WoTスクリプトAPIのセキュリティとプライバシーに関するリスク §

WoTランタイムの実装とWoTスクリプトAPIには、システムへの悪意のあるアクセスを防ぎ、マルチテナント⽅式のServientにおけるスクリプトを分離するメカニズムがあるべきである。より具体的には、WoTスクリプトAPIをWoTランタイムの実装と併⽤する場合は、以下のセキュリティとプライバシーに関するリスクを考慮し、推奨される軽減策を実装すべきである。

10.2.1 クロススクリプトのセキュリティとプライバシーに関するリスク §

基本的なWoTのセットアップにおいては、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と⾒なしたうえで製造者が配信を⾏うため、実⾏する各スクリプトのインスタンス間で厳密な分離を⾏う必要はない。しかし、デバイスの性能、展開の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、あるスクリプトで機密性の⾼いプライバシーに関わるPIIデータが扱われていて、⼗分な監査が⾏われていれば、同じシステム内の他のスクリプトがランタイム中に侵害された場合にデータが露出するリスクを最⼩限に抑えるために、そのスクリプトを残りのスクリプトのインスタンスから分離することが望ましいかもしれない。別の例は、⼀つのWoTデバイス上に異なるテナントが共存している場合である。このケースでは、WoTランタイムのインスタンスでそれぞれ異なるテナントが提供されているため、それらを分離する必要がある。

軽減策:

プライバシーに関わるデータや、その他のクリティカルなセキュリティデータをスクリプトで扱う場合には、WoTランタイムはスクリプトのインスタンスとそのデータを分離すべきである。同様に、WoTランタイムの実装では、あるWoTデバイスに複数のテナントがある場合には、WoTランタイムのインスタンスとそのデータの分離を⾏うべきである。このような分離は、デバイスで利⽤できるプラットフォームのセキュリティメカニズムを⽤いてWoTランタイム内で実⾏できる。詳細に関しては、WoTセキュリティとプライバシーに関するガイドライン仕様 [WOT-SECURITY] の「WoT Servientシングルテナント」と「WoT Servientマルチテナント」の章を参照のこと。

10.2.2 物理デバイス直接アクセスのセキュリティとプライバシーに関するリスク §

直接公開されているネイティブなデバイスのインターフェースをスクリプトで使⽤できる場合は、スクリプトが侵害されたり誤動作したりすると、基盤となる物理デバイス (および潜在的に周辺環境) が被害を受ける可能性がある。そのようなインターフェースで、⼊⼒に対する安全性のチェックが不⾜していれば、基盤となる物理デバイス (または環境) は安全ではない状態となる可能性がある。

軽減策:

WoTランタイムは、ネイティブなデバイスのインターフェースをスクリプトの開発者に直接公開することを避けるべきである。代わりに、WoTランタイムの実装では、ネイティブなデバイスの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化レイヤーを提供すべきである。このようなハードウェア抽象化レイヤーは、デバイス (または環境) を安全でない状態にする可能性のあるコマンドの実⾏を拒否すべきである。さらに、スクリプトが侵害された場合の物理的なWoTデバイスへの被害を減少させるために、特定のスクリプトに対して公開またはアクセスできるインターフェースの数を、スクリプトの機能に応じて最⼩限に抑えることが重要である。

## 10.3 WoTランタイムのセキュリティとプライバシーに関するリスク §

10.3.1 プロビジョニングと更新のセキュリティリスク §

WoTランタイムの実装で、製造後のプロビジョニングや、WoTランタイム⾃⾝、スクリプ

ト、または関連するデータ (セキュリティ証明書を含む) の更新がサポートされている場合、それは主要な攻撃ベクトル (attack vector) になる可能性がある。攻撃者は、更新やプロビジョニングの⼯程の間に上記の要素を変更しようとしたり、攻撃者のコードとデータのプロビジョニングを直接⾏ったりすることができる。

軽減策:

製造後のプロビジョニングやスクリプト、WoTランタイム⾃⾝、または関連するデータの更新は、安全な⽅法で⾏うべきである。安全な更新と製造後のプロビジョニングに関

する推奨は、WoTセキュリティとプライバシーに関するガイドライン仕様 [WOT-

SECURITY] にある。

10.3.2 セキュリティ証明書保管のセキュリティとプライバシーに関するリスク §

通常、WoTランタイムは、ネットワークで動作するために、WoTデバイスにプロビジョニングを⾏ったセキュリティ証明書を保管する必要がある。攻撃者がこれらの証明書の機密性や完全性を侵害できれば、資産にアクセスできるようになったり、他のWoTモノ、デバイス、またはサービスになりすましたり、DoS (Denial-Of-Service) 攻撃を仕掛けたりすることができる。

軽減策:

WoTランタイムは、プロビジョニング済みのセキュリティ証明書を安全に保管し、完全性と機密性を保証すべきである。⼀つのWoT対応デバイスに複数のテナントがある場合、WoTランタイムの実装では、各テナントのプロビジョニング済みのセキュリティ証明書の分離を保証すべきである。さらに、プロビジョニング済みのセキュリティ証明書が侵害されるリスクを最⼩限に抑えるために、WoTランタイムの実装は、プロビジョニング済みのセキュリティ証明書にクエリを実⾏するスクリプトのAPIを公開すべきではない。そのような証明書 (または、それを利⽤するけれども公開はしないという、ずっとましな抽象的操作であっても) には、それらを⽤いるプロトコルバインディングの実装のみがアクセスできるべきである。

A. 最近の仕様変更 §

# 勧告案からの変更 §

規定的な変更はなく、軽微な編集上の修正あり、外部参照には変更なし。

# 最初の勧告候補からの変更 §

⽤語の項を参考情報に。

要約とはじめにを再編成。

プライバシーに関する議論と軽減策を拡⼤。

図中のテキストを、⽂中のテキスト変更に適合。

定義を更新・拡張。参考⽂献を更新。

規定的な参考⽂献を追加: RFC2046

規定的な参考⽂献を削除または参考情報の項に移動: IANA-RELATIONS, MQTT,

RFC4395, RFC6838, RFC7049, RFC7231, RFC7252

アクセシビリティのフィードバックに基づいて図の⾊を調整。

誤植や⼤⽂字の⽤法など、編集上の軽微な修正。

# 最初の公開草案からの変更 §

ユースケースを改訂・拡張。

要件を再編成。

抽象的なアーキテクチャの定義。

⽤語を改訂・明確化。実装と展開への追加。

セキュリティとプライバシーに関する留意点の追加。

# B. 謝辞 §

この⽂書への貢献に対し、Michael McCool、Takuki Kamiya、Kazuyuki Ashimura、

Sebastian Käbisch、Zoltan Kis、Elena Reshetova、Klaus Hartke、Ari Keränen、

Kazuaki Nimura、Philippe Le Hegaretに特に感謝する。

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) のすべての関係者に感謝する。

WoT WGは、 [WOT-PIONEERS-1] [WOT-PIONEERS-2] [WOT-PIONEERS-3] [WOT-

PIONEERS-4] などの刊⾏物の形で学術的イニシアチブとして開始された「Web of [Things」の概念に関する先駆的な取り組みにも感謝する。これにより、2010年からWeb of Thingsの国際ワークショップが毎年開催されようになった。](https://webofthings.org/events/wot/)

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。

# C. 参考⽂献 §

## C.1 規定的な参考⽂献 §

[RFC2046]

[Multipurpose Internet Mail Extensions (MIME) Part Two: Media Types.](https://tools.ietf.org/html/rfc2046) 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](https://tools.ietf.org/html/rfc2119). S. Bradner. IETF. March 1997. Best Current Practice. URL: <https://tools.ietf.org/html/rfc2119>

[RFC3986]

[Uniform Resource Identifier (URI) : Generic Syntax](https://tools.ietf.org/html/rfc3986). T. Berners-Lee; R. Fielding; L. Masinter. IETF. January 2005. Internet Standard. URL: <https://tools.ietf.org/html/rfc3986>

[RFC3987]

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

[RFC5234]

[Augmented BNF for Syntax Specifications: ABNF.](https://tools.ietf.org/html/rfc5234) 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](https://tools.ietf.org/html/rfc8174). B. Leiba. IETF. May 2017. Best Current Practice. URL: <https://tools.ietf.org/html/rfc8174>

[RFC8259]

[The JavaScript Object Notation (JSON) Data Interchange Format](https://tools.ietf.org/html/rfc8259). T. Bray, Ed..

IETF. December 2017. Internet Standard. URL: <https://tools.ietf.org/html/rfc8259>

[RFC8288]

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

<https://httpwg.org/specs/rfc8288.html>

## C.2 参考情報の参考⽂献 §

[CoRAL]

[The Constrained RESTful Application Language (CoRAL) .](https://tools.ietf.org/html/draft-hartke-t2trg-coral) Klaus Hartke. IETF. [March 2019. Internet-Draft. URL: https://tools.ietf.org/html/draft-hartke-t2trgcoral](https://tools.ietf.org/html/draft-hartke-t2trg-coral)

[CoRE-RD]

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

[ECMAScript]

[ECMAScript Language Specification.](https://tc39.es/ecma262/) Ecma International. URL:

<https://tc39.github.io/ecma262/>

[HCI]

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

[HTML]

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

[IANA-RELATIONS]

[Link Relations.](https://www.iana.org/assignments/link-relations/) IANA. URL: <https://www.iana.org/assignments/link-relations/>

[IANA-URI-SCHEMES]

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

[IEC-FOTF]

[Factory of the future.](https://www.iec.ch/whitepaper/pdf/iecWP-futurefactory-LR-en.pdf) IEC. October 2015. URL: <https://www.iec.ch/whitepaper/pdf/iecWP-futurefactory-LR-en.pdf>

[IOT-SCHEMA-ORG]

[Schema Extensions for IoT Community Group](https://www.w3.org/community/iotschema/). URL: <https://www.w3.org/community/iotschema/>

[ISO-IEC-2382]

[Information technology — Vocabulary.](https://www.iso.org/obp/ui/#iso:std:iso-iec:2382:ed-1:v1:en) 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](https://www.iso.org/obp/ui/#iso:std:iso-iec:27000:ed-5:v1:en)

[ISO-IEC-29100]

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

[JSON-LD11]

[JSON-LD 1.1](https://www.w3.org/TR/2020/CR-json-ld11-20200316/). 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/>

[LINKED-DATA]

[Linked Data Design Issues.](https://www.w3.org/DesignIssues/LinkedData.html) 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.](http://openmobilealliance.org/release/LightweightM2M/V1_1-20180710-A/OMA-TS-LightweightM2M_Core-V1_1-20180710-A.pdf) OMA SpecWorks.

August 2018. Approved Version: 1.1. URL: [http://openmobilealliance.org/release/LightweightM2M/V1\_1-20180710-A/OMATS-LightweightM2M\_Core-V1\_1-20180710-A.pdf](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.](http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqtt-v3.1.1.html) Andrew Banks; Rahul Gupta. OASIS Standard. [December 2015. URL: http://docs.oasis-open.org/mqtt/mqtt/v3.1.1/mqttv3.1.1.html](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](https://openconnectivity.org/developer/specifications). 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](https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)

[Architectures. Roy Thomas Fielding. University of California, Irvine. 2](https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf)000. PhD thesis. URL: <https://www.ics.uci.edu/~fielding/pubs/dissertation/fielding_dissertation.pdf>

[RFC4301]

[Security Architecture for the Internet Protocol.](https://tools.ietf.org/html/rfc4301) 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](https://tools.ietf.org/html/rfc6202)

[Bidirectional HTTP. S. Loreto; P. Saint-Andre; S. Salsano; G. Wilkins. IETF. Apr](https://tools.ietf.org/html/rfc6202)il 2011. Informational. URL: <https://tools.ietf.org/html/rfc6202>

[RFC6347]

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

[RFC6690]

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

[RFC6749]

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

[RFC7049]

[Concise Binary Object Representation (CBOR)](https://tools.ietf.org/html/rfc7049) . 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](https://httpwg.org/specs/rfc7231.html). R. Fielding, Ed.; J. Reschke, Ed.. IETF. June 2014. Proposed Standard. URL: <https://httpwg.org/specs/rfc7231.html>

[RFC7252]

[The Constrained Application Protocol (CoAP) .](https://tools.ietf.org/html/rfc7252) 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)](https://tools.ietf.org/html/rfc7641) . K. Hartke.

IETF. September 2015. Proposed Standard. URL: <https://tools.ietf.org/html/rfc7641>

[RFC7744]

[Use Cases for Authentication and Authorization in Constrained Environments.](https://tools.ietf.org/html/rfc7744) 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.](https://tools.ietf.org/html/rfc8446) E. Rescorla. IETF. August 2018. Proposed Standard. URL: <https://tools.ietf.org/html/rfc8446>

[SAREF]

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

[VOCAB-SSN]

[Semantic Sensor Network Ontology](https://www.w3.org/TR/2017/REC-vocab-ssn-20171019/). 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-ssn20171019/](https://www.w3.org/TR/2017/REC-vocab-ssn-20171019/)

[WOT-BINDING-TEMPLATES]

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

[WOT-PIONEERS-1]

[Mobile Service Interaction with the Web of Things](https://pdfs.semanticscholar.org/3ee3/a2e8ce93fbf9ba14ad54e12adaeb1f3ca392.pdf). 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.pd f](https://pdfs.semanticscholar.org/3ee3/a2e8ce93fbf9ba14ad54e12adaeb1f3ca392.pdf)

[WOT-PIONEERS-2]

[Putting Things to REST](http://dret.net/netdret/docs/wilde-irep07-015-restful-things.pdf). Erik Wilde. UCB iSchool Report 2007-015, UC Berkeley, [Berkeley, CA, USA. November 2007. URL: http://dret.net/netdret/docs/wildeirep07-015-restful-things.pdf](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 Romer; Michael Fahrma](https://www.vs.inf.ethz.ch/publ/papers/ostermai-poster-2008.pdf)ir; Wolfgang Kellerer. Proceedings of ACM SenSys 2008, Raleigh, NC, USA.

[November 2008. URL: https://www.vs.inf.ethz.ch/publ/papers/ostermai-poster2008.pdf](https://www.vs.inf.ethz.ch/publ/papers/ostermai-poster-2008.pdf)

[WOT-PIONEERS-4]

[A Resource Oriented Architecture for the Web of Things.](https://ieeexplore.ieee.org/abstract/document/5678452) 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](https://www.w3.org/TR/2019/WD-wot-scripting-api-20191028/). Zoltan Kis; Daniel Peintner; Johannes Hund; Kazuaki Nimura. W3C. 28 October 2019. W3C Working Draft. URL: <https://www.w3.org/TR/2019/WD-wot-scripting-api-20191028/>

[WOT-SECURITY]

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

[WOT-THING-DESCRIPTION]

[Web of Things (WoT) Thing Description](https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/). Sebastian Kabisch; Takuki Kamiya; Michael McCool; Victor Charpenay; Matthias Kovatsch. W3C. 9 April 2020. W3C

[Recommendation. URL: https://www.w3.org/TR/2020/REC-wot-thing-description20200409/](https://www.w3.org/TR/2020/REC-wot-thing-description-20200409/)

[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.](https://www.itu.int/rec/T-REC-Y.2070-201501-I) Recommendation. URL: <https://www.itu.int/rec/T-REC-Y.2070-201501-I>

↑