【注意】 この文書は、2019年5月16日付の W3C勧告候補「Web of Things (WoT) Architecture」(原文は英語)を、WoT JP Communityが翻訳と修正をおこなって公開しているものです。この文書の正式版は、あくまでW3Cのサイト内にある英語版であり、この文書には翻訳上の間違い、あるいは不適切な表現が含まれている可能性がありますのでご注意ください。

W3C

Web of Things (WoT) Architecture

W3C Candidate Recommendation 16 May 2019

This version:
https://www.w3.org/TR/2019/CR-wot-architecture-20190516/
Latest published version:
https://www.w3.org/TR/wot-architecture/
Latest editor's draft:
https://w3c.github.io/wot-architecture/
Preliminary implementation report:
https://w3c.github.io/wot-thing-description/testing/report.html
Previous version:
https://www.w3.org/TR/2017/WD-wot-architecture-20170914/
Editors:
Matthias Kovatsch (Huawei)
Ryuichi Matsukura (Fujitsu Ltd.)
Michael Lagally (Oracle Corp.)
Toru Kawaguchi (Panasonic Corp.)
Kunihiko Toumura (Hitachi, Ltd.)
Kazuo Kajimoto (Former Editor, when at Panasonic)
Participate:
GitHub w3c/wot-architecture
File a bug
Commit history
Pull requests
Contributors:
In the GitHub repository

摘要

W3C Web of Things (WoT) は、IoT プラットフォームとアプリドメイン間の相互運用性を実現するために作られた。

WoTは、IoTデバイスおよびサービスが、その基本インプリメンテーションに影響されることなく、複数のネットワークプロトコルを超えて相互に通信できるように、IoTインターフェースを形式的に記述するメカニズムを提供するものである。さらに、WoTは、IoTビヘイビアを定義・プログラムするための標準方法も提供する。

このWoTアーキテクチャ仕様は、W3C WoTのアーキテクチャを抽象的に記述するものであり、複数のアプリドメインの使用事例から導き出された一連の要求事項をもとに作り上げられた。本アーキテクチャは、様々な具体的な展開シナリオにマッピングすることができ、そのいくつかのパターン例が提示されている。

本仕様では、いわゆるビルディングブロックに分解されるW3C WoT標準化のスコープに重点がおかれ、4つの初期WoTビルディングブロックを紹介している。これら４つのビルディングブロックはそれぞれの仕様において定義・詳細説明される。また、それらのインターワーキングも本仕様で解説される。

WoT Thing Descriptionは、モノのメタデータおよびネットワークインターフェースを詳述する中心的なビルディングブロックである。

情報WoTバインディングテンプレートでは、これらのネットワークインターフェースを記述するためのいわゆるプロトコルバインディングの定義方法に関するガイドラインを提供し、既存IoTエコシステムおよび標準例を提示している。

任意のWoTスクリプティングAPIは、WebブラウザAPIと同様の共通JavaScript APIを使用するモノのアプリロジックのインプ

リメンテーションを可能にし、IoTアプリ開発を簡素化し、ベンダおよびデバイス間のポータビリティを可能にする。

WoTセキュリティおよびプライバシーに関する考慮事項では、一つの分野横断的なビルディングブロックの解説をしているが、これはW3C WoTを実装するすべてのシステムに適用されるべきである。ここでは、モノの安全なインプリメンテーションおよび構成に焦点を当てる。

本仕様は、また、WoTシステム展開のための基準に基づかないアーキテクチャ上の側面および条件を解説している。これらのガイドラインは、展開シナリオとして説明される。

全体として、W3C WoTは、既存のIoT規格およびソリューションを維持し、補完することを目標としており、何を実装すべきかを規定するというよりは、何が存在するかを記述するために設計されている。

本文書のステータス

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

「編集者注記:W3C WoT WGへのフィードバックをお願いいたします。

WoTアーキテクチャリポジトリのGitHub Issue機能を使用して、本ドラフトへの投稿をお願いいたします。セキュリティとプライバシー考慮事項についてのフィードバックに関しては、全仕様にわたり分野横断的ですので、WoTセキュリティとプライバシー考慮事項リポジトリを使用して問題をファイル化してください。

本文書は、アーキテクチャ設計を抽象的に記述している。他に、関連するWoT TD仕様に基づいて具体的にインプリメンテーションを説明する事前インプリメンテーションレポートが用意されている。ワーキンググループでは、推奨フェーズ候補の終了基準として各機能で少なくとも2つのインプリメンテーションという要件を設定し、インプリメンテーションのフィードバックを求めている。ワーキンググループは、できれば、各機能について、1つのTDプロデューサと1つのTDコンシューマからレポートを取得することを目指している。インプリメンテーションの定義、TDプロデューサ、およびTDコンシューマなどの詳細については、事前インプリメンテーションレポート参照。

本文書は、WoTワーキンググループによって推奨候補として公開されたものである。本文書は、W3C推奨となるものである。

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

W3Cは、本文書が信頼できるものであると考えられていることを示し、開発者コミュニティによるインプリメンテーションを奨励するために、推奨候補を公開するものである。この推奨候補は、2019年6月13日以前に提案推奨として更新される予定である。

関連するWoT Thing Description仕様については、ワーキンググループの事前インプリメンテーションレポートを参照。

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

本文書は、W3C特許ポリシーに従い運営しているグループが作成したものである。W3Cでは、ワ

ーキンググループの成果物に関係する公的な特許公開リストを管理しており、そのページには

特許開示にあたっての指示も掲載されている。必須特許請求を含むと信じる特許について実知識を持つ者は、W3C特許ポリシー第6項に従って、その情報を開示しなければならない。

本文書は、2019 年3 月1 日のW3Cプロセス文書に準拠する。

もくじ

1. 機能要件
    1. 共通原則
    2. モノの機能
    3. 検索と検出
    4. 記述メカニズム
    5. 属性の記述
    6. 機能の記述
    7. ネットワーク
    8. 展開
    9. アプリ
    10. レガシー採用
2. 技術要件
    1. Web of ThingsとWeb of Thingsアーキテクチャを構成するコンポーネント
    2. デバイス
    3. アプリ
    4. デジタルツイン
    5. 検出
    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. 対話モデル
        1. プロパティ
        2. アクション
        3. イベント
    5. ハイパーメディアコントロール
        1. リンク
        2. フォーム
    6. プロトコルバインディング
        1. ハイパーメディア主導
        2. URI
        3. 標準メソッド
        4. メディアタイプ
    7. WoTシステムコンポーネントとその相互接続性
        1. 直接通信
        2. 間接通信

7. **WoTビルディングブロック**

1. WoT Thing Description

2.

8.

8.1

8.2

8.3

8.4

8.5

8.6

8.7

8.8

8.8.1

8.8.2

9.

9.1

9.2

9.2.1

9.2.2

9.2.2.1

9.2.2.2

9.2.3

9.2.4

9.2.5

9.2.5.1

9.2.5.2

10.

10.1

10.1.1

10.1.2

10.1.3

10.2

10.2.1

10.2.2

10.3

10.3.1

10.3.2

A.

WoTバインディングテンプレートWoTスクリプティングAPI

1. はじめに

WoT(Web of Things)の目標は、IoT(Internet of Things)の相互運用性および有用性を改善することである。過去数年間にわたる多くの関係者による共同作業を通じて、これらの課題に対処する多数のビルディングブロックが明らかになってきた。WoTビルディングブロックの最初のセットは、以下のように定義されている。

- Web of Things (WoT) Thing Description [wot-thing-description]
- Web of Things (WoT) バインディングテンプレート [wot-binding-templates]
- Web of Things (WoT) スクリプティングAPI [wot-scripting-api]
- Web of Things (WoT) セキュリティとプライバシー考慮事項 [wot-security]

本仕様は、W3C WoT仕様の包括であり、用語およびW3C WoTの基礎となる抽象アーキテクチャなどの基本事項を定義する。

特に、本仕様の目的は、以下を提供することである。

- 「第４項　使用事例」（W3C WoTアーキテクチャにつながる使用事例）
- 「第５項　要件」（WoTインプリメンテーションのための要件）
- 「第６項　WoTアーキテクチャ」（抽象アーキテクチャの定義）
- 「第７項　WoTビルディングブロック」（利用可能なWoTビルディングブロックと対話の概要）
- 「第８項　サービアントインプリメンテーション」（抽象アーキテクチャをソフトウェアスタック及びハードウェア構成部品にマッピングするためのガイドライン）
- 「第９項　WoT展開」（展開シナリオ）
- 「第１０項　セキュリティとプライバシー考慮事項」（W3CWoTアーキテクチャインプリメンテーション時の注意するセキュリティ考慮事項）

2. 適合性

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

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

3. 用語

本項は標準である。

本仕様では、ここで定義されているように、以下の用語を使用する。WoTという接頭辞が、Web of Things概念のために特に(再)定義されている用語の曖昧さを避けるために使用される。

アクション

状態を操作する(例えば、ランプをオンまたはオフに切り替える)、あるいは、モノのプロセスをトリガーする(例えば、次第にランプを暗くする) モノの関数の呼び出しを可能にする対話アフォーダンス。

バインディングテンプレート

異なるIoTプラットフォームとの通信のためのブループリントの再利用可能なコレクション。これらのブループリントは、WoT TDを介してプラットフォーム固有のメッセージに対話アフォーダンスをマッピングするための情報、ならびに、必要なプロトコルスタックまたは専用通信ドライバのインプリテーションノートを提供する。

モノの消費

TDドキュメントの構文解析および処理を行い、そこから、ローカルランタイム環境におけるアプリのためのインターフェースとして、消費されるモノのソフトウェアブストラクションを作成する。

消費されるモノ

ローカル・アプリによって使用されたリモートのモノを表すソフトウェアブストラクション。このアブストラクションは、ネイティブWoTランタイムによって作成されるか、あるいは、WoTスクリプティングAPIを介してオブジェクトとして事例を上げて裏付けられることがある。

コンシューマ

WoT TD(JSON ベースの表現形式を含む) を処理し、モノと対話する(つまり、モノを消費する) ことができるエンティティ。

デジタルツイン

デジタルツインは、クラウドまたはエッジノード上に存在するデバイスまたは一群のデバイスの仮想表現であり、連続的にオンライン状態でない可能性がある現実世界のデバイスを表すために、あるいは、新しいアプリおよびサービスのシミュレーションを、それらが現実のデバイスに展開される前に実行するために使用することができる。

ドメイン固有の語彙

WoT TDで使用できるがW3C WoTでは定義されていないリンクされたデータ語彙。

エッジデバイス

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

イベント

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

モノを公開する

モノの状態を管理し、ビヘイビアインプリテーションとのインターフェースとなるために、ローカルランタイム環境で公開されたモノのソフトウェアブストラクションを作成する。

公開されたモノ

リモートコンシューマがネットワーク上でアクセスできるローカルに提供されたモノを表すソフトウェアブストラクション。このアブストラクションは、ネイティブWoTランタイムによって作成されるか、またはWoTスクリプティングAPIを介してオブジェクトとして事例を上げて裏付けられることがある。

ハイパーメディアコントロール

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Webリンク[RFC8288]か、他の操作を実行するためのWebフォームのいずれかである。フォームは、コンシューマによって完成・送信される対象のモノが提示するリクエストテンプレートとしてみることができる。

対話アフォーダンス

コンシューマに実行可能な選択肢を提示・説明するモノのメタデータであり、コンシューマが対象のモノとどのように対話することができるかを示唆している。たくさんのタイプの潜在的なアフォーダンスがあるが、W3C WoTでは、3タイプの対話アフォーダンス、すなわち、プロパティ、アクション、イベントを定義している。第4の対話アフォーダン

スは、ナビゲーションであるが、リンク時点にウェブ上で利用可能となっている。

対話モデル

アプリの意図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、絞り込む仲介者アブストラクション。W3C WoTでは、定義された一連の対話アフォーダンスが対話モデルを構成する。

仲介者

モノを代理、増補、または構成し、元のモノの代わりに仲介者上のWoTインターフェースを示すWoTTDを再掲載することができるコンシューマとモノの間のエンティティ。コンシューマの場合、仲介者はRESTの階層構造制約のためモノと区別できないことがある。

IoTプラットフォーム

OCF、oneM2M、または、Mozilla Project Thingsなどの特定のIoTエコシステムで、アプリAPI、データモデル、および、プロトコルあるいはプロトコル構成に関する独自の仕様を保有する。

個人情報(PII)

特定の個人に関連付けることができる情報。

プライバシー

本システムは、個人情報の機密性を維持しなければならない。

特性

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

プロトコルバインディング

対話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をすることで、その対話アフォーダンスの起動方法をコンシューマに通知する。W3C WoTは、プロトコルバインディングをハイパーメディアコントロールとしてシリアライズする。

セキュリティ

本システムは、攻撃を受けやすい状況下にあっても、その整合性および機能性を維持しなければならない。

サービアント

WoTビルディングブロックを実装するソフトウェアスタック。サービアントは、モノを提供し、公開することができ、かつ/または、モノを消費するコンシューマを提供することができる。サービエントは、異なるIoTプラットフォームと対話できるようにするために複数のプロトコルバインディングをサポートすることができる。

サブプロトコル

対話可能状態を確認されていなければならない転送プロトコルへの拡張メカニズム。一例としては、HTTPのためのロングポーリングである。

TD

WoT Thing Description の省略形。

TD語彙

WoTバインディングテンプレートの通信メタデータなどWoTTD内のモノのメタデータにタグを付けるためのW3C WoTによる制御されたリンクデータ語彙。

モノまたはWeb Thing

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

モノディレクトリ

TD([CoRE-RD]と同様)を登録し、それらを検索(例えば、SPARQLクエリまたはCoRE RD検索インターフェース[CoRE-RD]を使用して)するためのウェブインターフェースを提供するTDのためのディレクトリサービス。

転送プロトコル

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

仮想のモノ

別のシステムコンポーネントに存在するモノを表すモノの実例。

WoT インターフェース

WoTTDによって記述されるモノのネットワークインターフェース。

WoTランタイム

アプリの実行環境を維持し、また、モノを公開および/あるいは消費し、WoTTDを処理し、プライベートセキュリティメタデータを維持管理し、プロトコルバインディングインプリメンテーションと接続する事ができるランタイムシステム。WoTランタイムは、独自のAPIを有するか、または任意のWoTスクリプティングAPIを使用する場合がある。

WoTスクリプティングAPI

WoTランタイム内で実行されるビヘイビアまたはアプリのインプリメンテーションを容易にするために、サービアントが提供するアプリプログラミングインターフェース。これは、ウェブブラウザAPIに当たる。WoTスクリプティングAPIは、W3C WoTのためのオプションのビルディングブロックである。

WoT サーバ

サービアントと同義。

WoT TDまたはTD

モノを記述する構造データ。WoT TDは、一般的メタデータ、ドメイン固有メタデータ、対話アフォーダンス(サポートされるプロトコルバインディングを含む)、および関連するモノへのリンクで構成されている。WoTTDフォーマットは、W3C WoTの中心的なビルディングブロックである。

4. 使用事例

本項は標準ではない。

本項では、W3C WoT対象であり、かつ、第7項WoT ビルディングブロックで考察される抽象アーキテクチャを導出するために使用されるアプリドメインおよび使用事例を提示する。

WoTアーキテクチャは、使用事例やアプリドメインに関し制限を置いていない。様々なアプリドメインが抽象アーキテクチャによって満たされなければならない共通パターンを収集するために考究されてきた。

以下に続く項は網羅的ではない。むしろ、接続されたモノが付加的な利益を提供する、あるいは、新たなシナリオを可能にすることができるという実例となっている。

1. アプリドメイン

1. コンシューマ

コンシューマ空間には、接続されることで利益を得る複数のアセットが存在する。部屋の占有率に基づいて、照明およびエアコンをオフにすることができ、ウィンドウブラインドは、気象条件および人の存在に基づいて自動的に閉じられる。エネルギーおよび他のリソース消費は使用パターンおよび予測に基づいて最適化することができる。

本項のコンシューマ使用事例にはスマートホーム使用事例が含まれている。

図1は、スマートホームの一例である。この場合、ゲートウェイは、KNX、ECHONET、ZigBee、DECT ULE、およびWi–SUNなどの対応するローカル通信プロトコルを介してセンサ、カメラ、および家電などのエッジデバイスに接続されている。複数のゲートウェイが1つの家庭で存在でき、各ゲートウェイは複数のローカルプロトコルをサポートすることができる。

ゲートウェイは、インターネットでクラウドに接続することができ、いくつかの器具がクラウドに直接接続することができる。クラウドで動作するサービスは、エッジデバイスからデータを収集し、そのデータを分析し、エッジデバイスおよび他のUXデバイスを介してユーザに価値を提供する。

図 1 スマートホーム

スマートホームは、リモートアクセスおよび制御、音声制御、ならびにホームオートメーションなどの便宜をコンシューマに提供する。又、スマートホームは、デバイス製造者によるデバイスをリモートで監視・維持管理も可能にする。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することができる。

2. 産業

本項の産業使用事例は、多様な業種に適用可能である。

アプリシナリオ内での重複という性質上、多様な業種で同様の使用事例が発生している。

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。ここでは、フィールドレベル、セル、および回線コントローラは、PROFINET、Modbus、OPC UA TSN、EtherCAT、またはCANなどの産業通信プロトコルに基づいて、異なる工場機器を自動化している。産業用エッジデバイスは、様々なコントローラから選択されたデータを収集し、例えば、ダッシュボードを介した遠隔監視ようにクラウドバックエンドサービスが利用できるようにする、あるいは、予防保守のために分析を行う。

図 2 スマートファクトリー

スマートファクトリーは、接続された製造装置及び製造された製品の高度な監視を必要とする。手痛いダウンタイム及びメンテナンス作業を防止するために、機会故障の予測及び異常の早期検出することで利益を得る。

さらに、接続された製造装置および生産設備の環境を、有害なガス、過剰なノイズまたは熱の存在を監視することは、作業者の安全性を高め、事故または事故のリスクを低減することになる。

生産設備のリアルタイム監視および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有益である。

3. 運輸・物流

車両、燃料コスト、保守ニーズ、および、割り当ての監視は、車両フリートのフル活用を最適化するのに役立つ。

輸送される物品の一貫した品質および状態を保証するために、輸送物はルート上で追跡することができる。これは、倉庫から冷蔵トラックそして納品までコールドチェーンの完璧性を表明するのに特に有用である。

倉庫およびヤードにおける在庫の集中監視および管理は、在庫切れおよび過剰な在庫状況を防ぐことができる。

4. ユーティリティ

住居用およびC&I (商業用および産業用)メータの自動読み取りと課金は、リソース消費および潜在的な問題に対する不断の洞察を提供する。

分散型再生可能エネルギー発生装置の状態および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。

配電装置の監視および遠隔制御は、配電プロセスの自動化に役立つ。

発電と配電インフラを継続的に監視することで、現場のユーティリティ作業員の安全性を向上させている。

5. 石油・ガス

オフショアプラットフォームの監視、パイプラインの漏れ検出および予測、ならびに、タンクおよび貯蔵施設のレベルの監

視および制御は、労働力ならびに環境の産業安全性を向上させるのに役立つ。

さまざまな貯蔵タンクおよび配送管・トラックを使った分散在庫の自動計算は、計画の向上および資源最適化を可能にする。

6. 保険

接続された構造物、フリート車両などの高価値資産のプロアクティブな資産監視は、インシデントの予測および早期検出することで深刻な損害および高コストのリスクを緩和する。

使用量ベースの保険は、使用追跡およびカスタマイズされた保険ポリシーを使って提供することができる。

予測気象監視および覆いのある車庫へのフリート車両のルート変更により雹や樹木による損傷が引き起こす損失を抑えることができる。

7. 土木建築

産業上の安全性を監視することにより、セキュリティハザードのリスクが軽減される。建設現場で資産を監視することで、被害や損失を防ぐことができる。

8. 農業

農産物の状態監視ばかりでなく、土地状態のモニタリングや水やりや施肥の最適計画策定は、農産物の品質と収穫を最適化する。

9. ヘルスケア

臨床試験データのデータ収集および分析は新領域への洞察を得るのに役立つ。

遠隔患者モニタリングは、高齢者および入院後の患者にとって検出されなかった深刻な状況リスクを軽減する。

10. 環境モニタリング

環境監視は、通常、共通のゲートウェイ、エッジデバイス、およびクラウドサービスに測定データを送信する多数の分散されたセンサに依存する。

深刻な環境条件を検出するために、大気汚染、水汚染、および微細なダスト、オゾン、揮発性の有機複合体、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は、回復不能な健全性または環境の損傷を防ぐことができる。

11. スマートシティ

橋、ダム、堤防、運河の材料状態、劣化、振動を監視することで保守修理作業の必要を検出し、重大な損害を防ぐ。高速道路を監視し、適切な標識を設置することにより、交通の流れを最適化することができる。

スマートパーキングは、パーキングスペースの使用および利用可能性を最適化および追跡し、請求/予約を自動化する。

人の存在検出、気象予測などに基づく街灯のスマート制御はコストを削減する。

ごみ容器は廃棄物管理運営およびごみ回収ルートを最適化するために監視することができる。

12. スマートビル

建物全体のエネルギー使用量監視は、資源消費を最適化と無駄の削減に役立つ。

HVAC、エレベータなどのビル内の機器を監視と問題の早期修正により、居住者の満足度が改善される。

13. コネクティッドカー

運用状況の監視、サービスニーズの予測により保守ニーズとコストが最適化される。運転者の安全性は危険な道路および交通状況の早期警報システムの通知によって高められる。

1. コネクティッドカーの例

図3は、コネクティッドカーの例である。ここでは、ゲートウェイがCANを介して自動車の部品に、また、車上のインターフェースを介してカーナビに接続されている。クラウド上のサービスは、車の部品から送られるデータを収集し、複数の車からのデータを分析して交通パターンを決定する。このゲートウェイは、また、クラウドサービスを利用して交通データを取得し、それをカーナビ上で運転者に通知する。

図 3 コネクティッドカー

運用状況の監視、サービスニーズの予測により、保守ニーズとコストが最適化される。運転者の安全性は、危険な道路および交通状況についての早期警報システムの通知によって高められる。

2. 共通パターン

本項では、デバイス/モノがコントローラ、他のデバイス、エージェント、およびサーバとどのように対話するかを示す一般的な使用事例パターンを紹介する。クライアント役という用語をトランスポートプロトコルのイニシエータとして使用し、サーバ役という用語をトランスポートプロトコルの受動コンポーネントとして使用する。しかし、個々のシステム構成要素に対して特定の役割が決められているということではない。一つのデバイスがクライアントおよびサーバの役を同時に果たすことができる。

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

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

1. デバイスコントローラ

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

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

図 4 デバイスコントロール

2. モノ対モノ

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

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

図 5 コントロールエージェント

3. リモートアクセス

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

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

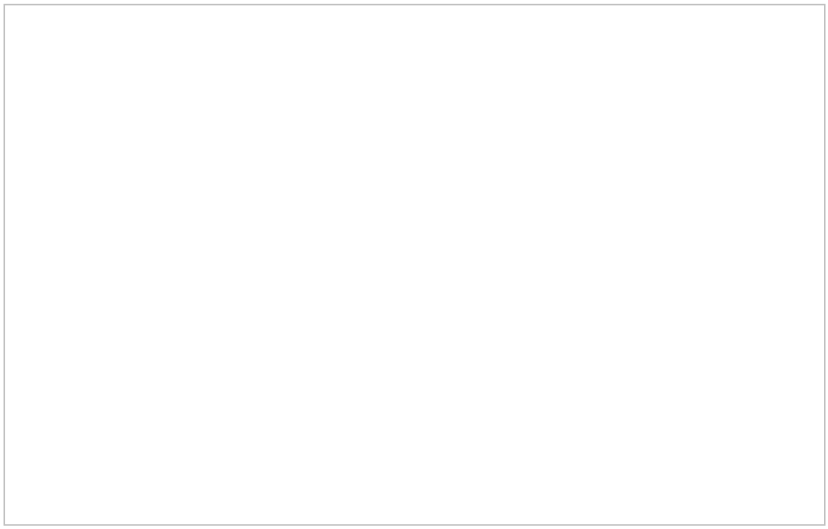

図 6 複数のネットワークインターフェース

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

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

図 7 スマートホームゲートウェイ

5. エッジデバイス

エッジデバイスあるいはエッジゲートウェイは、スマートホームゲートウェイに似ている。この用語は、エッジゲートウェイによって実行される追加のタスクを指すために使用される。図8のホームゲートウェイは、本来、公衆ネットワークと信頼できる

ネットワーク間を単にブリッジするだけであるが、エッジデバイスは、ローカルでの電算機計算能力を有し、通常、異なるプロトコルをブリッジする。エッジデバイスは、基本的に、産業ソリューションで使用され、接続されたデバイスとセンサによって提供されるデータの前処理、フィルタリング、および集約を行うことができる。

図 8 エッジデバイス

6. デジタルツイン

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

図9 デジタルツイン

デジタルツインは、単一のデバイスを作ることができ、または、組み合わされたデバイスの仮想表示において複数のデバイスを集約することができる。

図10 複数デバイスのためのデジタルツイン

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

1. クラウド対応デバイス

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

要画像挿入

図11 クラウド対応デバイス用機器ツイン

2. レガシーデバイス

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

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

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

要画像挿入

図12 レガシーデバイス用のデジタルツイン

7. マルチクラウド

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

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

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

8. クロスドメインコラボレーション

図13は、クロスドメインコラボレーションの例である。各システムは、スマートシティーとスマートファクトリー、スマートホームとスマートシティーというように、他のドメイン内の他のシステムを巻き込んでいる。この種のシステムは、[IEC-FOTF]に示されているように、「共生」エコシステムと呼ばれる。直接コラボレーションと間接コラボレーションの2つのコラボレーションモデルがあり、直接コラボレーションモデルでは、システムは、ピアツーピア方式で互いに直接情報を交換する。間接コラボレーションでは、システムは何らかのコラボレーションプラットフォームを介して情報を交換する。このコラボレーションを維持・継続するためには、各システムは、その能力およびインターフェースのメタデータを提供して他のシステムに順応する。

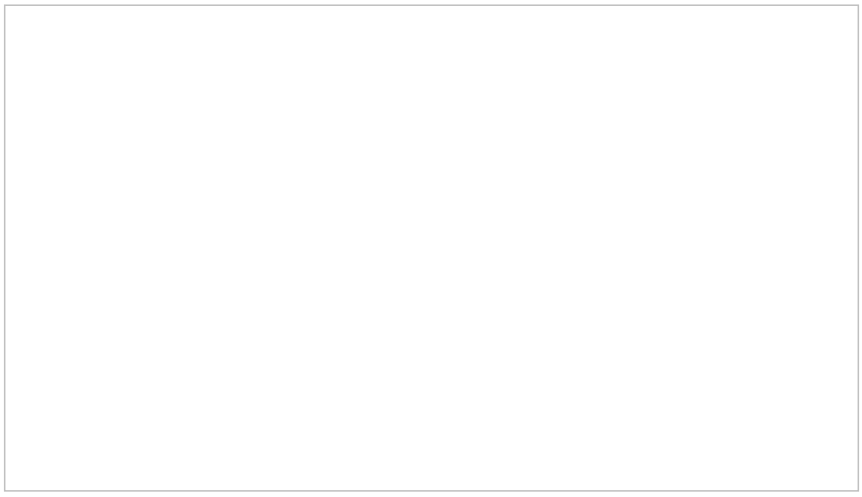

図13 クロスドメインコラボレーション

4.3 まとめ

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

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

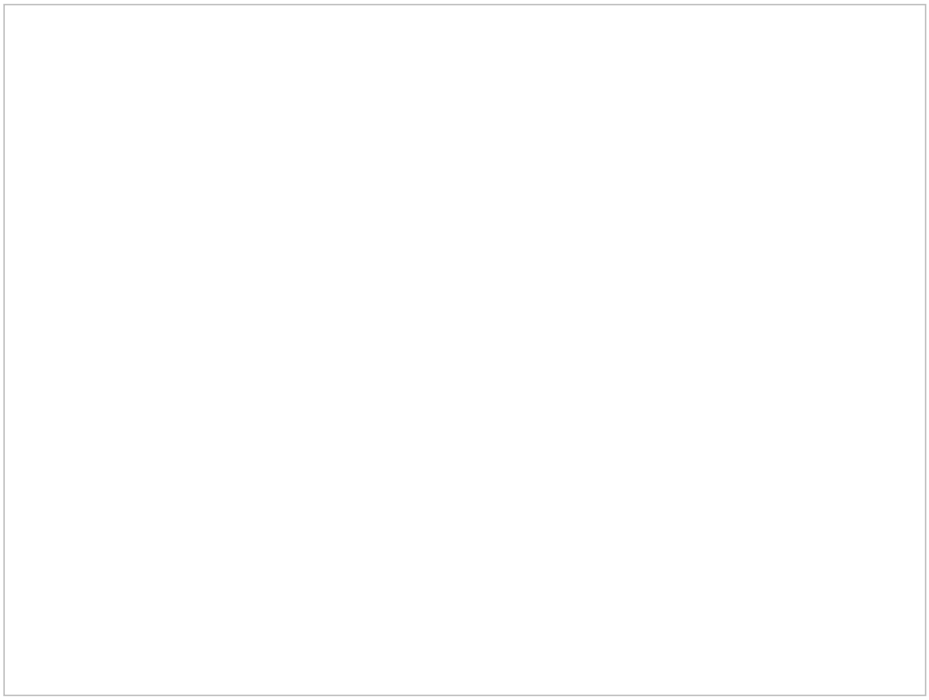

図14 使用事例の概要

5. 要件

本項は標準である。

1. 機能要件

本項は、抽象WoT(Web of Things)アーキテクチャで必要となるプロパティを定義する。

1. 共通原則

- WoTアーキテクチャは、Web技術を用いて異なるエコシステムの相互作用を可能にすべきである。
- WoT アーキテクチャは、RESTful API を使用したWeb アーキテクチャに基づくべきである。
- WoTアーキテクチャは、ウェブで一般的に使用される複数のペイロードフォーマットの使用が可能であるべきである。

- WoTアーキテクチャは、異なるデバイスアーキテクチャを使用可能にしなければならず、また、システムコンポーネントのクライアントあるいはサーバインプリメンテーションを強制してはならない。
- 柔軟性

    WoTインプリメンテーションのための多様な物理的デバイス構成が存在する。WoT抽象アーキテクチャは、すべてのバリエーションにマッピングされることができ、そのバリエーションすべてに対応することができなければならない。

- 互換性

  多くのビジネス分野において多くのIoTソリューションがあり、また、IoT標準化活動が進められている。WoTは、これらの既存の開発中のIoTソリューションとWoTコンセプトに基づくウェブ技術とをブリッジすべきである。WoTは、既存のIoTソリューションおよび現在の標準との上位互換性がなければならない。

- スケーラビリティ

  WoTは、数千〜数百万のデバイスを組み込んだIoTソリューションに対応できなければならない。これらのデバイスは、たとえ異なるメーカーによって製造されたものでも同じ機能を提供する。

- 相互運用性

  WoTは、デバイスメーカーとクラウドメーカーを超えた相互運用性を提供しなければならない。WoT対応デバイスを使って異なるメーカーのクラウドサービスに接続することが可能でなければならない。

2. モノの機能

- WoTアーキテクチャは、モノに下記の機能を与えること
- モノのステータス情報を読み取る
- 動作を引き起こすモノのステータス情報の更新
- モノのステータス情報への変更通知を規則的に受け取り、受信、配信停止
- 特定の動作または計算をトリガーする入出力パラメータを有する関数を呼び出す
- 単なる状態遷移レポートよりもより全般的なイベント通知の受け取り、受信、配信停止

3. 検索と検出

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

4. 記述メカニズム

- WoTアーキテクチャは、モノ及びそれらの機能の記述を可能にする共通の記述メカニズムをサポートすべきである。
- このような記述は、人間による読み取りが可能であるだけでなく、機械による読み取りが可能であるべきである。
- このような記述は、その構造および記述された内容の意味論的注釈付けを可能にすべきである。
- このような記述は、ウェブで一般に使用される複数のフォーマットを使用して交換できるべきである。

5. 属性の記述

- WoTアーキテクチャは、下記のようなモノの属性を記述することができるべきである。
- 名称

- 説明
- スペック、フォーマット、記述自体のバージョン
- その他の関連するモノへのリンクとメタデータ情報
- このような記述は、国際化をサポートすべきである。

6. 機能の記述

- WoTアーキテクチャは、第5.1.2項モノの機能に掲載されているモノの機能の記述を可能にすべきである。

7. ネットワーク

- WoTアーキテクチャは、一般に使用される複数のウェブプロトコルをサポートすべきである。

- そのようなプロトコルには、下記が含まれる。

1. インターネット上で一般的に使用されているプロトコル
2. ローカルエリアネットワーク上で一般的に使用されているプロトコル

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

8. 展開

WoTアーキテクチャは、同じモデルに基づいて、リソース制限のあるエッジデバイスやクラウド上の仮想物など、多様なモノの能力をサポートすべきである。

WoTアーキテクチャは、ゲートウェイおよびプロキシなどの仲介者エンティティを有する複数レベルの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変換を考慮して、ローカルネットワーク(インターネットあるいはもう一つのローカルネットワーク)外部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ある。

9. アプリ

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

10. レガシー採用

- WoTアーキテクチャは、レガシープロトコルが終了され変換される場合、様々なトポロジをサポートしているレガシーIPおよび非IPプロトコルのウェブプロトコルへのマッピングを可能にすべきである。
- WoTアーキテクチャは、RESTfulアーキテクチャに従う既存のIPプロトコルの変換することなくトランスペアレントな使用を可能にすべきである。
- WoT アーキテクチャは、デバイスおよびサービスに対してクライアントまたはサーバ役を強制してはならない。IoTデバイスは、システムアーキテクチャに従い、クライアントまたはサーバあるいはその両方になることができ、これは、エッジサービスおよびクラウドサービスについても同様である。

2. 技術要件

§ 4.2 共通パターンは様々な使用事例を提示し、アーキテクチャのコンポーネントを組み合わせるためのパターンを列挙することによって、WoT抽象アーキテクチャを定義している。本項では、抽象アーキテクチャから導き出された技術要件について説明する。

1. WoTとWoTアーキテクチャで構成されるコンポーネント

本使用事例は、デバイス、デバイス間に位置するプロキシ(すなわち、ゲートウェイやエッジデバイス)にアクセスし制御するデバイスやアプリなど基本的なコンポーネントの識別に有用である。いくつかの使用事例の中で有用な追加コンポーネントは検出を支援するディレクトリである。

これらのコンポーネントは、オフィス、工場、その他施設内のインターネットあるいはフィールドネットワークに接続される。それらコンポーネントはすべて、場合によっては単一のネットワークに接続されることがあるが、通常は、複数のネットワークにわたって展開することができることに留意されたい。

2. デバイス

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

一般的なメタデータには、デバイス識別子(URI)、シリアル番号、製造日、製造地、その他人間が読み取り可能な情報といったデバイス情報が含まれる。

情報モデルは、デバイス属性を定義し、デバイスの内部設定、制御機能、および通知機能を表すものである。同じ機能を有するデバイスは、使用されるトランスポートプロトコルにかかわらず、同じ情報モデルを有する。

WoTアーキテクチャに基づくシステムの多くはシステムドメインを横断するので、情報モデルで使用される語彙およびメタデータ(例えば、オントロジ)は、関係当事者が問題なく理解できるものであるべきである。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。

セキュリティ情報には、認証、認可、および安全な通信に関する記述が含まれている。デバイスは、TDをその内部または外部のいずれかに置き、TDへのアクセスを可能にして、他のコンポーネントがTDを検出しアクセスできるようにすること必要である。

3. アプリ

アプリは、メタデータ(記述)に基づいてネットワークおよびプログラムインターフェースを生成・使用することができなければならない。

アプリは、ネットワークを介してこれらの記述を取得することができなければならず、したがって、ネットワークを介して検索動作を実行し、必要な記述を取得することができなければならない。

4. デジタルツイン

デジタルツインは、メタデータ(記述)に基づいて内部的にプログラムインターフェースを生成し、それらプログラムインターフェースを使用して仮想デバイスを表すことが必要となる。ツインは仮想デバイスの記述を生成しそれが外部的に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、元デバイスとは異なる。これにより、仮想デバイスと元デバイスが別々のエンティティとして明確に認識されることが保証される。仮想デバイスのトランスポートおよびセキュリティメカニズムおよび設定は、必要に応じて元デバイスとは異なるものであってもよい。仮想デバイスは、ツインによって直接提供される記述を有するか、または外部ロケーションで記述を利用可能にする必要がある。いずれの場合も、他のコンポーネントが関連デバイスを検出し使用できるように、記述を利用可能にする必要がある。

5. 検出

デバイス、アプリ、ツインからアクセス可能であるデバイスおよび仮想デバイスのTDにとっては、TD共通の共有方法が必要となる。ディレクトリは、デバイスおよびツイン自体が自動的に、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によってこの要件を満たすことができる。

デバイスおよび仮想デバイスの記述は、外部エンティティが検索可能である必要がある。ディレクトリは、デバイス記述あるいは情報モデル内の一般的な記述から得るキーワードのようなサーチキーを使ってサーチオペレーションを処理することができなければならない。

6. セキュリティ

デバイスおよび仮想デバイスに関するセキュリティ情報は、デバイス記述に記述されることが必要とされる。これには、認証/認可およびペイロード暗号化のための情報が含まれる。

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

7. アクセシビリティ

WoTは、主に、マシン対マシンの通信をターゲットとする。関与する人間は、通常、モノをアプリに結合する開発者である。エンドユーザは、アプリのフロントエンド、またはデバイス自体によって提供される物理ユーザインターフェースを使用する。両方とも、W3C WoT仕様の範囲外である。本仕様はユーザではなくIoTに焦点を当てているため、アクセシビリティは直接的な要求事項ではなく、したがって本仕様では取り上げられていない。

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

6. WoTアーキテクチャ

本項は標準である。

第4項の使用事例に対処し、第5項の要件を満たすために、WoTは、いわゆるコンシューマが使用できるWeb Things (通常は単にモノと呼ばれる)の概念の上に構築される。本項では、W3C WoTアーキテクチャ全体を定義するための背景と標準アサーションについて説明する。WoTは、異なるドメインからの利害関係者に対処するので、ウェブ技術について、特に、ハイパーメディアの概念についてより詳細に説明する。

1. 概要

モノは、物理エンティティまたは仮想エンティティ(例えば、デバイスまたは部屋)の抽象であり、標準化されたメタデータによって記述される。W3C WoTでは、記述メタデータは、WoT TD [wot-thing-description]で**なければならない。**コンシューマは、JSON [RFC8259]に基づくTD表現フォーマットを解析し、処理で**きなければならない。**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-ld-syntax] と互換性があるため、フォーマットは従来のJSON ライブラリまたはJSON-LD プロセッサのいずれかで処理できる。TDは、インスタンス固有であり(すなわち、モノのタイプではなく、個々のモノを記述する)、モノのデフォルトの外部テキスト(ウェブ)表現である。HTMLベースのユーザインターフェース、単に物理的エンティティの画像、あるいはクローズドシステムにおける非ウェブ表現などでモノが表現**されることもある。**

しかし、モノであるためには、TD表現が少なくとも1つ利用可能**でなければならない。**WoTTDは、コンシューマがモノと対話するときにモノの能力を(意味論的注釈を介して)検出・解釈し、異なるインプリメンテーション(例えば、異なるプロトコルまたはデータ構造)に適応することを可能にし、それによって、異なるIoTプラットフォーム、すなわち、異なるエコシステムおよび標準との相互運用性を可能にする標準化されたマシン理解可能な表現フォーマットである。

図15 コンシューマとモノの対話

モノは、仮想エンティティの抽象とすることもできる。仮想エンティティは、1つまたは複数のモノ(例えば、いくつかのセンサおよびアクチュエータがある部屋)で構成される。その構成のオプションの一つは、仮想エンティティの能力のスーパーセットを持つ単一の統合されたWoT TDを提供することである。構成がやや複雑である場合、そのTDは、構成内の階層的にサブのモノにリンクすることができる。メインTDは、エントリポイントとととなり、一般的なメタデータおよび潜在的な包括的能力のみを記述する。これは、より複雑なモノの特定の様態のグループ化を可能にする。

リンクは、階層的なモノに適用されるだけでなく、モノと他のリソースとの関係にも適用される。リンクの関係タイプは、モノが、例えば、運動センサによって監視される光あるいは部屋を制御するスイッチなどとどのように関連するかを表す。モノに関連する他のリソースは、マニュアル、予備部品のカタログ、CADファイル、GUI、またはウェブ上のドキュメントとすることができる。概して、モノ同士のウェブリンクにより、人間と機械の両方にとってWoTのナビゲートが可能となる。これは、利用可能なモノのカタログを管理するモノのディレクトリをそれらのTD表現をキャッシュすることによって提供すれば、さらに促進される。つまり、WoT TDは、ウェブ上の他のモノや他のリソースにリンクして、モノのウェブを形成**するかもしれない。**

図16 リンクされたモノ

モノは、ネットワークインターフェース、すなわちモノのWoTインターフェースを介して対話を実現するために、ソフトウェアスタックを有するネットワーク化されたシステムコンポーネント上で提供されなければならない。一例としては、モノのアブストラクションの背後にある物理エンティティとインターフェースするセンサおよびアクチュエータを持つ組み込みデバイス上で実行されるHTTPサーバである。ただし、W3C WoTはモノが提供されている場所を指定しない。直接IoTデバイスかもしれないし、ゲートウェイなどのエッジデバイスの場合も、またはクラウド上にかもしれない。

代表的な展開上の課題は、通常はIPv4ネットワークアドレス変換(NAT)またはファイアウォールデバイスのために、ローカルネットワークにインターネットから入れないというシナリオである。この状況を打開するために、W3C WoTは、モノとコンシューマとの間の仲介者を許容している。

仲介者は、仲介者が提供するWoTインターフェースの方を向いている元のモノと同様のWoTTD を持っている場合、モノのプロキシとして作用することができる。仲介者は、また、既存のモノを追加の能力で増補するか、または複数の利用可能なモノから新しいモノを構成し、それによって仮想エンティティを形成することもできる。コンシューマにとって、仲介者は、WoTTDを持ちWoTインターフェースを提供しているのでモノのように見え、したがって、Web [REST]のようなレイヤードシステムアーキテクチャ上ではモノと区別できない可能性がある。WoTTD中の識別子は、同じ元のモノまたは最終的に一意の物理エンティティを表す複数のTDの相関を考慮**しなければならない。**

図17 仲介者

制限されたローカルネットワークに対する別の打開策は、WoTインターフェースを、ローカルネットワーク内のモノからに明らかに到達可能なコンシューマ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。

モノとモノの対話を可能にするために、モノはコンシューマとバンドル**してもよい。**通常、コンシューマビヘイビアは、モノのビヘイビアを実行もするソフトウェアコンポーネントに埋め込まれる。コンシューマビヘイビアの構成は、「モノ」を介して公開**されることになるかもしれない。**

W3C WoTの概念は、IoTアプリに関連するすべてのレベル、すなわち、デバイスレベル、エッジレベル、およびクラウドレベルにおいて適用可能である。これは、異なるレベルにわたって共通のインターフェースおよびAPIを促進し、IoTアプリのために、モノトモノ、モノとゲートウェイ、モノとクラウド、ゲートウェイとクラウド、更には、クラウドフェデレーション、すなわち、2つ以上のサービスプロバイダの相互接続されたクラウドコンピューティング環境といった多様な結合パターンを可能にする。図18は、上記で紹介されたWoT概念が、どのように適用され、組み合わされて、第4.3項要約の中で要約された使用事例に対応することができるのか、その概要である。

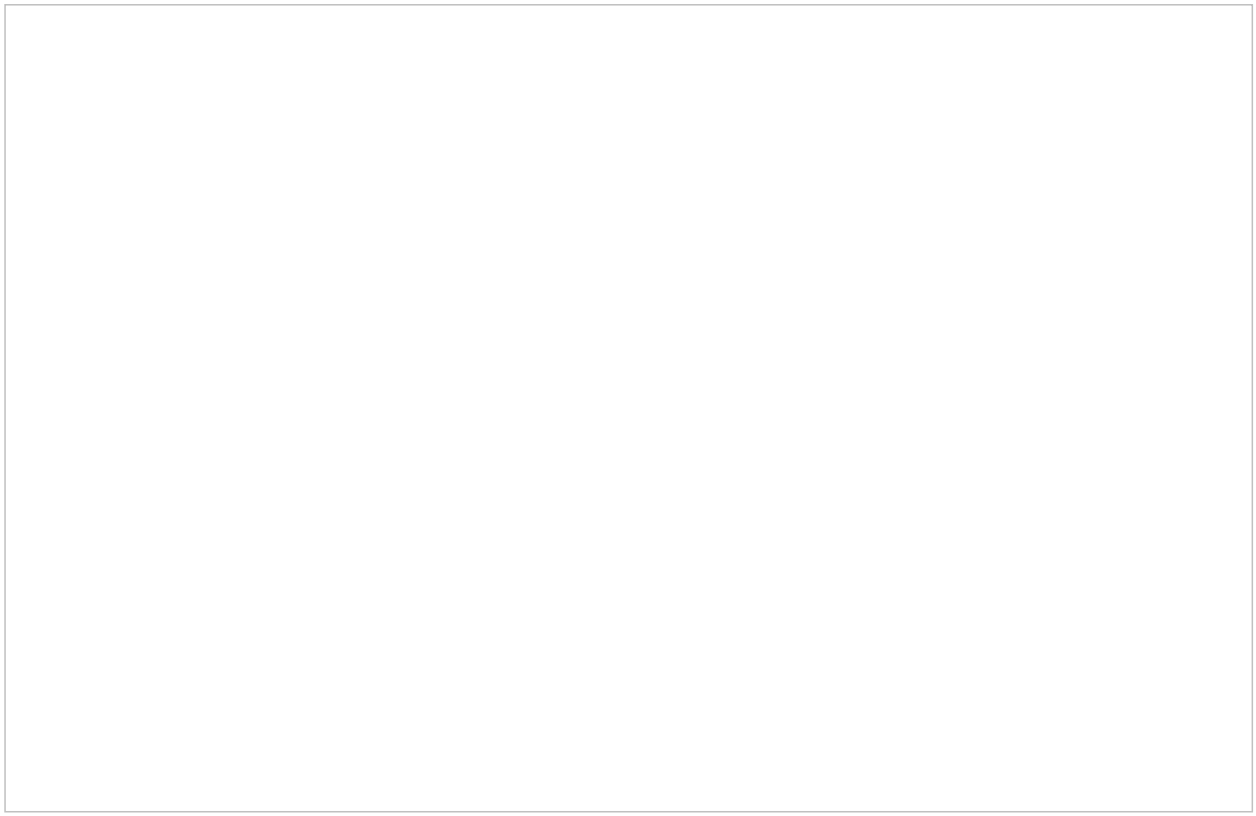

図18 W3C WoTの抽象アーキテクチャ

2. アフォーダンス

W3C WoTにおける中心的な側面は、マシン理解可能なメタデータ(すなわち、WoT TDの提供である。理想的には、そのようなメタデータは、コンシューマは、モノがどのような能力を提供するのか、提供された能力をどのように使用するのかを判断するこ

とができるように自己記述的である。この自己記述性の鍵は、アフォーダンスの概念にある。

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

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

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

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

全体として、このW3C WoT定義は、一般にウェブサービスに取り組んでいるRESTおよびマイクロサービスコミュニティは、もちろん物理的モノを作成するHCIや対話設計者と整合が取れている。

3. Web Thing

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

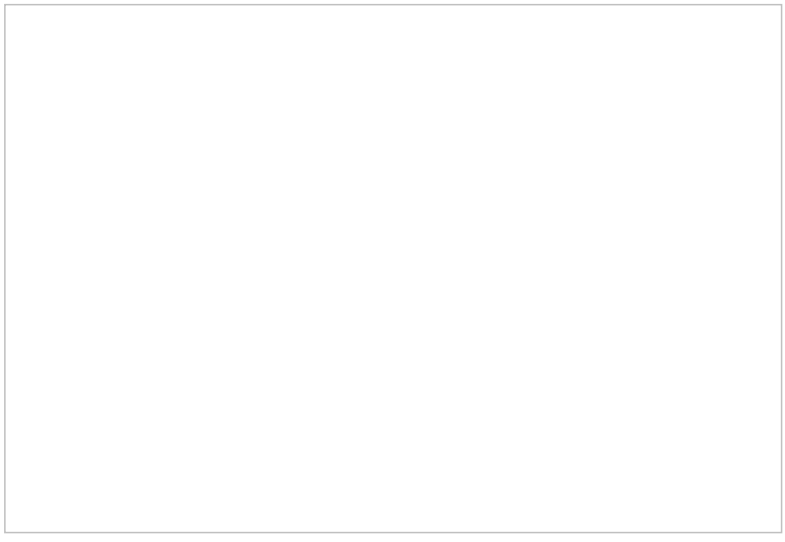

図19 モノのアーキテクチャ側面

4. 対話モデル

元来、ウェブリソースは、ウェブクライアントが簡単に取り出すことができるWorld Wide Web上のドキュメントを意味していた。ウェブサービスの導入により、リソースは、どのような種類のビヘイビアも実施することができるより総称的対話エンティティとなった。この非常に高いレベルのアブストラクションは、その多数の対話可能性ゆえに、アプリとリソース間の緩い結合を提供することが困難となっている。その結果、書込み時には、代表的なAPI記述は、アプリの意図からリソース・アドレス、メソッド、要求ペイロード構造、応答ペイロード構造、予想されるエラーまでの静的マッピングで構成される。このため、ウェブクライアントとウェブサービス間に緊密な結合が求められる。

このW3C WoTの対話モデルは、アプリの意図から具体的なプロトコル操作までのマッピング

を形式化し、対話アフォーダンスをどのようにモデル化できるかの可能性を制限する中間抽象化を導入している。

ナビゲーションアフォーダンス(すなわち、ウェブリンク)に加えて、モノは、本仕様によって定義される3つの他のタイプの対話アフォーダンス、すなわち、プロパティ、アクション、およびイベントを

提供**することもできる。**この範囲の狭さが、コンシューマとモノを分離することを可能にするが、これらの4つのタイプの対話アフォーダンスは、事実上、IoTデバイスおよびサービスに見られるすべての対話可能性をモデル化することもやはり可能である。

1. プロパティ

プロパティとは、モノの状態を公開する対話アフォーダンスである。プロパティによって公開される状態は、検索可能(読み取り可能)**でなければならない。**プロパティによって公開される状態は更新**することができる**(書き込み可能)。モノは、変更後に新しい状態をプッシュして、プロパティを観察可能にする選択**もできる**(リソースの観察 [RFC7641]参照)。書き込み専用状態は、アクションを介して更新されるべきである。

データが使用されるプロトコルバインディングによって(例えば、メディアタイプを介して)十分に指定されていない場合、プロパティは公開された状態のために1つのデータスキーマを持つ**ことができる。**

プロパティの例としては、センサ値(読み取り専用)、ステートフルアクチュエータ(読み取り–書き込み)、構成パラメータ(読み取り–書き込み)、モノのステータス(読み取り専用または読み取り–書き込み)、または、計算結果(読み取り専用)が挙げられる。

2. アクション

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

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

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

3. イベント

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

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

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

5. ハイパーメディアコントロール

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

W3C WoTでは、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ための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コントロールを利用している。リンクは、CoREリンクフォーマット[RFC6690]、OMA LWM2M [LWM2M]、OCF [OCF]など他のIoT規格およびIoTプラットフォームで既に使用されている。フォームは、W3C WoTに加えて、IETFによって定義されたConstrained RESTful Application Language (CoRAL) [CoRAL]も導入している新しい概念である。

1. リンク

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

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

- リンクコンテキスト、
- リレーションタイプ、
- リンクターゲット、
- 任意で、ターゲット属性

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

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

2. フォーム

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

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

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

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

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

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

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

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

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

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

表 1 Web of Things の周知操作タイプ

操作タイプ	説明
readproperty	対応するデータを取得するためにプロパティアフォーダンスで読み取り操作を識別する。
writeproperty	対応するデータを更新するためにプロパティアフォーダンスでの書き込み操作を識別する。
observeproperty	プロパティが更新されたときに新しいデータで通知を受けるためにプロパティアフォーダンスでオブザーブ操作を識別する。
unobserveproperty	対応する通知を停止するためにプロパティアフォーダンスで非オブザーブ操作を識別する。
invokeaction	対応するアクションを実行するためにアクションアフォーダンス でインボーク操作を識別する。
subscribeevent	イベントが発生したときにモからの通知を受けるためにイベントアフォーダンスでサブスクライブ操作を識別する。

unsubscribeevent 対応する通知を停止するためにイベントアフォーダンスサブスクライブ解除操作を識別する。

readallproperties 単一の対話ですべてのプロパティデータを取得するためにモノでreadallproperties 操作を識別する。

writeallproperties 単一の対話ですべての書き込み可能なプロパティのデータを更新するためにモノでwriteallproperties操作を識別する。

readmultipleproperties 単一のインタラクションで選択したプロパティのデータを取得するためにモノに対するreadmultipleperties 操作を識別する。

writemultipleproperties 単一のインタラクションで選択した書き込み可能プロパティのデータを更新するためにモノに対するwritemultipleperties 操作を識別する。

編集者注

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

6. プロトコルバインディング

プロトコルバインディングは、対話アフォーダンスから、HTTP [RFC7231]、CoAP [RFC7252]、MQTT [MQTT]などの特定のプロトコルの具象メッセージまでのマッピングである。これは、ネットワークインターフェースを介して、対話アフォーダンスをどのようにアクティブ化するかという情報をコンシューマ与えるものである。プロトコルバインディングは、相互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制約を遵守している。したがって、すべての通信プロトコルがW3C WoTのためのプロトコルバインディングを実行する資格を有するわけではなく、本要件は以下のアサーションにより提示されている。

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

1. ハイパーメディア主導

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

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

2. URIs

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

3. メソッドの標準セット

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

4. メディアタイプ

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

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

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

7. WoTシステムコンポーネントとその相互接続性

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

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

モノは、サービアントによって実施することができる。モノの中で、サービアントソフトウェアスタッ

クは、公開されたモノと呼ばれるモノの描出を含み、そのWoTインターフェースをモノのコンシューマが利用できるようにする。この公開されたモノは、サービアント上の他のソフトウェアコンポーネント(例えば、アプリ)が使用でき、モノのビヘイビアを実装することができる。

図 20 モノとしてのサービアント

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

コンシューマ側では、サービアントソフトウェアスタックは、消費されるモノと呼ばれるモノの描出

を提供し、それを、モノと対話するためにTDを処理する必要があるサービアント上で実行されるアプリが利用できるようにする。

図21 コンシューマとしてのサービアント

サービアントソフトウェアスタック内の消費されるモノインスタンスは、プロトコルレベルの複雑さ

をアプリから分離する役割を果たし、アプリに代わって公開されたモノと通信している。

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

図22 仲介者としての役割

1. 直接通信

図23は、TDを介して対話アフォーダンスを公開しているモノと、対話アフォーダンスによ

ってモノを使用するコンシューマとの直接通信を示している。直接通信は、両方のサービアントが同じネットワークプロトコルを使用し、互いにアクセス可能であるときに適用される。

図23 コンシューマとモノの高レベルアーキテクチャ

公開されたモノは、モノによって提供される対話アフォーダンスのWoTインターフェースであり、モ

ノのアブストラクションのソフトウェア表現である。

消費されるモノは、コンシューマによって消費されているリモートのモノのソフトウェア表現であり、

アプリのためのリモートのモノへのインターフェースとなる。コンシューマは、TDドキュメントを構

文解析および処理することによって、消費されるモノのインスタンスを生成することができる。コンシューマとモノとの対話は、コンシューマと公開されたモノの直接的なネットワーク接続を上でメッセージを交換するコンシューマと公開されたモノが行う。

2. 間接通信

図24において、コンシューマとモノは、仲介者を介して互いに接続している。サービアントが異

なるプロトコルを使用する場合、あるいは、認証を要求し、アクセス制御を提供する異なるネットワーク上にある場合(ファイアウォールなど)は、仲介者が必要となる。

図 24 仲介者を使用した高レベルアーキテクチャー

仲介者は、公開されたモノと消費されるモノの機能を組み合わせる。仲介者の機能は、コンシュ

ーマとモノとの間の対話アフォーダンスのためのメッセージを中継することと、任意でモノのデー

タをより速い応答のためにキャッシュすることとモノの機能を仲介者が拡張するるときに通信を

変換することなどである。仲介者側では、消費されるモノがモノの公開されたモノの代理オブジェクトを作成し、コンシューマは、それ自体の消費されるモノを介して代理オブジェクト(すなわち、仲介の公開されたモノ)にアクセスすることができる。

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

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

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

7. WoTビルディングブロック

本項は標準である。

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

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

図25 WoTビルディングブロックとモノのアーキテクチャ的側面との関係

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

1. WoT Thing Description

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

TDは、名前、ID、説明などの一般的なメタデータを有するモノのインスタンスを記述し、関係する

モノまたは他のドキュメントへのリンクを介して関係メタデータを提供することもできる。TDは、ま

た、パブリックセキュリティ構成メタデータとなる第6.4　対話モデルで定義された対話モデルに基づく対話アフォーダンスメタデータと、プロトコルバインディングを定義する通信メタデータとを含む。TDは、ハイパーメディアコントロールを使用して記述・提供されるサービスおよび関連するリソースを知るためのエントリポイントを提供するので、モノのindex.htmlとみなすことができる。

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

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

セマンティックな相互運用性のために、TDは、明らかな拡張ポイントが提供されるドメイン

固有の語彙を利用することができる。しかしながら、特定のドメイン固有語彙の開発は、現在、W3C WoT標準化活動の範囲外である。

潜在的に有用な外部IoT語彙の3つの例は、SAREF [SAREF]　iot.schema.org[iot-schema-org]、

W3C Semantic Sensor Network ontology [vocab-ssn]となっている。TDにおけるこのような外部語彙の使用は任意である。将来、追加のドメイン特有の語彙が開発され、TDと共に使用される可能性がある。

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

2. WoTバインドテンプレート

本項は標準ではない。

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

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

図26 バインディングテンプレートからプロトコルバインディングへ

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

プロトコルバインディングの通信メタデータには5つ要素がある。

- **IoT プラットフォーム:**

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

- **メディアタイプ:**

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

- **転送プロトコル:**

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

- **サブプロトコル:**

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

- **セキュリティ:**

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

3. WoTスクリプトAPI

本項は標準ではない。

WoTスクリプティングAPIは、WebブラウザAPIと同様のECMAScriptベースのAPI [ECMAScript]を提供することによってIoTアプ

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

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

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

4. WoTセキュリティとプライバシーのガイドライン

本項は標準ではない。

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

8. サービアントのインプリメンテーション

本項は標準ではない。

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

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

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

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

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

図27の各モジュールの役割および機能は、以下の項で説明される。

1. ビヘイビアのインプリメンテーション

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

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

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

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

2. WoTランタイム

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

あらゆるWoTランタイムは、ビヘイビアインプリメンテーションのためのアプリインターフェ

ース(すなわち、API)を持っている。図27のオプションWoTスクリプティングAPIビルディングブロッ

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

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

WoTランタイムは、ビヘイビアインプリメンテーションをプロトコルバインディングの詳細から切り

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

3. WoTスクリプトAPI

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

詳細は　[wot-scripting-api]参照。

4. 公開されたモノと消費されるモノのアブストラクション

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

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

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

WoTスクリプティングAPI を使用する場合、公開されたモノ および消費されるモノはJavaScript オブジェクトであり、アプリス

クリプトによって生成、操作、および破棄することができる。しかしながら、アクセスは、例えば、マルチテナントサービエントにおいて、セキュリティ構成により制限され得る。

5. プライベートセキュリティ構成

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

6. プロトコルスタックのインプリメンテーション

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

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

詳細は　[wot-binding-templates] 参照。

7. システムAPI

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

デバイスは、また、物理的にサービアントの外部にあってもよいが、独自のプロトコル、または、

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

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

8. 代替サービアントおよびWoTインプリメンテーション

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

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

1. ネイティブWoT API

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

この場合、WoTランタイムは依然として使用することができるが、WoTスクリプティングAPIの代わ

りに、代替のアプリインターフェースを使用して公開される同等のアブストラクションおよび機能性が必要となる。その点を除き、第８項　サービアントインプリメンテーションのブロック記述はすべて図28でも有効である。

図28 ネイティブWoT API を使用したサービアントのインプリメンテーション

2. TDで記述される既存デバイス

既存のIoTデバイスまたはサービスをモノのW3Cウェブに結合し、これらのデバイスまたはサービスのTDを作成することによって、それらをモノとして使用することも可能である。このようなTDは、手動で、または、ツールまたはサービスを使って作成することができる。例えば、TDは、別のエコシステム依存の機械読み取り可能フォーマットによって提供されるメタデータの自動翻訳を提供するサービスによって生成することができる。ただし、これは、ターゲットデバイスがプロトコルバインディングを使用して記述できるプロトコルを使用している場合にのみ行うことができる。その要件は、第6.6項　プロトコルバインディングで述べられている。前述の考察の多くが、モノそれ自体のTDを提供することを意味する。これは有用なパターンであるが、必須ではない。特に、そのTDを直接提供するために既存のデバイスを改修することはできないであろう。この場合、TDは、ディレクトリまたは他の外部的な別個の配布メカニズムなどのサービスを使用して別途提供されなければならない。

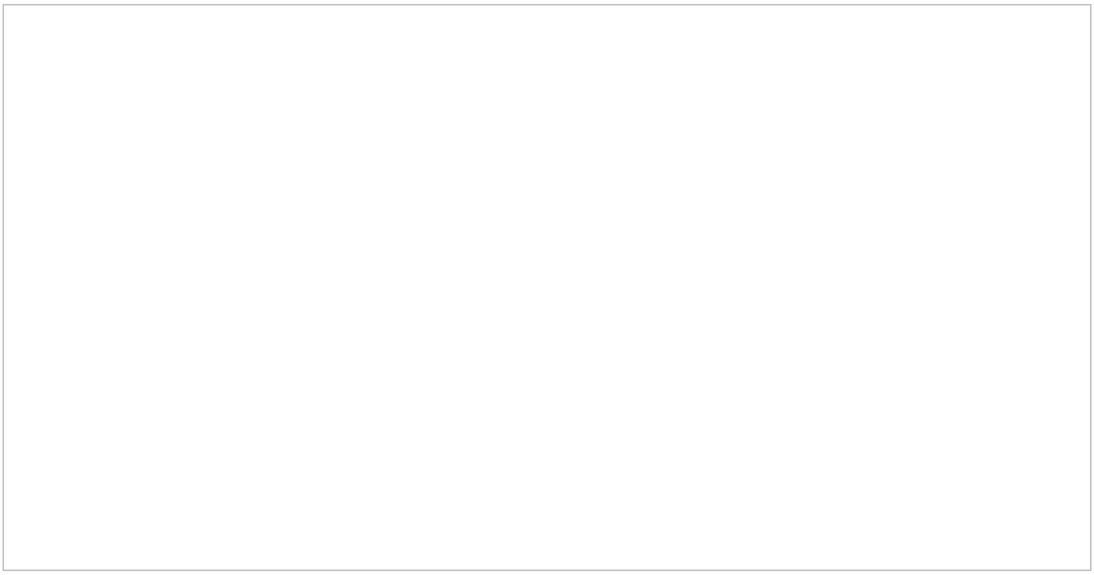

図29 既存IoT デバイスの W3C WoT への結合

9. WoTの導入

本項は標準ではない。

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

特定のトポロジを論じる前に、まず、WoTネットワークにおいてモノとコンシューマが果たすこと

ができる役割、および、公開されたモノと消費されるモノのソフトウェアブストラクションとの関係を検討する。公開されたモノと消費されるモノは、サービアントのビヘイビアインプリメンテーションがそれぞれ、モノおよびコンシューマ役として内部的に利用できる。

1. モノとコンシューマの役割

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

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

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

モノは、必ずしも物理的デバイスを表すということではない。モノは、ゲートウェイまたはクラウ

ド上で実行されるデバイスまたは仮想サービスの集合を表すこともできる。同様に、コンシュ

ーマは、ゲートウェイまたはクラウド上で実行されるアプリまたはサービスを代理することができる。コンシューマをエッジデバイス上に実装することもできる。仲介者側では、単一のサービアントが、単一のWoTランタイムを共用しているモノとコンシューマ両方の役割を同時に実行する。

2. WoTシステムのトポロジと展開シナリオ

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

1. 同一ネットワーク上のコンシューマとモノ

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

図30同一ネットワーク上にあるコンシューマとモノ

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

2. 仲介者を介して接続されたコンシューマとモノ

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

1. プロキシとしての仲介者

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

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

図31 プロキシとして仲介者を介したコンシューマとモノの接続

2. デジタルツインとしての仲介者

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

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

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

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

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

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

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

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

4. モノディレクトリを使用した検出

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

しかしながら、リモート仲介者が、クライアントアプリ用クラウドプラットフォームの一部である場合、クライアントは、例えば、接続されたデバイスのディレクトリにアクセスすることによって、デバイス情報を見つけることができる必要が出てくる。

以下の図では単純化のために、すべてのローカルデバイスがモノとして実装され、すべてのクラウドアプリがコンシューマとして実装されていると想定した。モノとして実装されたローカルデバイスのメタデータをクラウドアプリが利用できるようにするために、それらのメタデータはモノディレクトリサービスに登録することができる。このメタデータは、具体的には、リモート仲介者によって提供されるセキュリティ構成および通信メタデータを反映するように修正されたローカルデバイスのTDである。クライアントアプリは、モノディレクトリに問い合わせることによってその機能を達成するためにローカルデバイスと通信するのに必要なメタデータを取得することができる。

図33 モノディレクトリ を使用したクラウドサービス

図には示されていないが、より複雑な状況では、モノとして働くクラウドサービスも存在し得る。これらは、モノディレクトリにそれ自身を登録することもできる。モノディレクトリはウェブサービスで

あるため、NATまたはファイアウォールデバイスを介してローカルデバイスが見ることができなければならず、そのインターフェースには、それ自体のTDを設けることさえできる。コンシューマとして働くローカルデバイスは、モノディレクトリを介してクラウド内のモノを検出し、直接に、または

、例えば、プロトコル変換が必要な場合、ローカル仲介者を介してモノに接続することができる。

5. 複数ドメイン間のサービス対サービス接続

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

1. ディレクトリ同期による接続

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

図34 ディレクトリ同期による複数のクラウド接続

2. プロキシ同期による接続

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

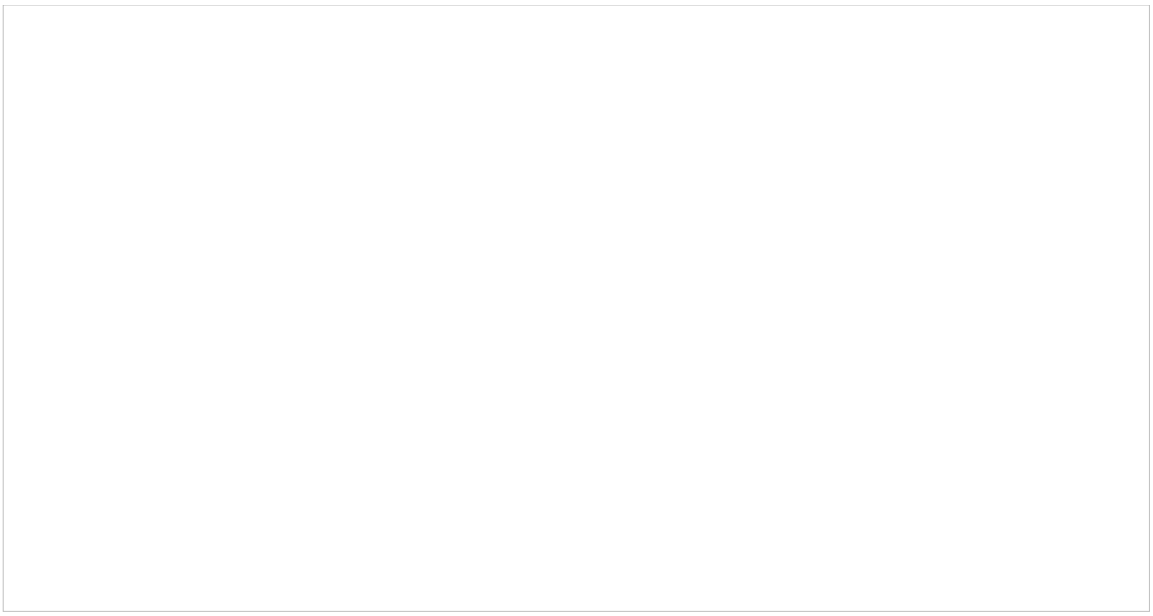

図35 仲介者同期による複数のクラウド接続

10. セキュリティとプライバシーに関する考慮事項

このセクションは標準ではない。

セキュリティは、すべてのWoTビルディングブロックおよびWoTインプリメンテーションにおいて考慮される必要がある分野横断的な問題である。本章では、WoTインプリメンテーションのセキュリティとプライバシーを保護するのに役立ついくつかの一般的な問題とガイドラインを要約する。セキュリティとプライバシーの問題のより詳細で完全な分析については、WoTセキュリティとプライバシーポリシー考慮事項仕様[wot-security] 参照。

全体として、WoTの目標は、セキュリティを含みIoTデバイスおよびサービスの既存のアクセスメカニズムおよびプロパティを記述することである。一般に、W3C WoTは、何を実装すべきかを規定するのではなく、何が存在するかを記述するように設計されている。

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

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

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

1. WoTDリスク

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

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

1. TDのプライベートセキュリティデータリスク

TDはパブリックセキュリティデータのみを持つように設計されている。TDの制作者は、プライベートセキュリティ情報が一切TDに含まれていないことを保証しなければならない。パブリックセキュリティデータとプライベートセキュリティデータは厳密に分離されるべきである。TDは、コンシューマに権限が与えられている場合に限りシステムにアクセスするために何をする必要であるかを理解させるパブィックセキュリティ情報のみを持つべきである。許可は、別個に管理されたプライベート情報に置かれるべきである。

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

軽減対策:

TDおよびTDでの使用を意図した拡張の作成者は、プライベートセキュリティデータがTDに格納されていないことを保証しなければならない。

2. TDの個人情報リスク

TDは、潜在的に、様々なタイプの個人情報を含むことができる。明示的でない場合であっても、TD内のセマンティック情報の存在およびその人物との関連付けは、その人物に関する情報を推論するために使用可能である。例えば、位置を判別できるモバイルデバイスによって露出される特異な識別可能TDの関連付けは、追跡リスクとなりうる。

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

軽減対策:

TDへのPIIの保存は、可能な限り最小限にすべきである。TD内に明示的なPIIがなくても、追跡および識別プライバシーリスクが存在する可能性がある。このリスクを最小限に抑えるために、TDは、一般的に、あたかもPIIを持つと捉え、他のPIIと同じ管理ポリシー遵守すべきである。許可されたコンシューマにのみ提供されるべきである。

3. TD通信メタデータリスク

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

軽減対策:

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

2. WoTスクリプティングAPIのセキュリティとプライバシーのリスク

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

1. クロススクリプトセキュリティとプライバシーリスク

基本的なWoTセットアップでは、WoTランタイム内で実行されるすべてのスクリプトは、信頼できるるものであるとみなされ、製作業者によって配布される。したがって、絶対に各実行中のスクリプトインスタンス間で厳密な分離を実行するという必要はない。ただし、デバイスの機能、展開使用事例シナリオ、リスクレベルによっては、そうすることが望ましい場合がある。例えば、1つのスクリプトが、機密のプライバシー関連のPIIデータを処理し、十分に審査されている場合、同じシステム内の他のスクリプトがランタイム中に危険にさらされる場合に、データ露出リスクを最小限に抑えるために、スクリプトインスタンスを残りのスクリプトインスタンスから分離することが望ましい場合がある。別の例としては、単一のWoTデバイス上の異なるテナントの相互共存がある。この場合、各WoTランタイムインスタンスは異なるテナントを提供することになり、それらの間の分離が必要とされる。

軽減対策:

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

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

2. 物理デバイスダイレクトアクセスセキュリティとプライバシーリスク

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

軽減対策:

WoTランタイムは、ネイティブデバイスインターフェースをスクリプト開発者に直接公開すること

を避けるべきである。その代わりに、WoTランタイムインプリメンテーションは、ネイティブデバイスインターフェースにアクセスするためのハードウェアブストラクションレイヤを提供すべきである。そのようなハードウェアブストラクションレイヤは、デバイス(または環境)を安全でない状態に晒す可能性があるコマンドの実行を拒否するべきである。さらに、スクリプトが危険にさらされる場合に物理WoTデバイスへの損害を低減するために、その機能に基づいた特定のスクリプトに公開されるかまたはアクセス可能なインターフェースの数を最小限に抑えることが重要である。

2. WoTランタイムセキュリティとプライバシーリスク

1. セキュリティリスクのプロビジョニングと更新

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

軽減対策:

WoTランタイム自体、または関連データのスクリプト製作後のプロビジョニングまたは更新は安全な方法で行われるべきで

ある。安全な更新と製作後のプロビジョニングに関する推奨事項は、WoTセキュリティとプライバシー考慮事項仕様 [wot-security] に掲載されている。

2. セキュリティ証明書ストレージのセキュリティとプライバシーリスク

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

軽減対策:

WoTランタイムは、プロビジョニングされたセキュリティ証明書を安全に格納し、それらの完

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

A. 最近の仕様変更

第一回公開作業原案からの変更

- 使用事例の改訂・拡大
- 再編成された要件
- 抽象アーキテクチャの定義
- 用語の改訂と明確化
- インプリメンテーションおよびデプロイメントへの追加
- セキュリティとプライバシーの考慮事項追加

以前の出版物

- 2017年9月:第1回公開作業原案
- 2019年3月:TAGレビューのための公開作業原案。

B. 謝辞

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

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

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

C. References

C.1 Normative references

[IANA-RELATIONS]
Link Relations. IANA. URL: https://www.iana.org/assignments/link-relations/
[MQTT]
MQTT Version 3.1.1. OASIS Standard. 2014.
[RFC2119]
Key words for use in RFCs to Indicate Requirement Levels. S. Bradner. IETF. March 1997. Best Current Practice. URL: https://tools.ietf.org/html/rfc2119
[RFC3986]
Uniform Resource Identifier (URI): Generic Syntax. T. Berners-Lee; R. Fielding; L. Masinter. IETF. January 2005. Internet Standard. URL: https://tools.ietf.org/html/rfc3986
[RFC3987]
Internationalized Resource Identifiers (IRIs). M. Duerst; M. Suignard. IETF. January 2005. Proposed Standard. URL: https://tools.ietf.org/html/rfc3987
[RFC4395]
Guidelines and Registration Procedures for New URI Schemes. T. Hansen; T. Hardie; L. Masinter. IETF. February 2006. Best Current Practice. URL: https://tools.ietf.org/html/rfc4395
[RFC5234]
Augmented BNF for Syntax Specifications: ABNF. D. Crocker, Ed.; P. Overell. IETF. January 2008. Internet Standard. URL: https://tools.ietf.org/html/rfc5234
[RFC6838]
Media Type Specifications and Registration Procedures. N. Freed; J. Klensin; T. Hansen. IETF. January 2013. Best Current Practice. URL: https://tools.ietf.org/html/rfc6838
[RFC7049]
Concise Binary Object Representation (CBOR). C. Bormann; P. Hoffman. IETF. October 2013. Proposed Standard. URL: https://tools.ietf.org/html/rfc7049
[RFC7231]
Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content. R. Fielding, Ed.; J. Reschke, Ed.. IETF. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7231
[RFC7252]
The Constrained Application Protocol (CoAP). Z. Shelby; K. Hartke; C. Bormann. IETF. June 2014. Proposed Standard. URL: https://tools.ietf.org/html/rfc7252
[RFC8174]
Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words. B. Leiba. IETF. May 2017. Best Current Practice. URL: https://tools.ietf.org/html/rfc8174
[RFC8259]
The JavaScript Object Notation (JSON) Data Interchange Format. T. Bray, Ed.. IETF. December 2017. Internet Standard. URL: https://tools.ietf.org/html/rfc8259
[RFC8288]
Web Linking. M. Nottingham. IETF. October 2017. Proposed Standard. URL: https://tools.ietf.org/html/rfc8288
[wot-thing-description]
Web of Things (WoT) Thing Description. Sebastian Käbisch; Takuki Kamiya. W3C. 21 October 2018. W3C Working Draft. URL: https://www.w3.org/TR/wot-thing-description/

C.2 Informative references

[CoRAL]
The Constrained RESTful Application Language (CoRAL). Klaus Hartke. IETF. March 2019. Internet-Draft. URL: https://tools.ietf.org/html/draft-hartke-t2trg-coral
[CoRE-RD]
CoRE Resource Directory. IETF. 11 March 2019. Internet-Draft. URL: https://tools.ietf.org/html/draft-ietf-core-resource-directory-20
[ECMAScript]
ECMAScript Language Specification. Ecma International. URL: https://tc39.github.io/ecma262/
[EVENTSOURCE]
Server-Sent Events. Ian Hickson. W3C. 3 February 2015. W3C Recommendation. URL: https://www.w3.org/TR/eventsource/
[HCI]
The Encyclopedia of Human-Computer Interaction, 2nd Ed. Interaction Design Foundation. 2013.
[IEC-FOTF]
Factory of the future. IEC. October 2015. URL: https://www.iec.ch/whitepaper/pdf/iecWP-futurefactory-LR-en.pdf
[iot-schema-org]
iot.schema.org. URL: https://iot.schema.org/
[json-ld-syntax]
JSON-LD 1.0. Manu Sporny; Gregg Kellogg; Markus Lanthaler. W3C. 16 January 2014. W3C Recommendation. URL: https://www.w3.org/TR/json-ld/
[JSON-LD11]
JSON-LD 1.1. Gregg Kellogg; Pierre-Antoine Champin. W3C. 10 May 2019. W3C Working Draft. URL: https://www.w3.org/TR/json-

ld11/
[LINKED-DATA]
Linked Data Design Issues. Tim Berners-Lee. W3C. 27 July 2006. W3C-Internal Document. URL: https://www.w3.org/DesignIssues/LinkedData.html
[LWM2M]
Lightweight Machine to Machine Technical Specification: Core. OMA SpecWorks. August 2018. Approved Version: 1.1. URL: http://openmobilealliance.org/release/LightweightM2M/V1_1-20180710-A/OMA-TS-LightweightM2M_Core-V1_1-20180710-A.pdf
[NORMAN]
The Psychology of Everyday Things. Basic Books. 1988.
[OCF]
OCF Core Specification. Open Connectivity Foundation. April 2019. Version 2.0.2. URL: https://openconnectivity.org/developer/specifications
[REST]
REST: Architectural Styles and the Design of Network-based Software Architectures. University of California, Irvine. 2000. PhD thesis.
[RFC4301]
Security Architecture for the Internet Protocol. S. Kent; K. Seo. IETF. December 2005. Proposed Standard. URL: https://tools.ietf.org/html/rfc4301
[RFC6202]
Known Issues and Best Practices for the Use of Long Polling and Streaming in Bidirectional HTTP. S. Loreto; P. Saint-Andre; S. Salsano; G. Wilkins. IETF. April 2011. Informational. URL: https://tools.ietf.org/html/rfc6202
[RFC6347]
Datagram Transport Layer Security Version 1.2. E. Rescorla; N. Modadugu. IETF. January 2012. Proposed Standard. URL: https://tools.ietf.org/html/rfc6347
[RFC6690]
Constrained RESTful Environments (CoRE) Link Format. Z. Shelby. IETF. August 2012. Proposed Standard. URL: https://tools.ietf.org/html/rfc6690
[RFC6749]
The OAuth 2.0 Authorization Framework. D. Hardt, Ed.. IETF. October 2012. Proposed Standard. URL: https://tools.ietf.org/html/rfc6749
[RFC7641]
Observing Resources in the Constrained Application Protocol (CoAP). K. Hartke. IETF. September 2015. Proposed Standard. URL: https://tools.ietf.org/html/rfc7641
[RFC7744]
Use Cases for Authentication and Authorization in Constrained Environments. L. Seitz, Ed.; S. Gerdes, Ed.; G. Selander; M. Mani; S. Kumar. IETF. January 2016. Informational. URL: https://tools.ietf.org/html/rfc7744
[RFC8446]
The Transport Layer Security (TLS) Protocol Version 1.3. E. Rescorla. IETF. August 2018. Proposed Standard. URL: https://tools.ietf.org/html/rfc8446
[SAREF]
Smart Appliances REFerence (SAREF) ontology. ETSI. November 2015. URL: https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology
[vocab-ssn]
Semantic Sensor Network Ontology. Armin Haller; Krzysztof Janowicz; Simon Cox; Danh Le Phuoc; Kerry Taylor; Maxime Lefrançois. W3C. 19 October 2017. W3C Recommendation. URL: https://www.w3.org/TR/vocab-ssn/
[wot-binding-templates]
Web of Things (WoT) Protocol Binding Templates. Michael Koster. W3C. 5 April 2018. W3C Note. URL: https://www.w3.org/TR/wot-binding-templates/
[wot-pioneers-1]
Mobile Service Interaction with the Web of Things. E. Rukzio, M. Paolucci; M. Wagner, H. Berndt; J. Hamard; A. Schmidt. Proceedings of 13th International Conference on Telecommunications (ICT 2006), Funchal, Madeira island, Portugal. May 2006.
[wot-pioneers-2]
Putting Things to REST. Erik Wilde. UCB iSchool Report 2007-015, UC Berkeley, Berkeley, CA, USA. November 2007.
[wot-pioneers-3]
Poster Abstract: Dyser – Towards a Real-Time Search Engine for the Web of Things. Benedikt Ostermaier; B. Maryam Elahi; Kay Römer; Michael Fahrmair; Wolfgang Kellerer. Proceedings of ACM SenSys 2008, Raleigh, NC, USA. November 2008.
[wot-pioneers-4]
A Resource Oriented Architecture for the Web of Things. Dominique Guinard; Vlad Trifa; Erik Wilde. Proceedings of Internet of Things 2010 International Conference (IoT 2010). Tokyo, Japan. November 2010.
[wot-scripting-api]
Web of Things (WoT) Scripting API. Zoltan Kis; Kazuaki Nimura; Daniel Peintner; Johannes Hund. W3C. 29 November 2018. W3C Working Draft. URL: https://www.w3.org/TR/wot-scripting-api/
[wot-security]
Web of Things (WoT) Security and Privacy Considerations. Elena Reshetova; Michael McCool. W3C. 3 December 2018. W3C Note. URL: https://www.w3.org/TR/wot-security/
[Y.4409-Y.2070]
ITU-T Rec. Y.4409/Y.2070 (01/2015) Requirements and architecture of the home energy management system and home network services . ITU-T. January 2015. Recommendation.

↑