IoTのためのWebAPI
要件定義書

概要

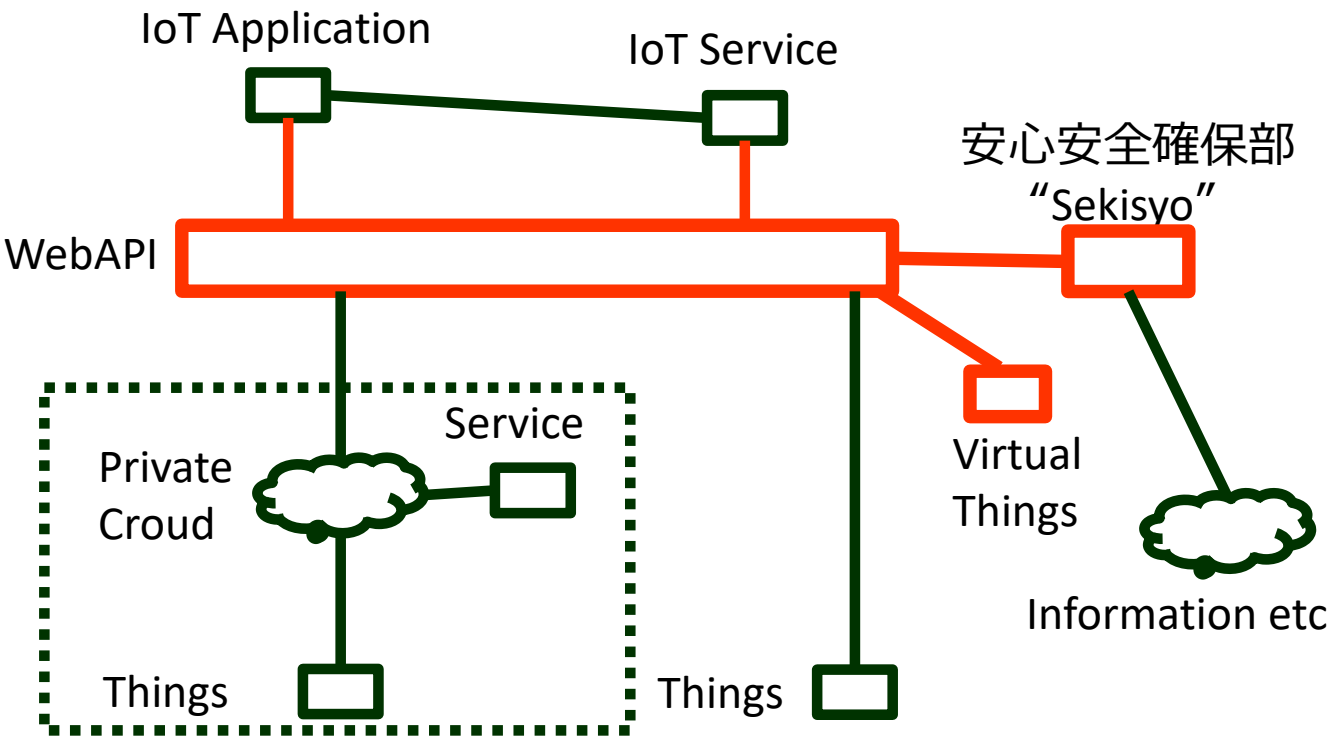
本要件定義書の目的

IoT社会の実現に向けて、サイロ化するIoTインフラに対し、

① **Interconnectivity（相互接続性）**
② **Interoperability（相互運用性）**
③ **相互に接続した際の安全性**

を提供するためのIXインフラとしてのWebAPIについて述べる。

凡例
本書の定義範囲
本書の定義範囲外



Things(モノ)からサービスが生まれるのはよいことであるが、サイロ化してしまうこともある。
自社製品内で安全を確保できる代わりに
相互接続性や相互運用性の阻害要因となっている。
WebAPI を共通のIXとして活用することで解決が可能

点線：サイロ化モデルの例

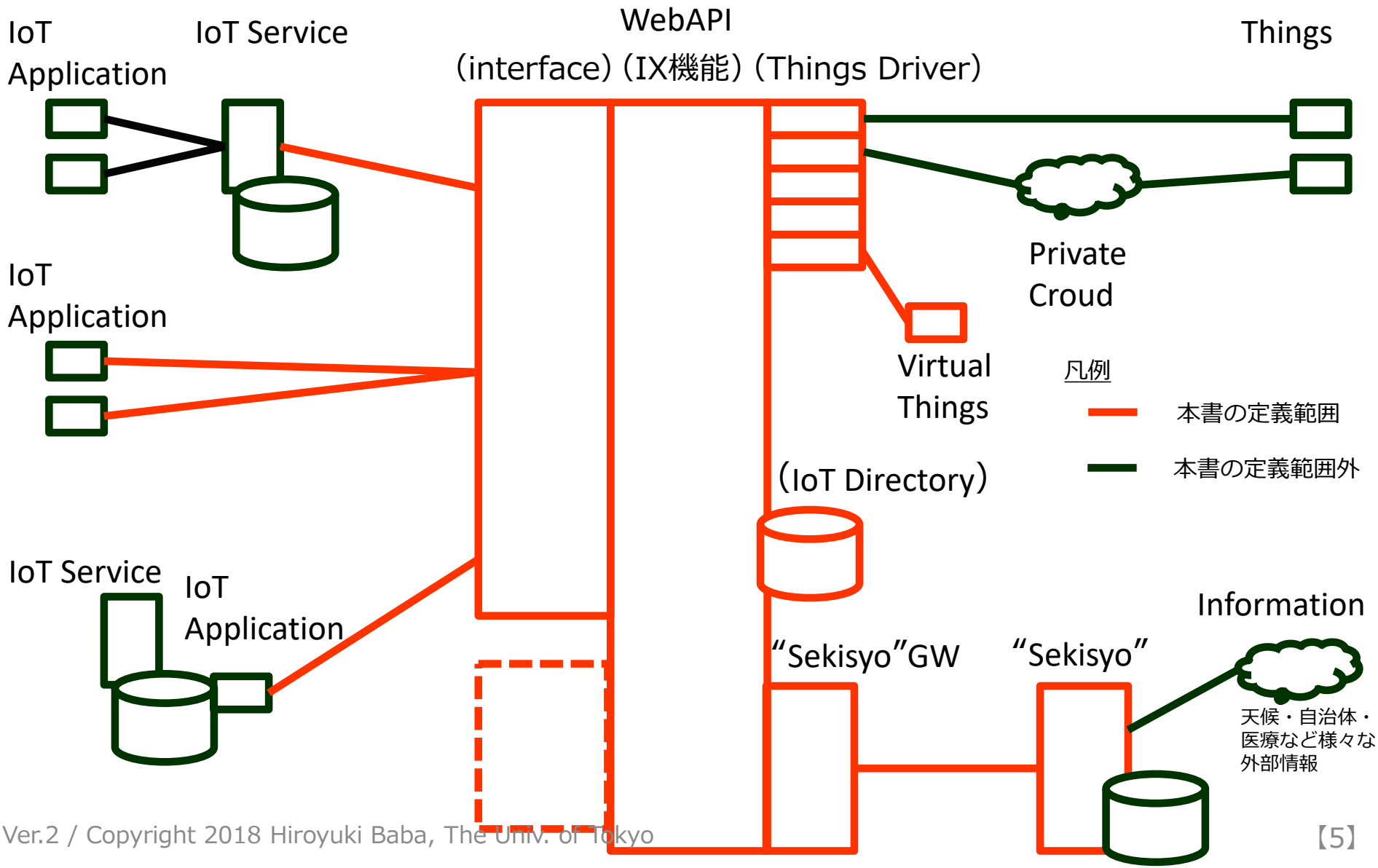
本要件定義書の用語定義－１

用語	内容
管理者	本仕組みを管理する事業者もしくはシステム
利用者	本仕組みを利用してIoTサービスを利用する個人などの利用者
サービス事業者やアプリベンダー	IoT Applicationの提供もしくは、見守りなどサービスを提供する事業者
Things メーカー	ネットワーク家電やセンサーなどを提供するメーカー
IoT Application	利用者のスマホなどにインストールされたり、サービス事業者がサービスを提供するために動かすIoT用アプリケーション
IoT Service	サービス事業者などにより、利用者を束ね、利用者情報を保有し、提供されるIoTサービス
Sekisyo	IoT由来の脅威から守るべく、家の中のThingsや外部情報などを用いてコマンド認証判定や通知、競合動作の解決を行う、安心安全確保部
WebAPI	IoT ApplicationやIoT ServiceがつながるInterface機能、IX機能・通知などの機能も持つ
IoT Directry	IX機能に必要なThings関連情報、利用者認証情報、紐づけ情報などを保持
IX機能	紐づけ情報を元にIoT Applicationと対象のPrivate CroudやThingsと相互に通信させ、また、安心安全確保部への確認や、必要に応じコマンドの置き換えを行う
Things Driver	独自プロトコルに対応し、それぞれのプロトコルと相互に変換するインターフェイス
Private Croud	企業などにて自社Thingsや、自社の規定するプロトコルでネットワークされたThingsと、IoT Applicationをつなぐ独自Cloud
Things	ネットワーク家電やセンサーなどの”モノ”

本要件定義書の用語定義－２

用語	内容
Information	天気予報情報やヘルスケア情報、地域情報など、外部情報
サイロ化	企業などが独自プロトコルを用いて相互接続性や相互運用性の低い状態
APIKey	IoT Service , IoT Application, Private Croud を特定する情報
UserKey	利用者を特定する情報や認証情報
HouseKey	場所を特定する情報
ThingsKey	機器を特定する情報
操作種別	操作(Command)・状態取得などの種別
操作(Command)	Thingsに対して状態を変更させる（例：エアコンをONにするなど）
操作優先レベル	操作必要性の度合い
状態取得	Thingsや外部情報に対して状況を取得する（例：電子錠の開閉情報など）
操作内容	操作の時の動作に関する内容
汎用Things Driver	通信規格や通信先のAPI仕様に合わせて作られたThings Driver Things側の実装依存まではフォローできない場合がある （例：エアコンの風量のレベルはメーカーごとに実装が異なるなど）

システム・アプリケーション構成イメージ



概要

独自プロトコルに対し、共通の通過点（WebAPI）を設けることで
①　Interconnectivity(相互接続性)
②　Interoperability(相互運用性)
を実現。

IoT ApplicationやIoT Serviceから見て、Thingsのベンダーやプロトコルフリーにするとともに、簡易に接続できる環境を提供

安心安全確保部による確認により
③　相互に接続した際の安全性
を実現。

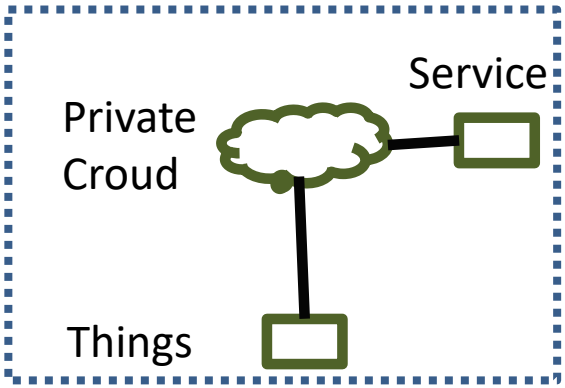
システムの構想

背景

IoT社会の実現には、サービスとインフラ、Thingsすべてがビジネスとして成り立ち、利用者も安心して使える仕組みや制度が必要

- ✓ Thingsを製作しているメーカーからすると、Thingsの魅力向上となるIoT Application創出されてほしいものの、他社のIoT Applicationなどからコントロールされると動作保証が困難であり、また、対応不可能な問い合わせにつながるなど課題が多く、相互接続のニーズはあるものの、自社IoT Application ＋自社PriveteCroudとThingsの組み合わせでのIoT化(サイロ化)が進んでいる

- ✓ また、サービス事業者やアプリベンダーからすると、IoTでビジネス化したい夢はあるものの、ThingsやPrivate　Croud側の多様さや仕様変更などの影響により、制作やメンテナンスコストが高額となり、参入障壁となっている



点線：サイロ化モデルの例

現状　利用者から見た課題

特定のサービスを特定の機器だけで利用するなど制限されてしまう
（見守り、帰宅時の照明　など）

利用者は、生活を豊かにしたり、健康維持、家族とのつながりなど、
さまざまなシーンで、IoTを活用可能であるが、下記のような課題が存在

□　実際の　モノ　が動くことにより、情報セキュリティーのみでなく
IoTならではの「IoT由来の脅威」が存在
（階段を移動中、遠隔操作で照明が消える　など）

□　使いたいサービスに合わせたThingsを買いそろえるのは大変

□　設定含め専門的な知識が求められる場合が多い

□　詳しくないと、どうして、だれが機器が動かしたのかわからない
動かなくてもだれに連絡していいのか分からない

現状　サービス事業者やアプリベンダーから見た課題

ビジネス化には様々な障壁が存在

サービス事業者やアプリベンダーは、IoTによるビジネス化を図りたいが
下記のような課題が存在

- □　特定のメーカー／機器だけだとサービスや利用者数に広がりがなく、
  サービスが成り立ちにくい。逆に多くのメーカー対応とするとコスト増大
  Interconnectivity（相互接続性）の欠如

- □　アプリ側ですべての利用者の環境を知って操作処理するのは難しい
  （例：ゲリラ豪雨の中、窓が開く）
  Interoperability（相互運用性）の欠如

- □　通信規格や、API仕様などが乱立。それぞれの規格や仕様に合わせても
  IT系は流れが速く、変更など継続的なメンテナンスが必要となってしまう。

- □　機種依存があり通信規格に合わせても動作する保証がない

- □　アプリベンダーが利用者の情報を管理したり、管理し続けるのは困難

- □　実機がないとアプリ開発ができない

現状　Thingsメーカーから見た悩み

オープン化に障壁があり、サイロ化してしまう
利用者も　サービスの内容も限定されてしまう

Things側は、魅力的なIoT Applicationが増えることでThingsとしての
魅力向上につながるものの、下記のような課題が存在

- □　ThingsとIT界の寿命の違い
- □　3rdパーティーのサービスや機器がつながっても、メーカーとしては
動作保証ができない（すべての動作確認は困難）
- □　同一メーカーで揃えてほしい　お客さまを囲い込みたいという要望はある
ものの、自社製品だけでは、サービスの範囲が限定されてしまう
- □　他社IoT Applicationなどから操作される場合、問い合わせが来ても自社製品の
故障であるか判断が難しく、メンテナンス員が出向しても解決しない
- □　独自サービスは開示したくないという思いがある場合がある

課題解決へのプロセス

IX機能を持つインフラとしてのWebAPIを設けることで、
前述した課題のうち、ほとんどが解決可能となる

WebAPIや、それを活用して安心安全に利用できる仕組みや制度を作ることで
魅力的なサービス創出などによりビジネス化が進み、IoT社会の実現に期待ができると考える

WebAPIの特徴

① **Interconnectivity(相互接続性)**
- 自社Thingsに限らずWebAPIのIX機能を活用して、他のThingsとも連動が可能となり、自社製品だけではなりえないサービスも提供可能にする
- 異なるThingsの仕様間差異を吸収
- Thingsメーカー特有の特殊機能による差別化も守秘性を保ちながら使用可能

② **Interoperability(相互運用性)**
- 他社製品のセンシング情報やヘルスケアや天候などの外部情報なども活用可能となり、3rdパーティーによる魅力的なアプリケーションの創出や接続も可能に
- 責任分界点の明確化や切り分けが可能

③ **相互に接続した際の安全性**
- IoTが利用者から見て安心・安全に使える

WebAPIの機能　①　Interconnectivity(相互接続性)

（１）IoT用のドライバー構造の採用

- ドライバー構造とすることで、様々な通信規格への対応や、PriveteCroud との接続を可能とすること（マルチベンダー・マルチプロトコル）
- ThingメーカーがIoT用ドライバーを提供することで、特殊機能などの内容はブラックボックス化を可能とする

（２）IX機能

- サービスやアプリから利用者と紐づく鍵で簡易に接続できるよう WebAPI側で、利用者の家情報、Things保有状況、Things機器種別などの情報を保有し、交換・接続可能とする サービスやアプリ側からの情報取得も可能とする
- WebAPI側でドライバを追加・更新。Things側の更新はサービスやアプリ側の変更なくWebAPIで吸収可能とする（メンテナンスフリー）

（３）実装依存の吸収

- 規格・基準だけでは吸収できない実装依存やバージョン違いなどの機種依存について、IoT用ドライバーで吸収しきれない部分もWebAPI側で差異を吸収。サービスや、アプリ側でのフォロー不要とする

（４）仮想機器のサポート

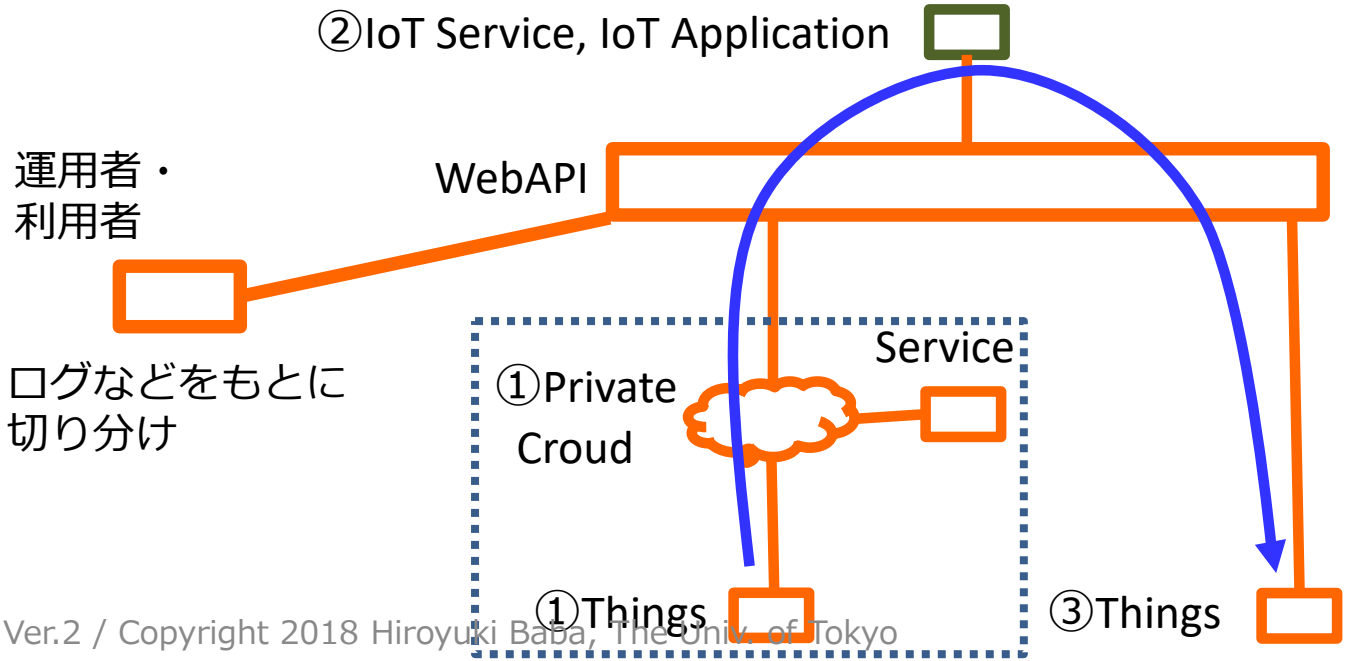
- IoT用ドライバーに、機器を模擬するVirtual Thingsを接続可能とする
- アプリ開発者がVirtual Thingsを用いることで、実機がなくても机上でアプリ開発を行うことを可能とする

WebAPIの機能　②　Interoperability(相互運用性)

（１）ThingsやPriveteCroudを含めた外部情報からの情報取得機能を持つ

（２）誰が操作したかなど切り分けや、故障に関する切り分け機能と通知機能をもつ

例）①のセンサーや外部情報（Private　Croud，Things，天気予報情報など）を
②IoT ServiceやIoT Serviceが受信もしくは情報取得。それをトリガーとして
③Thingsを動かす　など、相互に情報を活用することで、利用者負担を抑えるとともに
サービスに広がりができる



WebAPIの機能　③　相互に接続した際の安全性

（１）安心・安全確保機能

- 安心・安全確保部（関所“Sekisyo”と命名/確認時の判定内容は本仕様の定義範囲外）」とのAPIを介した通信が可能。関所は外部の情報と連携するなどし、コマンドレベルでの操作の良否判断などを実施し、IoT由来の脅威から安全を提供する
- 判断結果や一時保留通知をWebAPIへレスポンスすることで、レベルに合わせ、“動かさない“と”確認してから動かす（警告後承認）”で動作し柔軟性も保有する

IoT由来の脅威の例

パターン１）アプリとアプリの競合の一例

- 一つのアプリは窓を開け、一つのアプリはエアコンで快適な空間を作ろうとするなど、複数のアプリがThingsに接続し、不便益などが発生する

パターン２）アプリと人の操作の競合の一例

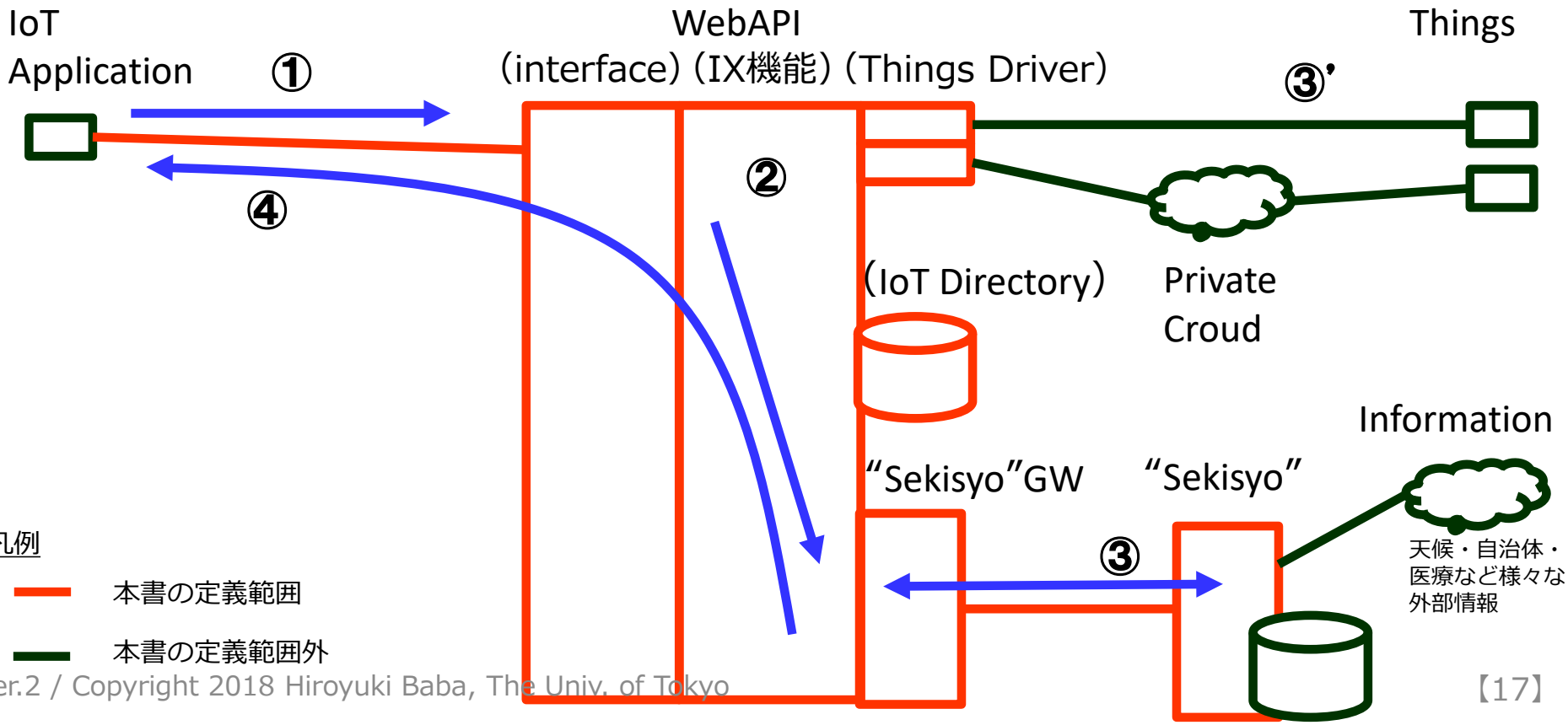
- 火を使っているのに、外出先から換気扇を止めてしまうなど、人の操作で危険などが発生する

パターン３）アプリと環境の競合の一例

- ゲリラ豪雨なのに、窓が開くなどアプリから環境や状況に合わない動作が発生してしまう

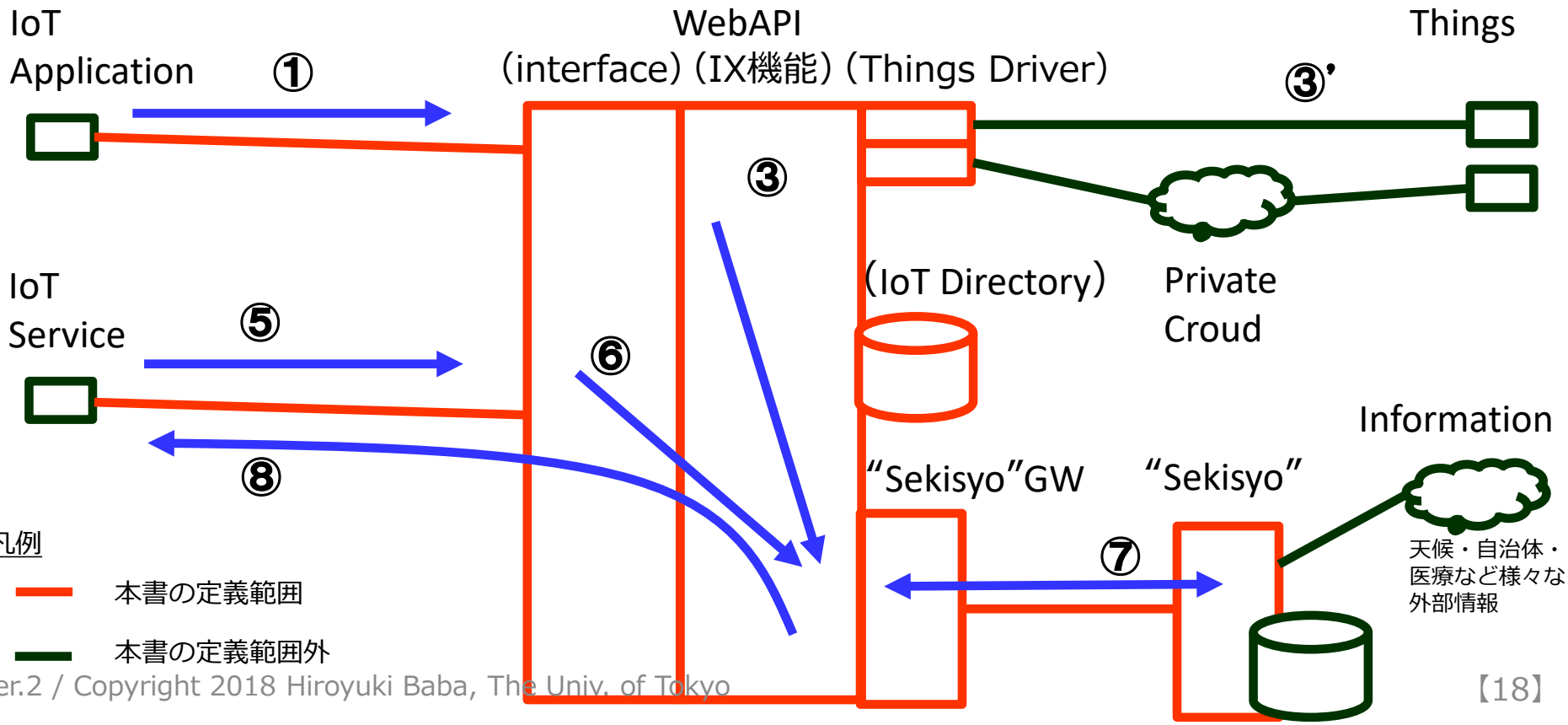
WebAPIの機能　③‘　Sekisyo動作のイメージ（例1）

① IoT Applicationにて、外から「換気扇」を含めThingsを止めるよう操作
② Web APIが、IoT Directoryの情報とともにSekisyoに確認
③ Sekisyoは、換気扇が止められて大丈夫な状態かセンサーなどのThingsから状態取得　結果例：“火器が使われている！”
④ Sekisyoは換気扇を止めるコマンドの判定をNGとし、IoT Applicationへ通知すべき内容をレスポンス



WebAPIの機能　③‘　Sekisyo動作のイメージ（例2）

①②　IoT Applicationにて、利用者がApplicationの優先順位を設定
（例：今日は赤ちゃんが寝ているから快適制御アプリを優先）
③④　Web APIが、Sekisyoに通知
⑤⑥　IoT Service(例：負荷制御アプリ）は、需要抑制信号を受信し需要抑制制御
⑦⑧　SekisyoはApplicationの優先順位情報を元にコマンドの判定をNGとし、
IoT Applicationへ通知すべき内容をレスポンス



解決される内容　利用者

課題	解決後
実際の　モノ　が動くことにより、情報セキュリティーのみでなく、IoTならではのIoT由来の脅威が存在。 （階段を移動中、遠隔操作で照明が消えるなど）	関所により安心・安全にアプリを利用可能。 また、WebAPIからの警告や通知をアプリ側で受信可能※１となるため、利用者にも状況が分かる
使いたいサービスに合わせたThingsを買いそろえるのは大変	Thingsのメーカーを問わずに使いたいIoT Applicationを利用可能となる※２ Thingsはニーズに合わせて用意すれば、全て揃えなくてもサービスを利用可能
詳しくないと、どうして、だれが機器が動かしたのかわからない	利用者や、カスタマーセンターなどが、履歴を確認できるため、 状況が分かるとともに、修理対応も依頼できる。 また、Thingsの新規・追加時に自動登録されるので、利用者は機器を選ぶだけで簡単に登録可能

※１　IoT Application側で実装した場合
※２　本書で提案するWebAPIを使用しているIoT Applicationのみ

⇒　使えるIoT Application・サービスの幅が広がるとともに、安心して使えるようになる

解決される内容　アプリベンダー・サービス事業者

課題	解決後
特定のメーカー／機器だけだとサービスや利用者数に広がりがなく、サービスが成り立ちにくい。逆に多くのメーカー対応とするとコスト増大	WebAPI用の簡単な通信をするだけで、マルチベンダー・マルチプロトコルとなり、開発コストの削減や、利用者の母数増につながる／Interconnectivity(相互接続性）が向上する
アプリ側ですべての利用者の環境を知って操作処理するのは難しい （例：ゲリラ豪雨の中、窓が開く）	センサーや天候情報など利用者に関する情報を共有したり、共通の安心安全確保部を経由することで、アプリ側の処理をシンプルに出来るなどInteroperability(相互運用性)を向上できる
通信規格や、API仕様などが乱立。それぞれの規格や仕様に合わせても、IT系は流れが速く、変更など継続的なメンテナンスが必要となってしまう。	WebAPI向けの通信のみ利用すれば、Things側の更新などに対してもメンテナンスフリーとなり、IoT Application側の費用削減並びに、IoT Applicationを長く使えるようになる（利用者メリット）
機種依存があり通信規格に合わせても動作する保証がない	利用者の使用している機器情報はWebAPI側で持てるので、Things専用ドライバの活用や、汎用ドライバであれば、WebAPI側で機器情報に合わせた変換を行うことで機種依存を吸収できる。 (例：エアコンの風量、JEMA規格の窓　など)
アプリベンダーが利用者の情報を管理したり、管理し続けるのは困難	利用者情報（Things情報含む）は、WebAPI側で保有するので、アプリを軽装化できるとともに利用者側も登録の一元化ができる
実機がないとアプリ開発ができない	Virtual Thingsを用いることで、実機がなくても机上でアプリ開発を行える

⇒　IoTサービスをビジネス化しやすくなる（＝IoTサービスの広がりにつながる）

解決される内容　Thingsメーカー

課題	解決後
3rdパーティーのサービスや機器がつながっても、メーカーとしては動作保証ができない（すべての動作確認は困難）	機種依存(※１)を含めて、WebAPI側でフォローできるので、3rdパーティのIoT Applicationや機器も接続しやすくなる。
同一メーカーで揃えてほしい　お客さまを囲い込みたいという要望はあるものの、自社製品だけでは、サービスが限定されてしまう	独自機能はそのまま活用できるので、差別化が可能。さらに、他社商品や自治体など外部も組み合わせたサービスの創出が可能
他社IoT Applicationなどから操作される場合、問い合わせが来ても自社製品の故障であるか判断が難しく、メンテナンス員が出向しても解決しない	責任分界点が明確になり、かつ、WebAPI側である程度、切り分けが可能となるので、無駄な出向を減らすことができ、メーカー・利用者ともにメリットとなる
独自サービスは開示したくない	Thingsメーカーがドライバを提供することで、ブラックＢＯＸ化可能

※１　機種依存に関する情報はWebAPI側に通知が必要
（独自機能は必要に応じて）

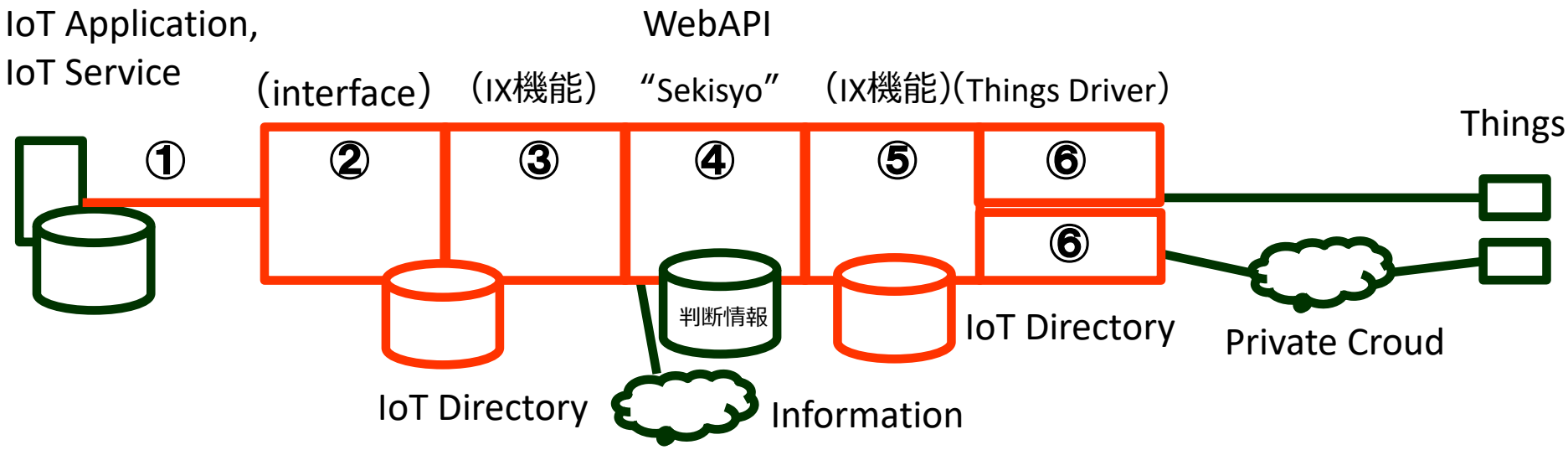
⇒　3rdパーティーのIoT Applicationやサービスがつながることで、Thingsの魅力向上になるとともに、各社独自サービスもそのまま活用でき、差別化が可能

通信の事例

1） Thingsへの操作・状態取得
2） Thingsからの通信（センサー情報など）
3） WebAPI API通信する情報の構成
4） 利用者登録
5） Things登録
6） ThingsDriver更新
7） ログ取得

1）Thingsへの操作・状態取得-1

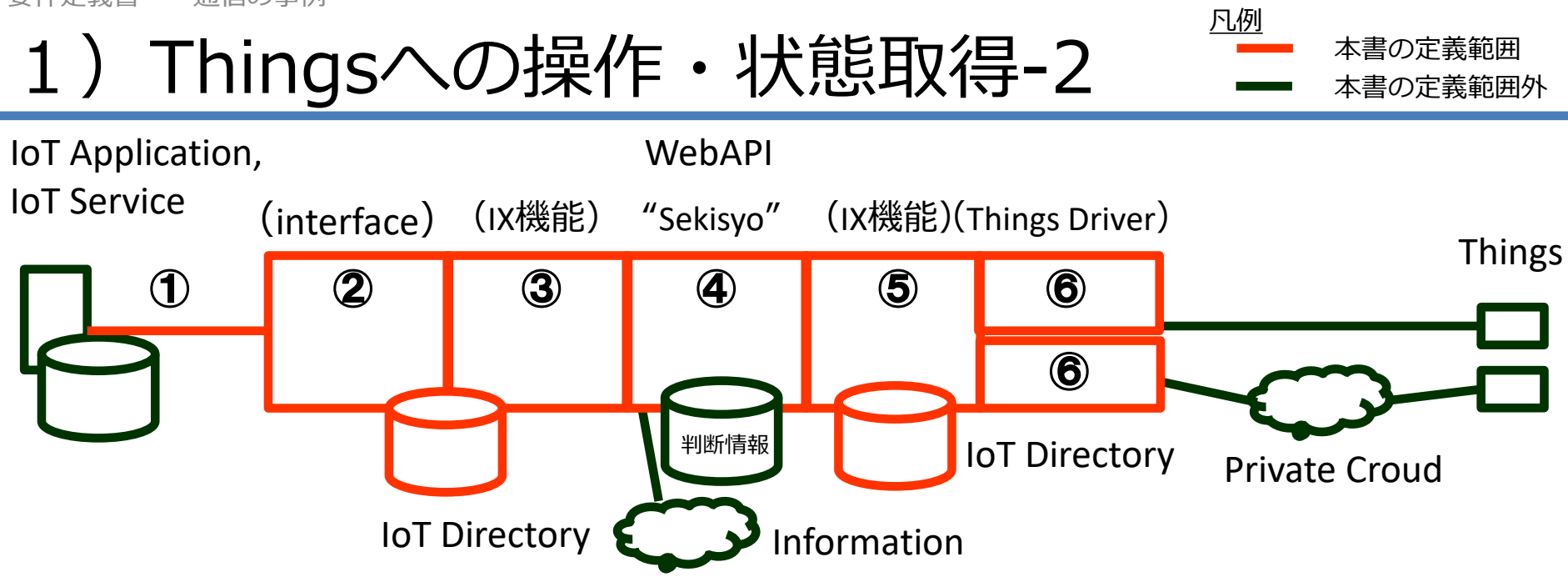
凡例
本書の定義範囲
本書の定義範囲外



	通信概要・処理
①	APIKey, UserKey, HouseKey, ThingsKey, 操作種別, 操作優先度 , 操作内容などをWebAPIへ要求
②	IX機能へ情報受け渡し・情報セキュリティーの確保 通信プロトコルは、JSONなどその時代に応じたプロトコル
③	IoT Directry の情報より個を特定し、操作(Command)であれば、Sekisyoへ確認
④	操作してよい状態か判断
⑤	ThingsDriverやThings情報から、必要に応じ実装依存などの吸収（操作の選択※１や置き替え） 接続先のThings情報を含め該当するThingsDriverへ通信
⑥	ThingsやPriveteCroudの仕様に合わせた通信プロトコルにて該当のThingsやPrivate　Croudへ通信

※１　Things１には、Command AとB を通すが　Things２には　Command Aのみを通すなど汎用ThingsDriverを用いた場合でも　Things毎の差異を吸収

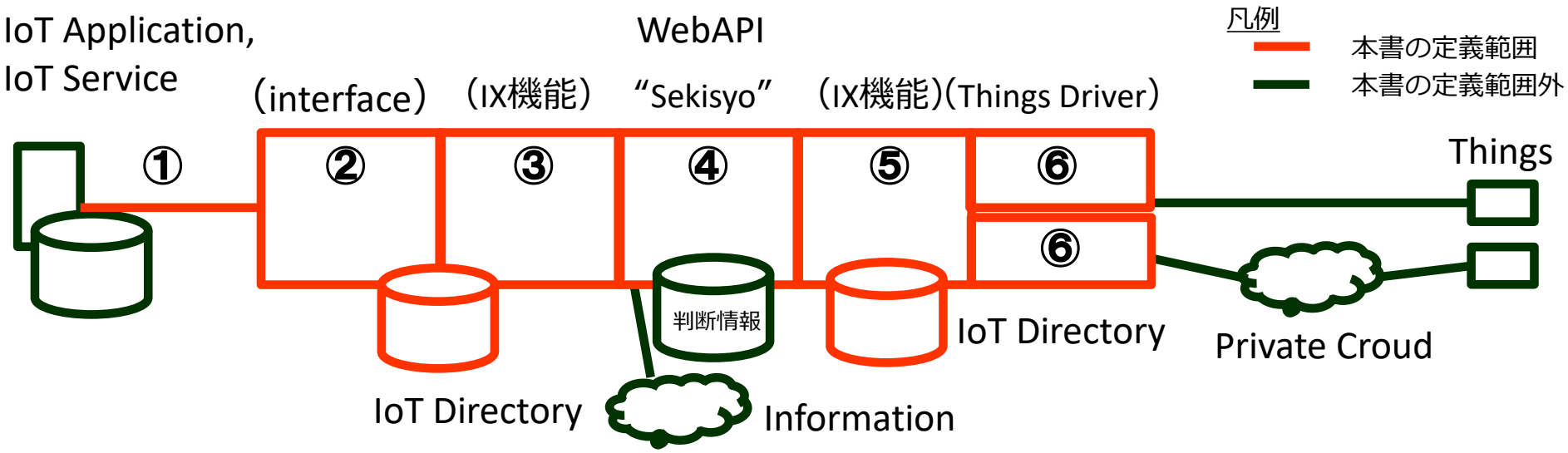
1）Thingsへの操作・状態取得-2



レスポンスは、⑥⇒⑤（③）⇒②⇒①の順で通信

Sekisyoにて確認NGの場合は
④⇒③⇒②⇒①の順でNGの分をレスポンス

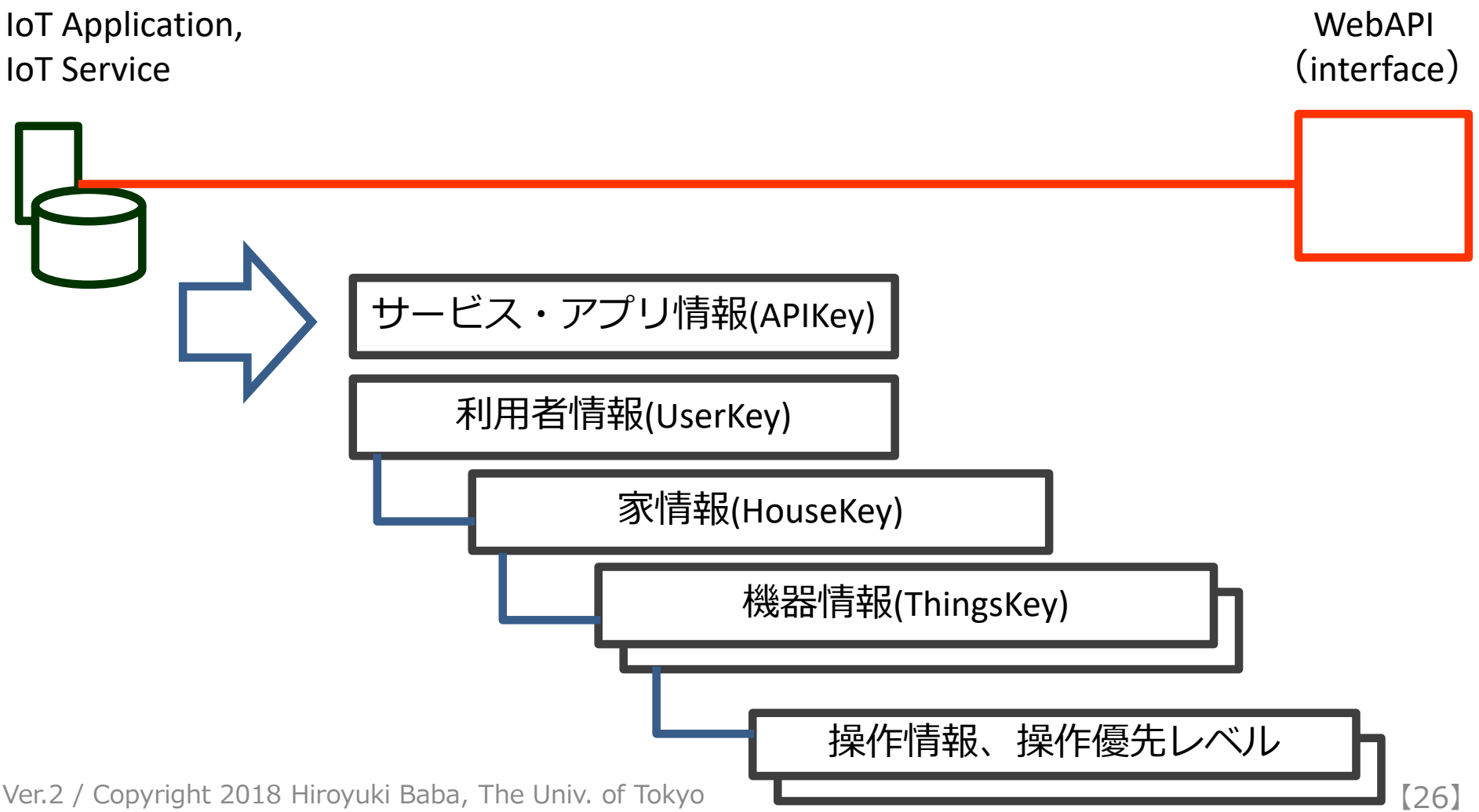
2）Thingsからの通信(センサー情報など)



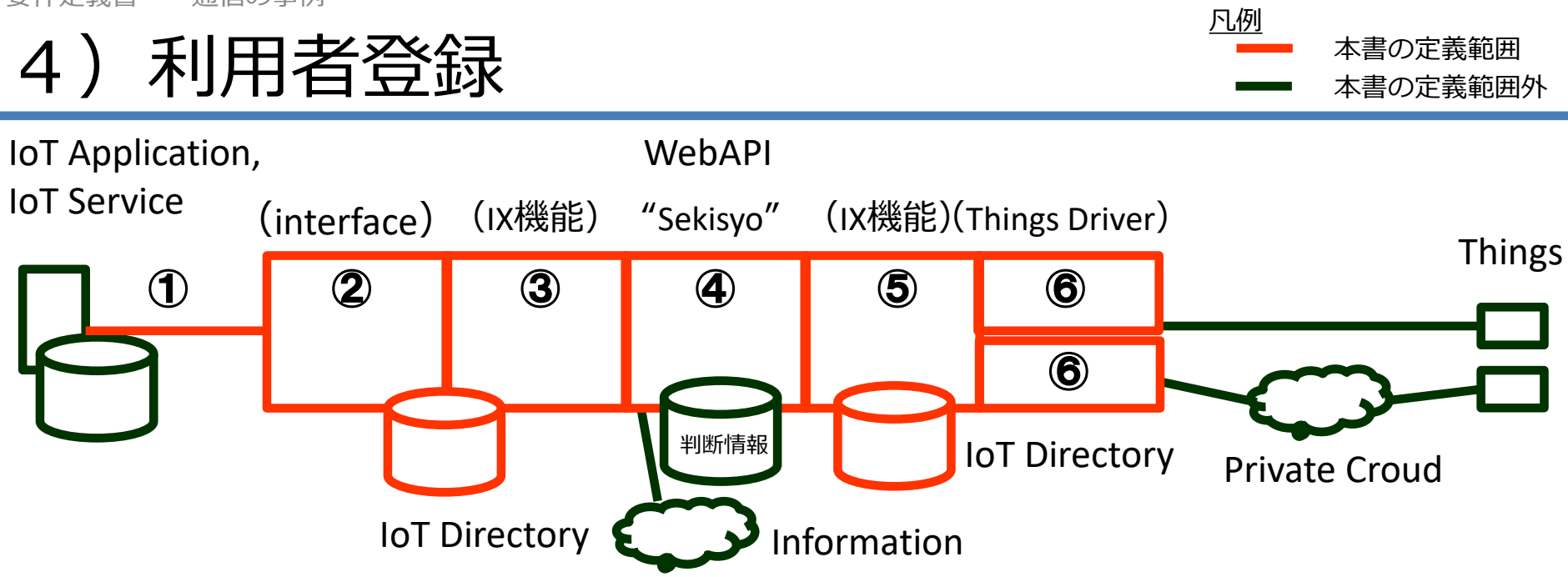
Thingsからの通信は、⑥⇒⑤（③）⇒②⇒①の順で通信

3）WebAPI　通信する情報の構成

階層化することでわかりやすく・交換しやすくなる



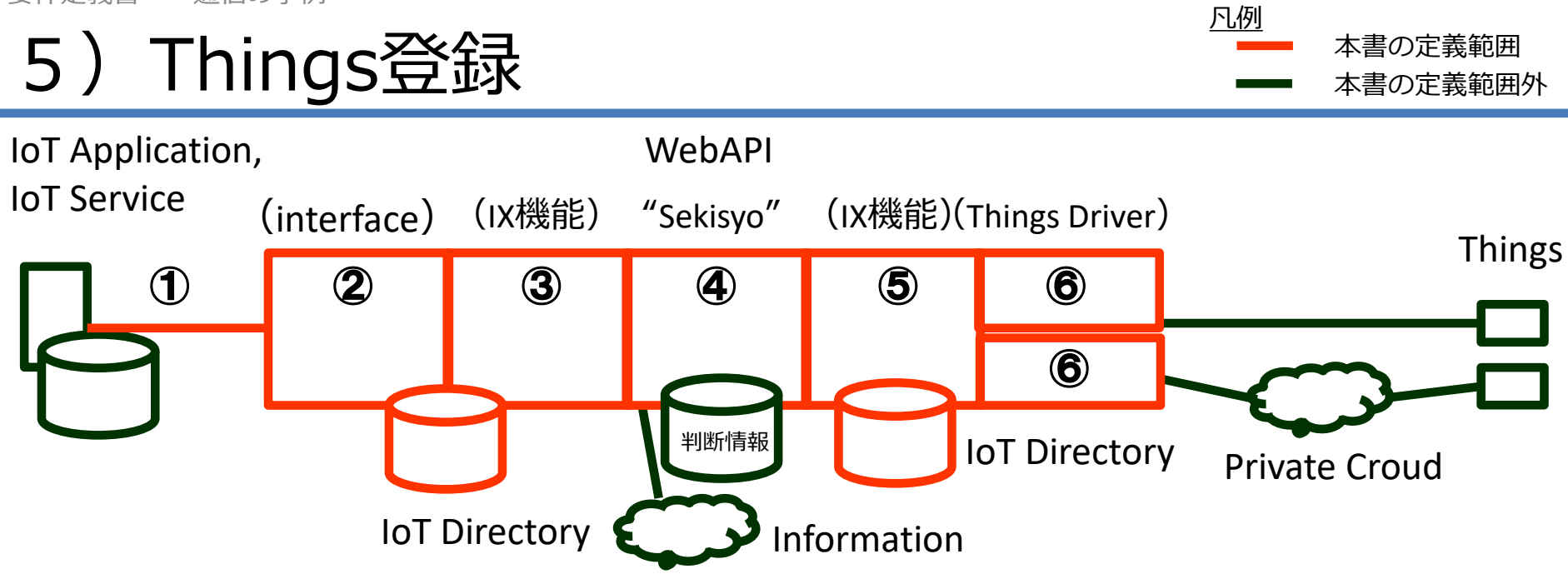
４）利用者登録



IoT Apprication もしくは　IoT Service（管理者機能を含む）より
利用者とその紐づけとなる情報を登録

紐づけとなる情報の例
　　Things Aは、寝室のエアコン

5）Things登録

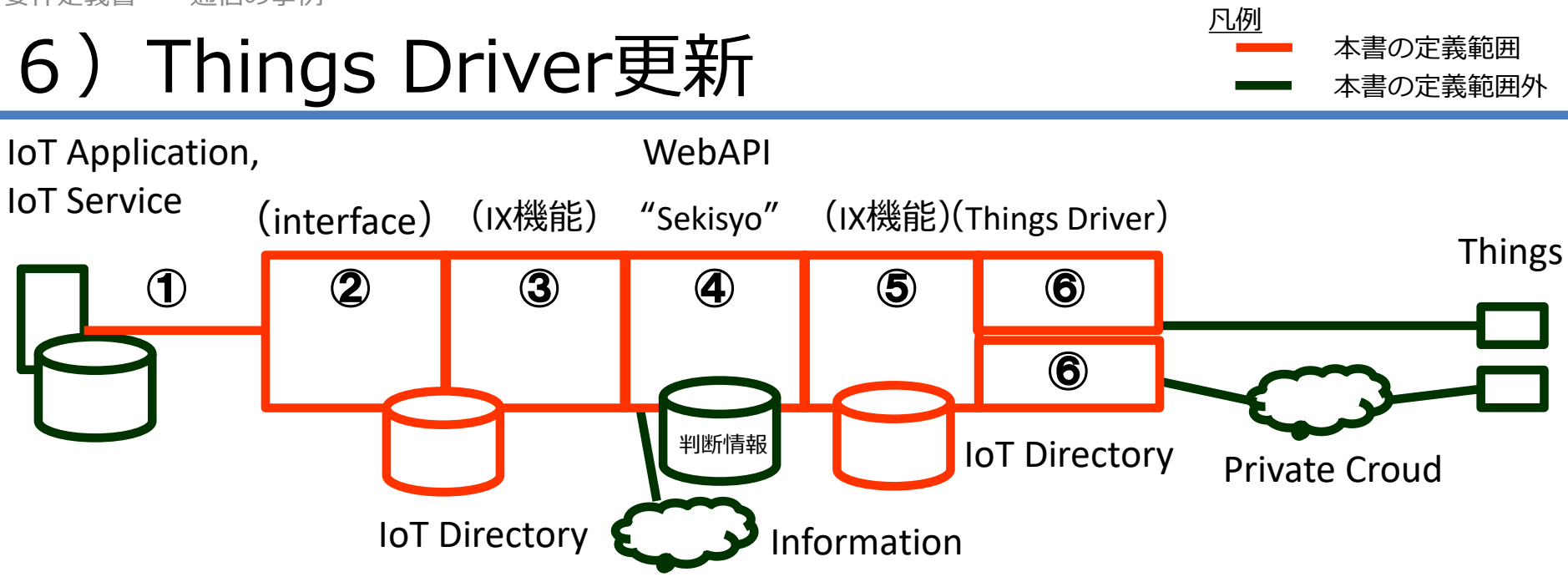


ThingsDriverがThings情報を取得（もしくはPrivete CroudからのPUSH情報など）

IoT Directoryは、Things関連情報と、それに紐づく情報を格納

紐づく情報の例
　Things A は　機器種別：エアコン　ThingsDriver：A社Private　Croud

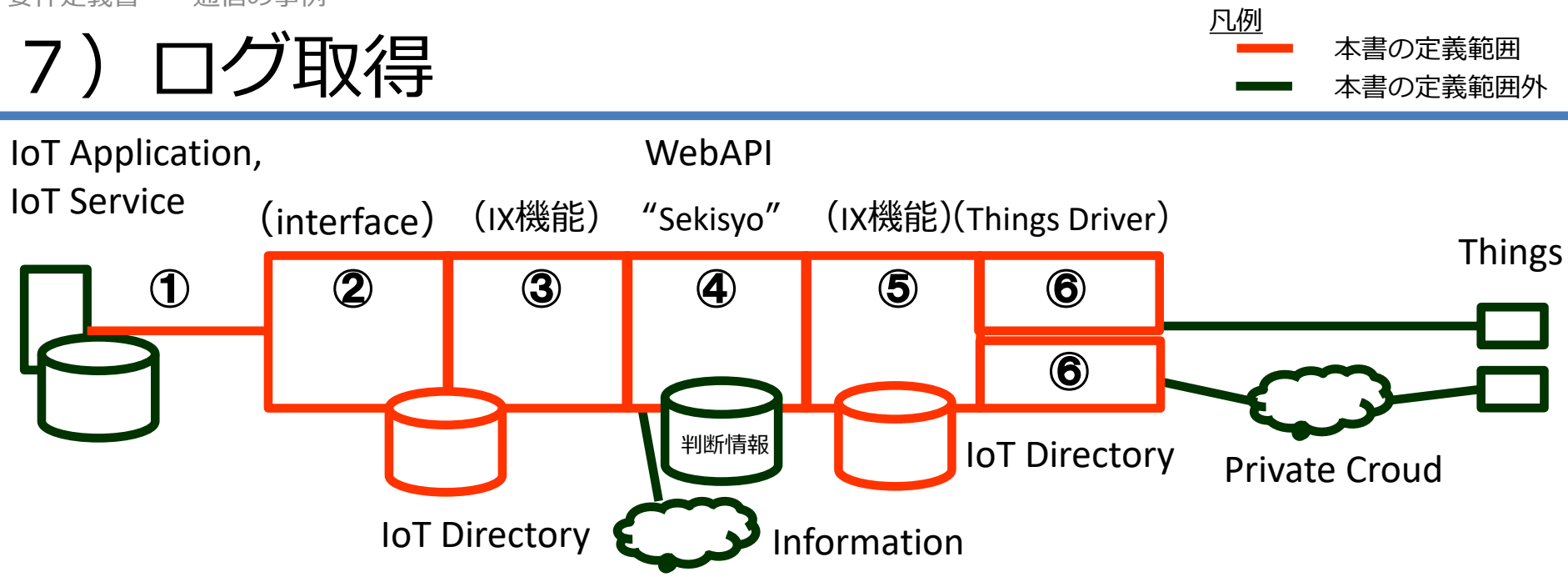
6）Things Driver更新



Things Driverの追加や更新を行う

パターン例１）
　システム管理者より各Things Driverに対して追加や更新
パターン例２）
　Private Croud側より自社Things Driverに対して更新
パターン例３）
　Things DriverがWebAPIより更新情報を取得し追加や更新

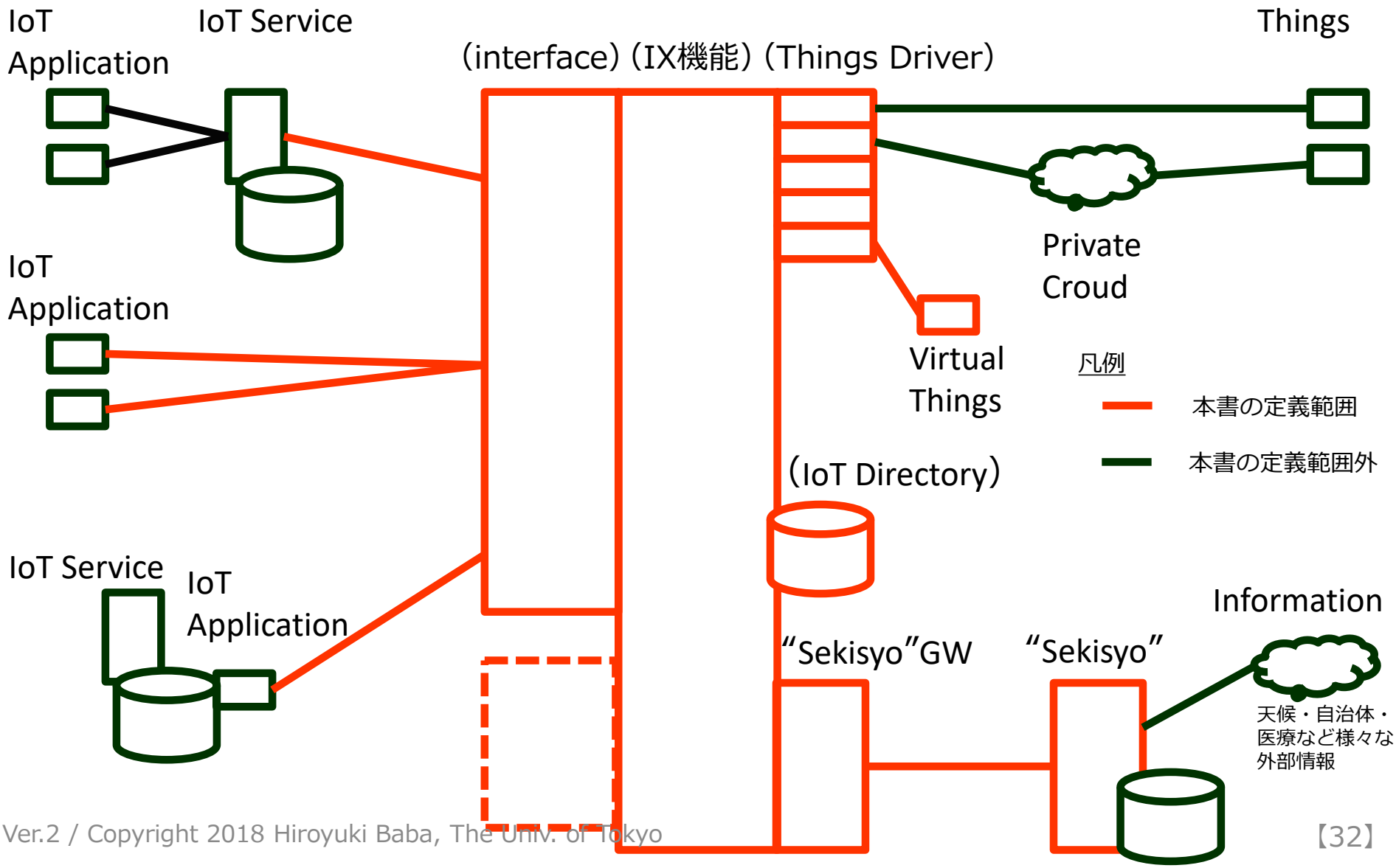
7）ログ取得



通信内容や結果はログとしてWebAPIにて保持
管理者もしくは利用者にて確認できるインターフェイスを保有し
切り分けなどに必要な情報が分かるようにする

機能要求

システム・アプリケーション構成イメージ（再掲）



WebAPI（Interface部）

機能）
- IoT ServiceやIoT Applicationとの通信インターフェイス（WebAPI）

用件）
- IoT ServiceやIoT Applicationと合った通信プロトコル（JSONなど）やセキュリティー施策を実装すること
- IoT ServiceやIoT ApplicationへのPUSH機能を実装すること
- 関所との通信（確認・応答からのアプリへのアクション）が出来ること

通信概要）
- IoT ServiceやIoT ApplicationからのThings（家電・センサーなど）情報取得・操作
- Things側からの情報PUSH通信
- 接続先ユーザー特定情報

WebAPI（IX機能部）-1

機能）
- ・通信のエクスチェンジ/ひも付けと判断
- ・IoT Driver等UpDate情報の配布

用件）

＜利用者機能＞
- ・利用申し込み・変更・解除を持つこと
- ・管理機能を持つこと
  - ・機器やアプリ管理のほか切り分けに必要な情報も提供
- ・機器情報の取得・更新を持つこと

＜通信機能＞
- ・ IoT ServiceやIoT Applicationと接続先ユーザー間のエクスチェンジができること
- ・関所への問い合わせと結果受信ができること
- ・機種依存等変換ができること
- ・ひも付け登録ができること

＜UpDate機能＞
- ・ドライバ等の配布ができること

＜管理・運用者機能＞
- ・管理機能を持つこと

WebAPI（IX機能部）-2

機能）
- IoT Interface部とIoT DriverやSekisyoとの通信IF
- 機器監視（Things再起動に伴う最新のIPアドレス情報取得など）
- ドライバ等のUpDate

用件）
- 通信エクスチェンジをできること
- 機器情報取得、差分はWebAPI(IoT Directory部）へ反映できること
- UpDate関連情報の取得と情報を元にUpDateをできること
- ドライバの実装をできること
- IoT Driver,Thingsとの適切なセッション維持（上り下りの通信を円滑にする）をできること

WebAPI（IoT Directory部）

機能）
・エクスチェンジに必要な情報を保持

用件）
＜利用者関係＞
・利用者情報の入出力機能を持つこと
・利用者接続先内機器情報（種別・固有情報等）の入出力機能を持つこと
＜機器情報関係＞
・該当のThingsのIoT Driver情報など、IXに必要な情報を持てること
・機種依存関連情報（プロパティー有無含む）を持てること
＜アプリ関係情報＞
・各種キー等の情報を持てること

WebAPI （Things Driver部）

機能）
- Thingsに合わせたプロトコル変換

用件）
- 必要な通信IFを実装(Things, Privete Croud向け）できること
- 必要なプロトコル変換をできること（例：Things独自のThings Driverの場合、そのThingsの実装依存問題の解消）
- プライベートプロパティー（例：Thingsメーカー独自プロパティー）や、変換（例：あるコマンドが来たら特定のコマンドも付加する）など、Thingsメーカー独自の魅力も発揮することが出来ること。
- 機器情報取得ができること

想定）

＜Private Croud＞
- ヘルスケアなどの外部情報
- 大手検索系、OS系企業APIとの接続
- Thingsメーカー独自Croudとの接続
- Things APIへの接続

Sekisyo部

機能）
- 安心・安全判断
- 結果通知・一時保留
- Things等の情報取得

用件）
- 安心・安全ルールエンジンに基づく判断ができること
- 安心・安全ルールエンジンの判定に必要な情報取得（Things・外部情報）ができること
- 学習機能を具備すること（本仕様の定義範囲外）
- 管理機能があること
- IoT Applicationの優先順位などの情報を持つこと

Virtual Things部

機能）
- 実機器の模擬

用件）
- 実機器と同様に、機器操作や情報取得ができること
- 実機器と同様に、機器登録ができること

更新履歴

ver	更新日	更新内容
1.00	2018/2	初版公開
2.00	2018/12	仮想機器追記修正