Open APIs for Open Minds

コンテキスト情報管理:イントロダクション (Managing Context Information at Large Scale: Introduction)

Fermín Galán Márquez - fermin.galanmarquez@telefonica.com

(Reference Orion Context Broker version: 3.8.0)

(Translated into Japanese by Kazuhito Suda k@fisuda.jp)

FIWARE

イントロダクション

- FIWARE におけるコンテキスト管理
- Orion Context Broker
- データの作成とプル
- データのプッシュと通知
- バッチ操作

“スマート”であることは、まず”アウェア”であることが必要

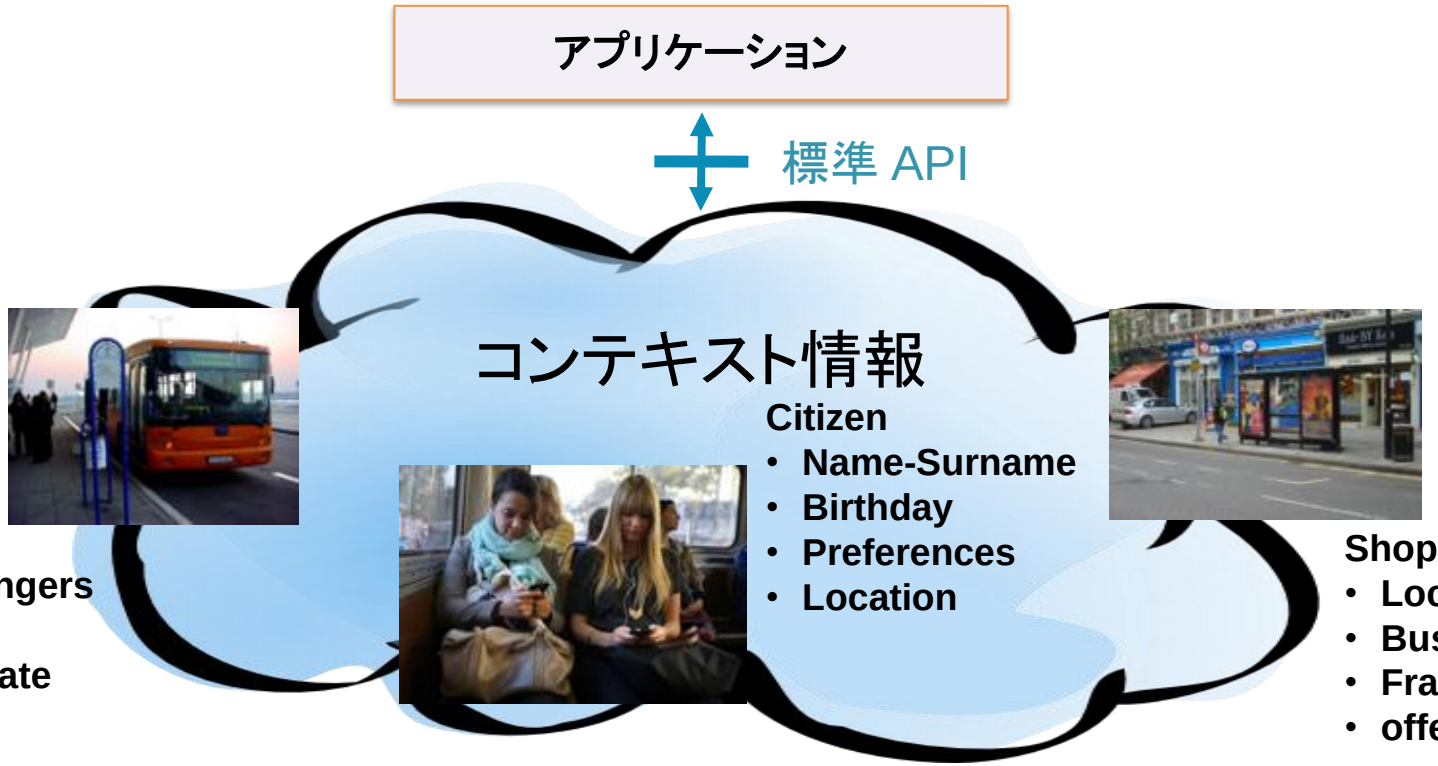
- スマート・アプリケーションを実装するには、関連するエンティティを特徴付ける**属性**の値を参照して、コンテキスト情報の収集と管理が必要です
- スマート・ホーム、スマート農業、スマート・インダストリー、スマートな物流、スマート・シティなど、コンテキスト情報の管理に関連する**アプリケーション・ドメイン**は数多くあります

“スマート”であることは、まず”アウェア”であることが必要

- スマート・アプリケーションを実装するには、関連するエンティティを特徴付ける**属性**の値を参照して、コンテキスト情報の収集と管理が必要です
- スマート・ホーム、スマート農業、スマート・インダストリー、スマートな物流、スマート・シティなど、コンテキスト情報の管理に関連する**アプリケーション・ドメイン**は数多くあります

“スマート”であることは、まず”アウェア”であることが必要

- スマート・アプリケーションを実装するには、関連するエンティティを特徴付ける**属性**の値を参照して、コンテキスト情報の収集と管理が必要です
- スマート・ホーム、スマート農業、スマート・インダストリー、スマートな物流、スマート・シティなど、コンテキスト情報の管理に関連する**アプリケーション・ドメイン**は数多くあります



スマート・シティにおけるコンテキスト情報管理

- 都市サービスやサードパーティ・アプリケーション(アクセス制御ポリシーに従う)の管理を扱うシステムは、コンテキスト情報を**消費・生成**することができます
- 全体の都市ガバナンスは、KPI を監視し、ビッグデータ分析を実行するために利用可能なコンテキスト情報(リアルタイムおよび履歴)に依存することができます

都市ガバナンスシステム

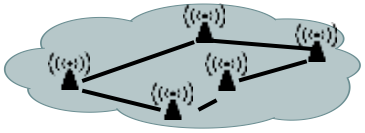
サードパーティ・アプリ

コンテキスト情報

Bus
- Location
- No. passengers
- Driver
- Licence plate

Citizen
- Name-Surname
- Birthday
- Preferences
- Location

Shop
- Location
- Business name
- Franchise
- offerings



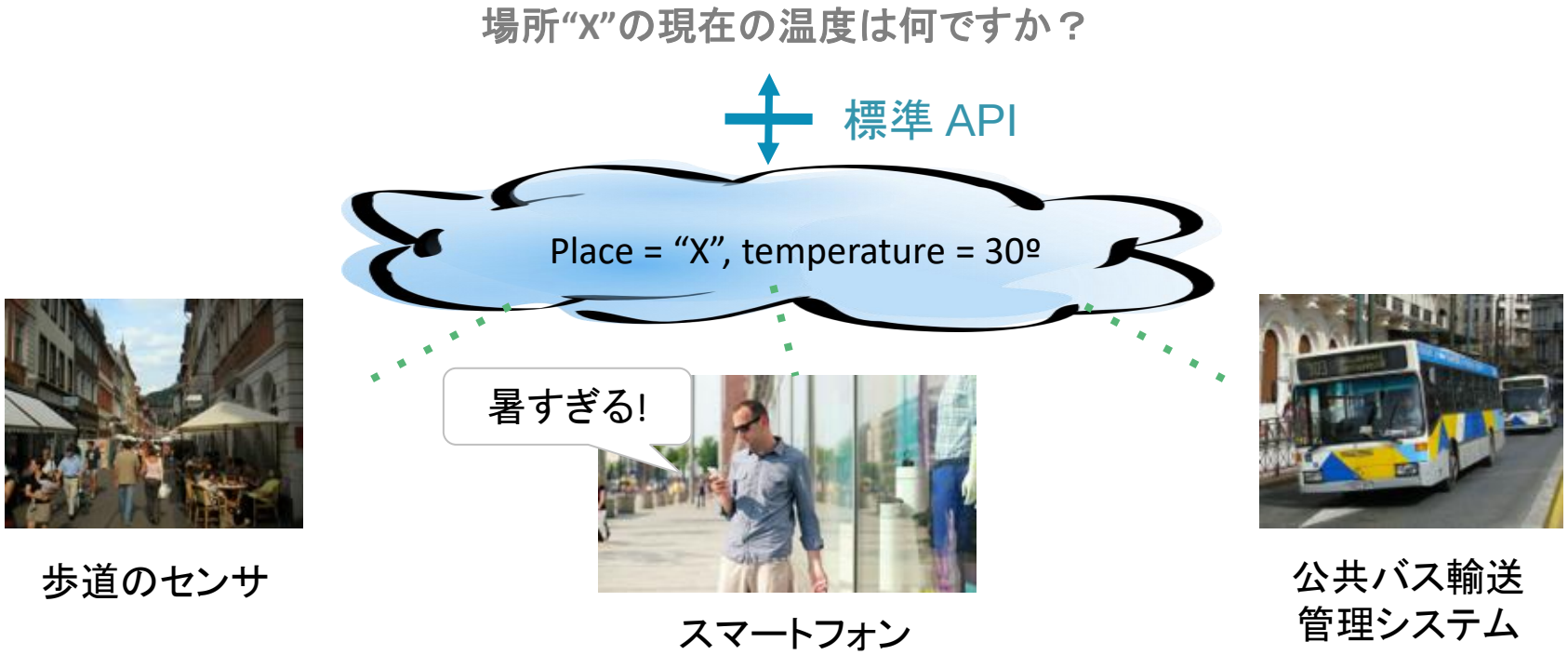
都市サービス

都市サービス

FIWARE

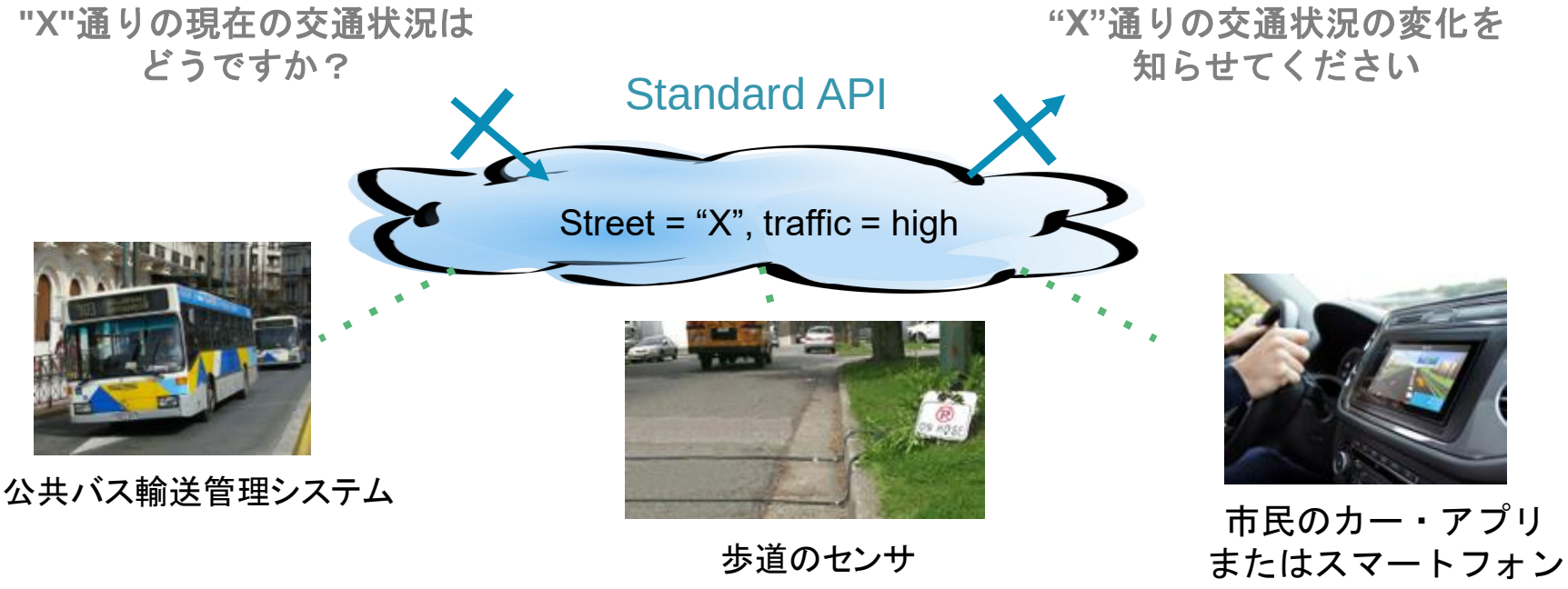
コンテキストの異なるソースを処理することが必要

- コンテキスト情報は、多くのソースからやってくる:
  - 既存システム
  - モバイル・アプリを使うユーザ
  - センサー・ネットワーク
- エンティティの情報源は時間とともに変化する可能性があります



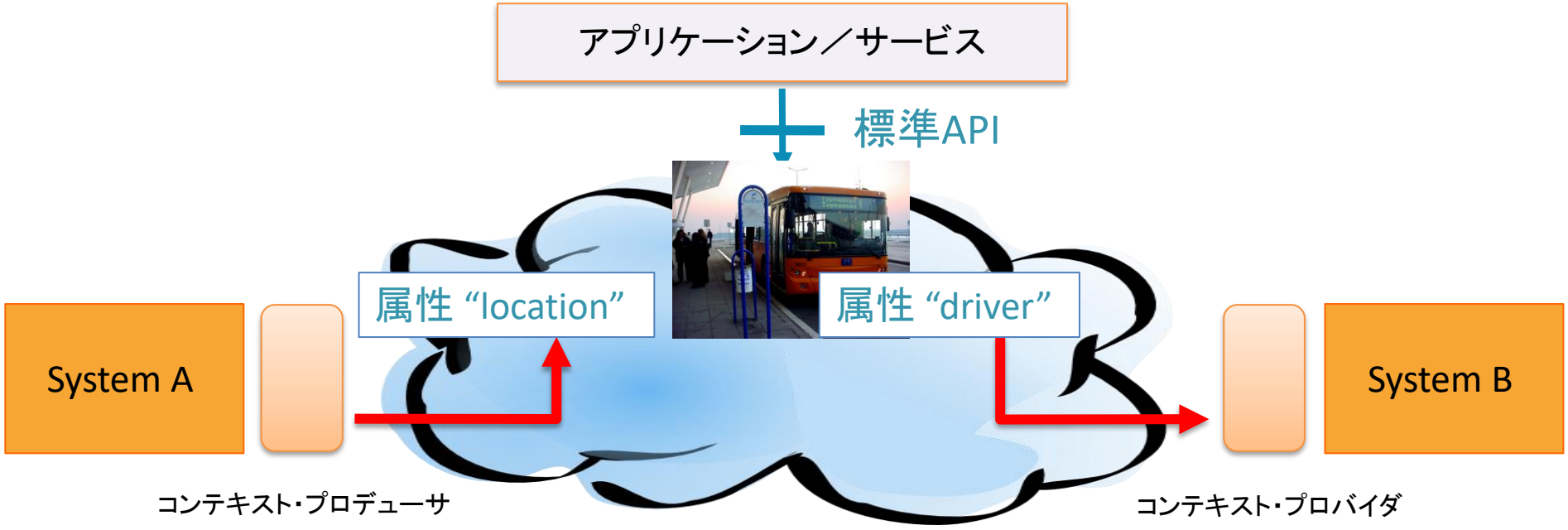
コンテキストの異なるソースを処理することが必要

- コンテキスト情報は、多くのソースからやってくる:
  - 既存システム
  - モバイル・アプリを使うユーザ
  - センサー・ネットワーク
- エンティティの情報源は時間とともに変化する可能性があります



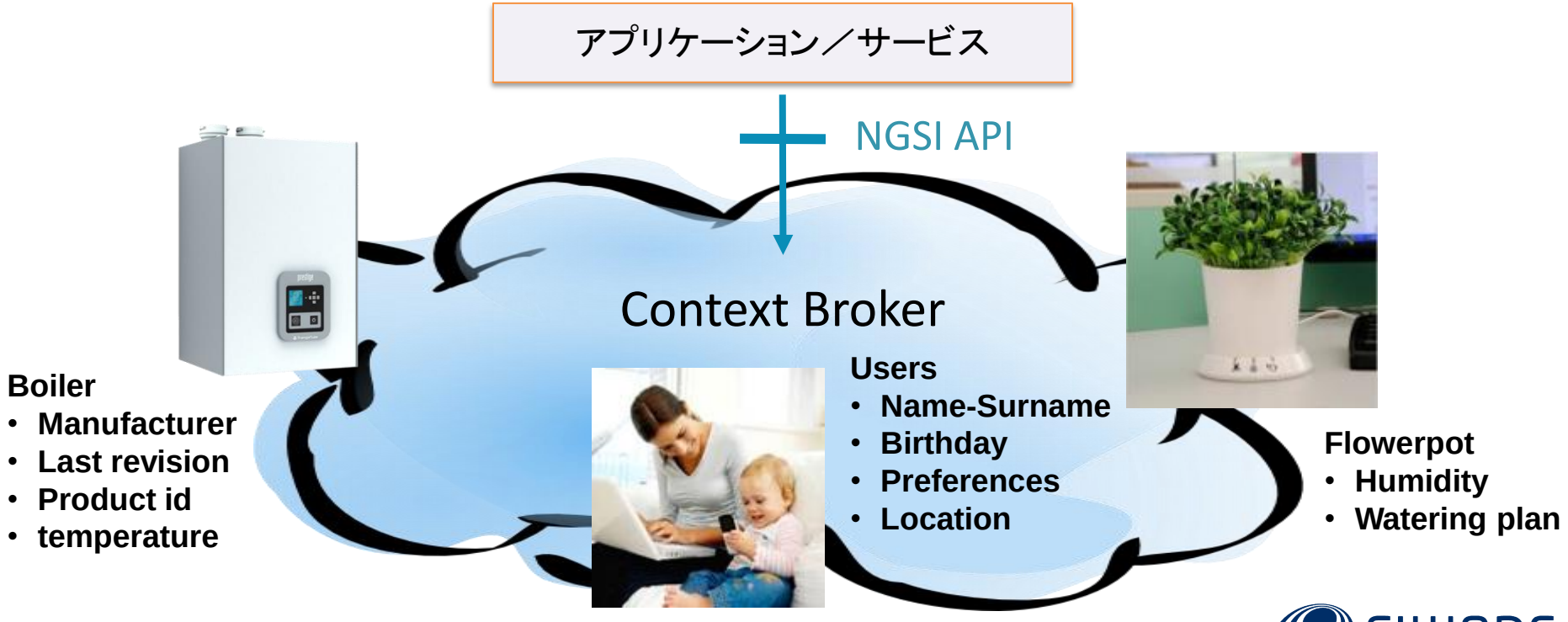
他システムに負担をかけないアプローチが必要

- アーキテクチャに影響を与えずに既存のシステムまたは将来のシステムと統合することができ、共通のコンテキスト情報ハブを実現します
- あるエンティティの属性に関する情報は、コンテキスト・プロデューサまたはコンテキスト・プロバイダのいずれかとして機能するさまざまなシステムからやってきます



FIWARE におけるコンテキスト管理

- FIWARE Context Broker GE は、スマート対応システムの要件に準拠したコンテキスト情報を管理するためのシンプルで強力な標準 API である **NGSI API** を実装しています
- FIWARE **NGSI API** は **Restful** です：Web /バックエンドのプログラマーはすぐに使いこなせます

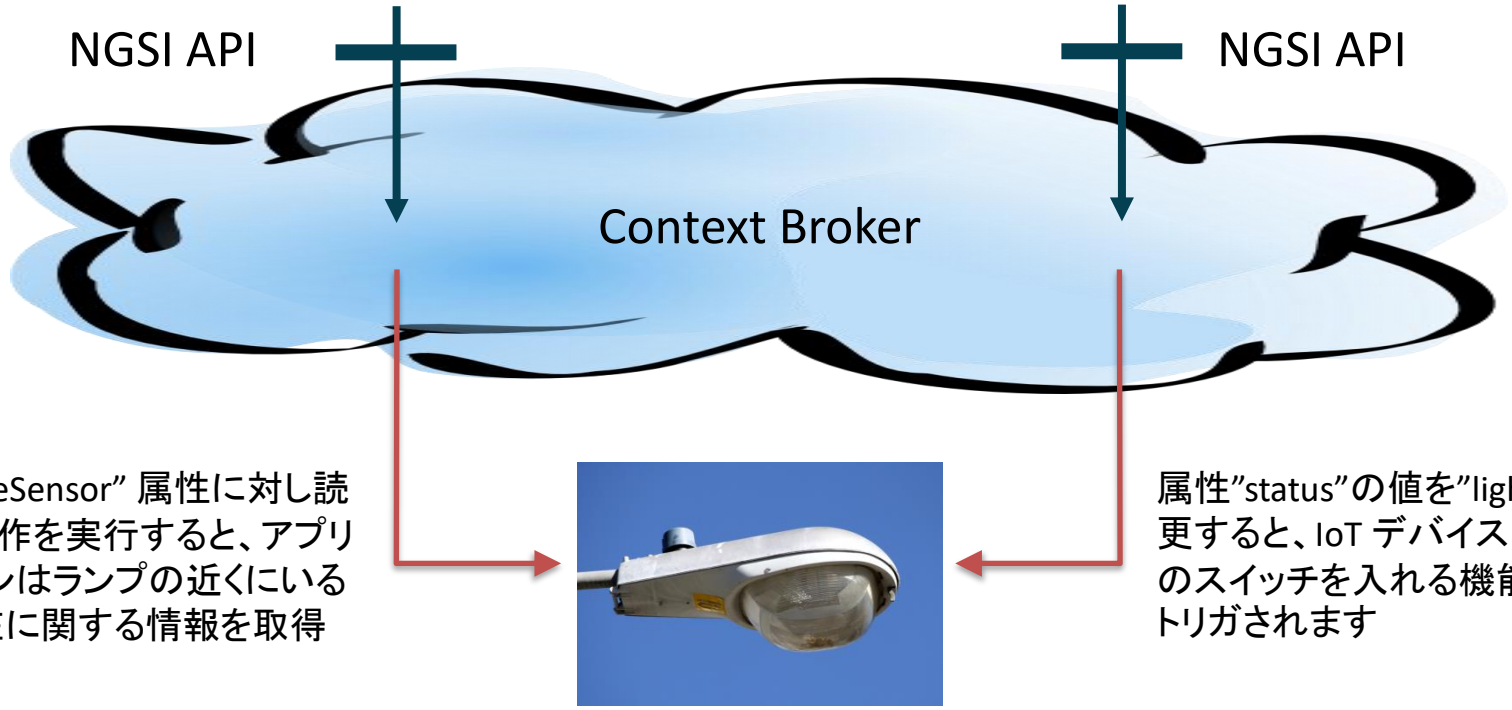


FIWARE NGSI: “IoT用 SNMP”

- IoT デバイスからのデータの取得、または IoT デバイス上での動作は、簡単になり、Context Broker を使用してコンテキスト・エンティティにリンクされた属性の値を読み取る/変更するだけです

GET /v2/entities/lamp1/attrs/presenceSensor

PUT /v2/entities/lamp1/attrs/status/value
“light on”



“presenceSensor” 属性に対し読み取り操作を実行すると、アプリケーションはランプの近くにいる人の存在に関する情報を取得できます

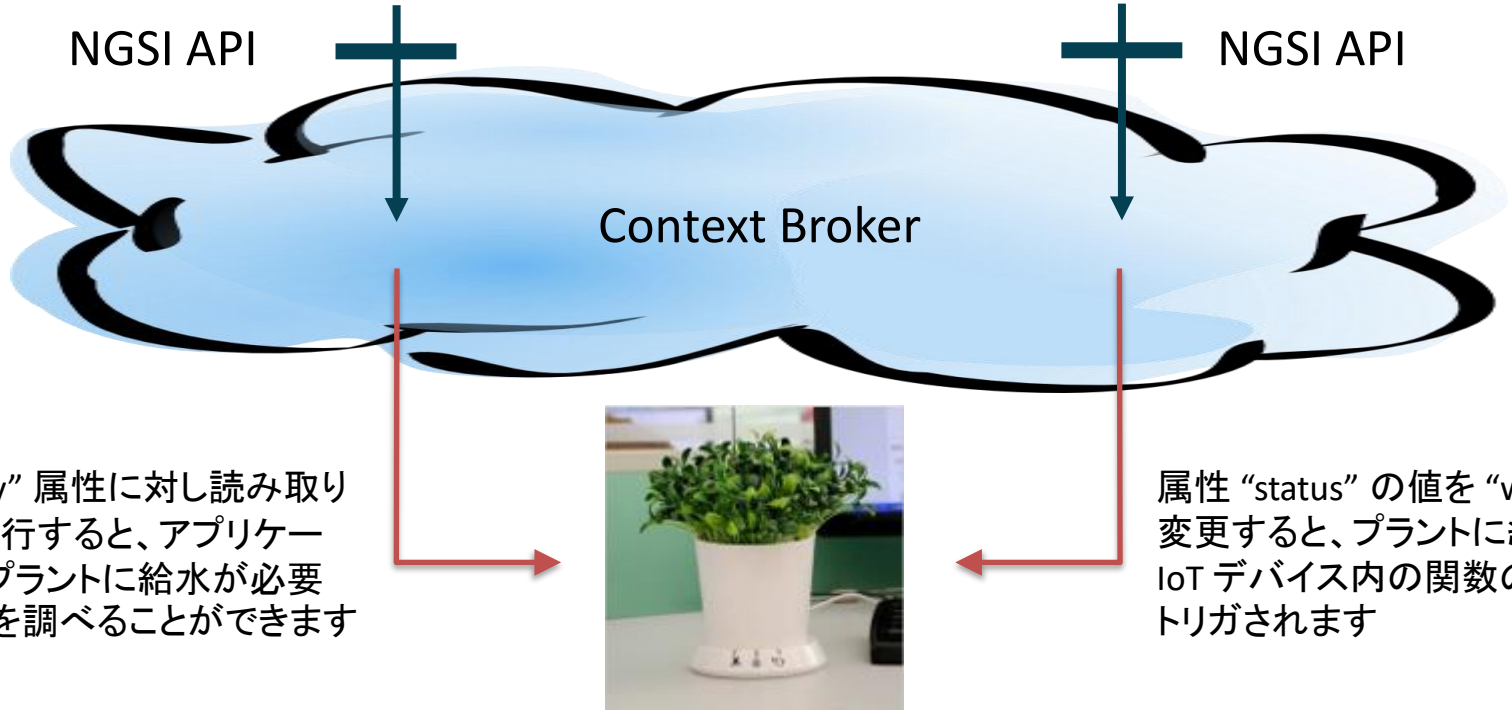
属性”status”の値を”light on”に変更すると、IoT デバイス内のランプのスイッチを入れる機能の実行がトリガされます

Internet of Things へ接続

- IoT デバイスからのデータの取得、または IoT デバイス上での動作は、簡単になり、Context Broker を使用してコンテキスト・エンティティにリンクされた属性の値を読み取る/変更するだけです

GET /v2/entities/lamp1/attrs/humidity

PUT /v2/entities/lamp1/attrs/status/value
“watering”

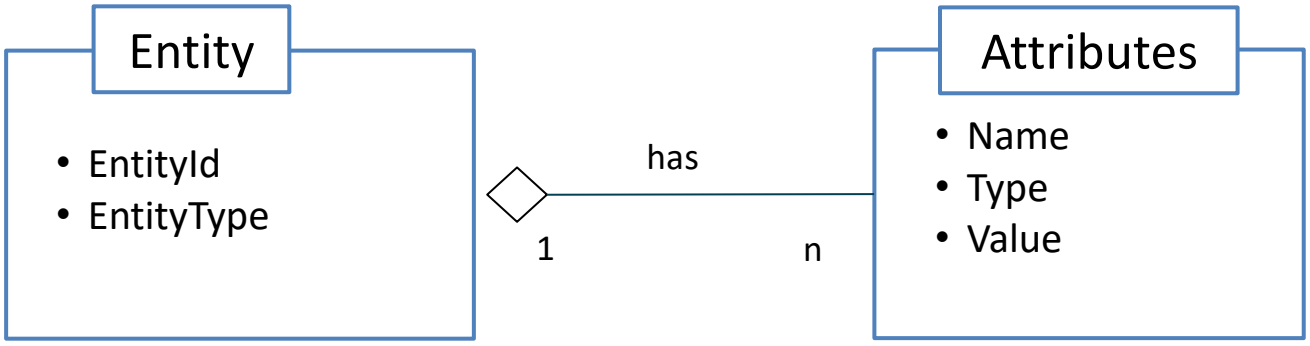


“humidity” 属性に対し読み取り操作を実行すると、アプリケーションはプラントに給水が必要かどうかを調べることができます

属性 “status” の値を “watering” に変更すると、プラントに給水する IoT デバイス内の関数の実行がトリガされます

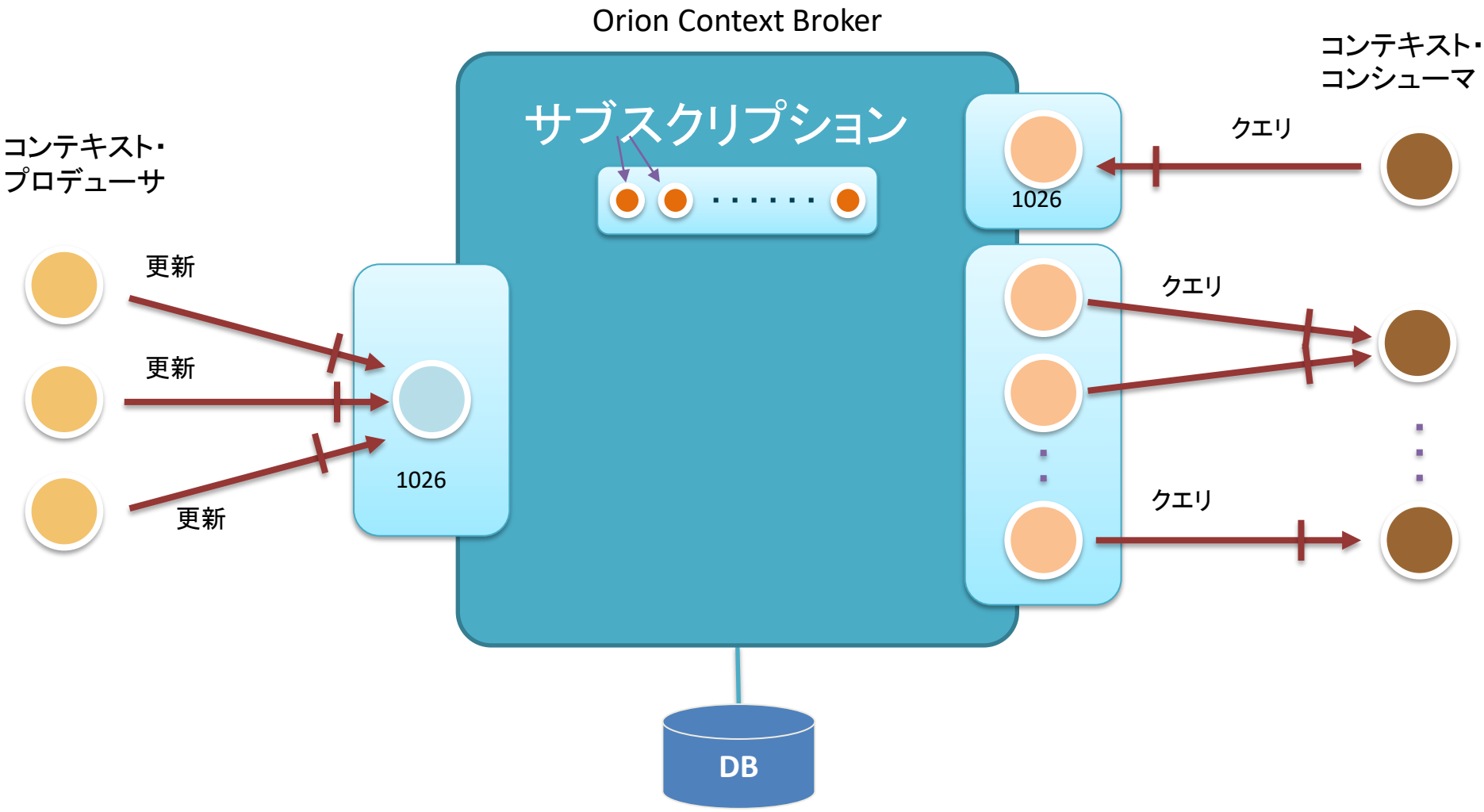
Orion Context Broker

- 主な機能:
  - コンテキスト管理
  - コンテキスト可用性管理 (高度なトピック) (*)
- HTTP と RESTベース
  - JSON ペイロードのサポート
- NGSIのコンテキストは、**entity-attribute** モデルに基づいています:



(*) With some limitations in current Orion version, see https://github.com/telefonicaid/fiware-orion/blob/master/doc/manuals.jp/orion-api.md#registration-implementation-differences

Orion Context Broker とは



Orion Context Broker – ヘルス･チェック

GET <cb_host>:1026/version

```
{
 "orion" : {
  "version" : "3.8.0",
  "uptime" : "0 d, 0 h, 0 m, 5 s",
  "git_hash" : "...",
  "compile_time" : "nodate",
  "compiled_by" : "fermin",
  "compiled_in" : "centollo",
  "release_date" : "nodate",
  "machine" : "x86_64",
  "doc" : "https://fiware-orion.readthedocs.org/en/3.8.0/"
  "libversions": …
 }
}
```

Orion Context Broker 基本オペレーション

エンティティ (Entities)

- GET /v2/entities
  - すべてのエンティティを取得
- POST /v2/entities
  - エンティティを作成
- GET /v2/entities/{entityID}
  - エンティティを取得
- [PUT|PATCH|POST] /v2/entities/{entityID}
  - エンティティを更新 (異なる “フレイバー”)
- DELETE /v2/entities/{entityID}
  - エンティティを削除

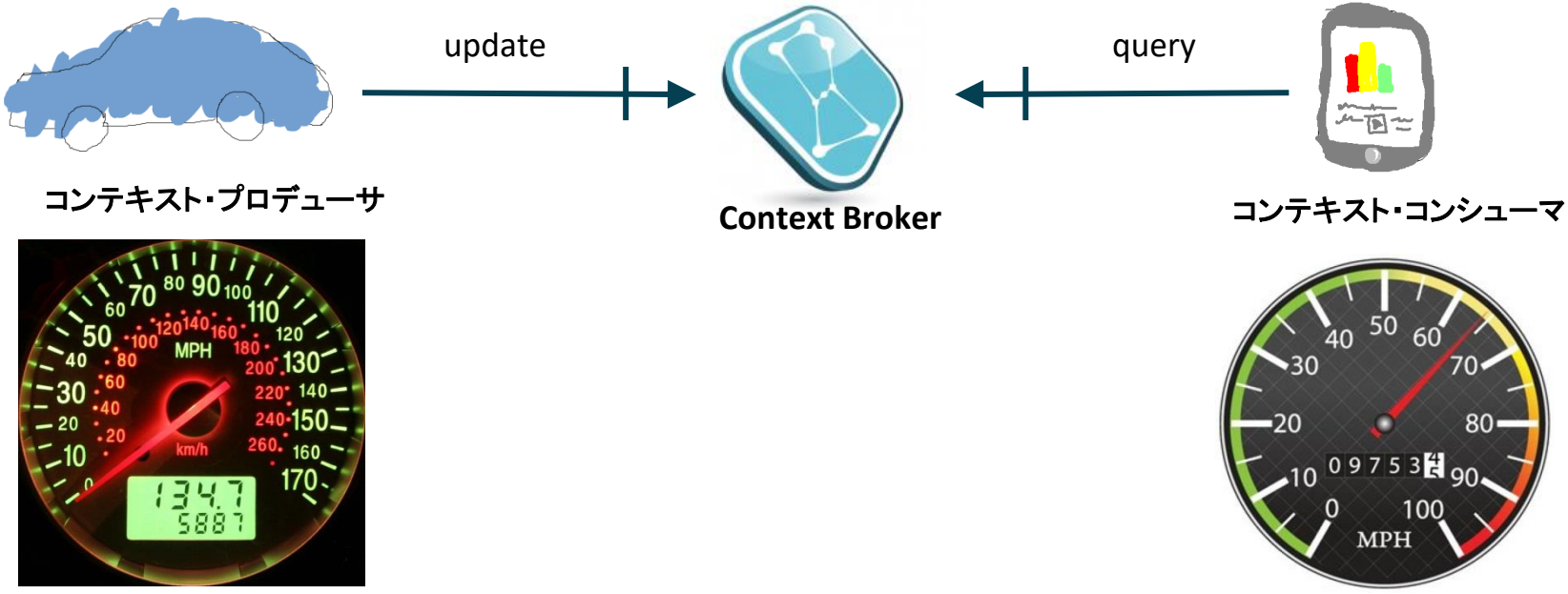
Orion Context Broker 基本オペレーション

属性 (Attributes)

- GET /v2/entities/{entityID}/attrs/{attrName}
  - 属性の取得
- PUT /v2/entities/{entityID}/attrs/{attrName}
  - 属性の更新
- DELETE /v2/entities/{entityID}/attrs/{attrName}
  - 属性の削除
- GET /v2/entities/{entityID}/attrs/{attrName}/value
  - 属性値の取得
- PUT /v2/entities/{entityID}/attrs/{attrName}/value
  - 属性値の更新

Context Broker operations: データの 作成 とプル

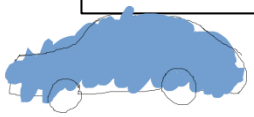
- **コンテキスト・プロデューサ**は、Context Broker に対する**更新(update)**操作を呼び出すことによって、データ/コンテキスト要素を公開します
- **コンテキスト・コンシューマ**は、Context Broker に対する**クエリ(query)**操作を呼び出すことによって、データ/コンテキスト要素を取得できます



FIWARE

使用例: Car の作成

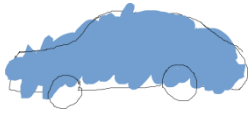
```
POST <cb_host>:1026/v2/entities
Content-Type: application/json
...

{
  "id": "Car1",
  "type": "Car",
  "speed": {
    "type": "Float",
    "value": 98
  }
}
```



URL に "?options=upsert" を追加できます。
その場合、エンティティが既に存在する場合に更新されます。

201 Created

使用例: Car Speed の更新 (1)



```
PUT <cb_host>:1026/v2/entities/Car1/attrs/speed
Content-Type: application/json
...

{
   "type": "Float",
   "value": 110
}
```

id があいまいなケースでは、"?type=Car"を使用してエンティティタイプを指定できます

```
204 No Content
...
```

FIWARE

使用例: Car Speed クエリ (1)

GET <cb_host>:1026/v2/entities/Car1/attrs/**speed**

200 OK
Content-Type: application/json
...

```
{
   "type": "Float",
   "value": 110,
   "metadata": {}
}
```

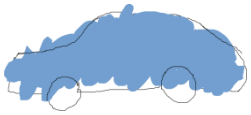
エンティティ URL を使用してエンティティのすべての属性を取得できます:
GET/v2/entities/Car1/attrs

FIWARE

使用例: Car Speed 更新 (2)

PUT <cb_host>:1026/v2/entities/Car1/attrs/speed/**value**
Content-Type: **text/plain**
...

115



204 No Content
...

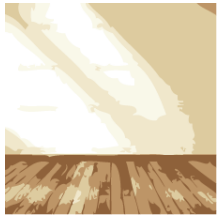
使用例: Car Speed クエリ (2)

```
GET <cb_host>:1026/v2/entities/Car1/attrs/speed/value
Accept: text/plain
```

```
200 OK
Content-Type: text/plain
...

115.000000
```

使用例: Room 作成 (1)

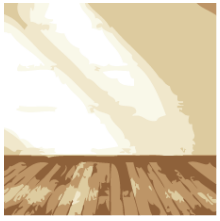
```
POST <cb_host>:1026/v2/entities
Content-Type: application/json
...

{
  "id": "Room1",
  "type": "Room",
  "temperature": {
    "type": "Float",
    "value": 24
  },
  "pressure": {
    "type": "Integer",
    "value": 718
  }
}
```



```
201 Created
...
```

使用例: Room 更新 (1)

```
PATCH <cb_host>:1026/v2/entities/Room1/attrs
Content-Type: application/json
...

{
  "temperature“: {
    "type": "Float",
    "value": 25
  },
  "pressure": {
    "type": "Integer",
    "value":  720
  }
}
```



```
204 No Content
...
```

FIWARE

使用例: Room クエリ (1)

GET <cb_host>:1026/v2/entities/**Room1**/attrs

200 OK
Content-Type: application/json
...


```
{
   "pressure": {
      "type": "Integer",
      "value": 720,
      "metadata": {}
   },
   "temperature": {
      "type": "Float",
      "value": 25,
      "metadata": {}
   }
}
```


使用例: Room クエリ (2)

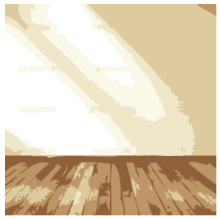
GET <cb_host>:1026/v2/entities/Room1/attrs?**options=keyValues**

```
200 OK
Content-Type: application/json
...

{
   "pressure": 720,
   "temperature": 25
}
```

FIWARE

使用例: Room 作成 (2)

```
POST <cb_host>:1026/v2/entities
Content-Type: application/json
...

{
  "id": "Room2",
  "type": "Room",
  "temperature": {
    "type": "Float",
    "value": 29
  },
  "pressure": {
    "type": "Integer",
    "value": 730
  }
}
```



```
201 Created
...
```

使用例: フィルタ (1)

```
GET <cb_host>:1026/v2/entities?options=keyValues&q=temperature>27
```

```
200 OK
Content-Type: application/json
...

[
   {
      "id": "Room2",
      "pressure": 730,
      "temperature": 29,
      "type": "Room"
   }
]
```

使用例: フィルタ (2)

```
GET <cb_host>:1026/v2/entities?options=keyValues&q=pressure==715..725
```

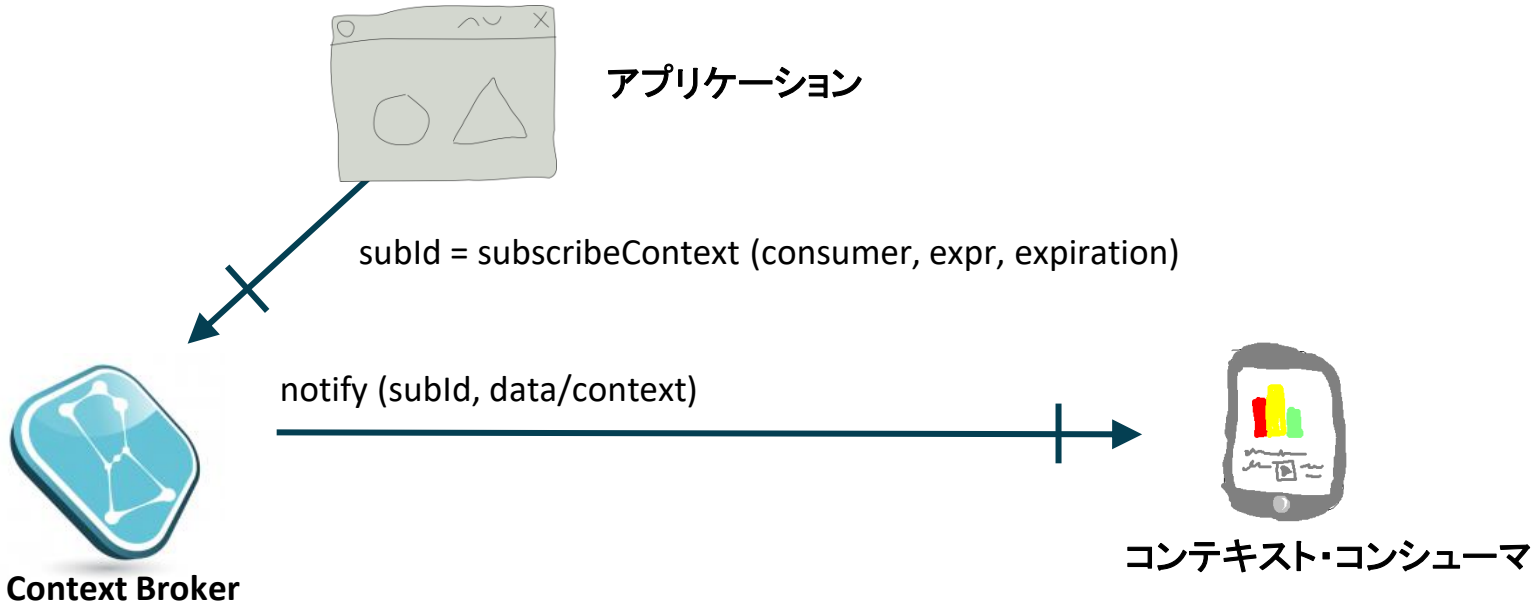
フィルタリングのためのシンプル・クエリ言語の完全な記述は、NGSIv2仕様書に記載されています。

```
200 OK
Content-Type: application/json
...

[
   {
      "id": "Room1",
      "pressure": 720,
      "temperature": 25,
      "type": "Room"
   }
]
```

Context Broker 操作: データのプッシュ

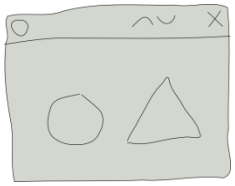
- **コンテキスト・コンシューマ**は、サブスクライブ(**subscribe**)操作を使用して特定の条件を満たすコンテキスト情報を受信するためにサブスクライブすることができます。このようなサブスクリプションには有効期限があります。

- Context Brokerは、サブスクライブ操作をしたコンテキスト・コンシューマに、通知(**notify**)操作により、コンテキスト情報の更新を通知します。



使用例: サブスクリプション (Subscription)

```
POST <cb_host>:1026/v2/subscriptions
Content-Type: application/json
...

{
  "subject": {
    "entities": [
      {
        "id": "Room1",
        "type": "Room"
      }
    ],
    "condition": {
      "attrs": [ "temperature"  ]
      }
    },
   "notification": {
    "http": {
      "url": "http://<host>:<port>/publish"
    },
    "attrs": [  "temperature"   ]
  },
  "expires": "2026-04-05T14:00:00.000Z"
}
```



```
201 Created
Location: /v2/subscriptions/51c0ac9ed714fb3b37d7d5a8
...
```

使用例: 通知 (Notification)

25

19

使用例: 通知 (Notification)

```
POST /publish HTTP/1.1
Content-type: application/json; charset=utf-8
Ngsiv2-AttrsFormat: normalized
…

{
   "subscriptionId": "574d720dbef222abb860534a",
   "data": [
      {
         "id": "Room1",
         "type": "Room",
         "temperature": {
            "type": "Float",
            "value": 19,
            "metadata": {}
         }
      }
   ]
}
```

既存のサブスクリプションの一覧表示

GET <cb_host>:1026/v2/subscriptions

サブスクリプション・オブジェクト(そのすべてのフィールドを含む)の完全な説明は、NGSIv2仕様書に記載されています

200 OK
Content-Type: application/json
…


```
[{
   "id": " 51c0ac9ed714fb3b37d7d5a8 ",
   "expires": "2026-04-05T14:00:00.000Z",
   "status": "active",
   "subject": {
      "entities": [{
         "id": "Room1",
         "type": "Room"
      }],
      "condition": {
         "attrs": ["temperature"]
      }
   },
   "notification": {
      "timesSent": 3,
      "lastNotification": "2016-05-31T11:19:32.000Z",
      "lastSuccess": "2016-05-31T11:19:32.000Z",
      "attrs": ["temperature"],
      "attrsFormat": "normalized",
      "http": {
         "url": "http://localhost:1028/publish"
      }
   }
}]
```

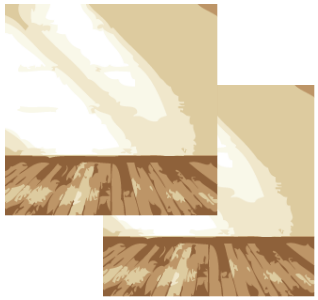
Orion Context Broker バッチ操作

- バッチ・クエリ と バッチ更新
- これらは、前述の RESTful 操作と機能的に同等です
- これらのすべては **POST** を動詞として使用し、***/v2/op*** URL プレフィックスを使用します。これには JSON ペイロードの操作パラメータも含まれます
- RESTful 操作では実現できない追加機能を実装しています。
  例えば、同じ操作で複数のエンティティを作成できます
- これらは代用ではなく、RESTful 操作を補完するものです

バッチ操作の例: 複数ルームの作成

```
POST <cb_host>:1026/v2/op/update
Conten-Type: application/json
...

{
  "actionType": "append",
  "entities": [
    {
      "type": "Room",
      "id": "Room3",
      "temperature": {
        "value": 21.2,
        "type": "Float"
      },
      "pressure": {
        "value": 722,
        "type": "Integer"
      }
    },
...
```

```
...
    {
      "type": "Room",
      "id": "Room4",
      "temperature": {
        "value": 31.8,
        "type": "Float"
      },
      "pressure": {
        "value": 712,
        "type": "Integer"
      }
    }
  ]
}
```



```
201 Created
...
```

Orion を使用するには? (Docker コンテナ)

- あなたのシステムに Docker をインストールします
- https://github.com/telefonicaid/fiware-orion/tree/develop/docker にあるドキュメントを見てください
- クイック・ガイド

```
git clone https://github.com/telefonicaid/fiware-orion.git
cd fiware-orion/docker
sudo docker-compose up
```

- これで完了!
  - curl localhost:1026/version
- Orion Docker を1分以内にインストールする
  - https://www.youtube.com/watch?v=6taR7e20H9U

これを試したいですか？

- FIWARE リファレンス・チュートリアル・アプリケーションを見てください
  - git clone https://github.com/Fiware/tutorials.TourGuide-App.git
  - cd tutorials.TourGuide-App/
  - docker-compose up orion
  - curl localhost:1026/version
- ルートディレクトリの README.md の説明
- Postman セッションを開いて、試します
  - Postman コレクション: https://github.com/Fiware/tutorials.TourGuide-App/blob/develop/contrib/CampusParty2016.postman_collection

NGSI Go

NGSI Go は、FIWARE NGSIv2 API と Orion Admin API をサポートするコマンドライン・インターフェイスです。これは強力なツールであり、簡単に利用できます。Orionのためのさまざまなコマンドがあります:

- NGSI エンティティ、サブスクリプション、および、レジストレーションを管理するためのNGSIコマンド
- FIWARE Orionの管理コマンド
- Orion のバージョン表示コマンド
- エンティティを一度にコピーまたは削除
- サブスクリプションまたはレジストレーションのテンプレートの作成
- ノーティフィケーションの受信
- Keystone (Thinking Cities プラットフォーム), Keyrock などのアクセストークンの統合管理

- ソースコード: https://github.com/lets-fiware/ngsi-go
- インストール方法: https://github.com/lets-fiware/ngsi-go#install
- ドキュメント: https://ngsi-go.letsfiware.jp/

* NGSI Go は、FIWARE コミュニティによって作成されたサードパーティのオープンソース・ソフトウェアです

もっと知りたいですか？

- リファレンス
  - プレゼンテーション
    https://github.com/telefonicaid/fiware-orion#introductory-presentations
  - NGSIv2 仕様
    - https://github.com/telefonicaid/fiware-orion/blob/master/doc/manuals.jp/orion-api.md
  - ドキュメント (ReadTheDocs):
    - https://fiware-orion.readthedocs.io/en/master/
    - https://fiware-orion.letsfiware.jp/
  - StackOverflow での Orion サポート
    - “fiware-orion” タグをつけて、質問してください
    - http://stackoverflow.com/questions/tagged/fiware-orion で既存のQ&Aを探してください

Open APIs for Open Minds

コンテキスト情報管理:高度なトピック (Managing Context Information at Large Scale: Advanced Topics)

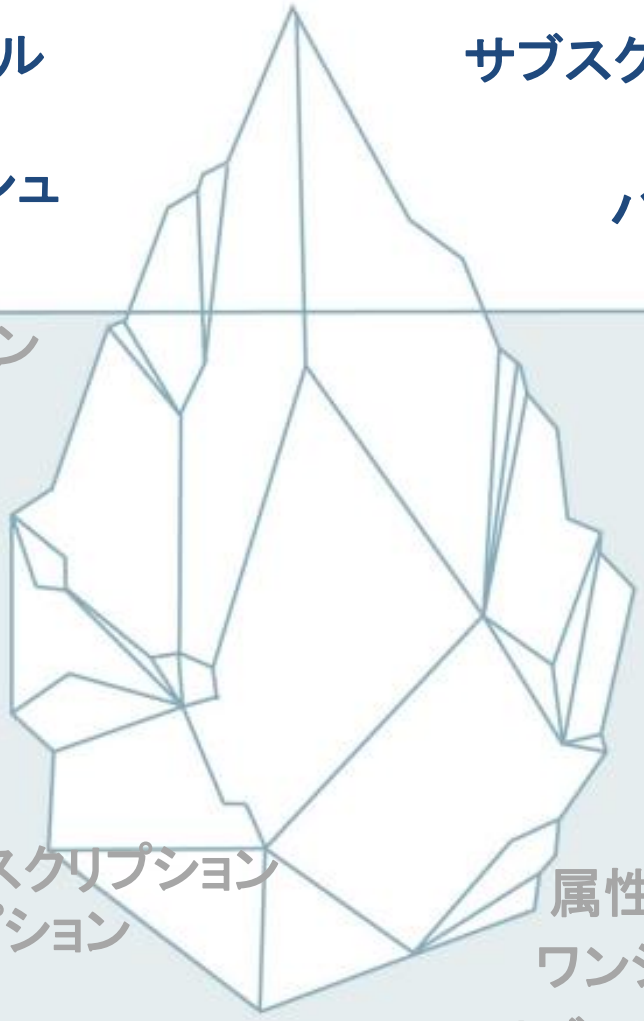
Fermín Galán Márquez - fermin.galanmarquez@telefonica.com

(Reference Orion Context Broker version: 3.8.0)

(Translated into Japanese by Kazuhito Suda k@fisuda.jp)

FIWARE

Orion の高度な機能



データの作成とプル

データのプッシュ

サブスクリプションと通知

バッチ操作

ページネーション
ジオ・ロケーション
更新演算子
DateTime サポート
クエリ・フィルタ
無制限のテキスト属性
フロー制御
変更タイプに基づくサブスクリプション
カバード・サブスクリプション
特別な属性/メタデータ
複合属性/メタデータ値
レジストレーションとコンテキスト・プロバイダ

タイプ・ブラウジング
メタデータ
カスタム通知
通知ステータス
MQTT 通知
一時的なエンティティ
マルチテナンシー
属性/メタデータのフィルタリング
ワンショット・サブスクリプション
サブスクリプションの自動無効化
サブスクリプションの強化

FIWARE

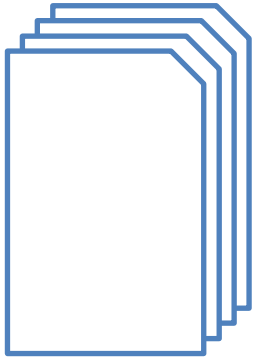
高度な機能

- ページネーション
- メータデータ
- 複合属性/メタデータ値
- タイプ・ブラウジング
- ジオ・ロケーション
- 更新演算子
- クエリ・フィルター
- DateTime サポート
- 一時的なエンティティ
- 無制限のテキスト属性
- フロー制御
- カスタム通知
- 通知ステータス
- ワンショット・サブスクリプション
- サブスクリプションの自動無効化
- 変更タイプに基づくサブスクリプション
- カバード・サブスクリプション
- MQTT 通知
- サブスクリプションの強化
- 属性/メタデータ・フィルタリングと特殊属性/メタデータ
- レジストレーションとコンテキストプロバイダ
- マルチテナンシー
- サービス・パス
- CORS
- プライベート・ネットワークでのサービスの通知

FIWARE

ページネーション (Pagination)

- ページネーションは、クライアントが、クエリおよびディスカバリのリクエストよる多数のレスポンスを整理するのに利用できます。
- 3つの URI パラメータ:
  - limit
    - ページあたりの要素数 (デフォルト: 20, 最大: 1000)
  - offset
    - スキップする要素の数 (デフォルト: 0)
  - count (オプション)
    - 合計要素を返します (デフォルト: 返しません)



ページネーション (Pagination)

- 例, 最初の100エントリをクエリする:
  - GET <orion_host>:1026/v2/entities?limit=100&options=count
- 最初の100個の要素が返され、レスポンスに次のヘッダーが返されます:
  - Fiware-Total-Count: 322
- 今、322の要素があります。Orion Context Broker にそれらをクエリし続けることができます:
  - GET <orion_host>:1026/v2/entities?offset=100&limit=100
  - GET <orion_host>:1026/v2/entities?offset=200&limit=100
  - GET <orion_host>:1026/v2/entities?offset=300&limit=100

ページネーション (Pagination)

- デフォルトでは、結果はエンティティの作成日順に並べられます
- この動作は **orderBy** URI パラメータを使用してオーバーライドできます
  - コンマで区切られた属性のリスト(システム/組み込み属性を含む)、 id (エンティティIDの場合) および、型 (エンティティタイプの場合)。結果は最初のフィールドで並べられます。続いて、結果は02番名のフィールドなどによって順序付けされます。フィールド名の前の"!" は、順序が逆になっていることを意味します。
- 例: 温度よる昇順で、次に湿度よる降順で最初の10エントリを取得します

  GET <orion_host>:1026/v2/entities?limit=20&offset=0&orderBy=temp,!humidity
- **dateCreated** と **dateModified** は、エンティティの作成および変更日ごとの順序付けにそれぞれ使用できます

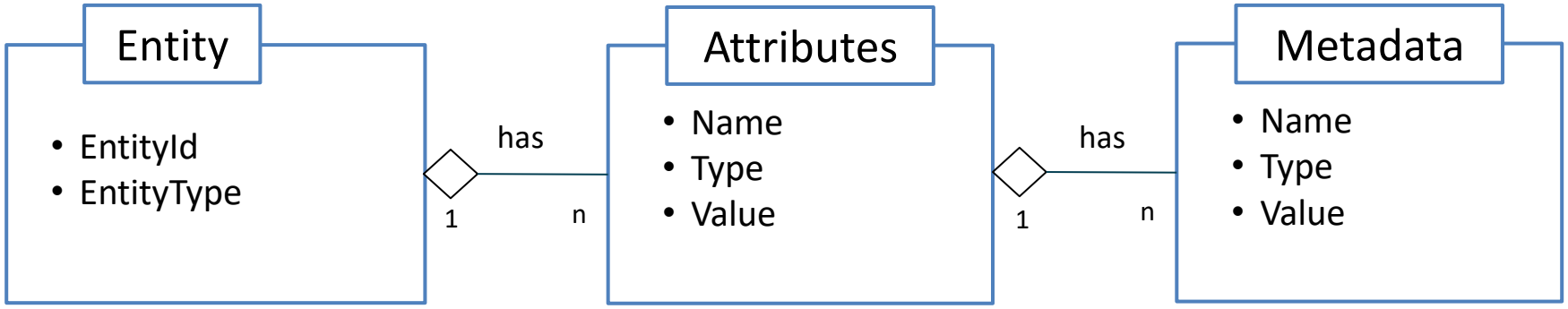
メタデータ (Metadata)

- ユーザは属性にメタデータを添付できます
- 予約されたメタデータ: dateCreated, dateModified, previousValue, actionType, ignoreType
- **overrideMetadata** オプションを使用した場合を除き、デフォルトでは、メタデータは更新時に削除されません
- 例:

```
...
"temperature": {
    "type": "Float",
    "value": 26.5,
    "metadata": {
     "accuracy": {
      "type": "Float",
      "value": 0.9
     }
    }
}
...
```

```
...
"temperature": {
    "type": "Float",
    "value": 26.5,
    "metadata": {
     "average": {
      "type": "Float",
      "value": 22.4
     }
    }
}
...
```

完全な NGSI モデル



複合属性/メタデータ値
(Compound Attribute/Metadata Values)

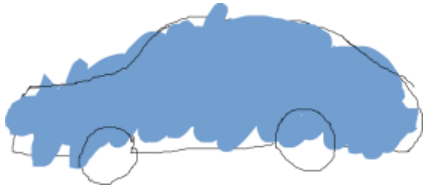
- 属性とメタデータは、構造化された値を持つことができます。Vectors と key-value マップをサポートしています
- JSONのオブジェクトと配列に直接マップされます

複合属性/メタデータ値

(Compound Attribute/Metadata Values)

- 例：4つのタイヤの空気を車のエンティティの複合属性として表現したい車があります。このような車のエンティティを作成します：

```
{
   "type": "Car",
   "id": "Car1",
   "tirePressure": {
      "type": "kPa",
      "value": {
         "frontRight": 120,
         "frontLeft": 110,
         "backRight": 115,
         "backLeft": 130
      }
   }
}
```



タイプ・ブラウジング (Type Browsing)

- GET /v2/types
  - Orionに現在あるすべてのエンティティ・タイプのリストを取得します。対応する属性とエンティティ数も含まれます
- GET /v2/types/{typeID}
  - エンティティタイプに関連付けられた属性およびエンティティの数を取得します

ヒント

GET /v2/contextTypes?options=values

追加情報なしですべてのエンティティ・タイプのリストを取得します

FIWARE

ジオ・ロケーション (Geo-location)

- エンティティは、その場所を指定する属性を持つことができます
- いくつかの属性タイプを使用できます
  - geo:point (ポイント)
  - geo:line (ライン)
  - geo:box (ボックス)
  - geo:polygon (ポリゴン)
  - geo:json (GeoJson 標準で任意のジオメトリ)

- 例:マドリードというエンティティを作成

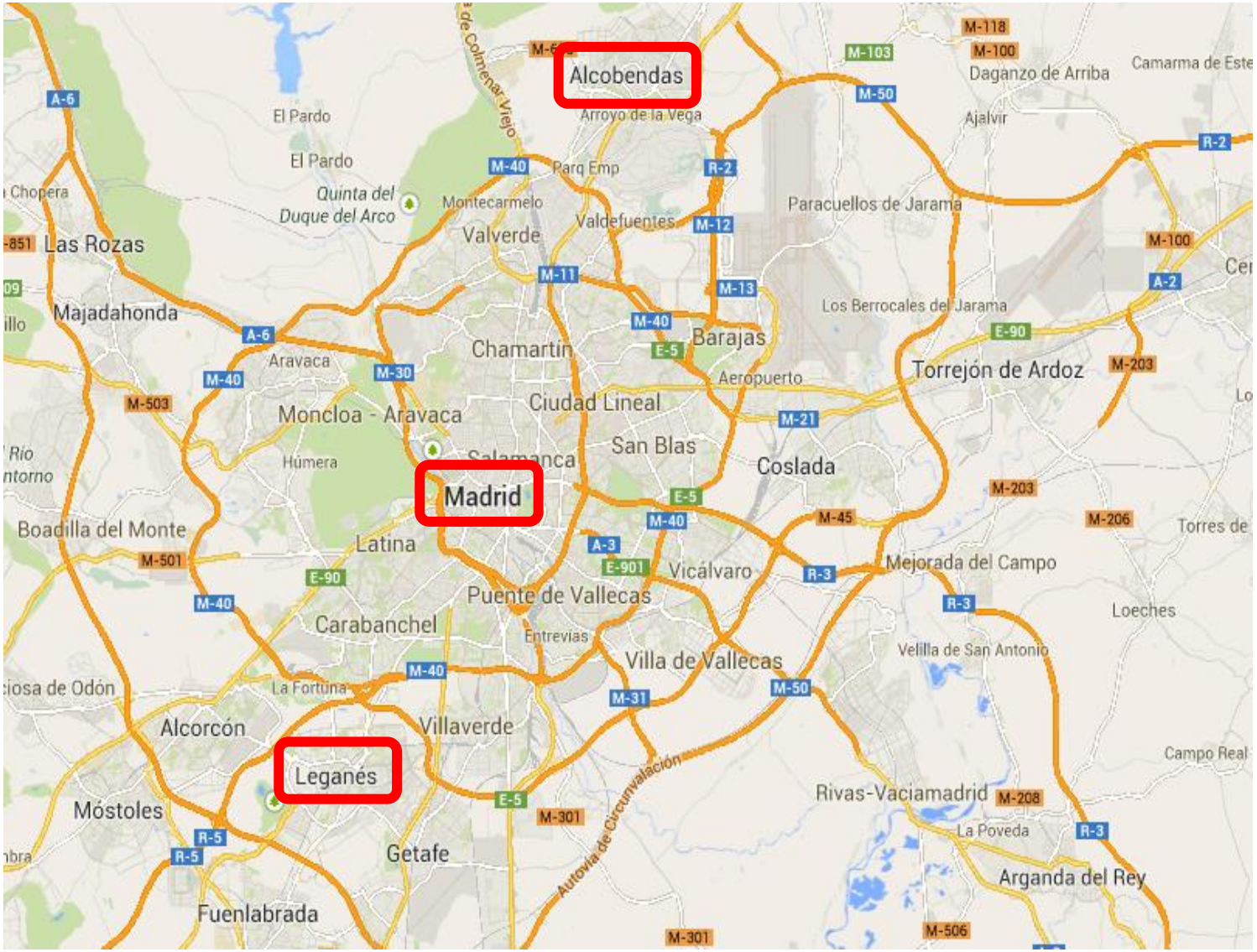
…そしていくつかの町を作成:

  - Leganés (レガネス)
  - Alcobendas (アルコベンダス)

```
POST <cb_host>:1026/v2/entities
{
   "type": "City",
   "id": "Madrid",
   "position": {
      "type": "geo:point",
      "value": "40.418889, -3.691944"
   }
}
```

null も値としてサポートされており、”場所がない“ことを意味します。これは、場合によっては役立つ場合があります。

ジオ・ロケーション – サークル



ジオ・ロケーション – 最大距離

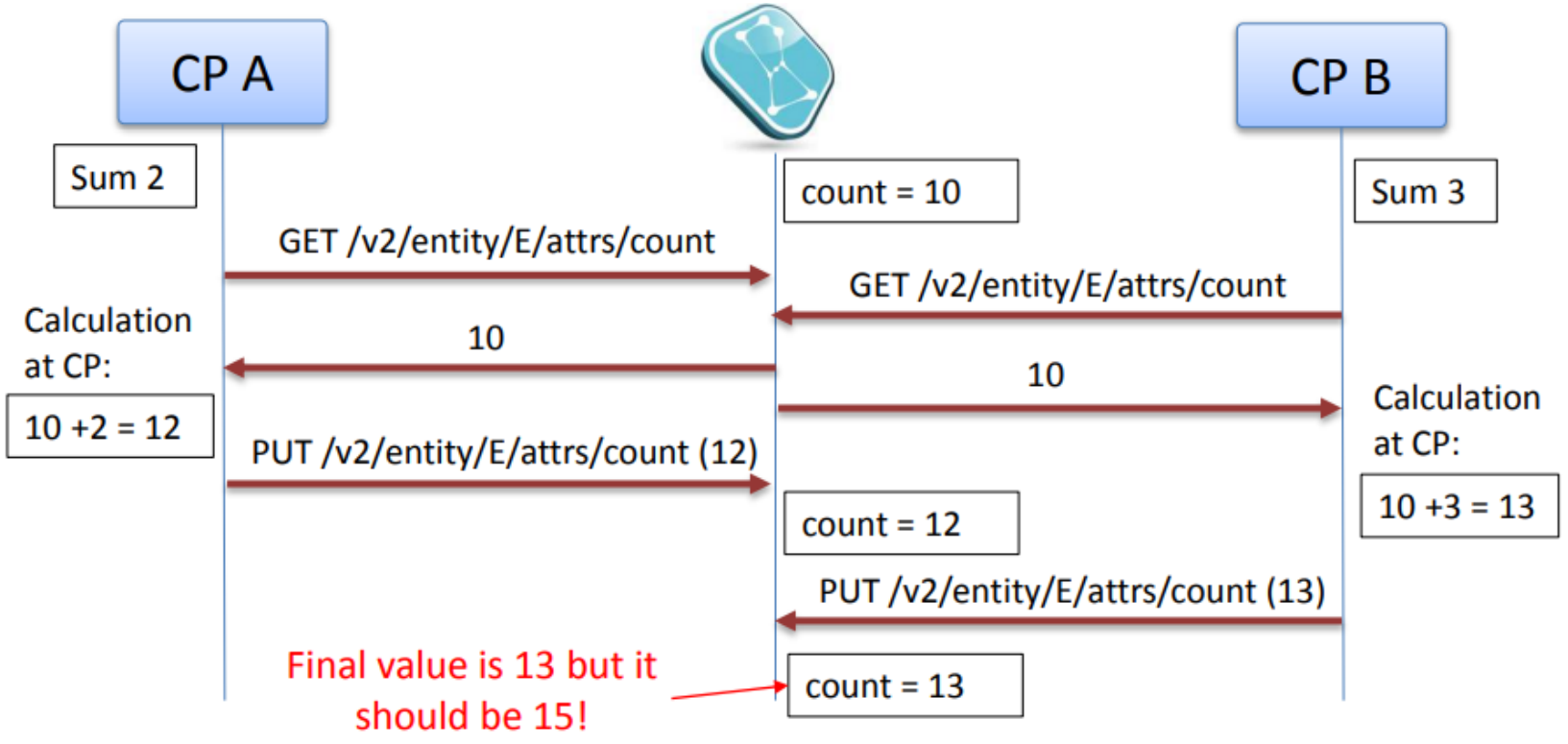
ジオ・ロケーション – 最小距離

さらなる、地理関係 (geo-relationships)

- **near** から離れて、次の georel を使用できます
  - georel=coveredBy
  - georel=intersects
  - georel=equals
  - georel=disjoint
- 詳細な説明については、”NGSIv2 仕様”を参照してください

更新演算子 (Update operators)

- 更新演算子を使用しない例：2つのコンテキスト・プロデューサが同じエンティティ属性を更新します
  - 複雑さ：コンテキスト・プロデューサはローカル計算を行う必要があります
  - 下の図に示すように、競合状態が発生する可能性があります



更新演算子 (Update operators)

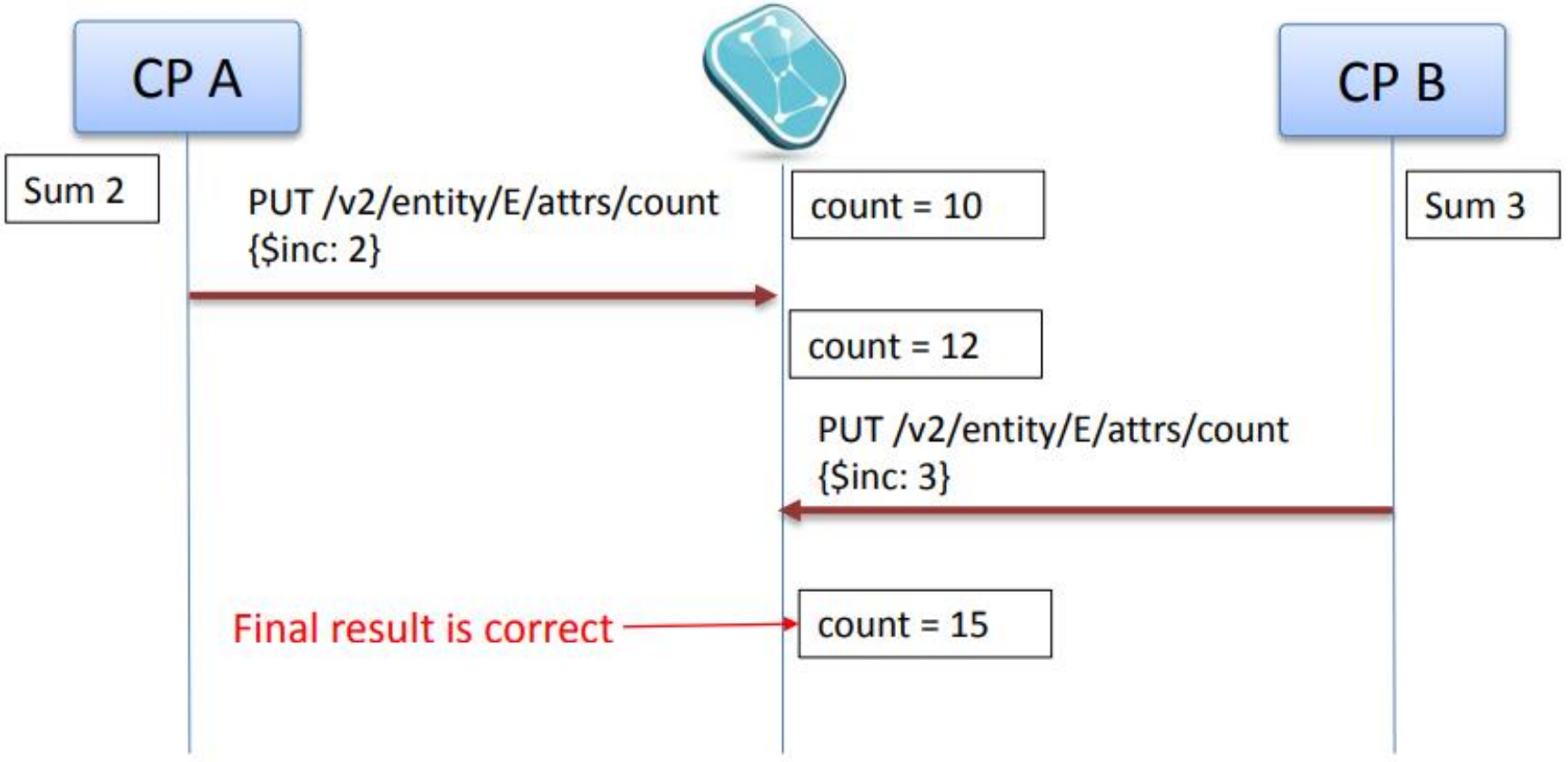
- 更新演算子は属性の更新に使用できます
- 例：”属性の値を2増やします”

```
…
"count": {
    "type": "Number",
    "value": { "$inc": 2 }
}
…
```

- 現在の演算子の完全なリスト：
  - $inc：指定された値だけ増加
  - $mul：指定された値を掛ける
  - $max, $min：提供された値と既存の値を比較した最大/最小
  - $push, $addToSet：配列に要素を追加
  - $pull, $pullAll：配列から要素を削除
  - $set, $unset：オブジェクトのサブキーを追加/削除

更新演算子 (Update operators)

- 更新演算子 **$inc** を使用した同じ例
  - コンテキスト･プロデューサでの計算の必要はありません (簡略化)
  - 競合状態は発生しない可能性があります



クエリ・フィルタ (Query filters)

- **GET /v2/entities** 操作
- エンティティ・タイプ によるフィルタ

  GET <cb_host>:1026/v2/entities?**type=Room**

- エンティティ **id** リスト によるフィルタ

  GET <cb_host>:1026/v2/entities?**id=Room1,Room2**

- エンティティ **id** のパターンによるフィルタ (正規表現)

  GET <cb_host>:1026/v2/entities?**idPattern=^Room[2-5]**

- エンティティ・タイプのパターンによるフィルタ (正規表現)

  GET <cb_host>:1026/v2/entities?**typePattern=T[ABC]**

- 地理上の位置によるフィルタ (**geographical location**)
  - 以前のスライドで詳しく説明しました
- フィルタは同時に使用できます (**AND**条件など)

クエリ・フィルタ (Query filters)

- **属性値によるフィルタ** (q)

GET <cb_host>:1026/v2/entities?**q=temperature>25**

属性名

GET <cb_host>:1026/v2/entities?**q=tirePressure.frontRight >130**

属性サブキー(複合属性値のみ)

- **メタデータ値によるフィルタ** (mq)

GET <cb_host>:1026/v2/entities?**mq=temperature.avg>25**

属性名

メタデータ名

GET <cb_host>:1026/v2/entities?**mq=tirePressure.accuracy.frontRight >90**

メタデータサブキー
(複合メタデータ値のみ)

- NGSIv2仕様の **q** および **mq** クエリ言語の詳細を参照してください

クエリ・フィルタ

- フィルタはサブスクリプションでも使用できます
  - id
  - タイプ
  - id パターン
  - タイプ・パターン
  - 属性値
  - メタデータ値
  - 地理上の位置

```
POST <cb_host>:1026/v2/subscriptions
...
{
  "subject": {
    "entities": [
      {
        "id": “Car5",
        "type": “Car"
      },
      {
        "idPattern": "^Room[2-5]",
        "type": "Room"
      },
      {
        "id": “D37",
        "typePattern": "Type[ABC]"
      },
    ],
    "condition": {
      "attrs": [ "temperature"  ],
      "expression":  {
        "q": "temperature>40",
        "mq": "humidity.avg==80..90",
        "georel": "near;maxDistance:100000",
        "geometry": "point",
        "coords": "40.418889,-3.691944"
      }
      }
    },
    ...
}
```

Datetime サポート

- Orion Context Broker は日付サポートを実装しています
  - ミリ秒単位の精度のISO ISO8601フォーマット (部分表現とタイムゾーンを含む) に基づいています
    - 構文の詳細については、https://github.com/telefonicaid/fiware-orion/blob/master/doc/manuals.jp/orion-api.md#datetime-support を参照ください
    - null もサポートされており、”日付なし”を意味します
  - 日時を表すには、予約属性タイプの **DateTime** を使用します
  - 日付ベースのフィルタをサポートします

Datetime サポート

```
POST /v2/entities
…
{
  "id": "John",
  "birthDate": {
    "type": "DateTime",
    "value": "1979-10-14T07:21:24.238Z"
  }
}
```

例: DateTime タイプを使用して *birthDate* 属性でエンティティ "John"を作成します

- 属性値の算術フィルタは、日付が数字であるかのように使用できます

```
GET /v2/entities?q=birthDate<1985-01-01T00:00:00
```

- エンティティの **dateModified** と **dateCreated** の特殊属性,エンティティの作成と最終変更のタイムスタンプを取得します
  - それらは以下を使用してクエリ・レスポンスで示されます **attrs=dateModified,dateCreated**
- エンティティの **dateModified** と **dateCreated** の特殊メタデータ, 属性の作成と最終変更のタイムスタンプを取得します
  - それらは以下を使用してクエリ・レスポンスで示されます **metadata=dateModified,dateCreated**

一時的なエンティティ (Transient entities)

```
POST /v2/entities
…
{
   "id": “Warning1",
   "dateExpires": {
   "type": "DateTime",
   "value": “2018-07-30T12:00:00Z"
  }
}
```

例: エンティティ Warning1 を作成し、2018年7月30日正午に有効期限を設定

- dateExpires 時間に達すると、Orion はエンティティを自動的に削除します
  - 実際、一時的なエンティティは、有効期限の60秒後まで (データベースの負荷が高い場合はもう少し)、データベースに残る可能性があります
- 注意してください！この方法で削除されたエンティティは回復できません
- dateModified および dateCreated (前述) と同様に、**dateExpires** はデフォルトでは取得されません
  - **attrs=dateExpires** を使用してクエリのレスポンスに表示されます
- **dateExpires** は、フィルタの他の DateTime 属性として使用できます。例:

```
GET /v2/entities?q=dateExpires<2018-08-01T00:00:00
```

無制限のテキスト属性 (Unrestricted text attributes)

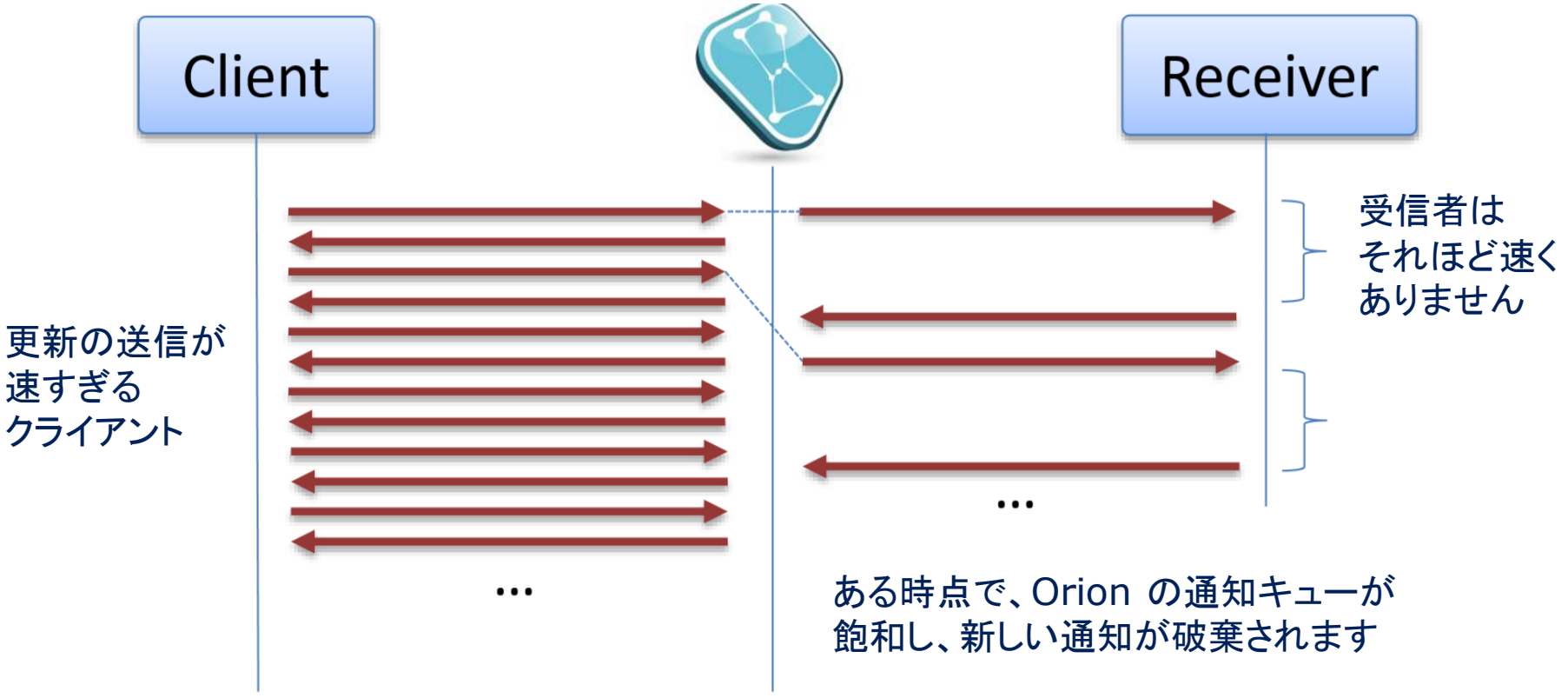
- デフォルトでは、Orion はインジェクション攻撃を回避するために許可される文字のセットを制限します
  - 禁止文字のリスト: https://github.com/telefonicaid/fiware-orion/blob/master/doc/manuals.jp/orion-api.md#general-syntax-restrictions
- 特殊な属性タイプ **TextUnrestricted** を使用して、属性値のこのチェックを回避できます

```
...
"description": {
    "type": "TextUnrestricted",
    "value": "Hey! I'm using <forbidden chars>"
}
...
```

- 自己責任で使用してください！

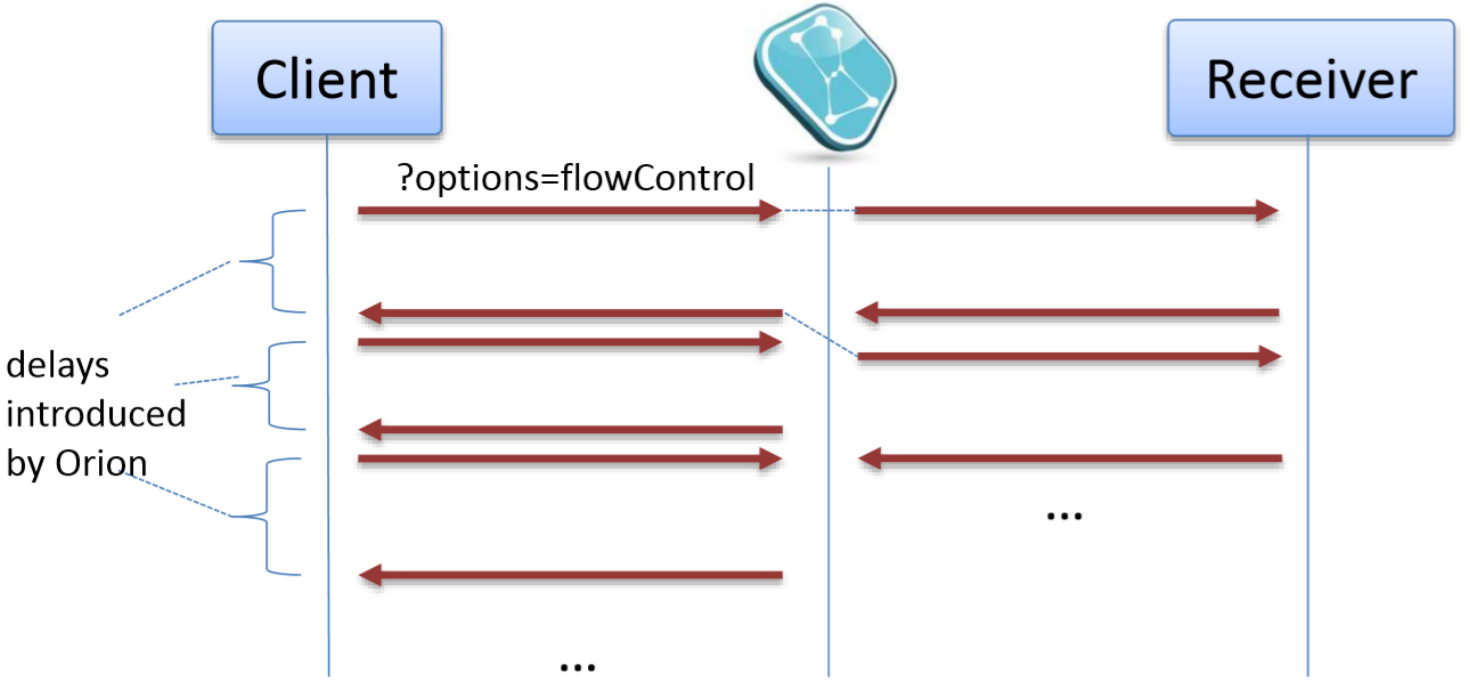
フロー制御 (Flow control)

- デフォルトでは、Orion の更新レスポンスは、これらの更新によってトリガーされる通知の送信から切り離されています
- これにより、更新を送信するクライアントが受信者の処理通知よりもはるかに速い場合、飽和が発生する可能性があります



フロー制御 (Flow control)

- Orion は、**-notifFlowControl** CLI オプションと更新リクエストの **flowControl** オプションに基づいて、フロー制御メカニズムを実装できます
- Orion はフロー制御を使用して、更新にすぐにレスポンスしません。通知キューの占有サイズに基づいて遅延が発生します (占有が多いほど、遅延も大きくなります)



フロー制御 (Flow control)

- フロー制御の詳細
  - 機能説明: https://fiware-orion.readthedocs.io/en/master/admin/perf_tuning/index.html#updates-flow-control-mechanism
  - 詳細な例/チュートリアル: https://github.com/telefonicaid/fiware-orion/blob/master/test/flowControlTest/README.md

カスタム通知 (Custom notifications)

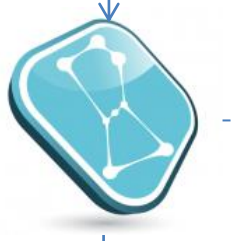
- 以前のスライドで定義された標準フォーマットとは別に、NGSIv2では**すべて**の通知の側面を再定義できます
- **http** の代わりに **httpCustom** が使用され、以下のサブフィールドがあります
  - URL クエリ・パラメータ
  - HTTP メソッド
  - HTTP ヘッダ
  - ペイロード (必ずしも JSON ではない!)
- **${..}** 構文に基づく単純なマクロ置換言語を使用して、**エンティティ・データ** (id, タイプ または 属性値) および**追加情報** (service, servicePath または authToken) で”ギャップを埋める”ことができます
  - 例外: これは HTTP メソッドのフィールドでは使用できません

カスタム通知 (Custom notifications)

Text based payload

```
PUT /v2/entities/DC_S1-D41/attrs/temp/value?type=Room
...
23.4
```

update



notification

```
PUT http://foo.com/entity/DC_S1-D41?type=Room
Content-Type: text/plain
Content-Length: 31

The temperature is 23.4 degrees
```

```
...
"httpCustom": {
 "url": "http://foo.com/entity/${id}",
 "headers": {
  "Content-Type": "text/plain"
 },
 "method": "PUT",
 "qs": {
  "type": "${type}"
 },
 "payload": "The temperature is ${temp} degrees"
 }
...
```

カスタム通知設定

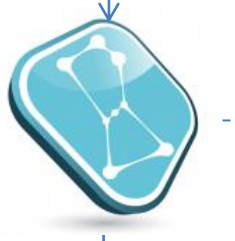
FIWARE

カスタム通知 (Custom notifications)

JSON based payload

```
PUT /v2/entities/DC_S1-D41/attrs?type=Room
...
{ "temp": {"value": 23.4, "type": "Number"},
"status": {"value": "on", "type": "Text"} }
```

update



notification

```
PUT http://foo.com/entity/DC_S1-D41?type=Room
Content-Type: application/json
Content-Length: 19

{ "t": 23.4, "s": "on"}
```

```
...
"httpCustom": {
 "url": "http://foo.com/entity/${id}",
 "method": "PUT",
 "qs": {
  "type": “${type}"
 },
 "json": {
  "t": “${temp}",
  "s": “${status}"
 }
}
...
```

カスタム通知設定

FIWARE

カスタム通知 (Custom notifications)

NGSI patching in payload

```
PUT /v2/entities/DC_S1-D41/attrs?type=Room
...
{ "A": {"value": "foo", "type": "Text"} }
```

update

notification

```
PUT http://foo.com/entity/DC_S1-D41
Content-Type: application/json
Content-Length: 140

{
 "id": "stamp:DC_S1-D41",
 "A": { "value": "foo", "type": "Text" },
 "A:map": { "value": "foo", "type": "Text" },
 "oldId": { "value": "DC_S1-D41", "type": "Text" }
}
```

```
...
"httpCustom": {
 "url": "http://foo.com/entity/${id}",
 "method": "PUT",
 "ngsi": {
  "id": "stamp:${id}"
  "A:map": { "value": “${A}", "type": "Text"}
  "oldId": { "value": “${id}", "type": "Text"}
 }
}
...
```

カスタム通知設定

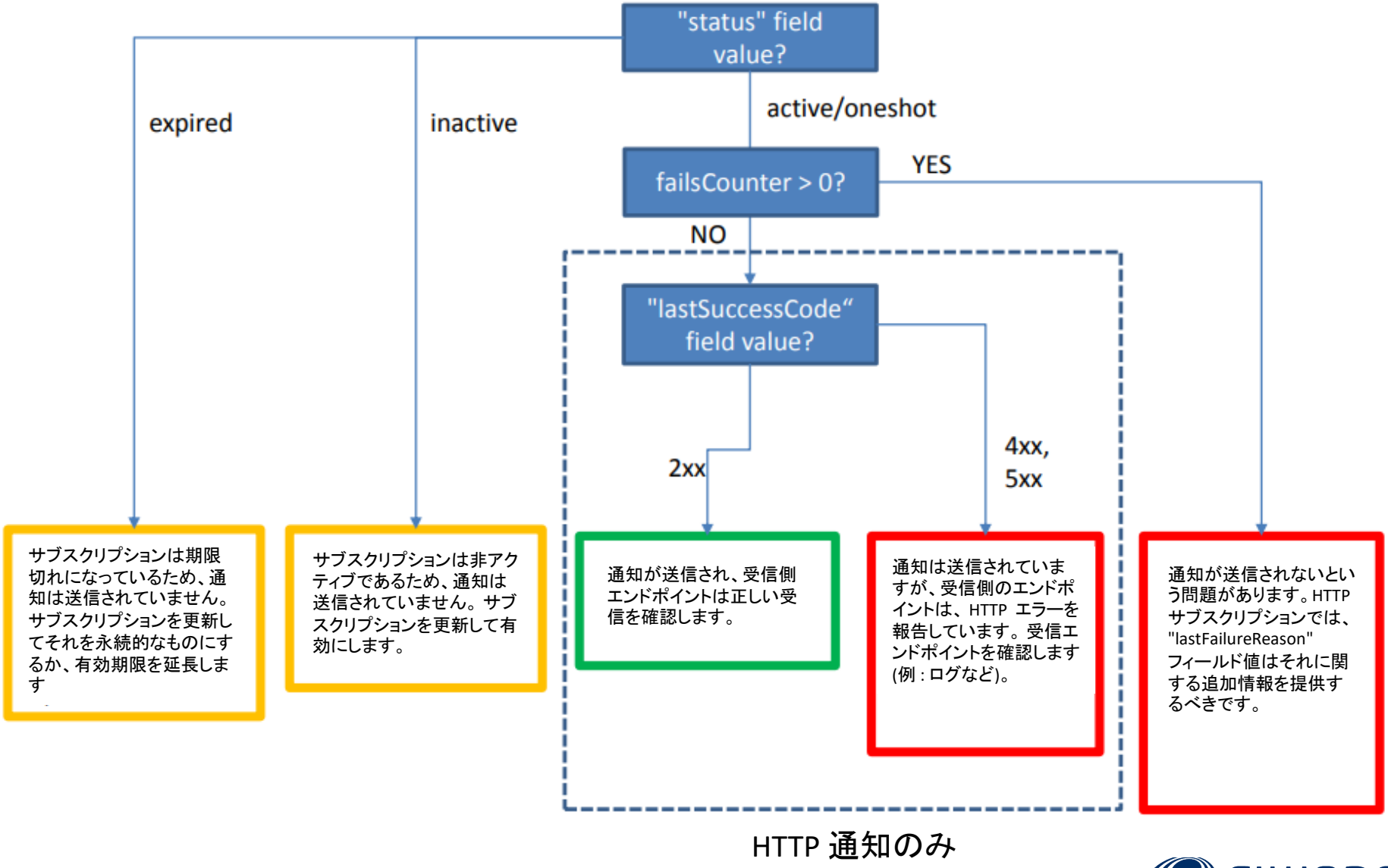
通知は NGSIv2 準拠の形式を
使用します (通常の通知と同様)!

通知ステータス (Notification status)

- **通知要素(notifications** element)の詳細情報
  - **timesSent**: 通知の合計回数(成功と失敗の両方)
  - **failsCounter:** 連続した通知の失敗数
    - failedCounter > 0 は、サブスクリプションが失敗状態にあることを意味します
  - **lastSuccess**: 通知が正常に送信された最後の時刻
  - **lastFailure**: 通知が試行され失敗した最後の時刻
  - **lastNotification**: 通知が送信された最後の時刻(成功または失敗のいずれか)
    - 結果として、lastNotification の値はlastFailure または lastSuccess と同じです
  - **lastFailureReason:** 最後の失敗の原因(テキスト)
  - **lastSuccessCode:** 最後に成功した通知が送信されたときに受信エンドポイントから返された http コード (数値)
- **lastSuccessCode** フィールドと **lastFailureReason** フィールドは HTTP 通知のみ(MQTT 通知ではない)

```
200 OK
Content-Type: application/json
…
[{
   "id": " 51c0ac9ed714fb3b37d7d5a8 ",
   "expires": "2026-04-05T14:00:00.000Z",
   "status": "failed",
   "subject": { … },
   "notification": {
      "timesSent": 3,
      "lastNotification": "2016-05-31T11:19:32.000Z",
      "lastSuccess": "2016-05-31T10:07:32.000Z",
      "lastFailure": "2016-05-31T11:19:32.000Z",
      …
   }
}]
```

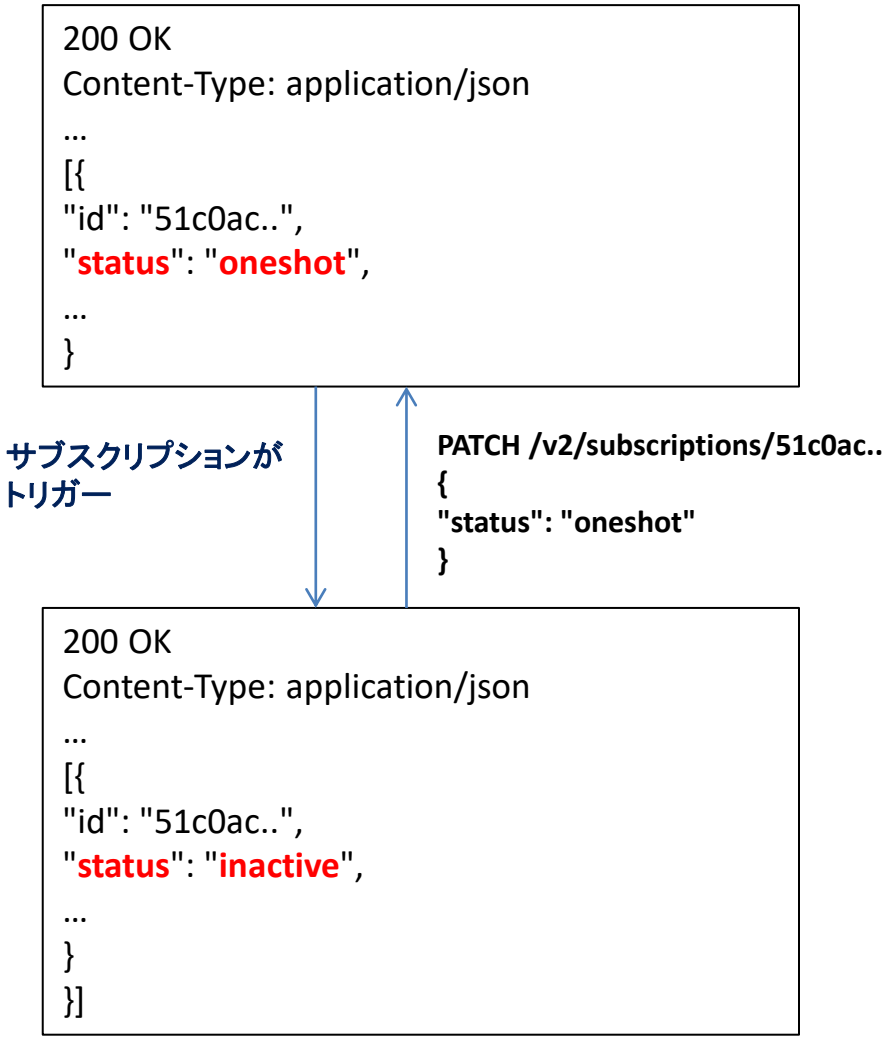
通知診断ワークフロー (Notification diagnosis workflow)



HTTP 通知のみ

ワンショット・サブスクリプション (Oneshot subscription)

- Active ステータスの変形です。したがって、サブスクリプションが1回トリガされると (つまり、通知が送信されると)、自動的に **Inactive** ステータスに移行します
- Inactive なサブスクリプションは、**oneshot** にステップして、プロセスをやり直すことができます
- この場合、サブスクリプションの作成または更新時の初期通知は送信されません



サブスクリプションの自動無効化 (Subscription auto-disabling)

- サブスクリプションに **maxFailsLimit** を指定できるため、連続する通知の数がその値を超えると、サブスクリプションは自動的に **Inactive** 状態に移行します

```
failsCounter > maxFailsLimit => status := inactive
```

- **failedCounter** は、成功した通知が送信されるたびに0にリセットされます
- これにより、通知リソース (プール・ワーカーなど) を消費する通知エンドポイントの障害から Orion を保護できます。
- サブスクリプションが自動無効化されると、サブスクリプションに関するトレースがログに出力されます:

```
time=... | lvl=WARN | ... | msg= Subscription 51c0ac9ed714fb3b37d7d5a8
automatically disabled due to failsCounter (4) overpasses maxFailsLimit (3)
```

```
POST /v2/subscriptions

{
  "subject": { ... },
  "notification": {
    "maxFailsLimit": 3
    ...
  }
}
```


FIWARE

変更タイプに基づくサブスクリプション (Subscriptions based in alteration type)

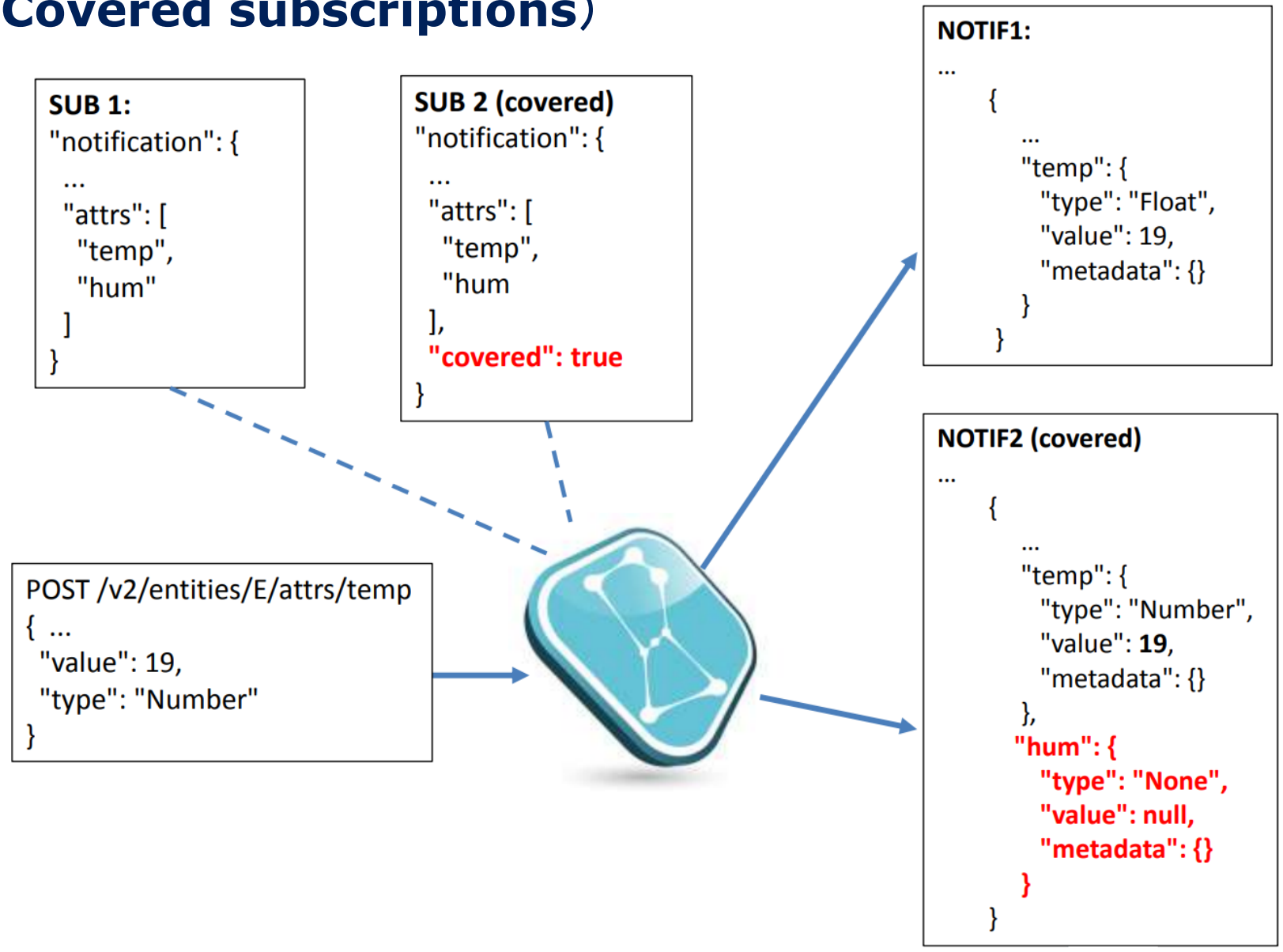
- デフォルトでは、サブスクリプションはエンティティが作成または変更されたときにトリガーされます（つまり、前の値とは異なる値で更新されます）
- これは、**alterationTypes** フィールド（条件内）で変更して、次の（OR **条件**）のリストを指定できます：
  - **entityUpdate:** エンティティの更新（実際に変更されたかどうかは関係ありません）
  - **entityChange:** エンティティの実際の変更
  - **entityCreate:** エンティティが作成された
  - **entityDelete:** エンティティが更新された
- サブスクリプションをトリガーした変更タイプは、特別な **alterationType** 属性を追加することで通知に含めることができます（可能な値は上記と同じトークンです）

```
POST /v2/subscriptions

{
 "subject": {
 ...
 "condition": {
 "alterationTypes":
  ["entityCreate","entityDelete"]
 }
},
"notification": {
 ...
 "attrs": [ "alterationType", "*" ]
 }
}
```

カバード・サブスクリプション（Covered subscriptions）

- デフォルトでは、エンティティに存在しない属性は通知されません
- これは、**カバーされた**フィールド（**通知**内）を true に設定して変更し、存在しない属性に通知する必要があることを指定できます
  - それらは存在しないため、値として null を使用します
- 通知が通知内の attrs フィールドのすべての属性を完全に「カバーする」(“covers”) という意味で、「カバード」("covered")という用語を使用します
- この機能は、可変の属性セットに対して十分な柔軟性がなく、受信したすべての通知で常に同じ着信属性 (incoming attributes) のセットを必要とする通知エンドポイントに役立ちます

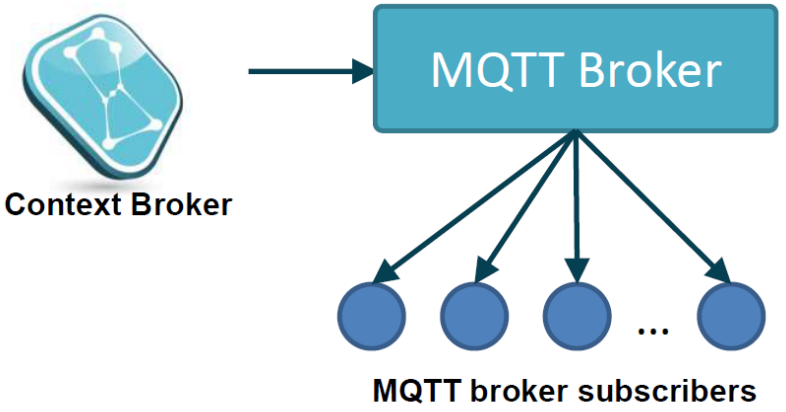
カバード・サブスクリプション（Covered subscriptions）



MQTT 通知

- サブスクリプションペイロードで **http** の代わりに **mqtt** を使用する
- MQTT機能を活用（共有サブスクリプション、QoSなど）
- ユーザ/パスワードに基づく MQTT 認証がサポートされています
- 効率的な接続管理 (MQTT ブローカーが使用されていない場合は、アイドル状態の接続を削除します。**-mqttMaxAge**)
- カスタム通知のサポート（**mqttCustom**）

```
POST <cb_host>:1026/v2/subscriptions
...
{
 "subject": {
  "entities": [
   { "idPattern": ".*" }
  ],
 },
 "notification": {
  "mqtt": {
   "url": "mqtt://<mqtt_broker>:1883",
   "topic": "/notif",
   "qos": 1
  }
 }
}
```



追加のサブスクリプションの強化 (Additional subscription improvements)

- デフォルトでは、実際には属性値を変更しない更新では通知は発生しませんが、この動作は forcedUpdate URI param オプションでオーバーライドできます。例：
  - PATCH /v2/entities/E/attrs?options=forcedUpdate
  - PUT /v2/entities/E/attrs/A?options=forcedUpdate
  - POST /v2/op/update?options=forcedUpdate
- **onlyChangedAttributes** (**通知**での) は、(**attrs** または **exceptAttrs** で指定されたものと組み合わせて) 変更された属性のみを通知するために **true** に設定します
- サブスクリプションごとの通知タイムアウトは、**timeout** フィールド(**通知**内の **http/httpCustom** 内)で HTTP サブスクリプションに指定できます。

FIWARE

属性フィルタリングと特殊属性

(Attributes filtering and special attributes)

- デフォルトでは、すべての属性はクエリのレスポンスまたは通知に含まれます
- GET操作のパラメータとして、およびサブスクリプションの**通知**サブフィールドとして、**attrs** フィールドを使用して、フィルタリング・リストを指定できます
- **attrs** フィールドは、特別な属性を明示的に含むためにも使用できます (デフォルトでは含まれていません)
  - **dateCreated, dateModified, dateExpires**:
    以前のスライドで説明しました
- "*"は、"すべての通常の属性"の別名として使用できます

属性フィルタリングと特殊属性

(Attributes filtering and special attributes)

- 例
  - temp と lum だけの属性を含めます
    - クエリで: GET /v2/entities?attrs=temp,lum
    - サブスクリプションで: "attrs": ["temp", "lum"]
  - 他の属性ではなく dateCreated を含めます
    - クエリで: GET /v2/entities?attrs=dateCreated
    - サブスクリプションで: "attrs": ["dateCreated"]
  - dateModified とその他すべての（通常の）属性を含めます
    - クエリで: GET /v2/entities?attrs=dateModified,*
    - サブスクリプションで: "attrs": ["dateModified", "*"]
  - すべての属性を含めます（attrsを使用しないのと同じ効果です）
    - クエリで: GET /v2/entities?attrs=*
    - サブスクリプションで: "attrs": ["*"]

メタデータのフィルタリングと特殊メタデータ (Metadata filtering and special metadata)

- デフォルトでは、すべての属性メタデータはクエリのレスポンスと通知に含まれています
- GET操作のパラメータとして、および、サブスクリプションの**通知**サブフィールドとして、**metadata** フィールドを使用して、フィルタリング・リストを指定できます
- **metadata** フィールドは、明示的には含まれていない特別なメタデータを明示的に含むためにも使用できます
  - **dateCreated, dateModified**: 以前のスライドで説明しました
  - **actionType**: どの値が通知をトリガする更新に対応するアクション・タイプであるか：update”, “append” または “delete” (*)
  - **previousValue**: 通知をトリガする更新を処理する前の属性値を提供します
- “*”は、“すべての通常のメタデータ"の別名として使用できます

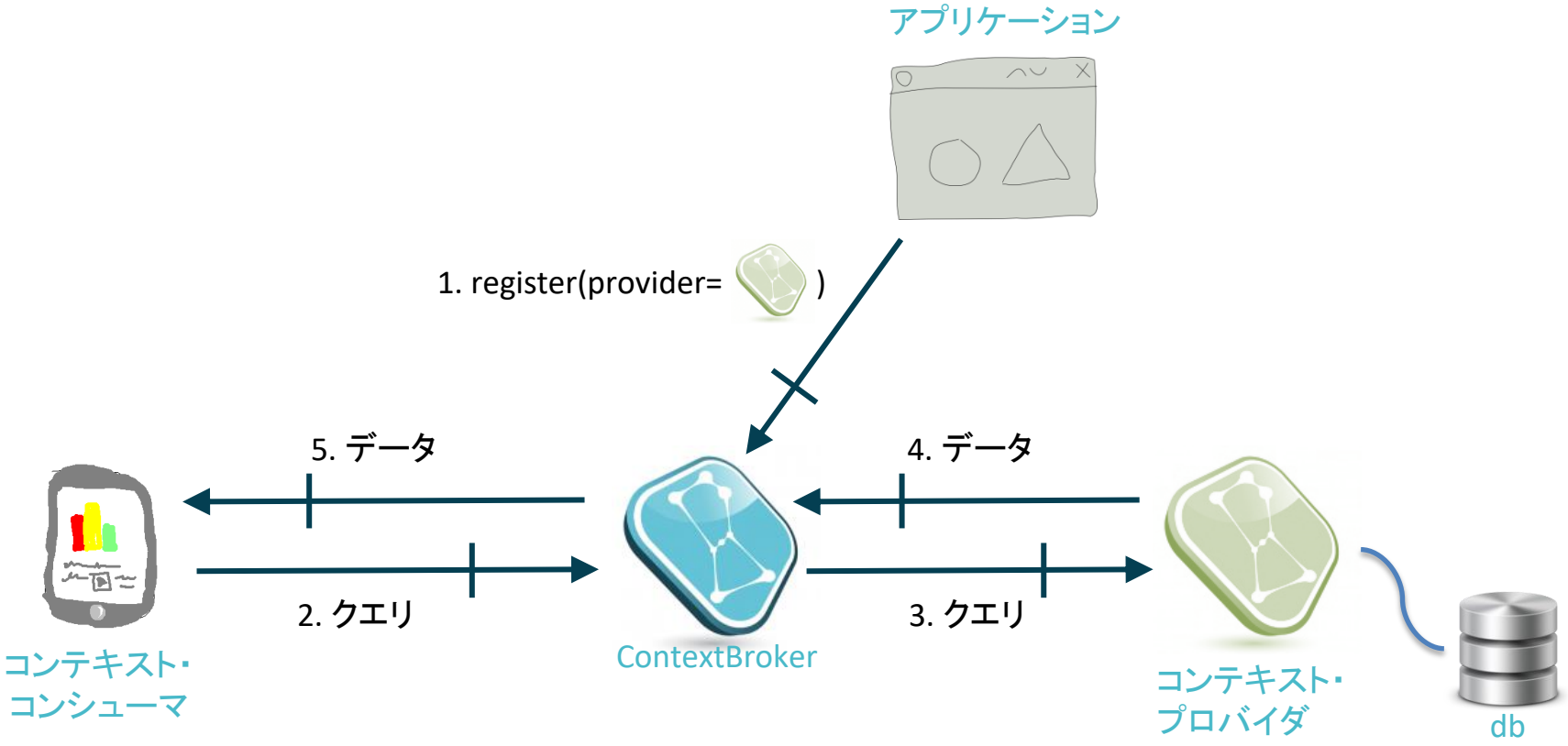
(*) actionType "delete" は、まだ、Orion 3.8.0 でサポートされていません

メタデータのフィルタリングと特殊メタデータ (Metadata filtering and special metadata)

- 例
  - メタデータ MD1 と MD2 のみを含めます
    - クエリで: GET /v2/entities?metadata=MD1,MD2
    - サブスクリプションで: "metadata": ["MD1", "MD2"]
  - previousValue を含み、他のメタデータは含みません
    - クエリで: GET /v2/entities?metadata=previousValue
    - サブスクリプションで: "attrs": ["previousValue"]
  - actionType とその他すべての通常のメタデータを含めます
    - クエリで: GET /v2/entities?metadata=actionType,*
    - サブスクリプションで: "attrs": ["actionType", "*"]
  - すべてのメタデータを含めます（メタデータを使用しない場合と同じ効果です）
    - クエリで: GET /v2/entities?metadata=*
    - サブスクリプションで: "metadata": ["*"]

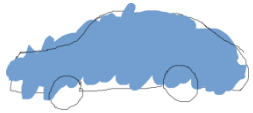
レジストレーション と コンテキスト・プロバイダ

- キャッシュされていないクエリと更新



レジストレーション と コンテキスト・プロバイダ

```
POST <cb_host>:1026/v2/registrations
...
{
  "description": "Registration for Car1",
  "dataProvided": {
    "entities": [
      {
        "id": "Car1",
        "type": "Car"
      }
    ],
    "attrs": [
      "speed"
    ]
  },
  "provider": {
    "http": {
      "url": "http://contextprovider.com/Cars"
    },
    "legacyForwarding": true
  }
}
```



```
201 Created
Location: /v2/registrations/5a82be3d093af1b94ac0f730
...
```

(*)現在のOrionバージョンにはいくつかの制限があります。https://github.com/telefonicaid/fiware-orion/blob/master/doc/manuals.jp/orion-api.md#registration-implementation-differences

FIWARE

レジストレーション と コンテキスト・プロバイダ

?options=skipForwarding を使用すると、クエリ転送を回避できます。この場合、クエリは Context Broker ローカル コンテキスト情報のみを使用して評価されます。

GET <cb_host>:1026/v2/entities/Car1/attrs

200 OK
Content-Type: application/json
...

```
{
   "type": "Float",
   "value": 110,
   "metadata": {}
}
```

データ

クエリ

ContextBroker

コンテキスト・
プロバイダ

db

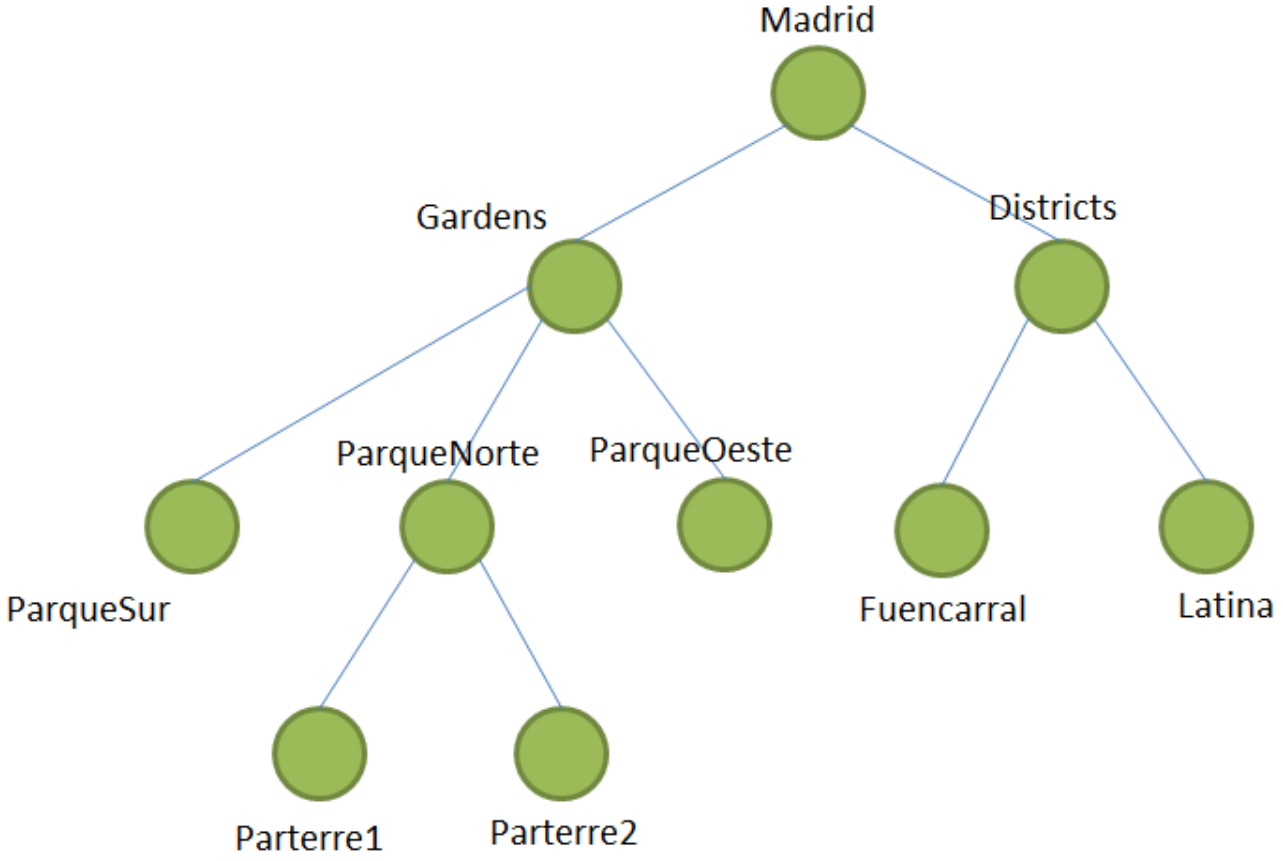
マルチテナンシー (Multitenancy)

- 論理データベース分離に基づく単純なマルチテナントモデル
- 他コンポーネントによって提供されるテナントベースの認証を容易にします
- “Fiware-Service” という追加の HTTP ヘッダにテナント名を指定して使用してください
  例:

  Fiware-Service: Tenant1

サービス･パス (Service Paths)

- サービス･パスは、作成時に POST /v2/entities でエンティティに割り当てられた階層スコープです

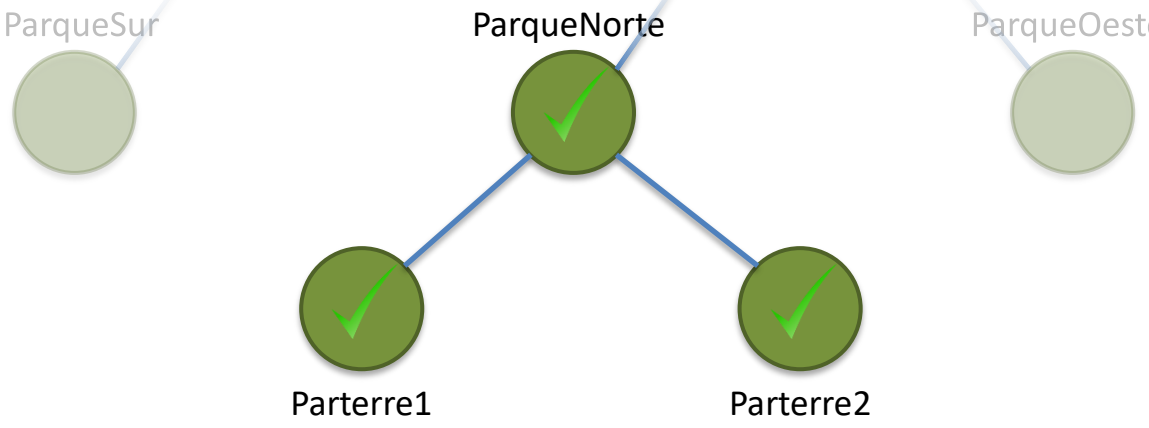


サービス・パス (Service Paths)

- サービス・パスを使用するために、“Fiware-ServicePath” と呼ばれる新しい HTTP ヘッダを挿入します。例:

  Fiware-ServicePath: /Madrid/Gardens/ParqueNorte/Parterre1

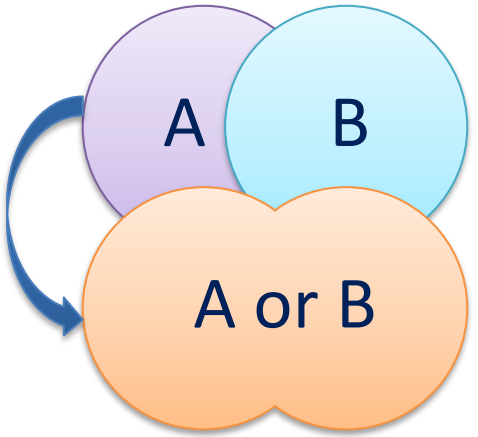
- プロパティ:
  - サービス・パスのクエリは、指定されたノードのみを検索します
  - すべての子ノードを含めるには "ParentNode/#" を使用します
  - Fiware-ServicePath のないクエリは “/#” と解釈します
  - エンティティはデフォルトで "/" ノードに入ります

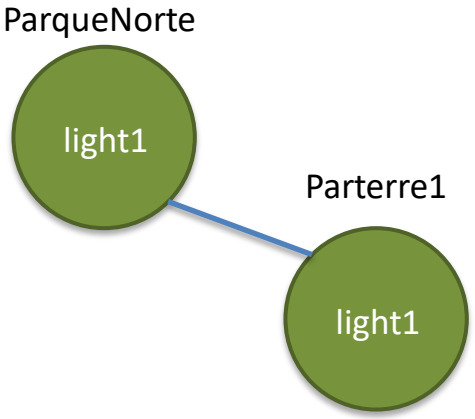


サービス・パス (Service Paths)

- プロパティ (続き):
  - ヘッダーにカンマ(,)演算子を使用してクエリをORできます
    - 例えば、ParqueSur または ParqueOesteにあるすべての街灯を照会するには:

      ServicePath: Madrid/Gardens/ParqueSur, Madrid/Gardens/ParqueOeste
    - 最大10個の異なるスコープを OR することができます
  - 最大スコープ・レベル: 10
    - Scope1/Scope2/.../Scope10
  - さまざまなスコープで同じ要素 ID を持つことができます (これに注意してください)
  - 要素が作成されると、範囲を変更することはできません
  - 1つのエンティティは1つのスコープにのみ属することができます
  - クエリだけでなく、サブスクリプション/通知でも機能します
  - NGSI10だけでなく、レジストレーション/ディスカバリー（NGSI9）でも機能します



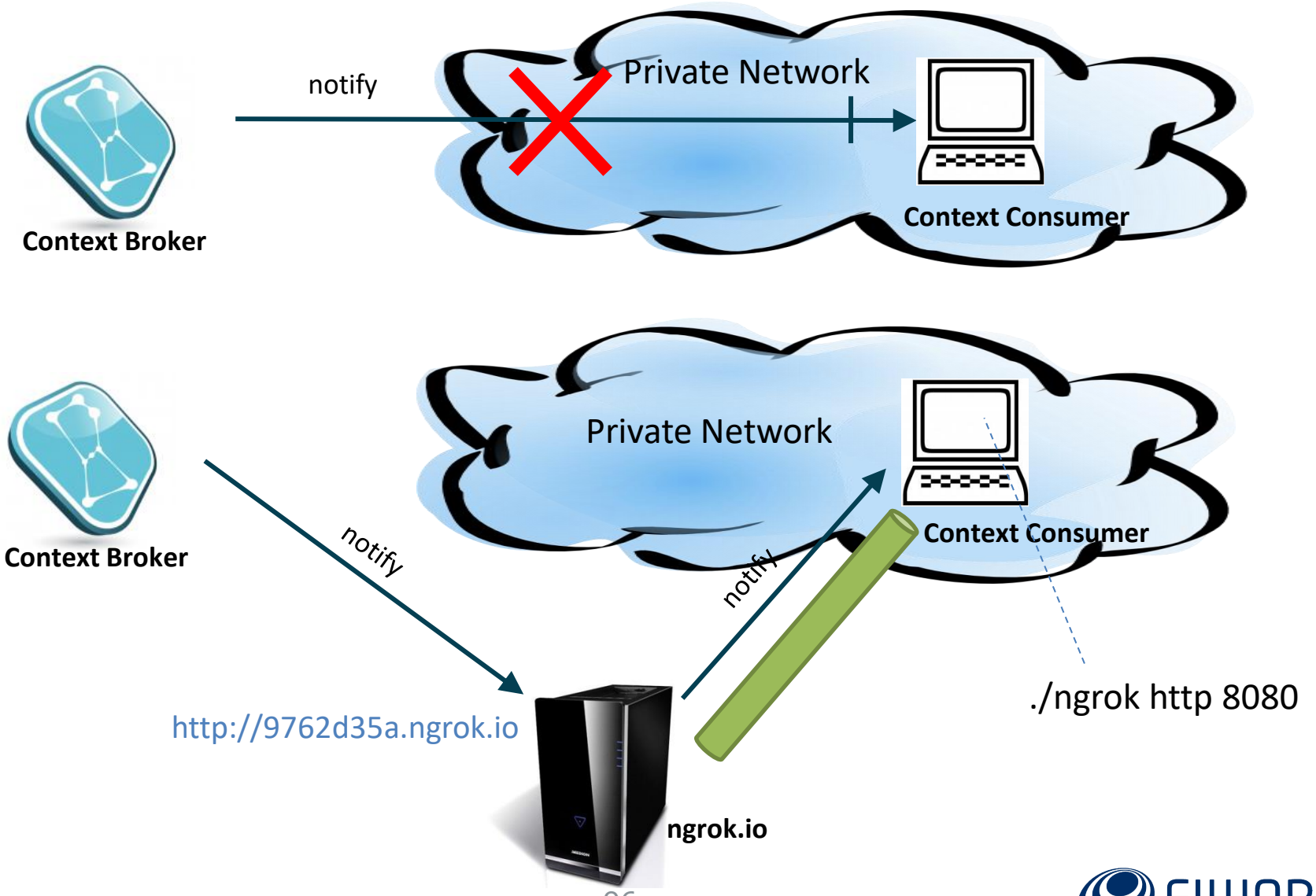


クロス・オリジン・リソース共有 (CORS)
(Cross-Origin Resource Sharing)

- バックエンドなしでブラウザで完全に実行されるプログラミング・クライアントに役立ちます
  - NGSIv2 での完全サポート (pre-flightのリクエストを含む)
  - NGSIv1 での部分的なサポート (GETリクエストのみ)
- CORS に関連する CLI パラメータ
  - **-corsOrigin**
  - **-corsMaxAge**
- 詳細
  - https://fiware-orion.readthedocs.io/en/master/user/cors/index.html

FIWARE

プライベート・ネットワークでのサービスの通知



もっと知りたいですか？

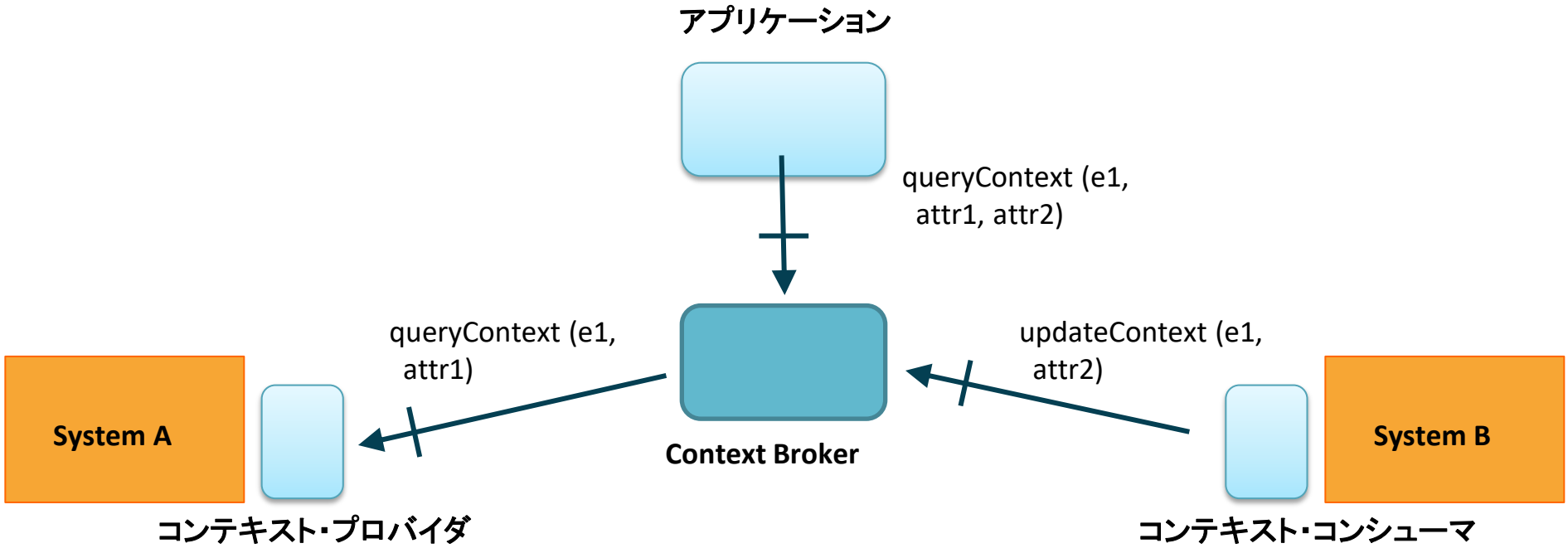
- リファレンス
  - プレゼンテーション
    https://github.com/telefonicaid/fiware-orion#introductory-presentations
  - NGSIv2 仕様
    - https://github.com/telefonicaid/fiware-orion/blob/master/doc/manuals.jp/orion-api.md
  - ドキュメント (ReadTheDocs):
    - https://fiware-orion.readthedocs.io/en/master/
    - https://fiware-orion.letsfiware.jp/
  - StackOverflow での Orion サポート
    - “fiware-orion” タグをつけて、質問してください
    - http://stackoverflow.com/questions/tagged/fiware-orion で既存のQ&Aを探してください

Thank you!

http://fiware.org
Follow @FIWARE on Twitter

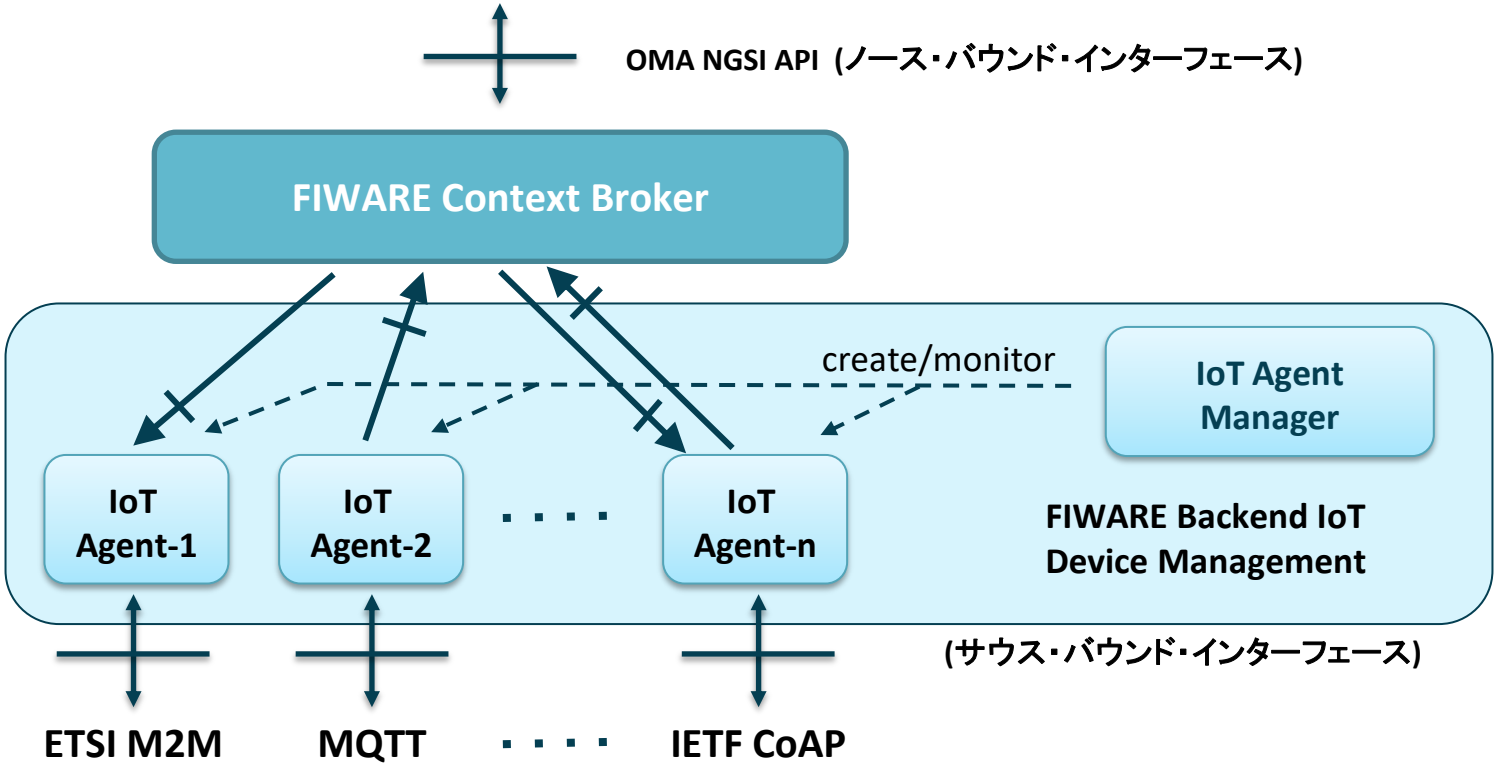
既存のシステムとの統合

- コンテキスト・アダプタは、コンテキスト・プロバイダ、コンテキスト・プロデューサ、またはその両方として機能する既存のシステム (例えば、スマート・シティの地方自治体サービス管理システム) とインターフェースするために開発さます
- 特定のエンティティからのいくつかの属性はコンテキストプロバイダにリンクされていてもよく、他の属性はコンテキストプロデューサにリンクされていてもよいです



センサ・ネットワークとの統合

- Backend IoT Device Management GE は、センサ・ネットワークに接続する NGSI IoT Agent の作成と設定を可能にします
- 各 NGSI IoT Agent は、コンテキスト・コンシューマ または コンテキスト・プロバイダ、またはその両方として動作できます

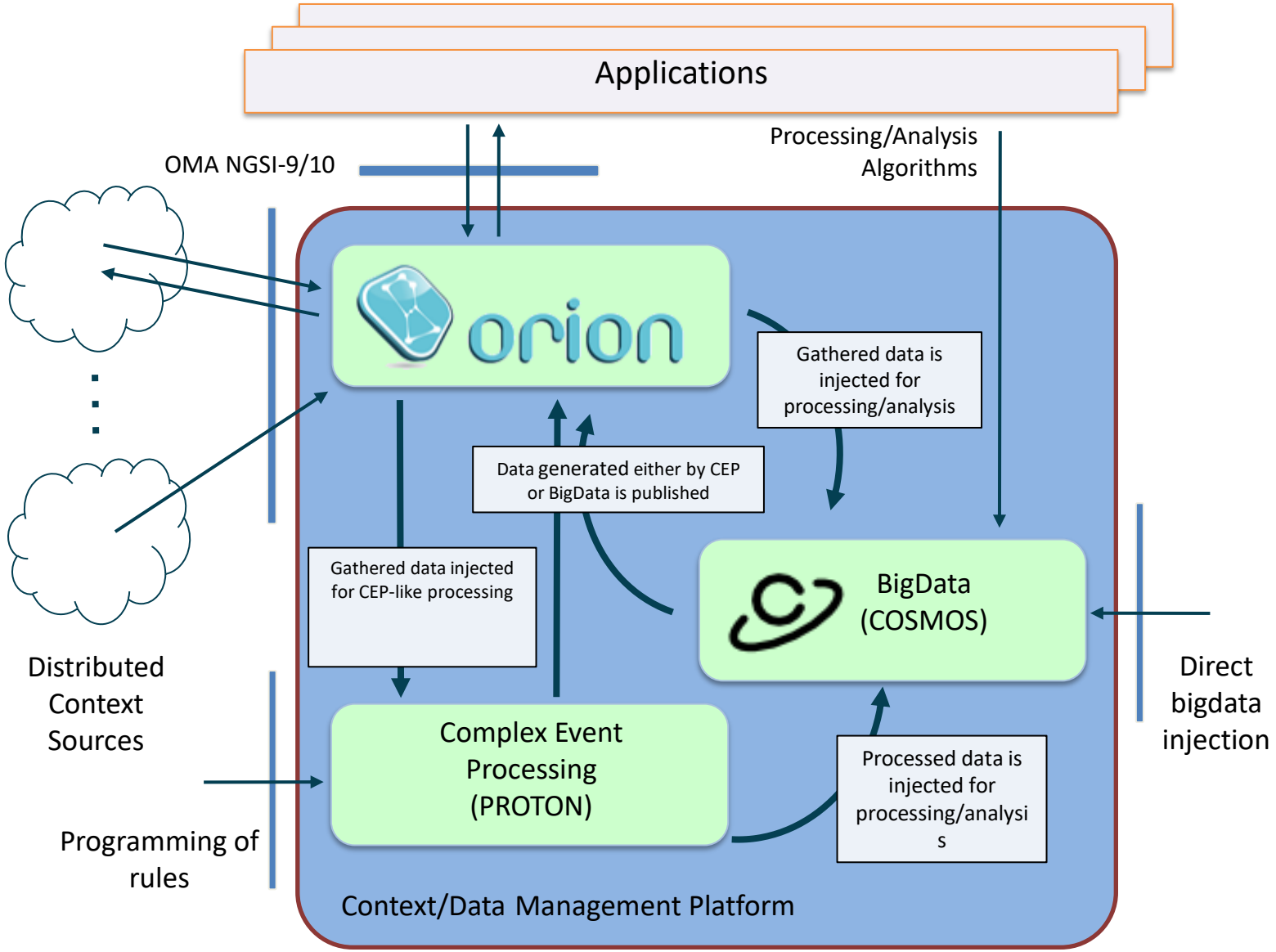


FIWARE におけるコンテキスト管理

Cloud		• Federation of infrastructures (private/public regions) • Automated GE deployment
Data		• Complete Context Management Platform • Integration of Data and Media Content
IoT		• Easy plug&play of devices using multiple protocols • Automated Measurements/Action ←→Context updates
Apps		• Visualization of data (operation dashboards) • Publication of data sets/services
Web UI		• Easy support of UIs with advanced web-based 3D and AR capabilities • Visual representation of context information.
I2ND		• Advanced networking capabilities (SDN) and Middleware • Interface to robots
Security		• Security Monitoring • Built-in Identity/Access/Privacy Management

FIWARE

FIWARE コンテキスト/データ管理プラットフォーム



特別な更新アクション・タイプ

- **/v2/op/update** によって使用 (バッチ操作)
- コンベンショナルな actionTypes
  - **append**: アペンド (または 属性がすでに存在するときは、更新)
  - **update**: 更新
  - **delete**: 削除
- 特別な actionTypes
  - **appendStrict**: 厳密な追加 (追加する属性がすでに存在する場合はエラーを返します)
  - **replace**: すべてのエンティティ属性を削除し、次に更新リクエスト内の属性を追加します