データ利活用基盤サービス

（FIWARE）

アプリケーション開発ガイド

データ収集蓄積編

第 1.5 版

2019 年 4 月

目次

第1章 はじめに

1.1 本ガイドの位置付け

本ガイドはデータ利活用基盤サービス（FIWARE）における「データ収集/蓄積レイヤ」の開発ガイドであり、データ収集/蓄積レイヤと連携するモジュールの開発者をターゲットとしています。

本ガイドに記載する内容は以下のとおりです。

- データ収集/蓄積レイヤの持つ機能（役割）
- データ収集/蓄積レイヤに関連する要素技術（FIWARE NGSI）
- データ収集/蓄積レイヤの連携モジュール開発ガイド
- データ収集/蓄積レイヤの API 仕様

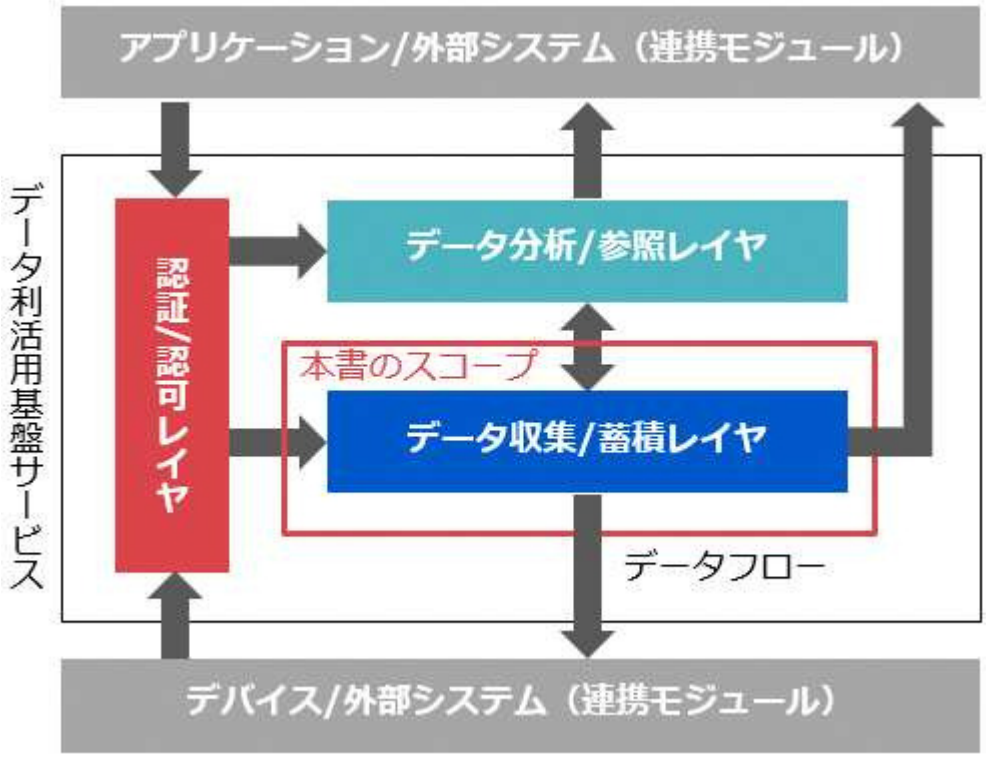

図 1-1 システム概要

本ガイドに掲載されている製品名やサービス名は、当社または各社、各団体の商標または登録商標です。

1.2 関連ガイド

本ガイドの関連ガイドを以下に示します。

表 1-1 関連ガイド

ガイド名	版数
データ利活用基盤サービス（FIWARE）アプリケーション開発ガイド	1.4 版
データ利活用基盤サービス（FIWARE）アプリケーション開発ガイド（認証認可編）	1.4 版
データ利活用基盤サービス（FIWARE）アプリケーション開発ガイド（データ分析参照編）	1.4 版

第2章 機能概要

本章ではデータ収集/蓄積レイヤが提供する機能（役割）について記載します。

データ収集/蓄積レイヤは Fiware-Orion（以降、Orion と記載）という Open Source Software（OSS）をベースにしており、下記3つの機能を提供します。下記機能によって、アプリケーションやデバイスなど、データ収集/蓄積レイヤと連携するモジュールはお互いの存在を意識せずにデータの収集/蓄積を行うことが可能です。本ガイドでは Orion バージョン 1.14.0 の情報を記載しています。Orion については、付録 A 参考情報 [1] もあわせて参照してください。

1. データ登録/更新
2. データ参照/問い合わせ

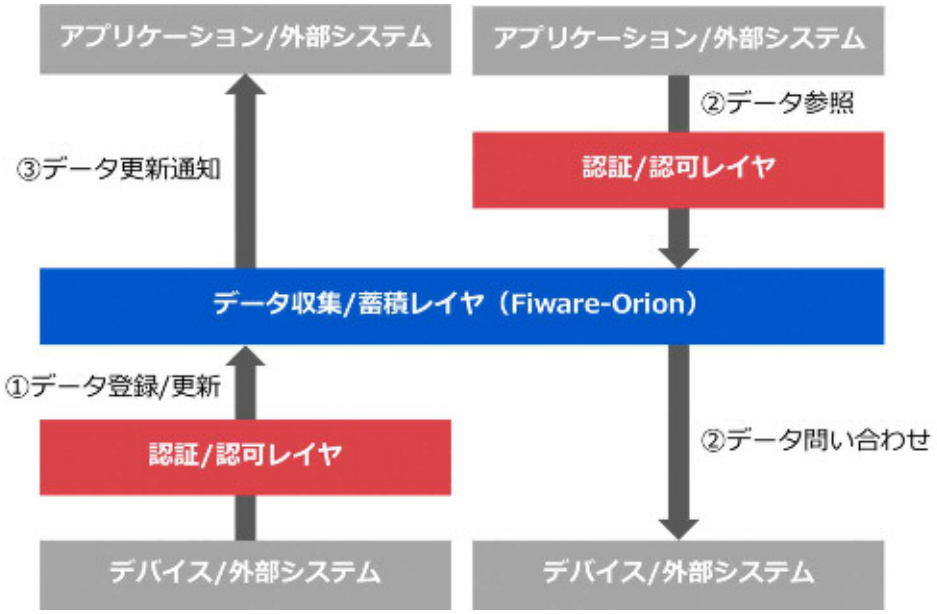

図 2-1 機能概要

「①データ登録/更新」は、デバイスや外部システムがデータ収集/蓄積レイヤにデータを登録もしくは更新する際に利用する機能です。

「②データ参照/問い合わせ」は、アプリケーションや外部システムがデータ収集/蓄積レイヤに蓄積されたデータを参照する際に利用する機能です。任意のタイミングでデータを参照する場合は、本機能を利用します。データ収集/蓄積レイヤに存在しないデータが参照された場合は、データ収集/蓄積レイヤがデータの提供元に問い合わせを行うことも可能です。

なお、「①データ登録/更新」および「②データ参照」を利用する場合には、必ず認証/認可レイヤを経由する必要がありますが、外部システムは認証/認可レイヤをほぼ意識することなく透過的にデータ収集/蓄積レイヤにアクセスすることが可能です。

以降の章では認証/認可レイヤを省略して記載します。

第3章 要素技術（FIWARE NGSI）

本章ではデータ収集/蓄積レイヤの要素技術である FIWARE NGSI について記載します。

FIWARE NGSI は Open Mobile Alliance（OMA）が策定した NGSI を拡張した IF であり、データ収集/蓄積レイヤが提供する機能はすべて FIWARE NGSI を利用します。なお、以降の章では FIWARE NGSI を NGSI と表記し、OMA が策定した NGSI を OMA-NGSI と表記します。

NGSI には NGSIv1 (Version1)と NGSIv2 (Version2)の2つのバージョンがあり、データ利活用基盤サービス（FIWARE）では NGSI v1、NGSIv2 の両方の API をサポートしています。

図 3-1 FIWARE NGSI

以降の章では、まず NGSI の利用者（アクタ）について記載します。次に NGSI のデータモデルを記載し、各アクタ間でどのようなデータをやり取りするのかを説明します。最後にデータをやり取りする際に実行する IF（RESTful API）の概要を記載します。

3.1 NGSI アクタ

本ガイドでは NGSI の利用者（アクタ）を分類しますが、単一の分類にのみ所属させる必要はありません。

たとえば、Context Producer と Context Consumer の両方に所属するアクタを定義しても問題ありません。なお、データ収集/蓄積レイヤは下記すべての分類に所属しており、状況に応じて異なるアクタとして振る舞います。

1. Context Producer
2. Context Consumer

3.1.1 Context Producer

Context Producer（CP）は、データ生産者の一種であり、データ収集/蓄積レイヤに対して能動的にデータを登録/更新するアクタのことです。CP の前提条件としてはデータ登録/更新用の NGSI を発行可能であること（NGSI クライアント機能を有すること）が挙げられます。

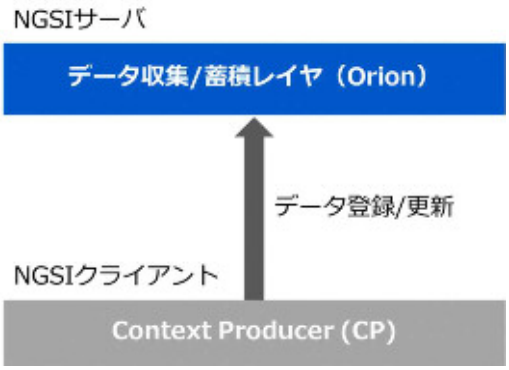

図 3-2 Context Producer

3.1.2 Context Consumer

Context Consumer（CC）は、データ消費者であり、データ収集/蓄積レイヤに対するデータ参照するアクタです。

- データ参照を行う場合
  データ参照用の NGSI を発行可能であること（NGSI クライアント機能を有すること）

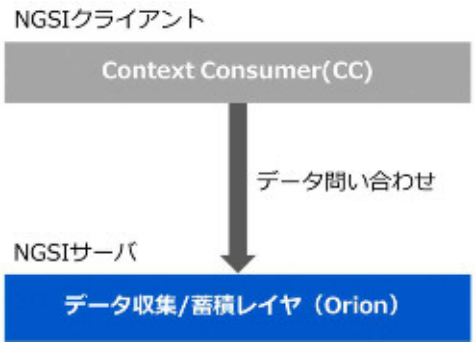

図 3-3 Context Consumer

3.2 NGSI データモデル

NGSI では Context Element（もしくは Context Entity）と呼ばれるデータモデルが定義されています。

Context Element は実世界の多様な情報を抽象化して表現できるよう柔軟な構造となっており、

EntityId、EntityType および Context Element Attribute（以降 Attribute と記載）から構成されます。

Attribute は Name、Type、Valume および Meta-data から構成されます。

Context Element は xml や json 形式で実装されていますが、データ利活用基盤サービス（FIWARE）ではデータ分析/参照レイヤとの互換性およびデータサイズを考慮して json 形式を採用しています。

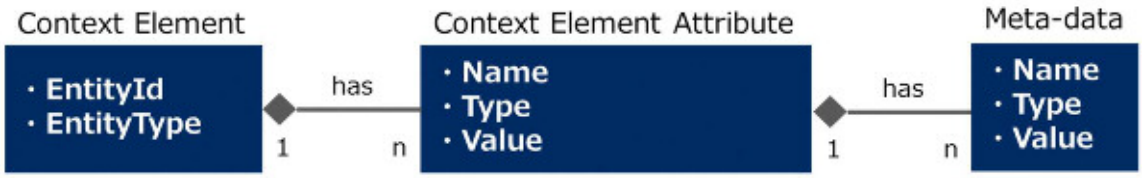

図 3-4 NGSI データモデル

以下に json 形式の Context Element の一例を記載します。下記例では、Room-1 という部屋の温度/湿度データを Context Element で表現しています。

[NGSIv1 Context Element 例(json 形式)]

```
{
"contextElements": [
        {
            "type": "Room",
            "isPattern": "false",
            "id": "Room-1",
            "attributes": [
                {
                    "name": "temperature",
                    "type": "float",
                    "value": "20.5",
                    "metadatas": [
                        {
                            "name": "accuracy",
                            "type": "float",
                            "value": "0.8"
                        }
                    ]
                },
                {
                    "name": "humidity",
                    "type": "integer",
                    "value": "50"
                }
            ]
        }
]
}
```

[NGSIv2 Context Element 例(json 形式)]

```
{
   "id": "Room-1",
   "type": "Room",
   "temperature": {
      "value": 20.5,
      "type": "Number",
      "metadata": {
         " accuracy ": {
            "value": 0.8,
            "type": "Number"
         }
      }
   },
   "humidity": {
      "value": 50,
      "type": "Number"
   }
}
```

3.3 NGSI v2

NGSI v2 は、データの登録、参照、更新、削除など、データのライフサイクル全体を管理することを目的としています。NGSI v1 から単純化され FIWARE-NGSI Simple API とも呼ばれます。

NGSI v2 は、表 3-1 のインタフェースで構成されます。各インタフェースの利用手順に関しては第 4 章 ～第 5 章 を、詳細については第 6 章 を参照してください。また、付録 A 参考情報 [7] もあわせて参照してください。

表 3-1 NGSI v2

NGSI v2	用途
entities	データリストの参照/データの登録
entities/{entityId}	データの参照/削除 (ID 指定)
entities/{entityId}/attrs	データの属性情報の参照/更新 (ID，属性指定)
entities/{entityId}/attrs/{attrName}	データの属性情報の参照/更新/削除 (ID，属性名称指定)
entities/{entityId}/attrs/{attrName}/value	データの属性値の参照/更新 (ID，属性名称指定)
types	タイプリストの参照
types/{entityType}	タイプ情報の参照 (タイプ指定)
op/update	バッチ更新オペレーションの実行
op/query	バッチクエリオペレーションの実行

3.3.1 NGSIv2 API を利用する利点

NGSIv2 API を採用することで以下のような利点があります。

- HTTP メソッドを変えて呼び出すことで、参照(GET)、作成(POST)、更新(PATCH)、置換(PUT)、削除(DELETE)の役割を持たせることができ、シンプルで一貫性のあるインタフェースを利用することができます。

3.3.2 NGSIv1 API との機能差分

NGSIv2 API は NGSIv1 API と比較すると以下のような機能差分があります。

- NGSIv2 では属性値を JSON ネイティブタイプ（数値、ブール値、文字列など）として登録します。NGSIv1 は文字列での指定だった、データの属性値の型の扱いが厳密化されました。

NGSIv1

```
{
    "name": "temperature",
    "type": "float",
    "value": "20.5"
}
```

NGSIv2

```
"temperature": {
      "type": "Number",
      "value": 20.5
}
```

- NGSIv2 では、フィールドの使用文字制限が追加されました。
  詳しくは付録 A 参考情報 [7]を参照してください。

3.4 Fiware Service / Fiware ServicePath

Fiware Service および Fiware ServicePath は Orion がマルチテナント/マルチサービスを提供するための機能です。Fiware Service により Context Element は論理的にグループ化され、隔離されます。また、Fiware ServicePath は、同一 Fiware Service に所属する Context Element を階層構造化する仕組みです。Fiware Service および Fiware ServicePath は NGSI 発行時に HTTP ヘッダとして付加することが可能です（任意）。

Fiware Service および Fiware ServicePath の指定ありで登録されたデータを参照する場合は、登録時と同じ Fiware Service および Fiware ServicePath を指定しないとデータを参照することはできません（論理的に隔離されているため）。

以下に Fiware Service および Fiware ServicePath の一例と命名規則を記載します。なお、Fiware Service については参考情報[3]を、Fiware ServicePath については、付録 A 参考情報 [4]をあわせて参照してください。

また、『データ利活用基盤サービス（FIWARE） アプリケーション開発ガイド（データ分析参照編）』の STH-Comet でのデータ参照も あわせて参照してください。

Fiware Service および Fiware ServicePath の一例

```
Fiware-Service: service1
Fiware-ServicePath: /city/street1, /city/street2
```

表 3-2 Fiware Sevice 命名規則

型	文字列
利用可能文字	・英字小文字 ・数字 ・アンダースコア（_）
備考	・最大 50 文字

表 3-3 Fiware SevicePath 命名規則

型	文字列
利用可能文字	・英数字 ・アンダースコア(_)
備考	・先頭文字はスラッシュ(/) ・最大 10 階層 ・各階層の文字数は 1 文字以上、50 文字以下 ・最大 10 個まで指定可能(updateContext 時は 1 個のみ) ・末尾にスラッシュ(/)を指定した場合は無視(破棄)

⚠注意

Fiware-Service、Fiware-ServicePath の付与は任意ですが、省略した場合は他事業者からデータ参照ができてしまうため、省略する場合は注意してください。

第4章 Context Producer 開発者向けガイド

本章では、Context Producer（CP）開発者向けガイドとして、CP がデータ収集/蓄積レイヤと連携する際に行う処理を記載します。

なお、本ガイドでは各処理の一例として curl コマンドのサンプルを記載します。

1. データ登録/更新（必須）

本ガイド中の${IP}はデータ利活用基盤サービス（FIWARE）のグローバル IP アドレス（ドメインを設定している場合は FQDN）を表しており、${TOKEN}は認証/認可レイヤを通過するためのアクセストークンを表しています。アクセストークンについては、『データ利活用基盤サービス（FIWARE） アプリケーション開発ガイド（認証認可編）』を参照してください。
なお、API リクエスト時に HTTP ヘッダに指定するアクセストークンには以下の 2 種類の形式を利用可能です。

- Authorization: Bearer ${TOKEN}
- X-Auth-Token: ${TOKEN}

4.1 データの登録/更新（必須）

CP はデータ収集/蓄積レイヤに対して能動的にデータの登録/更新を行うアクタであるため、本処理の実装は必須です。データの登録/更新タイミングに関して特に制約はかけていません。データに変更が発生したタイミングや閾値を超えたタイミング、事前に定めた周期で更新するなど CP 側で自由に規定することが可能です。

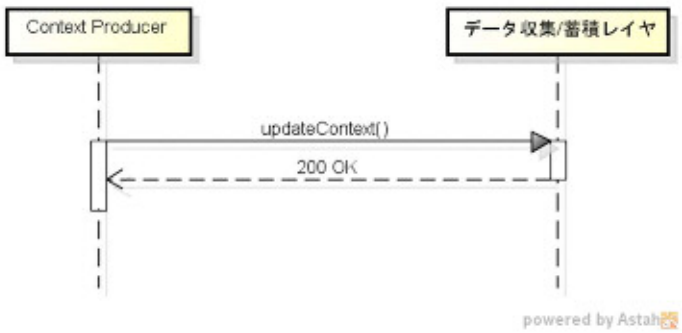

図 4-1 データの登録/更新

4.1.1 NGSIv2

NGSIv2 でのデータの登録には POST メソッドの entities、更新には PATCH メソッドの entities を利用します。POST メソッドの entities（リクエスト/レスポンス）の一例を以下に記載します。

下記例では、Room-1 という部屋の温度/湿度に関するデータを登録しています。パラメータの詳細や PATCH メソッドの entities については、第 6 章 を参照してください。

[entities リクエスト]

```
(curl -k -X POST "https://${IP}/orion/v2.0/entities" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" ¥
 -d @- ) <<EOF
{
   "id": "Room-1",
   "type": "Room",
   "temperature": {
      "value": 20.5,
      "type": "Float"
   },
   "humidity": {
      "value": 50,
      "type": "Integer"
   }
}
EOF
```

POST メソッドの entities のレスポンスには、ステータスコード 201 が返ります。

[entities レスポンス]

```
< HTTP/1.1 201 Created
< Connection: keep-alive
< Location: /v2/entities/Room-1?type=Room
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

4.2 NGSIv2 バッチオペレーションによるデータの登録/更新(任意)

本処理は NGSIv2 で追加された新しい処理です。NGSIv2の entities(POST メソッド)でのデータ登録処理は 1 つのデータにしか対応していません。そこでバッチオペレーション(/op/update)によって複数のデータの登録/更新を一度のリクエストで可能にしています。

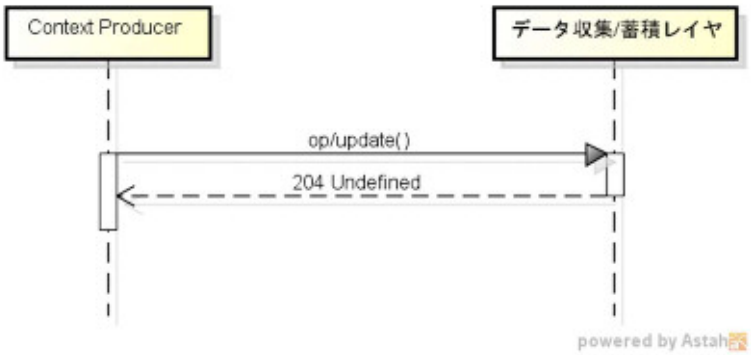

図 4-2 複数データの登録/更新

/op/update（リクエスト/レスポンス）の一例を以下に記載します。

下記例では、Room-1、Room-2 という部屋の温度/湿度に関するデータを登録しています。ボディに含まれる actionType はデータを登録するか更新するかを指定するパラメータであり、登録なら APPEND、更新なら UPDATE を指定します。ただし、データ収集/蓄積レイヤでは更新時に APPEND を指定しても問題ありません。その他パラメータの詳細は、第 6 章 を参照してください。

[/op/update リクエスト]

```
(curl -k -X POST "https://${IP}/orion/v2.0/op/update" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @-) <<EOF
{
  "actionType": "APPEND",
  "entities": [
    {
      "type": "Room",
      "id": "Room-1",
      "temperature": {
        "value": 21.7
       },
      "humidity": {
        "value": 60
      }
    },
    {
      "type": "Room",
      "id": "Room-2",
      "temperature": {
        "value": 22.9
      },
      "humidity": {
        "value": 85
      }
    }
  ]
}
EOF
```

レスポンスにはステータスコード 204 が返ります。

[/op/update レスポンス]

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

第5章 Context Consumer 開発者向けガイド

本章では Context Consumer（CC）開発者向けガイドとして、CC がデータ収集/蓄積レイヤと連携する際に行う処理を記載します。CC が行う処理は大まかに下記 6 つですが、目的に応じて必須処理が変わるため、ここではすべて任意としてあります。

なお、本ガイドでは各処理の一例として curl コマンドのサンプルを記載します。

1. データ参照（任意）
2. 拡張 IF によるデータ参照（任意）

本ガイド中の${IP}はデータ利活用基盤サービス（FIWARE）のグローバル IP アドレス（ドメインを設定している場合は FQDN）を表しており、${TOKEN}は認証/認可レイヤを通過するためのアクセストークンを表しています。アクセストークンについては、『データ利活用基盤サービス（FIWARE） アプリケーション開発ガイド（認証認可編）』を参照してください。
なお、API リクエスト時に HTTP ヘッダに指定するアクセストークンには以下の 2 種類の形式を利用可能です。

- Authorization: Bearer ${TOKEN}
- X-Auth-Token: ${TOKEN}

5.1 データ参照（任意）

データ参照は、CC 側でデータが必要になったタイミングで実行する処理です。データ参照前に CP によるデータの登録/更新（updateContext, entities）などが行われていなければ参照できません。

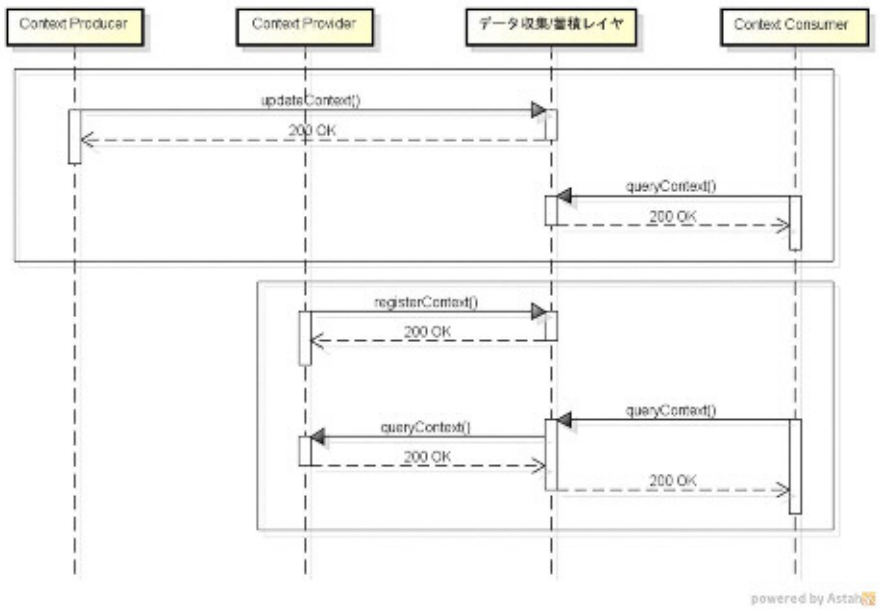

図 5-1 データ参照

5.1.1 NGSIv2

NGSIv2 でのデータ参照は GET メソッドの entities を利用します。entities（リクエスト/レスポンス）の一例を以下に記載します。

下記例では、Room というタイプに所属する Context Element が保持している温度データ（Attribute）を参照しています。type パラメータに対してタイプを指定でき、idPattern パラメータに対して正規表現を適用可能です。Context Element 内の全 Attribute 情報が必要であれば attrs パラメータを省略すればよいです。

その他パラメータの詳細は、第 6 章 を参照してください。

```
[entities リクエスト]
 (curl -k "https://${IP}/orion/v2.0/entities?type=Room&idPattern=Room.*&attrs=temperature" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
  | python -mjson.tool)
```

entities のレスポンスにはリクエストで指定した Context Element の情報が含まれます。

```
[entities レスポンス]
 [
   {
     "type": "Room",
     "id": "Room-1",
     "temperature": {
       "value": 35.6,
       "type": "Number",
       "metadata": {}
     }
   },
   {
     "type": "Room",
     "id": "Room-2",
     "temperature": {
       "value": 22.5,
       "type": "Number",
       "metadata": {}
     }
   }
 ]
```

5.2 拡張 IF によるデータ参照（任意）

NGSIv1 では、拡張 IF を利用することにより、任意のエンティティ ID（またはその属性）に対して POST メソッドによる追加、POST メソッドによる変更、GET メソッドによるデータ参照、DELETE メソッドによる削除を行うことができます。詳細は、「**エラー! 参照元が見つかりません。エラー! 参照元が見つかりません。**」の「**エラー! 参照元が見つかりません。**」を参照してください。

5.2.1 フィルタリング

拡張 IF によるデータ参照時に、以下の種類の条件でフィルタをかけることができます。

フィルタリングについては、付録 A 参考情報 [5]をあわせて参照してください。

- エンティティタイプが存在するエンティティ
- エンティティタイプが存在しないエンティティ
- 特定のエンティティタイプを持つエンティティ

使用例を以下に示します。たとえば、下記リクエストで異なるタイプの 3 つのエンティティ ID “Shop1“が登録されているものとします（エンティティタイプが異なれば、同一のエンティティ ID でエンティティを登録することが可能）。

```
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities " -s -S ¥
 --header "Authorization: Bearer ${TOKEN}"
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
     "id": "Shop1", "type": "",
     "attributes": [ { "name": "purpose", "type": "string", "value": "no type" } ]
}
EOF

(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities " -s -S ¥
 --header "Authorization: Bearer ${TOKEN}"
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
     "id": "Shop1", "type": "Workplace",
     "attributes": [ { "name": "purpose", "type": "string", "value": "type Workplace" } ]
}
EOF

(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities " -s -S ¥
 --header "Authorization: Bearer ${TOKEN}"
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
```

```
{
    "id": "Shop1", "type": "Favorite",
    "attributes": [ {"name": "purpose", "type": "string", "value": "type Favorite" }]
}
EOF
```

エンティティ ID が Shop1 のデータを取得する場合、標準 IF の queryContext を利用して 3 つの Shop1 エンティティのみの情報を取得することができます。

```
(curl -k -X POST "https://${IP}/orion/v1.0/queryContext" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "entities": [
        {
            "type": "",
            "isPattern": "false",
            "id": "Shop1"
        }
    ]
}
EOF
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "purpose",
                        "type": "string",
                        "value": "no type"
                    }
                ],
                "id": "Shop1",
                "isPattern": "false",
                "type": ""
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        },
        {
            "contextElement": {
```

```
                "attributes": [
                    {
                        "name": "purpose",
                        "type": "string",
                        "value": "type Workplace"
                    }
                ],
                "id": "Shop1",
                "isPattern": "false",
                "type": "Workplace"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        },
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "purpose",
                        "type": "string",
                        "value": "type Favorite"
                    }
                ],
                "id": "Shop1",
                "isPattern": "false",
                "type": "Favorite"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

以下の拡張 IF で URL にエンティティ ID“Shop1”を指定しても、3 つのエンティティを取得することはできないことに注意が必要です。

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntities/Shop1" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
```

```
    "contextElement": {
        "attributes": [
            {
                "name": "purpose",
                "type": "string",
                "value": "no type"
            }
        ],
        "id": "Shop1",
        "isPattern": "false",
        "type": ""
    },
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    }
}
```

異なるタイプの同一エンティティ ID が存在する可能性がある場合に、拡張 IF を使用して特定のエンティティを取得するには、エンティティタイプでフィルタリングする必要があります。

エンティティタイプが存在しない Shop1 を拡張 IF で取得するには、URL に” !exist=entity::type”を付加した下記リクエストで取得することが可能です。

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntities/Shop1?!exist=entity::type" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "purpose",
                        "type": "string",
                        "value": "no type"
                    }
                ],
                "id": "Shop1",
                "isPattern": "false",
                "type": ""
            },
            "statusCode": {
                "code": "200",
```

```
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

逆にエンティティタイプが存在する Shop1 を取得するには下記 URL で取得することは可能ですが、複数存在する場合、該当するすべてのエンティティ情報を取得することはできません。

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntities/Shop1?exist=entity::type" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "purpose",
                        "type": "string",
                        "value": "type Workplace"
                    }
                ],
                "id": "Shop1",
                "isPattern": "false",
                "type": "Workplace"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

エンティティタイプが定義された特定のエンティティを持つ Shop1 を取得する際には、以下のように URL にエンティティタイプ（ここでは”Favorite”）を指定します。

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntities?entity::type=Favorite" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "purpose",
                        "type": "string",
                        "value": "type Favorite"
                    }
                ],
                "id": "Shop1",
                "isPattern": "false",
                "type": "Favorite"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

5.3 NGSIv2 バッチオペレーションによるデータの参照（任意）

本処理は NGSIv2 で追加された新しい処理です。NGSIv2 の entities(GET メソッド)でのデータ参照処理は 1 つの条件にしか対応していません。そこでバッチオペレーション(/op/query)によって複数の条件を指定したデータの参照を一度のリクエストで可能にしています。

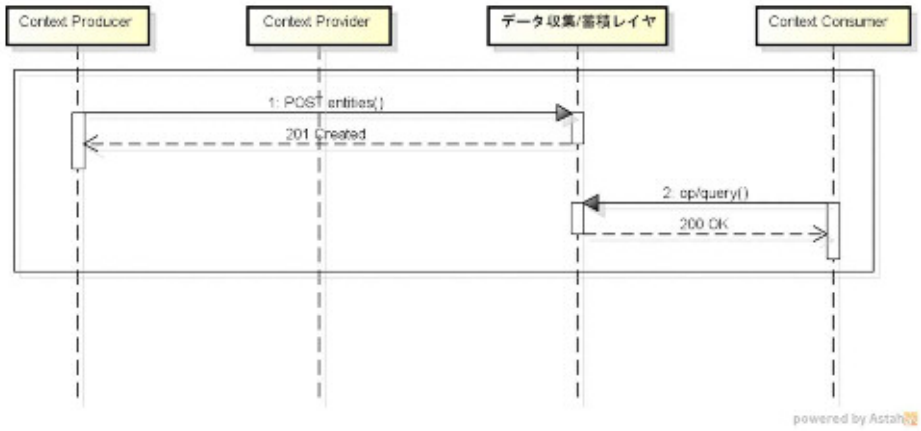

図 5-2 複数条件によるデータ参照

/op/query（リクエスト/レスポンス）の一例を以下に記載します。

下記例では、Room というタイプに所属する Context Element と Car というタイプに所属する Car-1 が保持している温度データ（Attribute）を参照しています。idPattern パラメータを true に設定することで id パラメータに対して正規表現を適用可能です。Context Element 内の全 Attribute 情報が必要であれば attribute パラメータを省略すればよいです。

その他パラメータの詳細は、第 6 章 を参照してください。

```
[/op/query リクエスト]
 (curl -k -X POST "https://${IP}/orion/v2.0/op/query" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @-) <<EOF
 {
    "entities": [
       {
          "idPattern": ".*",
          "type": "Room"
       },
       {
          "id": "Car-1",
          "type": "Car"
       }
    ],
    "attributes": [
       "temperature"
    ]
 }
 EOF
```

/op/query のレスポンスにはリクエストで指定した Context Element の情報およびデータ参照結果（statusCode）が含まれます。

[/op/query レスポンス]


```
[
  {
    "type": "Room",
    "id": "Room-1",
    "temperature": {
      "value": 35.6,
      "type": "Number"
    }
  },
  {
    "type": "Room",
    "id": "Room-2",
    "temperature": {
      "value": 22.5,
      "type": "Number"
    }
  },
  {
    "type": "Car",
    "id": "Car-1",
    "temperature": {
      "value": 35.6,
      "type": "Number"
    }
  }
]
```


レスポンスにはステータスコード 200 が返ります。

[/op/query レスポンス]

```
< HTTP/1.1 200 OK
< Connection: keep-alive
< Content-Type: application/json
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

第6章 API 一覧／仕様

API 一覧/仕様については、付録 A 参考情報 [2] 付録 C 注意事項もあわせて参照してください。

6.1 API 一覧

6.1.1 NGSI v2 インタフェース

No	API 名	HTTP	機能
1	entities	GET	全てのコンテキスト・エンティティに関する情報を検索する
		POST	新しいコンテキスト・エンティティを作成する
2	entities/{エンティティ ID}	GET	指定されたコンテキスト・エンティティに関する情報を検索する
		DELETE	指定されたコンテキスト・エンティティの全ての情報を削除する
3	entities/{エンティティ ID}/attrs	GET	指定されたコンテキスト・エンティティに関する全ての属性情報を検索する（ID, Type は省略される）
		POST	指定されたコンテキスト・エンティティに関する属性情報を追加または更新する
		PATCH	指定されたコンテキスト・エンティティに関する既存の属性情報を更新する
		PUT	指定されたコンテキスト・エンティティに関する全ての属性情報を置換する
4	entities/{エンティティ ID} /attrs/{属性名}	GET	指定されたコンテキスト・エンティティの任意の属性情報を検索する
		PUT	指定されたコンテキスト・エンティティの任意の属性情報を更新する
		DELETE	指定されたコンテキスト・エンティティの任意の属性情報を削除する
5	entities/{エンティティ ID} /attrs/{属性名}/value	GET	指定されたコンテキスト・エンティティの任意の属性値を取得する
		PUT	指定されたコンテキスト・エンティティの任意の属性値を更新する
6	types	GET	すべてのエンティティタイプの情報を取得する
7	types/{タイプ名}	GET	指定されたエンティティタイプの情報を取得する
8	op/update	POST	バッチ更新オペレーションを実行する
9	op/query	POST	バッチ検索オペレーションを実行する

6.2 API 仕様

6.2.1 NGSI v2 インタフェース

1. entities

機能	全てのコンテキスト・エンティティに関する情報を検索する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/entities	
ヘッダ	Accept: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	id	[任意]エンティティ ID idPattern パラメータとは排他的関係にある
	type	[任意]エンティティタイプ typePattern パラメータとは排他的関係にある
	idPattern	[任意]エンティティ ID の正規表現 id パラメータとは排他的関係にある
	typePattern	[任意]エンティティタイプの正規表現 type パラメータとは排他的関係にある
	q	[任意]属性値のクエリ式 詳細は公式ドキュメントの Simple Query Language 項目を参照のこと
	mq	[任意]メタデータのクエリ式 詳細は公式ドキュメントの Simple Query Language 項目を参照のこと
	georal	[任意]座標の位置関係 詳細は公式ドキュメントの Geographical Queries 項目を参照のこと
	geometry	[任意]座標が示すオブジェクトの形状 詳細は公式ドキュメントの Geographical Queries 項目を参照のこと
	coords	[任意]座標。WGS-84 に準拠 詳細は公式ドキュメントの Geographical Queries 項目を参照のこと
	limit	[任意]取得するエンティティの上限数
	offset	[任意]取得するエンティティのオフセット
	attrs	[任意]属性名称 ※カンマ区切りで複数指定可
	metadata	[任意]メタデータ名称

		※カンマ区切りで複数指定可
	orderBy	[任意]ソート条件
	options	[任意]オプション情報。カンマ区切りで複数指定可能 count: レスポンスヘッダにクエリ総数を表示 keyValues: レスポンスを keyvalue 形式で表示 values: レスポンスを属性値のみ表示 unique: レスポンスを重複していない属性値のみ表示
＜レスポンス＞		
ヘッダ	Content-Type: application/json Fiware-Total-Count: ${count}　※${count}はクエリ条件に合致するエンティティ総数	
ボディ	タグ名／キー名	説明
		※ラベル無し配列。以下要素を繰り返し表示する
	id	エンティティ ID
	type	エンティティタイプ
	${AttributeName}	※${AttributeName}は属性名称。 重複しない属性名称が複数表示の可能性有り
	value	属性値
	type	属性タイプ
	metadata	メタデータ情報
	${MetadataName}	※${MetadataName}はメタデータ名称。 重複しないメタデータ名称が複数表示の可能性有り
	type	メタデータタイプ
	value	メタデータ値

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/orion/v2.0/entities" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
 | python -mjson.tool)
```

＜レスポンス＞

```
[
  {
    "type": "Room",
    "id": "DC_S1-D41",
    "temperature": {
      "value": 35.6,
      "type": "Number",
      "metadata": {}
    }
  },
  {
    "type": "Room",
    "id": "Boe-Idearium",
    "temperature": {
```

```
        "value": 22.5,
        "type": "Number",
        "metadata": {}
      }
    },
    {
      "type": "Car",
      "id": "P-9873-K",
      "speed": {
        "value": 100,
        "type": "number",
        "metadata": {
          "accuracy": {
            "value": 2,
            "type": "Number"
          },
          "timestamp": {
            "value": "2015-06-04T07:20:27.378Z",
            "type": "DateTime"
          }
        }
      }
    }
]
```

機能	新しいコンテキスト・エンティティを作成する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v2.0/entities	
ヘッダ	Content-Type: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	options	[任意]オプション情報。カンマ区切りで複数指定可能 keyValues: リクエストボディを keyvalue 形式で指定
ボディ	タグ名／キー名	説明
	id	エンティティ ID
	type	エンティティタイプ
	${AttributeName}	※${AttributeName}は属性名称。 重複しない属性名称を複数指定可

		value				属性値
		type				属性タイプ
		metadata				メタデータ情報
			${MetadataName}			※${MetadataName}はメタデータ名称。 重複しないメタデータ名称を複数指定可
				type		メタデータタイプ
				value		メタデータ値
＜レスポンス＞						
ヘッダ	Location: /orion/v2.0/entities/${id}?type=${type} ※${id}はエンティティ ID, ${type}はエンティティタイプ					
	(ステータスコード)					※リクエスト結果の HTTP ステータスコード

curl コマンド実行例:

```
(curl -k -X POST "https://${IP}/orion/v2.0/entities" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" ¥
  -d @-) <<EOF
{
   "type": "Room",
   "id": "Bcn-Welt",
   "temperature": {
      "value": 21.7
   },
   "humidity": {
      "value": 60
   },
   "location": {
      "value": "41.3763726, 2.1864475",
      "type": "geo:point",
      "metadata": {
         "crs": {
            "value": "WGS84"
         }
      }
   }
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 201 Created
< Connection: keep-alive
< Location: /v2/entities/Bcn-Welt?type=Room
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

2. entities/{エンティティ ID}

機能	指定されたコンテキスト・エンティティに関する情報を検索する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}	
ヘッダ	Accept: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ
	attrs	[任意]属性名称 ※カンマ区切りで複数指定可
	metadata	[任意]メタデータ名称 ※カンマ区切りで複数指定可
	options	[任意]オプション情報。カンマ区切りで複数指定可能 keyValues: レスポンスを keyvalue 形式で表示 values: レスポンスを属性値のみ表示 unique: レスポンスを重複していない属性値のみ表示
＜レスポンス＞		
ヘッダ	Content-Type: application/json	
ボディ	タグ名／キー名	説明
	id	エンティティ ID
	type	エンティティタイプ
	${AttributeName}	※${AttributeName}は属性名称。 重複しない属性名称が複数表示の可能性有り
	value	属性値
	type	属性タイプ
	metadata	メタデータ情報
	${MetadataName}	※${MetadataName}はメタデータ名称。 重複しないメタデータ名称が複数表示の可能性有り
	type	メタデータタイプ
	value	メタデータ値

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/orion/v2.0/entities/Bcn-Welt" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
   | python -mjson.tool)
```

＜レスポンス＞

```
{
  "type": "Room",
  "id": "Bcn-Welt",
  "temperature": {
    "value": 21.7,
    "type": "Number"
  },
  "humidity": {
    "value": 60,
    "type": "Number"
  },
  "location": {
    "value": "41.3763726, 2.1864475",
    "type": "geo:point",
    "metadata": {
      "crs": {
        "value": "WGS84",
        "type": "Text"
      }
    }
  }
}
```

機能	指定されたコンテキスト・エンティティの全ての情報を削除する	
＜リクエスト＞		
HTTP メソッド	DELETE	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}	
ヘッダ	Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ
＜レスポンス＞		
	(ステータスコード)	※リクエスト結果の HTTP ステータスコード

curl コマンド実行例：

```
(curl -k -X DELETE "https://${IP}/orion/v2.0/entities/Bcn-Welt" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}")
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
```

```
< Fiware-Correlator: 8f5c38ba-59d7-11e9-ab8b-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

3. entities/{エンティティ ID}/attrs

機能	指定されたコンテキスト・エンティティに関する全ての属性情報を検索する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs	
ヘッダ	Accept: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ
	attrs	[任意]属性名称 ※カンマ区切りで複数指定可
	metadata	[任意]メタデータ名称 ※カンマ区切りで複数指定可
	options	[任意]オプション情報。カンマ区切りで複数指定可能 keyValues: レスポンスを keyvalue 形式で表示 values: レスポンスを属性値のみ表示 unique: レスポンスを重複していない属性値のみ表示
＜レスポンス＞		
ヘッダ	Content-Type: application/json	
ボディ	タグ名／キー名	説明
	${AttributeName}	※${AttributeName}は属性名称。 重複しない属性名称が複数表示の可能性有り
	value	属性値
	type	属性タイプ
	metadata	メタデータ情報
	${MetadataName}	※${MetadataName}はメタデータ名称。 重複しないメタデータ名称が複数表示の可能性有り
	type	メタデータタイプ
	value	メタデータ値

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
   | python -mjson.tool)
```

＜レスポンス＞

```
{
  "temperature": {
    "value": 21.7,
    "type": "Number"
  },
  "humidity": {
    "value": 60,
    "type": "Number"
  },
  "location": {
    "value": "41.3763726, 2.1864475",
    "type": "geo:point",
    "metadata": {
      "crs": {
        "value": "WGS84",
        "type": "Text"
      }
    }
  }
}
```

機能	指定されたコンテキスト・エンティティに関する属性情報を追加または更新する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs	
ヘッダ	Content-Type: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ
	options	[任意]オプション情報。カンマ区切りで複数指定可能 append: 属性の新規追加のみ有効とする。既存属性の更新となるクエリはエラーを返却する。 keyValues: リクエストボディを keyvalue 形式で指定する
ボディ	タグ名／キー名	説明
	${AttributeName}	※${AttributeName}は属性名称。 重複しない属性名称を複数指定可
	value	属性値
	type	属性タイプ
	metadata	メタデータ情報

			${MetadataName}		※${MetadataName}はメタデータ名称。 重複しないメタデータ名称を複数指定可
				type	メタデータタイプ
				value	メタデータ値
＜レスポンス＞					
	(ステータスコード)				※リクエスト結果の HTTP ステータスコード

curl コマンド実行例:

```
(curl -k -X POST "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @-) <<EOF
{
   "ambientNoise": {
      "value": 31.5
   }
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: fb04822c-59e9-11e9-a8eb-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

機能	指定されたコンテキスト・エンティティに関する既存の属性情報を更新する	
＜リクエスト＞		
HTTP メソッド	PATCH	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs	
ヘッダ	Content-Type: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ
	options	[任意]オプション情報。カンマ区切りで複数指定可能 keyValues: リクエストボディを keyvalue 形式で指定する
ボディ	タグ名／キー名	説明
	${AttributeName}	※${AttributeName}は属性名称。 重複しない属性名称を複数指定可

		value		属性値
		type		属性タイプ
		metadata		メタデータ情報
			${MetadataName}	※${MetadataName}はメタデータ名称。 重複しないメタデータ名称を複数指定可
			type	メタデータタイプ
			value	メタデータ値
＜レスポンス＞				
	(ステータスコード)			※リクエスト結果の HTTP ステータスコード

curl コマンド実行例:

```
(curl -k -X PATCH   "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @-) <<EOF
{
  "temperature": {
    "value": 25.5
  },
  "ambientNoise": {
    "value": 6
  }
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: fb04822c-59e9-11e9-a8eb-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

機能	指定されたコンテキスト・エンティティに関する全ての属性情報を置換する	
＜リクエスト＞		
HTTP メソッド	PUT	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs	
ヘッダ	Content-Type: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ

	options	[任意]オプション情報。カンマ区切りで複数指定可能 keyValues: リクエストボディを keyvalue 形式で指定する
ボディ	タグ名／キー名	説明
	${AttributeName}	※${AttributeName}は属性名称。 重複しない属性名称を複数指定可
	value	属性値
	type	属性タイプ
	metadata	メタデータ情報
	${MetadataName}	※${MetadataName}はメタデータ名称。 重複しないメタデータ名称を複数指定可
	type	メタデータタイプ
	value	メタデータ値
＜レスポンス＞		
	(ステータスコード)	※リクエスト結果の HTTP ステータスコード

curl コマンド実行例:

```
(curl -k -X PUT "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @-) <<EOF
{
   "temperature": {
      "value": 25.5
   },
   "ambientNoise": {
      "value": 6
   }
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: fb04822c-59e9-11e9-a8eb-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

4. entities/{エンティティ ID}/attrs/{属性名}

機能	指定されたコンテキスト・エンティティの任意の属性情報を検索する
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs/{属性名}
ヘッダ	Accept: application/json

	Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ
	metadata	[任意]メタデータ名称 ※カンマ区切りで複数指定可
＜レスポンス＞		
ヘッダ	Content-Type: application/json	
ボディ	タグ名／キー名	説明
	value	属性値
	type	属性タイプ
	metadata	メタデータ情報
	${MetadataName}	※${MetadataName}はメタデータ名称。 重複しないメタデータ名称が複数表示の可能性有り
	type	メタデータタイプ
	value	メタデータ値

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs/temperature" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
 | python -mjson.tool)
```

＜レスポンス＞

```
{
  "value": 21.7,
  "type": "Number",
  "metadata": {}
}
```

機能	指定されたコンテキスト・エンティティの任意の属性情報を更新する	
＜リクエスト＞		
HTTP メソッド	PUT	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs/{属性名}	
ヘッダ	Content-Type: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明

	type	[任意]エンティティタイプ
ボディ	タグ名／キー名	説明
	value	属性値
	type	属性タイプ
	metadata	メタデータ情報
	${MetadataName}	※${MetadataName}はメタデータ名称。 重複しないメタデータ名称を複数指定可
	type	メタデータタイプ
	value	メタデータ値
＜レスポンス＞		
	(ステータスコード)	※リクエスト結果の HTTP ステータスコード

curl コマンド実行例:

```
(curl -k -X PUT "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs/temperature" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @-) <<EOF
{
   "value": 25,
   "metadata": {
      "unitCode": {
         "value": "CEL"
      }
   }
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: fb04822c-59e9-11e9-a8eb-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

機能	**指定されたコンテキスト・エンティティの任意の属性情報を削除する**	
＜リクエスト＞		
HTTP メソッド	DELETE	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs/{属性名}	
ヘッダ	Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメー	パラメータ名	説明

タ		
	type	[任意]エンティティタイプ
＜レスポンス＞		
	(ステータスコード)	※リクエスト結果の HTTP ステータスコード

curl コマンド実行例：

```
(curl -k -X DELETE "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs/temperature" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}")
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: fb04822c-59e9-11e9-a8eb-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

5. entities/{エンティティ ID}/attrs/{属性名}/value

機能	**指定されたコンテキスト・エンティティの任意の属性値を取得する**	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs/{属性名}/value	
ヘッダ	Accept: */* ※value の値が JSON 形式の場合、application/json 指定が有効。その他は text/plain となる Authorization: Bearer ${TOKEN} ※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ
＜レスポンス＞		
ヘッダ	Content-Type: */* レスポンスボディが JSON 形式の場合、application/json。その他は text/plain となる	
ボディ	タグ名／キー名	説明
		※属性値

curl コマンド実行例①： JSON 形式の属性値を取得

```
(curl -k -X GET "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs/address/value" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
   "address": "Ronda de la Comunicacion s/n",
```

```
  "zipCode": 28050,
  "city": "Madrid",
  "country": "Spain"
}
```

curl コマンド実行例②： 数値型の属性値を取得

```
(curl -k -X GET "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs/temperature/value" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: text/plain")
```

＜レスポンス＞

```
21.7
```

機能	指定されたコンテキスト・エンティティの任意の属性値を更新する	
＜リクエスト＞		
HTTP メソッド	PUT	
URL	https://${IP}/orion/v2.0/entities/{エンティティ ID}/attrs/{属性名}/value	
ヘッダ	Content-Type: */*　※ボディの形式が JSON 形式の場合は application/json 指定が有効。その他は text/plain となる Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	type	[任意]エンティティタイプ
ボディ	タグ名／キー名	説明
		※属性値
＜レスポンス＞		
	(ステータスコード)	※リクエスト結果の HTTP ステータスコード

curl コマンド実行例①： JSON 形式の属性値で更新

```
(curl -k -X PUT "https://${IP}/orion/v2.0/entities/Bcn-Welt/attrs/address/value" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" ¥
 -d @-) <<EOF
{
  "address": "Ronda de la Comunicacion s/n",
  "zipCode": 28050,
  "city": "Madrid",
  "country": "Spain"
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: fb04822c-59e9-11e9-a8eb-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

curl コマンド実行例②: 数値型の属性値で更新

```
(curl -k -X PUT  "https://${IP}/orion/v2.0/entities/Bcn_Welt /attrs/temperature/value" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" ¥
 -d @-) <<EOF
25.5
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: fb04822c-59e9-11e9-a8eb-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

6. types

機能	すべてのエンティティタイプの情報を取得する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/types	
ヘッダ	Accept: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	limit	[任意]取得するタイプの上限数
	offset	[任意]取得するタイプのオフセット
	options	[任意]オプション情報。カンマ区切りで複数指定可能 count: レスポンスヘッダにクエリ総数を表示 values: レスポンスを属性値のみ表示
＜レスポンス＞		
ヘッダ	Content-Type: application/json Fiware-Total-Count: ${count}　※${count}はクエリ条件に合致するエンティティ総数	
ボディ	タグ名／キー名	説明
		※ラベル無し配列。以下要素を繰り返し表示する
	type	エンティティタイプ
	attrs	属性情報

		${AttributeName}				※${AttributeName}は属性名称 重複しない属性名称が複数存在する可能性あり
			types			属性タイプリスト(配列)
				${AttributeType}		※${AttributeType}は属性タイプ 重複しない属性タイプが複数存在する可能性あり
	count					このエンティティタイプを持つエンティティの総数

curl コマンド実行例:

```
(curl -k -X GET "https://${IP}/orion/v2.0/types" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
 | python -mjson.tool)
```

＜レスポンス＞

```
[
  {
    "type": "Car",
    "attrs": {
      "speed": {
        "types": [
          "Number"
        ]
      },
      "fuel": {
        "types": [
          "gasoline",
          "diesel"
        ]
      },
      "temperature": {
        "types": [
          "urn:phenomenum:temperature"
        ]
      }
    },
    "count": 12
  },
  {
    "type": "Room",
    "attrs": {
      "pressure": {
        "types": [
          "Number"
        ]
      },
      "humidity": {
        "types": [
```

```
            "percentage"
          ]
        },
        "temperature": {
          "types": [
            "urn:phenomenum:temperature"
          ]
        }
      },
      "count": 7
    }
]
```

7. types/{タイプ名}

機能	指定されたエンティティタイプの情報を取得する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/types/{タイプ名}	
ヘッダ	Accept: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
＜レスポンス＞		
ヘッダ	Content-Type: application/json	
ボディ	タグ名／キー名	説明
	attrs	属性情報
	${AttributeName}	※${AttributeName}は属性名称 重複しない属性名称が複数存在する可能性あり
	types	属性タイプリスト(配列)
	${AttributeType}	※${AttributeType}は属性タイプ 重複しない属性タイプが複数存在する可能性あり
	count	このエンティティタイプを持つエンティティの総数

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/orion/v2.0/types/Room" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
```

```
  "attrs": {
    "pressure": {
      "types": [
        "Number"
      ]
    },
    "humidity": {
      "types": [
        "percentage"
      ]
    },
    "temperature": {
      "types": [
        "urn:phenomenum:temperature"
      ]
    }
  },
  "count": 7
}
```

8. op/update

機能	バッチ更新オペレーションを実行する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v2.0/op/update	
ヘッダ	Content-Type: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	options	[任意]オプション情報。カンマ区切りで複数指定可能 keyValues: リクエストボディに含まれるエンティティを keyvalue 形式で指定する
ボディ	タグ／キー名	説明
	actionType	用途により、以下のいずれかを指定。 "APPEND" :エンティティの作成、既存エンティティの属性の作成／更新 "APPEND_STRICT":エンティティの作成、既存エンティティの属性の作成（既存エンティティの既存属性に対する操作時はエラー） "UPDATE":既存エンティティの既存属性の更新（存在しないエンティティ、および既存エンティティに存在しない属性に対する操作時はエラー）

							"DELETE"：エンティティ自体、もしくはエンティティの属性の削除 "REPLACE"：既存エンティティの属性置換（既存属性はすべて削除され、新たに指定した属性が追加される）
	entities						エンティティリスト(配列)
		type					エンティティタイプ
		id					エンティティ ID
		${AttributeName}					※${AttributeName}は属性名称 重複しない属性名称を複数指定可能
			value				属性値
			type				属性タイプ
			metadata				メタデータ情報
				${MetadataName}			※${MetadataName}はメタデータ名称 重複しないメタデータ名称を複数指定可能
					type		メタデータタイプ
					value		メタデータ値
＜レスポンス＞							
	(ステータスコード)						※リクエスト結果の HTTP ステータスコード

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v2.0/op/update" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @-) <<EOF
{
   "actionType": "APPEND",
   "entities": [
      {
         "type": "Room",
         "id": "Bcn-Welt",
         "temperature": {
            "value": 21.7
          },
         "humidity": {
            "value": 60
         }
      },
      {
         "type": "Room",
         "id": "Mad_Aud",
         "temperature": {
            "value": 22.9
         },
         "humidity": {
```

```
        "value": 85
      }
    }
  ]
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

9. op/query

機能	バッチ検索オペレーションを実行する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v2.0/op/query	
ヘッダ	Content-Type: application/json Accept: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意) Fiware-ServicePath: (任意)	
クエリパラメータ	パラメータ名	説明
	limit	[任意]取得するエンティティの上限数
	offset	[任意]取得するエンティティのオフセット
	orderBy	[任意]ソート条件
	options	[任意]オプション情報。カンマ区切りで複数指定可能 count: レスポンスヘッダにクエリ総数を表示 keyValues: レスポンスを keyvalue 形式で表示 values: レスポンスを属性値のみ表示 unique: レスポンスを重複していない属性値のみ表示
ボディ	タグ名／キー名	説明
	entities	エンティティリスト(配列)
	id	エンティティ ID idPattern パラメータとは排他的関係にある
	type	エンティティタイプ typePattern パラメータとは排他的関係にある
	idPattern	エンティティ ID の正規表現 id パラメータとは排他的関係にある
	typePattern	エンティティタイプの正規表現 type パラメータとは排他的関係にある

	attributes	属性リスト(配列)
	metadata	メタデータリスト(配列)
＜レスポンス＞		
ヘッダ	Content-Type: application/json Fiware-Total-Count: ${count}　※${count}はクエリ条件に合致するエンティティ総数	
ボディ	タグ名／キー名	説明
		※ラベル無し配列。以下要素を繰り返し表示する
	id	エンティティ ID
	type	エンティティタイプ
	${AttributeName}	※${AttributeName}は属性名称 重複しない属性名称が複数表示の可能性あり
	type	属性タイプ
	value	属性値
	metadata	メタデータ情報
	${MetadataName}	※${MetadataName}はメタデータ名称 重複しないメタデータ名称が複数表示の可能性あり
	type	メタデータタイプ
	value	メタデータ値

※op/query のボディに scopes フィールドを指定し、フィルタリングすることも可能ですが、将来利用できなくなります。op/query の scopes フィールドについては付録 A 参考情報 [7]を参照してください。

curl コマンド実行例:

```
(curl -k -X POST "https://${IP}/orion/v2.0/op/query" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @-) <<EOF
{
   "entities": [
      {
         "idPattern": ".*",
         "type": "myFooType"
      },
      {
         "id": "myBar",
         "type": "myBarType"
      }
   ],
   "attributes": [
      "temperature",
      "humidity"
   ],
   "metadata": [
      "accuracy",
      "timestamp"
```

```
  ]
}
EOF
```

＜レスポンス＞

```
[
  {
    "type": "Room",
    "id": "DC_S1-D41",
    "temperature": {
      "value": 35.6,
      "type": "Number"
    }
  },
  {
    "type": "Room",
    "id": "Boe-Idearium",
    "temperature": {
      "value": 22.5,
      "type": "Number"
    }
  },
  {
    "type": "Car",
    "id": "P-9873-K",
    "speed": {
      "value": 100,
      "type": "number",
      "accuracy": 2,
      "timestamp": {
        "value": "2015-06-04T07:20:27.378Z",
        "type": "DateTime"
      }
    }
  }
]
```

6.3 NGSI v2 API 利用時の仕様・制限等

6.3.1 データ収集/蓄積レイヤ単体

本開発ガイド中の NGSIv2 の API を実行する URL 中の /orion/v2.0 は、/orion/v2 でも利用可能です。

NGSIv2 では、属性値を数値、ブール値(true,false)で登録することも可能です。それにより、数値比較等の filter を活用することができますが、参照するデータは NGSIv2 で登録したものに限ります。NGSIv1 で登録したデータは文字列形式で判定されるため、正常に filter が動作しない可能性があります。詳しくは付録 A 参考情報 [8]を参照してください。

付録A　参考情報

項番	タイトル	URL
[1]	Fiware Orion	http://fiware-orion.readthedocs.io/en/1.14.0/
[2]	Fiware NGSI APIv1 Walkthrougn	http://fiware-orion.readthedocs.io/en/1.14.0/user/walkthrough_apiv1/index.html
[3]	Multi tenancy	http://fiware-orion.readthedocs.io/en/1.14.0/user/multitenancy/index.html
[4]	Entity service paths	http://fiware-orion.readthedocs.io/en/1.14.0/user/service_path/index.html
[5]	NGSIv1 filtering	http://fiware-orion.readthedocs.io/en/1.14.0/user/filtering/index.html#ngsiv1-filtering
[6]	HTTP and NGSI response codes	http://fiware-orion.readthedocs.io/en/1.14.0/user/http_and_ngsi_sc/index.html
[7]	FIWARE NGSI APIv2 Walkthrough	http://fiware-orion.readthedocs.io/en/1.14.0/user/walkthrough_apiv2/
[8]	NGSIv2 filtering	http://fiware-orion.readthedocs.io/en/1.14.0/user/filtering/index.html#ngsiv2-filtering

付録B　ステータスコード一覧

付録 A 参考情報 [6] もあわせて参照してください。

No	コード	内容
1	200	OK
2	201	Created
3	204	No Content
4	400	Bad Request
5	403	Forbidden
6	404	No context element found
7	405	Method Not Allowed
8	406	Not Acceptable
9	409	Too Many Results
10	411	Content Length Required
11	413	Request Entity Too Large
12	415	Unsupported Media Type
13	422	Invalid Modification
14	470	subscriptionId does not correspond to an active subscription
15	471	parameter missing in the request
16	472	request parameter is invalid/not allowed
17	473	Generic error in metadata
18	480	Regular Expression for EntityId is not allowed by receiver
19	481	EntityType required by the receiver
20	482	Attribute List required by the receiver
21	500	Internal Server Error
22	501	Not Implemented

付録C　注意事項

（1）Orion-API を呼び出す場合の注意

Orion-API を呼び出す場合に、リクエストのボディサイズが特定のサイズ（約 15KB）より大きいと HTTP ステータス：411 エラーが発生することがあります。下記の対応を行ってください。

・リクエストのボディサイズを 15KB 以下にしてください。大量の Attribute を保有するデータ(エンティティ)を登録する場合にリクエストのボディサイズが 15KB を超えてしまう場合は、当該エンティティに対する Attribute を複数回にわたり分割して登録してください。

・レスポンスの HTTP ステータスで 411 が返却された場合は、再度リクエストを送ってください。