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 2.0 版
2019 年 10 月

日本電気株式会社

データ利活用基盤-第 17-011 号

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

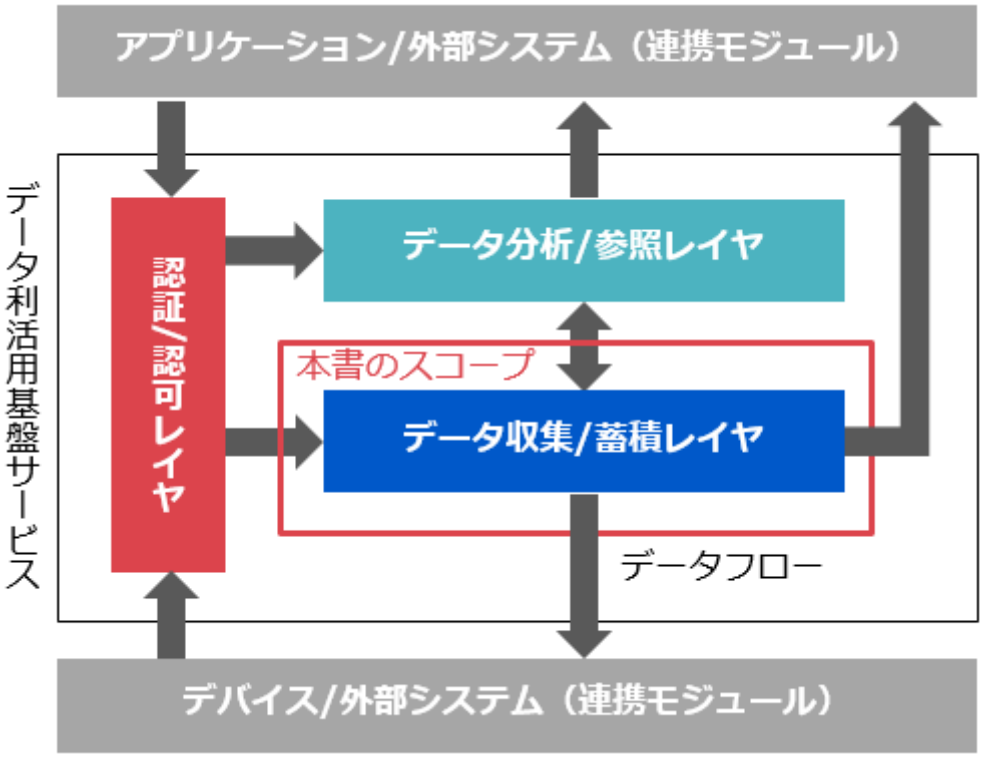

図 1-1 システム概要

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	2.0 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）	2.0 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）	2.0 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 2.1.0 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
3. データ更新通知

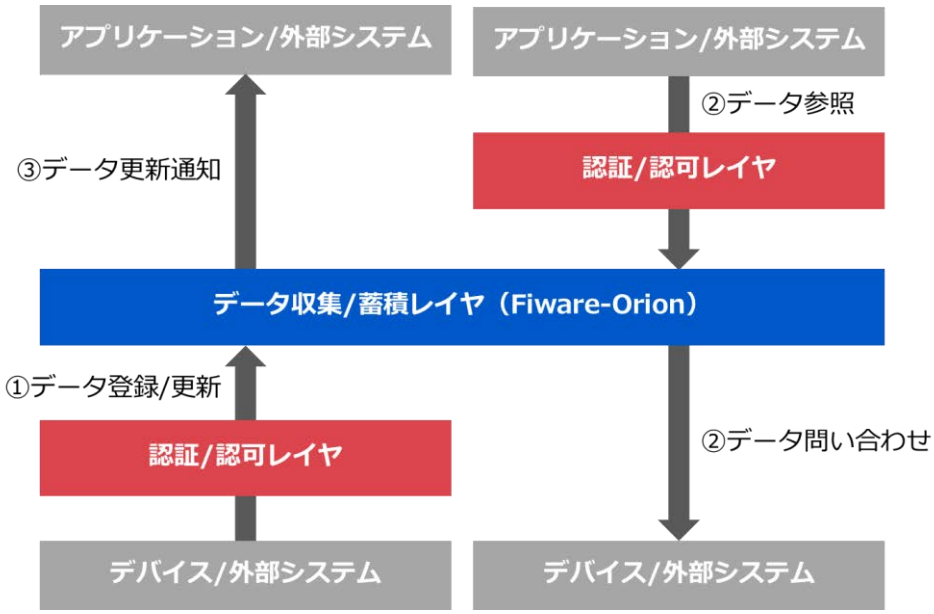

図 2-1 機能概要

「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。

「③データ更新通知」は、アプリケーションや外部システムに対してデータの更新通知を行う機能です。データ登録/更新をトリガとして即座にデータを参照したい場合は、本機能を利用します。

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3つに分類しますが、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 Provider の両方に所属するアクタや、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 Provider
3. 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。

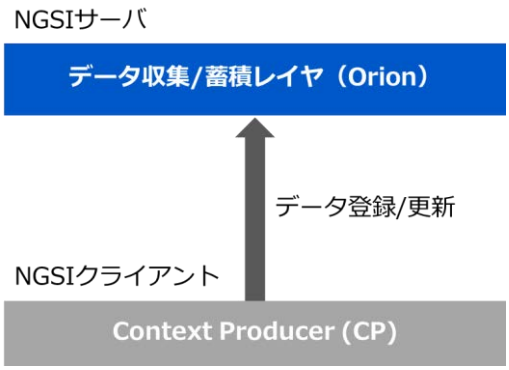

図 3-2 Context Producer

3.1.2 Context Provider

Context Provider（CPr）は、データ生産者の一種であり、データ収集/蓄積レイヤからのデータ問い合わせに応答するアクタです。CP と異なり、データ収集/蓄積レイヤに対して能動的にデータ登録/更新は行いませんが、データ収集/蓄積レイヤが問い合わせできるように事前にデータの所在（CPr がどのようなデータを提供可能か）を登録する必要があります。

したがって、CPr の前提条件としては下記2点が挙げられます。

- 事前にデータ生産者の登録を行うこと
- データ問い合わせ用の NGSI に応答可能であること（NGSI サーバー機能を有すること）

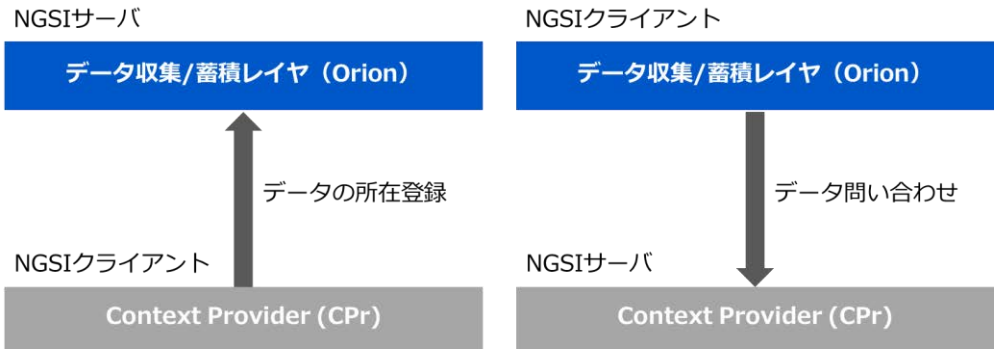

図 3-3 Context Provider

3.1.3 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やデータ収集/蓄積レイヤからのデータ更新通知を受信するアクタです。CC はデータ参照とデータ更新通知受信の両方に対応する必要はなく、用途に合わせて必要な機能が実装されていればよいです。

したがって、前提条件としては下記2点が挙げられます。

- 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）
- データ更新通知を受信する場合
  データ更新通知用の NGSI に応答可能であること（NGSI サーバー機能を有すること）

図 3-4 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、Value 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。

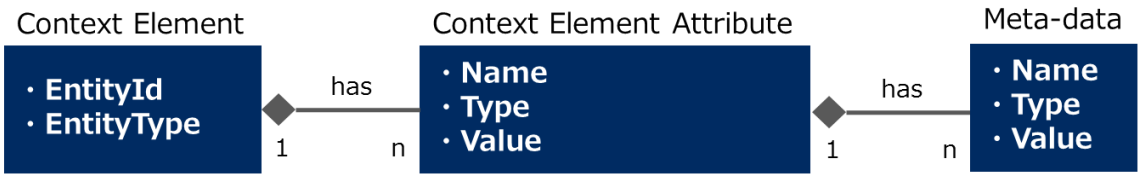

図 3-5 NGSI データモデル

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 v1

⚠注意

NGSI v1 API は、現在は互換性維持のために利用が可能ですが、将来利用できなくなります。

NGSIv1 の API は用途により NGSI-10 と NGSI-9 に分けられます。

3.3.1 NGSI-10

NGSI-10 は「データそのもの」をやりとりするインタフェース(RESTful API)であり、全 NGSI アクタ(CP/CPr/CC)で利用されます。NGSI-10 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 6 章 を、詳細については第 8 章 を参照してください。また、付録 A 参考情報 [2]もあわせて参照してください。

表 3-1 NGSI-10

NGSI-10	用途
updateContext	データ登録/更新
queryContext	データ参照
subscribeContext	データ更新通知予約
updateContextSubscription	データ更新通知予約の変更
unsubscribeContext	データ更新通知予約の削除
notifyContext	データ更新通知
拡張 IF	データ参照/追加/更新/削除など

3.3.2 NGSI-9

NGSI-9 は「データそのもの」ではなく「データの所在(入手先)」をやりとりするインタフェース(RESTful API)であり、主に CPr と連携する際に利用されます。CPr はデータ収集/蓄積レイヤに対して能動的にデータを登録/更新しないため、CC はまず NGSI-9 によりデータの所在を確認し、NGSI-10 によりデータそのものを入手する必要があります。

NGSI-9 は、表 3-2 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 6 章 を、詳細については第 8 章 を参照してください。また、付録 A 参考情報 [2]もあわせて参照してください。

表 3-2 NGSI-9

NGSI-9	用途
registerContext	データの所在登録/更新/削除
discoverContextAvailability	データの所在検索
subscribeContextAvailability	データの所在更新通知予約
updateContextAvailability	データの所在更新通知予約の変更

unsubscribeContextAvailability	データの所在更新通知予約の削除
notifyContextAvailability	データの所在更新通知
拡張 IF	データの所在登録/検索、 データの所在更新通知予約/更新/削除など

3.4 NGSI v2

NGSI v2 は、データの登録、参照、更新、削除、更新通知など、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 3-3 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 6 章 を、詳細については第 8 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。

データの所在検索機能、データの所在更新通知機能(NGSI-9)についてはまだ開発されていません。このため、データの所在検索、データの所在更新通知機能を使用したい場合は NGSIv1 の API を使用する必要があります。

表 3-3 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)
subscriptions	データ更新通知予約の参照/登録
subscriptions/{subscriptionId}	データ更新通知予約の参照/更新/削除 (ID 指定)
registrations	データの所在参照/登録
registrations/{registrationId}	データの所在参照/更新/削除
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

3.4.1 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。

- NGSIv1 は文字列だったデータの属性値の型の扱いが厳密化され、NGSIv2 で登録、更新されたデータであれば、検索条件への属性値の範囲指定や、更新通知に条件を指定して、ある属性値が指定した値を超えたら通知を行うことができます。

3.4.2 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
}
```

- データの所在検索、データの所在更新通知機能を使用したい場合は NGSIv1 の API を使用する必要があります。
- 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。

3.5 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)。データ更新通知予約(subscribeContext)を行う場合も同様です。

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、QuantumLeap でのデータ参照や、CKAN へのデータ登録なども あわせて参照してください。

メモ

QuantumLeap は、Managed 版のみ利用することができます。

Fiware Service および Fiware ServicePath の一例

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

表 3-4 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 3-5 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。

3.6 NGSI 非対応デバイスを用いたデータ収集

データ利活用基盤サービス（FIWARE）では NGSI 形式以外のデータを収集することが可能です。これは、データ変換アダプタとして IDAS という OSS を使用しており、NGSI 非対応デバイスから送信された NGSI 形式以外のデータを NGSI へ変換してデータ収集/蓄積レイヤへデータ登録を実施するためです。NGSI 非対応デバイスとデータ変換アダプタ間は MQTT プロトコルを用いて通信することができます。

IDAS を使用して NGSI 形式以外のデータを収集するためにはデータ変換アダプタに対して NGSI 非対応デバイスが通知してくるデータの型と名前などを予め登録しておくことが必要です。

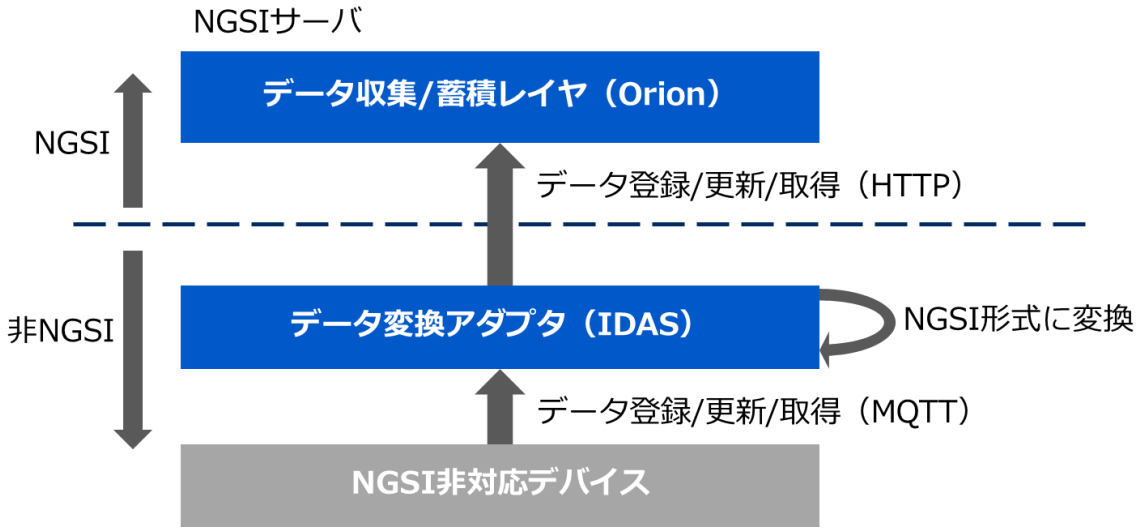

図 3-6 データ変換アダプタを用いた非 NGSI データの変換

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。CP が行う処理は大まかに下記２つですが、データの所在登録は任意であり、必須なのはデータ登録/更新のみです。

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. データの所在登録（任意）
2. 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 データの所在登録（任意）

CP にとってデータの所在登録は必須ではありませんが、Context Consumer（CC）によるデータの所在検索を許可する場合は事前に登録しておく必要があります。

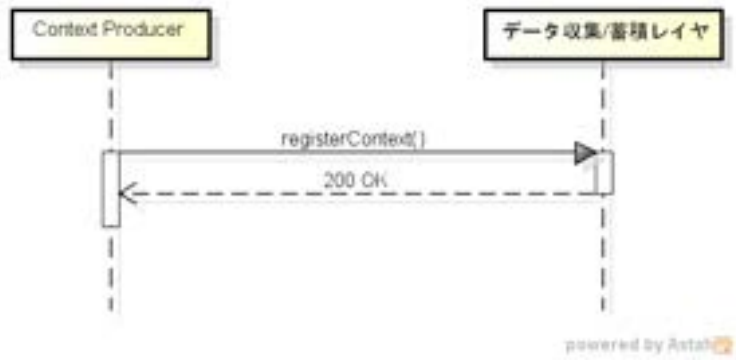

図 4-1 データの所在登録（Context Producer）

4.1.1 NGSIv1

NGSIv1 でのデータの所在登録には registerContext を利用します。registerContext（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/湿度を提供する CP（センサ）を想定しています。CP が registerContext を行う際に注意する必要があるのは、「providingApplication」のパラメータで

す。providingApplication は、CC（データ収集/蓄積レイヤを含む）がデータの問い合わせ時などに利用するベース URL であり、下記の例では「http://example.com:1026/v1/queryContext」というURL（リソース）に対して CC がデータ問い合わせ（queryContext）を発行します。

providingApplication に設定すべき値は、「CP が NGSI-10 のサーバー機能を有しているか」で決まります。

- CP が NGSI-10 のサーバー機能を有している場合
  providingApplication: CP の NGSI-10 サーバーURL
- CP が NGSI-10 のサーバー機能を有していない場合
  providingApplication: データ収集/蓄積レイヤのベース URL
  （https://${IP}/orion/v1.0）

その他パラメータの詳細は、「5.1 データの所在登録（必須）」および第 8 章 を参照してください。

[registerContext リクエスト]

```
 (curl -k -X POST "https://${IP}/orion/v1.0/registry/registerContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
 {
      "contextRegistrations": [
           {
                "entities": [
                     {
                          "type": "Room",
                          "isPattern": "false",
                          "id": "Room-1"
                     }
                ],
                "attributes": [
                     {
                          "name": "temperature",
                          "type": "float"
                     },
                     {
                          "name": "humidity",
                          "type": "integer"
                     }
                ],
                "providingApplication": "http://example.com:1026/v1"
           }
      ],
      "duration": "P1M"
 }
 EOF
```

[registerContext レスポンス]

```
{
    "duration": "P1M",
    "registrationId": "58d74820e46f5af904162a00"
}
```

4.1.2 NGSIv2

NGSIv2 でのデータの所在登録には POST メソッドによる registrations を利用します。POST メソッドによる registrations（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/湿度を提供する CP（センサ）を想定しています。CP が registrations を行う際に注意する必要があるのは、「url」のパラメータです。url は、CC（データ収集/蓄積レイヤを含む）がデータの問い合わせ時などに利用するベース URL であり、下記の例では「http://example.com:1026/v1/queryContext」という URL（リソース）に対して CC がデータ問い合わせ（queryContext）を発行します。legacyForwarding パラメータには true を指定する必要があります。また、NGSIv2 の registrations の有効期限指定はできません。

url に設定すべき値は、「CP が NGSI-10 のサーバー機能を有しているか」で決まります。

- CP が NGSI-10 のサーバー機能を有している場合
  url: CP の NGSI-10 サーバーURL
- CP が NGSI-10 のサーバー機能を有していない場合
  url: データ収集/蓄積レイヤのベース URL
  （https://${IP}/orion/v1.0）

その他パラメータの詳細は、「5.1 データの所在登録（必須）」および第 8 章 を参照してください。

```
[registrations リクエスト]
 (curl -k -X POST "https://${IP}/orion/v2.0/registrations" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" ¥
  -d @- ) <<EOF
 {
   "description": "Registration for Room-1",
   "dataProvided": {
      "entities": [
         {
            "id": "Room-1",
            "type": "Room"
         }
      ],
      "attrs": [
         "temperature",
         "humidity"
      ]
   },
   "provider": {
      "http": {
         "url": "http://example.com:1026/v1"
      },
      "legacyForwarding": true
   }
 }
 EOF
```

registrations はレスポンスヘッダの Location に、registrationId を含む URL をセットしてステータスコード 201 を返却します。

```
[registrations レスポンス]
 < HTTP/1.1 201 Created
 < Connection: keep-alive
 < Location: /v2/registrations/5a82be3d093af1b94ac0f730
 < Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
 < Date: Tue, 13 Feb 2018 10:30:21 GMT
```

4.2 データの所在更新/削除（任意）

データの所在登録と同様に必須ではありませんが、データの所在登録を行った場合は必須です。

データの所在更新は CP が提供可能なデータ（Context Element）に変更が生じた場合や有効期限の更新時に行う処理であり、データの所在削除は CP がデータ収集/蓄積レイヤとの連携を停止する場合に行う処理です。更新、削除いずれの処理においてもNGSI-9 の registerContext または NGSIv2 の registrations/{registrationId}を利用します。また、データの所在登録時に入手した registrationId が必要です。

データの所在更新/削除の詳細に関しては、「5.2 データの所在更新/削除（必須）」を参照してください。

4.3 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。

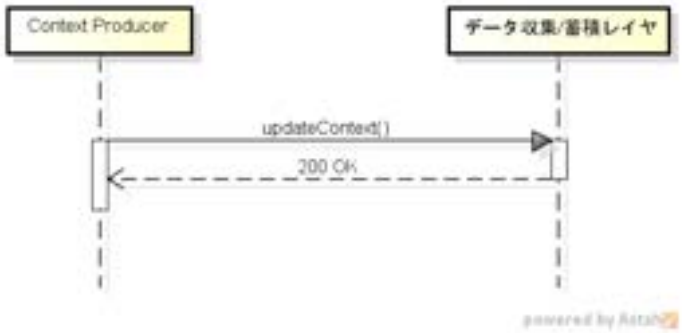

図 4-2 データの登録/更新

4.3.1 NGSIv1

NGSIv1 でのデータの登録/更新には updateContext を利用します。updateContext（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。updateAction 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 8 章 を参照してください。

[updateContext リクエスト]

```
(curl -k -X POST "https://${IP}/orion/v1.0/updateContext" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "contextElements": [
        {
            "type": "Room",
            "isPattern": "false",
            "id": "Room-1",
            "attributes": [
                {
                    "name": "temperature",
                    "type": "float",
                    "value": "20.5"
                },
                {
                    "name": "humidity",
                    "type": "integer",
                    "value": "50"
                }
            ]
        }
    ],
    "updateAction": "APPEND"
}
EOF
```

updateContext のレスポンスには、リクエストで指定した Context Element の情報および登録/更新結果（statusCode）が含まれます。なお、Context Element 内の value は空で返ってくることに注意しください。

[updateContext レスポンス]


```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": ""
                    },
                    {
                        "name": "humidity",
                        "type": "integer",
                        "value": ""
                    }
                ],
                "id": "Room-1",
                "isPattern": "false",
                "type": "Room"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```


4.3.2 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 8 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
```

4.4 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。

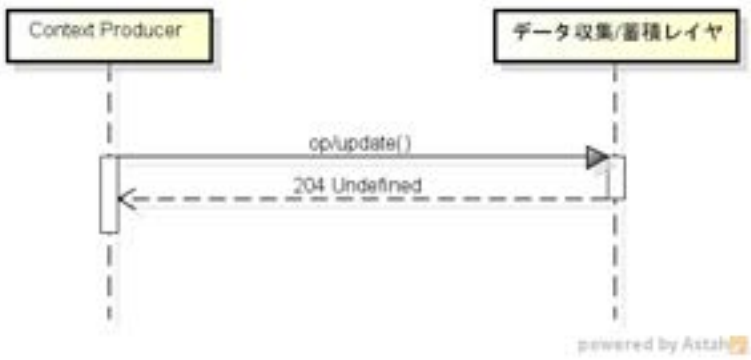

図 4-3 複数データの登録/更新

/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 8 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 Provider 開発者向けガイド

本章では Context Provider（CPr）開発者向けガイドとして、CPr 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。CPr が行う処理は大まかに下 7 つですが、必須なのは一部項目のみです。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. データの所在登録（必須）
2. データの所在更新/削除（必須）
3. データ問い合わせ応答（必須）
4. データ更新通知予約応答（任意）
5. データ更新通知予約の更新応答（任意）
6. データ更新通知予約の削除応答（任意）
7. データ更新通知（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 データの所在登録（必須）

CPr はデータ収集/蓄積レイヤに対して能動的にデータ登録/更新を行わないアクタであるため、データの所在登録は必須かつ最初に行う処理です。

図 5-1 データの所在登録（Context Provider）

5.1.1 NGSIv1

NGSIv1 でのデータの所在登録には registerContext を利用します。registerContext（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/湿度を提供する CPr（センサ）を想定しています。

providingApplication は CC（データ収集/蓄積レイヤを含む）がデータの問い合わせ時などに利用するベース URL であり、下記例ではデータ問い合わせ時に「http://example.com:1026/v1/queryContext」という URL（リソース）に対してデータ問い合わせ（queryContext）が発行されます。

データの所在登録には有効期限があり duration パラメータで期限を指定します。有効期限は下記計算式で算出されます。その他パラメータの詳細は、第 8 章 を参照してください。

有効期限＝現在時刻＋duration

```
[registerContext リクエスト]
 (curl -k -X POST "https://${IP}/orion/v1.0/registry/registerContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
 {
      "contextRegistrations": [
           {
                "entities": [
                     {
                          "type": "Room",
                          "isPattern": "false",
                          "id": "Room-1"
                     }
                ],
                "attributes": [
                     {
                          "name": "temperature",
                          "type": "float"
                     },
                     {
                          "name": "humidity",
                          "type": "integer"
                     }
                ],
                "providingApplication": "http://example.com:1026/v1"
           }
      ],
      "duration": "P1M"
 }
 EOF
```

registerContext のレスポンスには duration と registrationId が含まれます。registrationId はデータの所在登録の識別子（24 桁の 16 進数）であり、データの所在更新/削除処理で利用するため CPr 内で保持する必要があります。

[registerContext レスポンス]

```
{
    "duration": "P1M",
    "registrationId": "58d74820e46f5af904162a00"
}
```

5.1.2 NGSIv2

NGSIv2 でのデータの所在登録には POST メソッドによる registrations を利用します。POST メソッドによる registrations（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/湿度を提供する CPr（センサ）を想定しています。CPr が registrations を行う際に注意する必要があるのは、「url」のパラメータです。url は、CC（データ収集/蓄積レイヤを含む）がデータの問い合わせ時などに利用するベース URL であり、下記の例では「http://example.com:1026/v1/queryContext」という URL（リソース）に対して CC がデータ問い合わせ（queryContext）を発行します。legacyForwarding パラメータには true を指定する必要があります。また、NGSIv2 の registrations の有効期限指定はできません。

その他パラメータの詳細は、「5.1 データの所在登録（必須）」および第 8 章 を参照してください。

```
[registrations リクエスト]
 (curl -k -X POST "https://${IP}/orion/v2.0/registrations" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" ¥
  -d @- ) <<EOF
 {
   "description": "Registration for Room-1",
   "dataProvided": {
      "entities": [
         {
            "id": "Room-1",
            "type": "Room"
         }
      ],
      "attrs": [
         "temperature",
         "humidity"
      ]
   },
   "provider": {
      "http": {
         "url": "http://example.com:1026/v1"
      },
      "legacyForwarding": true
   }
 }
 EOF
```

registrations はレスポンスヘッダの Location に、registrationId を含む URL をセットしてステータスコード 201 を返します。

```
[registrations レスポンス]
 < HTTP/1.1 201 Created
 < Connection: keep-alive
 < Location: /v2/registrations/5a82be3d093af1b94ac0f730
 < Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
 < Date: Tue, 13 Feb 2018 10:30:21 GMT
```

5.2 データの所在更新/削除（必須）

データの所在登録には有効期限があるため、CPr は必要に応じてデータの所在更新/削除を行う必要があります。また、データの所在更新については CPr が提供可能なデータ（Context Element）に変更が生じた場合にも実行する必要があります。データの所在更新/削除には NGSI-9 の registerContext または NGSIv2 の registrations/{registrationId}を利用し、更新/削除対象を特定するため registrationId が必要です。

5.2.1 NGSIv1（更新）

NGSIv1 では、データの所在更新に registerContext を使用します。更新時の registerContext（リクエスト/レスポンス）の一例を以下に記載します。

データの所在更新リクエストは、データの所在登録リクエストに registrationId を追加して durationを更新した内容です。更新時の有効期限算出方法はデータの所在登録と同様です。また、提供可能なデータ（Context Element）に変更が生じた場合には entities や attributes の情報を必要に応じて更新します。

レスポンスにはデータの所在登録と同様、duration と registrationId が含まれます。

その他パラメータの詳細は第 8 章 を参照してください。

[registerContext リクエスト]

```
 (curl -k -X POST "https://${IP}/orion/v1.0/registry/registerContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
 {
      "contextRegistrations": [
           {
                "entities": [
                     {
                          "type": "Room",
                          "isPattern": "false",
                          "id": "Room-1"
                     }
                ],
                "attributes": [
                     {
                          "name": "temperature",
                          "type": "float"
                     },
                     {
                          "name": "humidity",
                          "type": "integer"
                     }
                ],
                "providingApplication": "http://example.com:1026/v1"
           }
      ],
      "duration": "P3M",
      "registrationId": "58d74820e46f5af904162a00"
 }
 EOF
```

[registerContext レスポンス]

```
{
    "duration": "P3M",
    "registrationId": "58d74820e46f5af904162a00"
}
```

5.2.2 NGSIv2（更新）

NGSIv2 のデータの所在の更新機能（PATCH メソッドの registrations/{registrationId}）は現在開発中です。NGSIv2 でデータの所在の更新を行いたい場合は DELETE メソッドの registrations/{registrationId}でデータの所在を削除し、再度 POST メソッドの registrations でデータの所在を作成する必要があります。

5.2.3 NGSIv1（削除）

NGSIv1 でデータの所在を削除する時の registerContext（リクエスト/レスポンス）の一例を以下に記載します。

データの所在削除リクエストは、データの所在登録リクエストに registrationId を追加して duration に 0 を指定した内容です。厳密にはデータの所在登録は削除することができないため、duration を 0 に指定して期限切れとすることで無効化します。

再登録する場合はデータの所在更新処理を行えばよいです（データの所在登録ではないことに注意）。したがって、再登録の可能性がある場合は registrationId を破棄しないようにしてください。

レスポンスにはデータの所在登録と同様、duration と registrationId が含まれます。

[registerContext リクエスト]

```
 (curl -k -X POST "https://${IP}/orion/v1.0/registry/registerContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
 {
      "contextRegistrations": [
           {
                "entities": [
                     {
                          "type": "Room",
                          "isPattern": "false",
                          "id": "Room-1"
                     }
                ],
                "attributes": [
                     {
                          "name": "temperature",
                          "type": "float"
                     },
                     {
                          "name": "humidity",
                          "type": "integer"
                     }
                ],
                "providingApplication": "http://example.com:1026/v1"
           }
      ],
      "duration": "PT0S",
      "registrationId": "58d74820e46f5af904162a00"
 }
 EOF
```

[registerContext レスポンス]

```
{
    "duration": "PT0S",
    "registrationId": "58d74820e46f5af904162a00"
}
```

5.2.4 NGSIv2（削除）

NGSIv2 でデータの所在を削除する時の registrations/{registrationId}（リクエスト/レスポンス）の一例を以下に記載します。

[registrations リクエスト]

```
(curl -k -X DELETE "https://${IP}/orion/v2.0/registrations/5a82be3d093af1b94ac0f730" -i ¥
-s -S --header "Authorization: Bearer ${TOKEN}" --header "Accept: application/json")
```

レスポンスにはステータスコード 204 が返ります。

[registrations 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
```

5.3 データ問い合わせ応答（必須）

CC からデータ収集／蓄積レイヤに対して、CPr が持つデータへの参照リクエストが発行されると、データ収集／蓄積レイヤは CC からのデータ参照リクエストを CPr に転送します（自レイヤにデータ蓄積されていない場合のみ）。問い合わせを受けた CPr は、収集／蓄積レイヤに対してデータ問い合わせ応答を返却する必要があります。CPr からの応答はデータ収集／蓄積レイヤによって CC に転送されます。

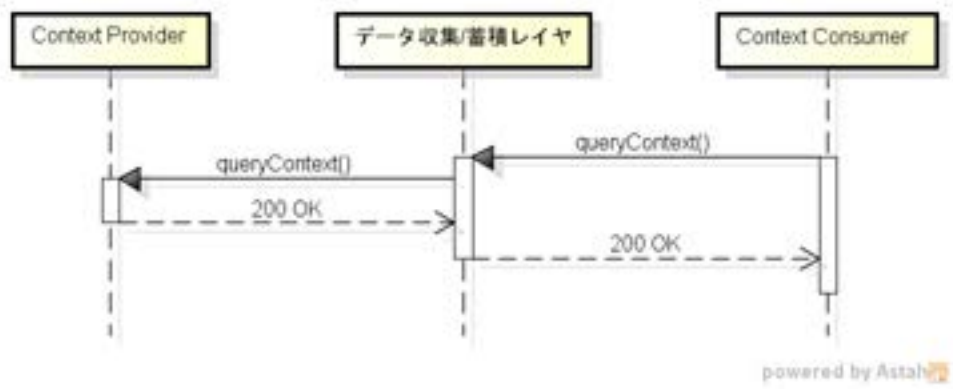

図 5-2 データ問い合わせ応答

NGSIv1 の場合 registerContext の providingApplication、NGSIv2 の場合 POST メソッドの registrations の url で「http://example.com:1026/v1」というベース URL が登録されている場合、データ収集／蓄積レイヤは「http://example.com:1026/v1/queryContext」にしてデータ問い合わせ（queryContext）を発行します。

下記は、CC から Room-1 の温度データがリクエストされた場合にデータ収集／蓄積レイヤから CPr に発行される例です。

```
[queryContext リクエスト]
 {
      "entities": [
           {
                "type": "Room",
                "isPattern": "false",
                "id": "Room-1"
           }
      ],
      "attributes": [
           "temperature"
      ]
 }
```

[queryContext レスポンス]

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": "20.5"
                    }
                ],
                "id": "Room-1",
                "isPattern": "false",
                "type": "Room"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

5.4 データ更新通知予約応答（任意）

データ更新通知予約は CC が利用する機能であり、データの更新（新規登録も含む）をトリガとして即座にデータを参照したい（通知を受けたい）場合に実行される処理です。

CPr はデータ収集/蓄積レイヤに対して能動的にデータ登録/更新を行わないアクタであるため、CPr からイベントドリブンでデータを取得した場合は CC 側から上記予約を実行します。

CC に対してデータの更新通知（新規登録も含む）を許可する場合、CPr はデータ更新通知予約を待ち受け、応答を返す必要があります。

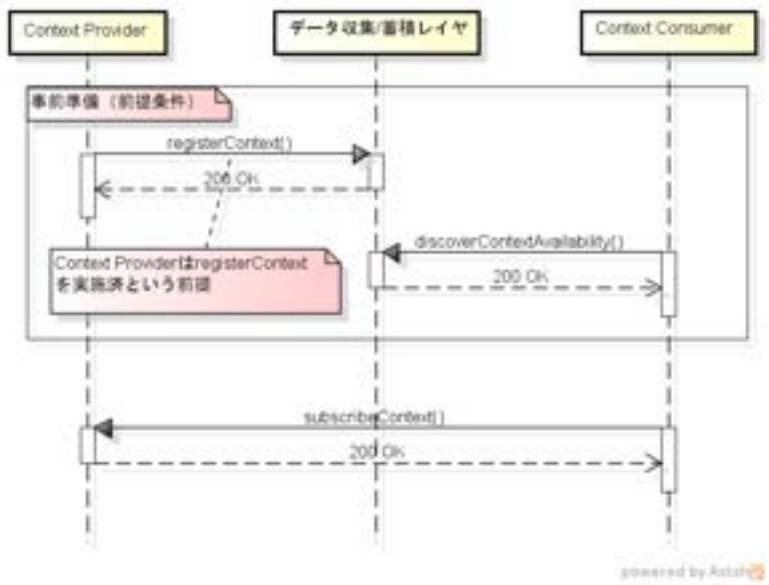

図 5-3 データ更新通知予約応答

5.4.1 NGSIv1

NGSIv1 でのデータ更新通知予約には、subscribeContext を利用します。subscribeContext（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 Context Element に含まれる温度データ（Attribute）に対するデータ更新通知予約を表現しています。

更新通知対象の Context Element は entities パラメータで、通知条件は notifyConditions で指定されます。notifyConditions で指定された条件が満たされた場合、CPr は reference で指定された URL（リソース）に対してデータ更新通知を行います。データ更新通知については、「5.7 データ更新通知（任意）」を参照してください。

また、データ更新通知予約には有効期限があり、duration パラメータで指定した値を元に下記計算式で算出されます。その他パラメータの詳細は、第 8 章 を参照してください。

有効期限＝現在時刻＋duration

```
[subscribeContext リクエスト]
 (curl -k -X POST "https://${IP}/orion/v1.0/subscribeContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "entities": [
            {
                  "type": "Room",
                  "isPattern": "false",
                  "id": "Room-1"
            }
      ],
      "attributes": [
            "temperature"
      ],
      "reference": "http://example.com:1026/v1/notify",
      "duration": "P1M",
      "notifyConditions": [
            {
                  "type": "ONCHANGE",
                  "condValues": [
                        "temperature"
                  ]
            }
      ],
      "throttling": "PT5S"
}
EOF
```

CPr は上記リクエストを受けるとデータ更新通知予約の識別子として subscriptionId を発行し、リクエストに含まれる duration などのパラメータとともに応答を返します。subscriptionId はデータ更新通知予約の更新/削除にて利用されるパラメータであり、24 桁の 16 進数（を文字列化したもの）です。

CPr 内部では subscriptionId と subscribeContext のリクエスト内容を紐づけて保管する必要があります。

[subscribeContext レスポンス]

```
{
    "subscribeResponse": {
        "duration": "P1M",
        "subscriptionId": "58d76733e46f5af904162a01",
        "throttling": "PT5S"
    }
}
```

5.4.2 NGSIv2

NGSIv2 でのデータ更新通知予約には、POST メソッドの subscriptions を利用します。subscriptions（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 Context Element に含まれる温度データ（Attribute）に対するデータ更新通知予約を表現しています。

更新通知対象の Context Element は entities パラメータで、通知条件は condition で指定されます。condition で指定された条件が満たされた場合、CPr は url で指定された URL（リソース）に対してデータ更新通知を行います。url で指定された URL（リソース）が、NGSIv2 の通知に対応していない場合、attrsFormat に legacy をセットすることで、NGSIv1 形式の更新通知を行うことができます。condition で指定する条件式については、付録 A 参考情報 [9] を参照してください。データ更新通知については、「5.7 データ更新通知（任意）」を参照してください。

また、データ更新通知予約には有効期限があり、expires パラメータで指定した日時となります。その他パラメータの詳細は、第 8 章 を参照してください。

```
[subscriptions リクエスト]
 (curl -k -X POST "https://${IP}/orion/v2.0/subscriptions" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" ¥
  -d @- ) <<EOF
{
   "description": "One subscription to rule them all",
   "subject": {
      "entities": [
         {
            "id": "Room-1",
            "type": "Room"
         }
      ],
      "condition": {
         "attrs": [
            "temperature"
         ],
         "expression": {
            "q": "temperature>40"
         }
      }
   },
   "notification": {
      "http": {
         "url": "http://localhost:1234"
      },
      "attrs": [
         "temperature"
      ]
   },
   "expires": "2016-04-05T14:00:00.00Z",
   "throttling": 5
}
EOF
```

CPr は上記リクエストを受けるとデータ更新通知予約の識別子として subscriptionId を発行し、レスポンスヘッダの Location に subscriptionId を含む URL を格納してステータスコード 201 を返します。subscriptionId はデータ更新通知予約の更新/削除にて利用されるパラメータであり、24 桁の 16 進数（を文字列化したもの）です。

CPr 内部では subscriptionId と subscriptions のリクエスト内容を紐づけて保管する必要があります。

```
[subscriptions レスポンス]
  < HTTP/1.1 201 Created
  < Connection: keep-alive
  < Location: /v2/subscriptions/5a82be3d093af1b94ac0f730
  < Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
  < Date: Tue, 13 Feb 2018 10:30:21 GMT
```

5.5 データ更新通知予約の更新応答（任意）

データ更新通知予約の更新は、データ更新通知予約の期限延長や更新通知条件の変更時に利用される機能であり、CPr はリクエストの内容に応じて内部で保管しているデータ更新通知予約を更新する必要があります。

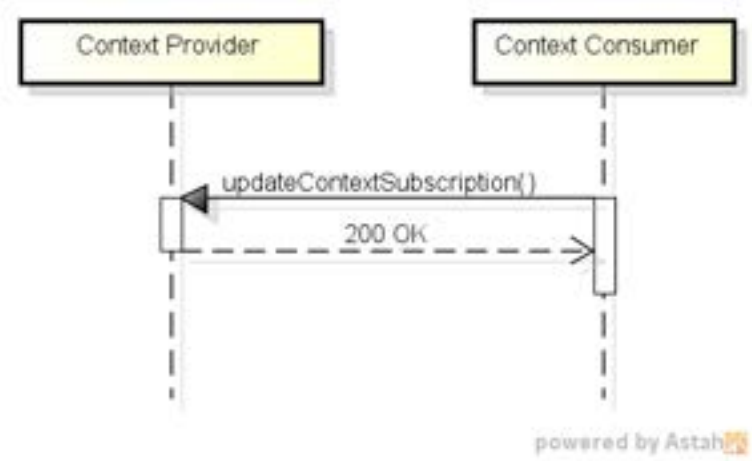

図 5-4 データ更新通知予約の更新応答

5.5.1 NGSIv1

NGSIv1 のデータ更新通知予約の更新には NGSI-10 の updateContextSubscription が利用されます。updateContextSubscription（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、有効期限の更新を想定しています。更新時の有効期限算出方法はデータの所在登録と同様です。

```
[updateContextSubscription リクエスト]
 (curl -k -X POST "https://${IP}/orion/v1.0/updateContextSubscription" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
 {
       "subscriptionId": "58d76733e46f5af904162a01",
       "duration": "P3M"
 }
 EOF
```

CPr は上記リクエストを受けると subscriptionId を元にデータ更新通知予約の内容を更新し、リクエストに含まれる duration などのパラメータとともに応答を返します。

```
[updateContextSubscription レスポンス]
 {
       "subscribeResponse": {
             "duration": "P3M",
             "subscriptionId": "58d76733e46f5af904162a01"
       }
 }
```

5.5.2 NGSIv2

NGSIv2 のデータ更新通知予約には PATCH メソッドの subscriptions/{subscriptionId}が利用されます。PATCH メソッドの subscriptions/{subscriptionId}（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、有効期限の更新を想定しています。更新時の有効期限算はデータの所在登録と同様日時指定です。

[subscriptions リクエスト]

```
(curl -k -X PATCH "https://${IP}/orion/v2/subscriptions/5a82be3d093af1b94ac0f730" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @-) <<EOF
{
  "expires": "2016-04-05T14:00:00.00Z"
}
EOF
```

CPr は上記リクエストを受けると subscriptionId を元にデータ更新通知予約の内容を更新し、応答を返します。

[subscriptions 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
```

5.6 データ更新通知予約の削除応答（任意）

データ更新通知予約の削除は、データ更新通知が不要になった場合に利用される機能であり、CPr は内部で保管しているデータ更新通知予約を削除する必要があります。

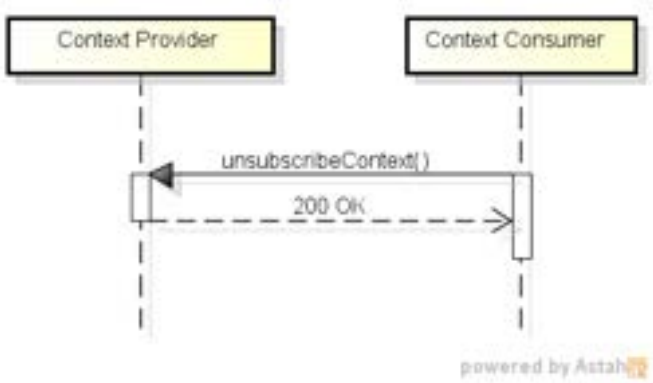

図 5-5 データ更新通知予約の削除応答

5.6.1 NGSIv1

NGSIv1 でのデータ更新通知予約の削除には NGSI-10 の unsubscribeContext が利用されます。unsubscribeContext（リクエスト/レスポンス）の一例を以下に記載します。

```
[unsubscribeContext リクエスト]
 (curl -k -X POST "https://${IP}/orion/v1.0/unsubscribeContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "subscriptionId": "58d76733e46f5af904162a01"
}
EOF
```

CPr は上記リクエストを受けると subscriptionId を元にデータ更新通知予約の内容を削除し、削除結果（statusCode）などのパラメータとともに応答を返します。

[unsubscribeContext レスポンス]

```
{
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    },
    "subscriptionId": "58d76733e46f5af904162a01"
}
```

5.6.2 NGSIv2

NGSIv2 でのデータ更新通知予約の削除には DELETE メソッドの subscriptions/{subscriptionId} が利用されます。DELETE メソッドの subscriptions/{subscriptionId}（リクエスト/レスポンス）の一例を以下に記載します。

[subscriptions リクエスト]

```
(curl -k -X DELETE "https://${IP}/orion/v2.0/subscriptions/5a82be3d093af1b94ac0f730" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" --header "Accept: application/json")
```

CPr は上記リクエストを受けると subscriptionId を元にデータ更新通知予約の内容を削除し、ステータスコードを返します。

[subscriptions 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
```

5.7 データ更新通知（任意）

CPr が提供するデータに更新が発生し、かつ更新データがデータ更新通知予約における条件に該当する場合は、データ更新通知を行う必要があります。

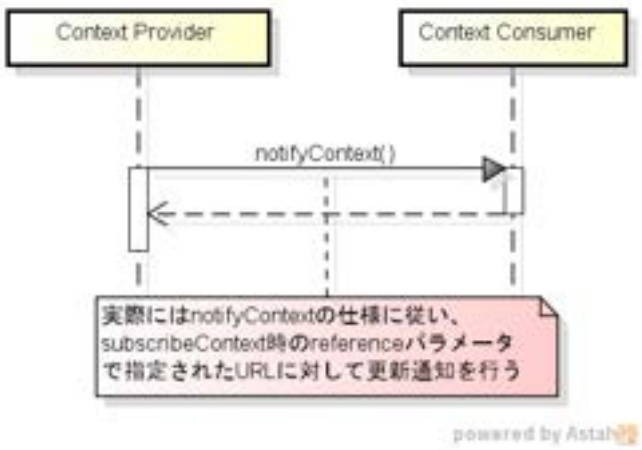

図 5-6 データ更新通知

5.7.1 NGSIv1

NGSIv1 のデータ更新通知は NGSI-10 の notifyContext の仕様に従います。たとえば、下記のような subscribeContext リクエストを事前に受けていた場合、通知対象のデータは type が Room で id が Room-1 の Context Element です。

通知条件は notifyConditions で指定されており、temperature という Attribute が変化（ONCHANGE）した場合です。さらに、throttling パラメータにより通知条件には制約がかかります。

下記の例では「notifyConditions の条件には合致しますが、過去 5 秒以内にデータ更新通知を行った場合は通知をスキップする」という制約が指定されています。

完全に条件が合致した場合、reference パラメータで指定された URL（リソース）に対して notifyContext の仕様に従いデータ更新通知を行います。下記例では、「http://example.com:1026/v1/notify」という URL が通知先を示します。

パラメータの詳細は、第 8 章 を参照してください。

[subscribeContext リクエスト]

```
{
    "entities": [
        {
            "type": "Room",
            "isPattern": "false",
            "id": "Room-1"
        }
    ],
    "attributes": [
        "temperature"
    ],
    "reference": "http://example.com:1026/v1/notify",
    "duration": "P1M",
    "notifyConditions": [
        {
            "type": "ONCHANGE",
            "condValues": [
                "temperature"
            ]
        }
    ],
    "throttling": "PT5S"
}
```

notifyContext（リクエスト/レスポンス）の一例を以下に記載します。

[notifyContext リクエスト]

```
{
   "subscriptionId" : "58d79959dc7d443c8e6ad54d",
   "originator" : "localhost",
   "contextResponses" : [
      {
         "contextElement" : {
            "type" : "Room",
            "isPattern" : "false",
            "id" : "Room-1",
            "attributes" : [
               {
                  "name" : "temperature",
                  "type" : "float",
                  "value" : "20.5"
               }
            ]
         },
         "statusCode" : {
            "code" : "200",
            "reasonPhrase" : "OK"
         }
      }
   ]
}
```

notifyContext リクエストを送信後、CPr は notifyContext レスポンス（データ更新通知応答）を待ち受ける必要があります。

[notifyContext レスポンス]

```
{
   "responseCode" : {
      "code" : "200",
      "reasonPhrase" : "OK"
   }
}
```

5.7.2 NGSIv2

NGSIv2 のデータ更新通知は/op/notify の仕様に従います。たとえば、下記のような subscriptions リクエストを事前に受けていた場合、通知対象のデータは type が Room で id が Room-1 の Context Element です。

通知条件はconditionで指定されており、temperatureというAttributeが40を超えた場合です。さらに、throttling パラメータにより通知条件には制約がかかります。

下記の例では「condition の条件には合致しますが、過去 5 秒以内にデータ更新通知を行った場合は通知をスキップする」という制約が指定されています。

完全に条件が合致した場合、url パラメータで指定された URL（リソース）に対して/op/notify の仕様に従いデータ更新通知を行います。下記例では、「http://example.com:1026/v2/notify」という URL が通知先を示します。

パラメータの詳細は、付録 A 参考情報[9]を参照してください。

```
[subscriptions リクエスト]
 (curl -k -X POST "https://${IP}/orion/v2.0/subscriptions" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" ¥
  -d @- ) <<EOF
 {
   "description": "One subscription to rule them all",
   "subject": {
     "entities": [
       {
         "id": "Room-1",
         "type": "Room"
       }
     ],
     "condition": {
       "attrs": [
         "temperature"
       ],
       "expression": {
         "q": "temperature>40"
       }
     }
   },
   "notification": {
     "http": {
       "url": "http://example.com:1026/v2/notify"
     },
     "attrs": [
       "temperature"
     ]
   },
   "expires": "2016-04-05T14:00:00.00Z",
   "throttling": 5
 }
 EOF
```

/op/notify（リクエスト/レスポンス）の一例を以下に記載します。

[/op/notify リクエスト]

```
{
  "subscriptionId": "5aeb0ee97d4ef10a12a0262f",
  "data": [
    {
      "type": "Room",
      "id": "Room-1",
      "temperature": {
        "value": 41,
        "type": "Number"
      }
    }
  ]
}
```

/op/notify リクエストを送信後、CPr は/op/notify レスポンス（データ更新通知応答）を待ち受ける必要があります。

[/op/notify レスポンス]

```
< HTTP/1.1 200 OK
< Connection: Keep-Alive
< Content-Length: 0
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

第6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. データ更新通知予約（任意）
3. データ更新通知予約の更新（任意）
4. データ更新通知予約の削除（任意）
5. データ更新通知応答（任意）
6. 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}

6.1 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）や CPr によるデータの所在登録（registerContext, registrations）などが行われていなければ参照できません。

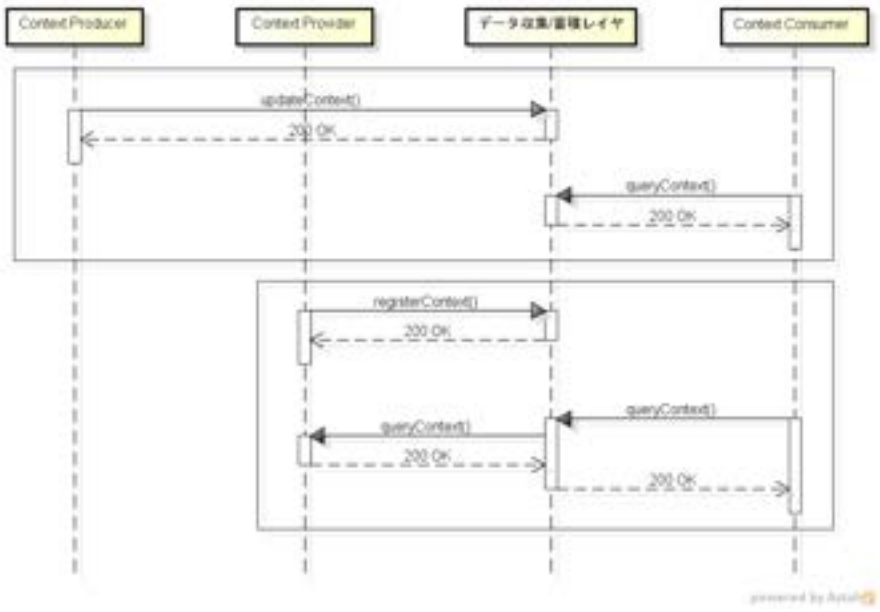

図 6-1 データ参照

6.1.1 NGSIv1

NGSIv1 でのデータ参照は NGSI-10 の queryContext を利用します。queryContext（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。isPattern 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 8 章 を参照してください。

```
[queryContext リクエスト]
 (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": "Room",
                "isPattern": "true",
                "id": ".*"
           }
      ],
      "attributes": [
           "temperature"
      ]
 }
 EOF
```

quertContext 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。

```
[queryContext レスポンス]
 {
      "contextResponses": [
           {
                "contextElement": {
                     "attributes": [
                          {
                               "name": "temperature",
                               "type": "float",
                               "value": "20.5"
                          }
                     ],
                     "id": "Room-1",
                     "isPattern": "false",
                     "type": "Room"
                },
                "statusCode": {
                     "code": "200",
                     "reasonPhrase": "OK"
                }
           }
      ]
 }
```

6.1.2 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 8 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": {}
     }
   }
 ]
```

6.2 データ更新通知予約（任意）

データ更新通知予約は、データ収集/蓄積レイヤに蓄積されたデータの更新（新規登録も含む）をトリガとして即座にデータを参照したい（通知を受けたい）場合に実行する処理です。データ更新通知予約はあくまで「予約」であるため、実際に通知が発生するのは予約後にデータ更新が発生したタイミングであることに注意してください。さらに、CC 側で参照タイミングを制御できないため、CC 側に NGSI-10 サーバー機能または NGSIv2 サーバー機能を実装して通知を待ち受ける必要があります。

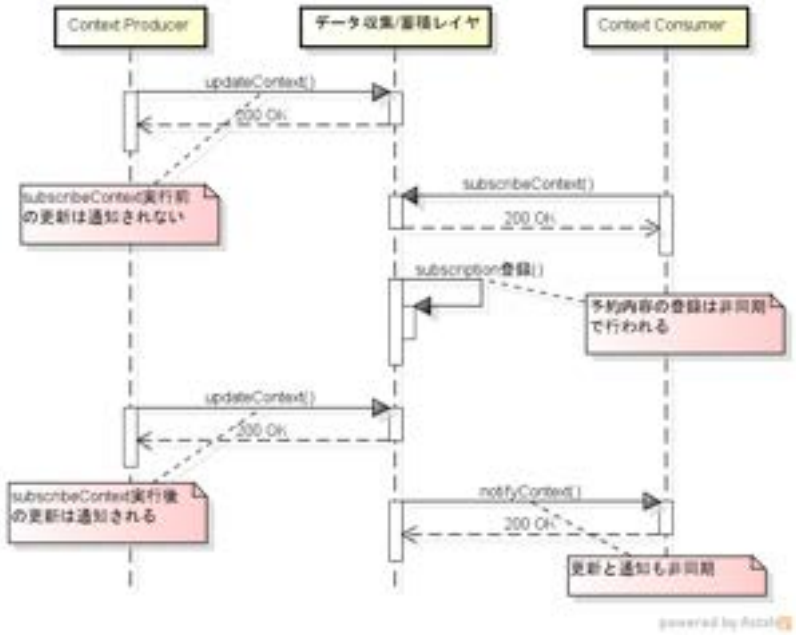

図 6-2 データ更新通知予約

6.2.1 NGSIv1

NGSIv1 でのデータ更新通知予約は NGSI-10 の subscribeContext を利用します。subscribeContext（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 Context Element に含まれる温度データ（Attribute）に更新が発生した場合に通知を受けるための予約を想定しています。

通知条件は notifyConditions パラメータにて指定し、通知先（CC が待ち受ける URL）は reference にて指定します。下記例において、データ収集/蓄積レイヤでは温度データが更新（新規登録含む）された場合に「http://example.com:1026/v1/notify」という URL に対してデータ更新通知を行います。

データ更新通知の詳細は、「5.7 データ更新通知（任意）」および「6.5 データ更新通知応答（任意）」を参照してください。

また、データ更新通知予約には有効期限があり、duration パラメータで指定した値を元に下記

計算式で算出されます。その他パラメータの詳細は、第 8 章 を参照してください。

有効期限＝現在時刻＋duration

```
[subscribeContext リクエスト]
 (curl -k -X POST "https://${IP}/orion/v1.0/subscribeContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "entities": [
            {
                  "type": "Room",
                  "isPattern": "false",
                  "id": "Room-1"
            }
      ],
      "attributes": [
            "temperature"
      ],
      "reference": "http://example.com:1026/v1/notify",
      "duration": "P1M",
      "notifyConditions": [
            {
                  "type": "ONCHANGE",
                  "condValues": [
                        "temperature"
                  ]
            }
      ],
      "throttling": "PT5S"
}
EOF
```

subscribeContext のレスポンスには、データ更新通知予約の識別子（subscriptionId）や有効期限（duration）などが含まれます。subscriptionId は 24 桁の 16 進数（を文字列化したもの）で、データ更新通知予約の更新/削除で必要になるため CC 内で保持する必要があります。

[subscribeContext レスポンス]

```
{
    "subscribeResponse": {
        "duration": "P1M",
        "subscriptionId": "58d76733e46f5af904162a01",
        "throttling": "PT5S"
    }
}
```

6.2.2 NGSIv2

NGSIv2 でのデータ更新通知予約は POST メソッドの subscriptions を利用します。subscriptions（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 Context Element に含まれる温度データ（Attribute）が 40 を超える値に更新された場合に通知を受けるための予約を想定しています。

通知条件は condition パラメータにて指定し、通知先（CC が待ち受ける URL）は url にて指定します。下記例において、データ収集/蓄積レイヤでは温度データが更新（新規登録含む）された場合に「http://example.com:1026/v2/notify」という URL に対してデータ更新通知を行います。

データ更新通知の詳細は、「5.7 データ更新通知（任意）」および「6.5 データ更新通知応答（任意）」を参照してください。

また、データ更新通知予約には有効期限があり、expires パラメータで指定した日時となります。その他パラメータの詳細は、第 8 章 を参照してください。

[subscriptions リクエスト]

```
(curl -k -X POST "https://${IP}/orion/v2.0/subscriptions" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" ¥
  -d @- ) <<EOF
{
    "description": "One subscription to rule them all",
    "subject": {
        "entities": [
            {
                "id": "Room-1",
                "type": "Room"
            }
        ],
        "condition": {
            "attrs": [
                "temperature"
            ],
            "expression": {
                "q": "temperature>40"
            }
        }
    },
    "notification": {
        "http": {
            "url": "http://localhost:1234"
        },
        "attrs": [
            "temperature"
        ]
    },
    "expires": "2016-04-05T14:00:00.00Z",
    "throttling": 5
}
EOF
```

subscriptions のレスポンスヘッダ Location には、データ更新通知予約の識別子（subscriptionId）をセットした URL が含まれます。subscriptionId は 24 桁の 16 進数（を文字列化したもの）で、データ更新通知予約の更新/削除で必要になるため CC 内で保持する必要があります。

```
[subscriptions レスポンス]
  < HTTP/1.1 201 Created
  < Connection: keep-alive
  < Location: /v2/subscriptions/5a82be3d093af1b94ac0f730
  < Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
  < Date: Tue, 13 Feb 2018 10:30:21 GMT
```

6.3 データ更新通知予約の更新（任意）

データ更新通知予約には有効期限があるため、CC は必要に応じて期限の更新を行う必要があります。また、予約内容（通知条件など）を変更する場合にも本処理を実行します。

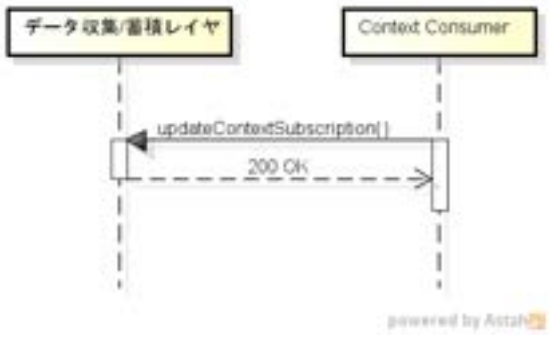

図 6-3 データ更新通知予約の更新

6.3.1 NGSIv1

NGSIv1でのデータ更新通知予約の更新には NGSI-10 の updateContextSubscription を利用します。更新対象を特定するため subscriptionId が必要です。updateContextSubscription（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、有効期限の更新を想定しています。更新時の有効期限算出方法はデータの所在登録と同様です。また、本処理で更新可能なパラメータは下記 4 つです。

1. notifyConditions
2. throttling
3. duration
4. restriction

データ更新通知予約の更新は、データ更新通知を一時的に無効化したい場合にも使える処理です。duration を 0 にして更新することで一時的に無効化し、再度通知を受けたい場合は duration に任意の期間を指定して更新すればよいです。

```
[updateContextSubscription リクエスト]
 (curl -k -X POST "https://${IP}/orion/v1.0/updateContextSubscription" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
 {
       "subscriptionId": "58d76733e46f5af904162a01",
       "duration": "P3M"
 }
 EOF
```

```
[updateContextSubscription レスポンス]
 {
       "subscribeResponse": {
            "duration": "P3M",
            "subscriptionId": "58d76733e46f5af904162a01"
       }
 }
```

6.3.2 NGSIv2

NGSIv2 でのデータ更新通知予約の更新には PATCH メソッドの subscriptions を利用します。更新対象を特定するため subscriptionId が必要です。PATCH メソッドの subscriptions（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、有効期限の更新を想定しています。更新時の有効期限指定方法はデータの所在登録と同様です。また、本処理で更新可能なパラメータは下記 6 つです。

1. description
2. subject
3. condition
4. notification
5. expires
6. throttling

データ更新通知予約の更新は、データ更新通知を一時的に無効化したい場合にも使える処理です。expiresを現在の日時以前にして更新することで一時的に無効化し、再度通知を受けたい場合は expires に任意の日時を指定して更新すればよいです。

```
[subscriptions リクエスト]
 (curl -k -X PATCH "https://${IP}/orion/v2.0/subscriptions/5a82be3d093af1b94ac0f730" -s -S -i ￥
  --header "Authorization: Bearer ${TOKEN}" ￥
  --header "Content-Type: application/json" ￥
  -d @- | python -mjson.tool) <<EOF
{
   "expires": "2016-04-05T14:00:00.00Z"
}
EOF
```

```
[subscriptions 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
```

6.4 データ更新通知予約の削除（任意）

データ更新通知が不要になった場合はデータ更新通知予約の削除を実行します。

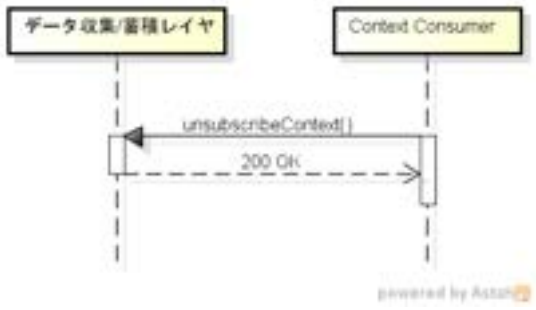

図 6-4 データ更新通知予約の削除

6.4.1 NGSIv1

NGSIv1 でのデータ更新通知予約の削除には NGSI-10 の unsubscribeContext を利用します。更新対象を特定するため subscriptionId が必要です。unsubscribeContext（リクエスト/レスポンス）の一例を以下に記載します。

```
[unsubscribeContext リクエスト]
 (curl -k -X POST "https://${IP}/orion/v1.0/unsubscribeContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "subscriptionId": "58d76733e46f5af904162a01"
}
EOF
```

unsubscribeContext のレスポンスには削除結果（statusCode）が含まれます。

削除に成功した場合は、CC 内に保持していた subscriptionId を破棄して問題ありません。

```
[unsubscribeContext レスポンス]
 {
      "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
      },
      "subscriptionId": "58d76733e46f5af904162a01"
}
```

6.4.2 NGSIv2

NGSIv2 でのデータ更新通知予約の削除には DELETE メソッドの subscriptions を利用します。更新対象を特定するため subscriptionId が必要です。subscriptions（リクエスト/レスポンス）の一例を以下に記載します。

```
[subscriptions リクエスト]
 (curl -k -X DELETE "https://${IP}/orion/v2.0/subscriptions/5a82be3d093af1b94ac0f730" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}")
```

subscriptions のレスポンスはステータスコード 204 が返ります。

削除に成功した場合は、CC 内に保持していた subscriptionId を破棄して問題ありません。

[subscriptions 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

6.5 データ更新通知応答（任意）

データ更新通知予約の実行後に CC 側でデータ更新通知を受信した場合、CC はデータ更新通知の送信元にデータ更新通知応答を返す必要があります。

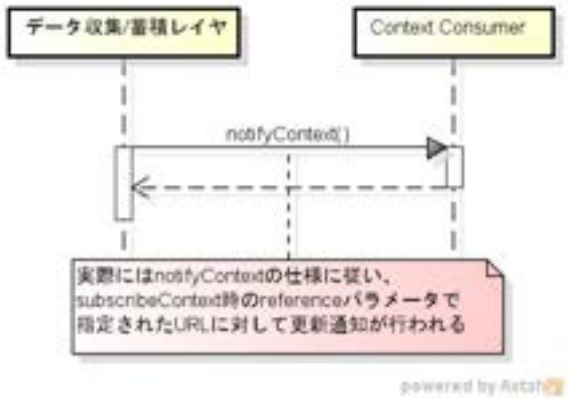

図 6-5 データ更新通知応答

6.5.1 NGSIv1

NGSIv1 のデータ更新通知およびデータ更新通知応答は NGSI-10 の notifyContext の仕様に従います。notifyContext（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、「6.2 データ更新通知予約（任意）」で行ったデータ更新通知予約に対するデータ更新通知を想定しています。

notifyContext のリクエストには、更新データ（contextElement）およびデータ登録/更新時の結果（statusCode）などが含まれています。

[notifyContext リクエスト]

```
{
   "subscriptionId" : "58d79959dc7d443c8e6ad54d",
   "originator" : "localhost",
   "contextResponses" : [
      {
         "contextElement" : {
            "type" : "Room",
            "isPattern" : "false",
            "id" : "Room-1",
            "attributes" : [
               {
                  "name" : "temperature",
                  "type" : "float",
                  "value" : "20.5"
               }
            ]
         },
         "statusCode" : {
            "code" : "200",
            "reasonPhrase" : "OK"
         }
      }
   ]
}
```

notifyContext のレスポンスにはレスポンス結果（responseCode）を含める必要があります。

CC は notifyContext のリクエストに応じて responseCode を設定し応答します。

[notifyContext レスポンス]

```
{
   "responseCode" : {
      "code" : "200",
      "reasonPhrase" : "OK"
   }
}
```

6.5.2 NGSIv2

NGSIv2 のデータ更新通知およびデータ更新通知応答は/op/notify の仕様に従います。パラメータの詳細は、付録 A 参考情報[9]を参照してください。

/op/notify（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、「6.2 データ更新通知予約（任意）」で行ったデータ更新通知予約に対するデータ更新通知を想定しています。

/op/notify のリクエストには、更新データ(type, id, 属性値)が含まれています。

```
[/op/notify リクエスト]
 {
    "subscriptionId": "5aeb0ee97d4ef10a12a0262f",
    "data": [
       {
          "type": "Room",
          "id": "Room-1",
          "temperature": {
             "value": 41,
             "type": "Number"
          }
       }
    ]
 }
```

CC は/op/notify のリクエストに応じて HTTP ステータスコードを設定し応答します。

```
[/op/notify レスポンス]
  < HTTP/1.1 200 OK
  < Connection: Keep-Alive
  < Content-Length: 0
  < Date: Tue, 13 Feb 2018 10:30:21 GMT
```

6.6 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、「8.2.1 NGSI v1 」の「NGSI-10 拡張 API」を参照してください。

6.6.1 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"
            }
        }
    ]
}
```

6.7 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。

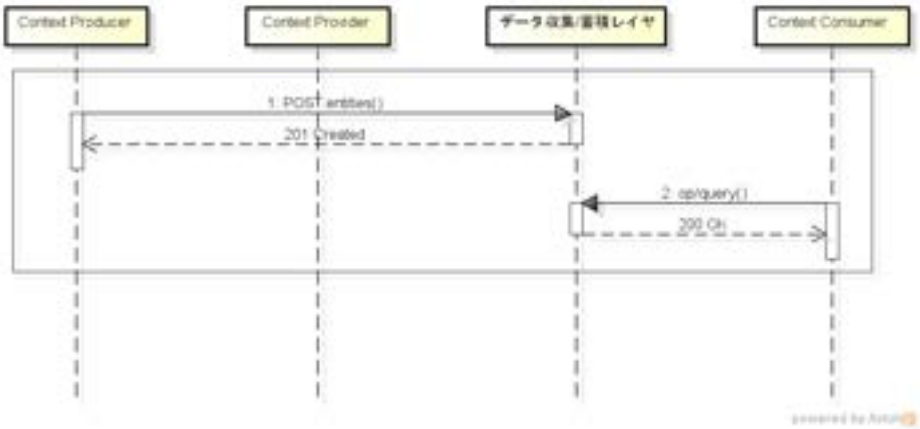

図 6-6 複数条件によるデータ参照

/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 8 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"
       }
    ],
    "attrs": [
       "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7章 NGSI 非対応デバイス開発者向けガイド

本章では NGSI 非対応デバイス開発者向けガイドとして、データ変換アダプタに対して行う処理を記載します。

データ変換アダプタを使って NGSI 非対応デバイスを接続する際は、事前準備としてデバイス情報の登録を行い、NGSI 非対応デバイスに認証情報の設定を行うことが必須です。また、NGSI 非対応デバイスの接続を終了する場合はデバイス情報の削除を実施してください。

1. NGSI 非対応デバイス情報登録（必須）
2. NGSI 非対応デバイスの認証設定（必須）
3. NGSI 非対応デバイス情報削除

NGSI 非対応デバイスをデータ変換アダプタに対して接続した場合、以下の 4 つの機能を扱えます。

1. NGSI 非対応デバイスからのデータ更新
2. NGSI 非対応デバイスからのデータ参照
3. NGSI 非対応デバイスへのデータ更新通知
4. NGSI 非対応デバイスのコマンド実行

7.1 NGSI 非対応デバイス情報登録（必須）

NGSI 非対応デバイスをデータ変換アダプタにする場合は接続するデバイスの情報をデータ変換アダプタに登録する必要があります。デバイス情報の登録には IDAS インタフェースを使用します。詳細は、第 8 章 を参照してください。

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}

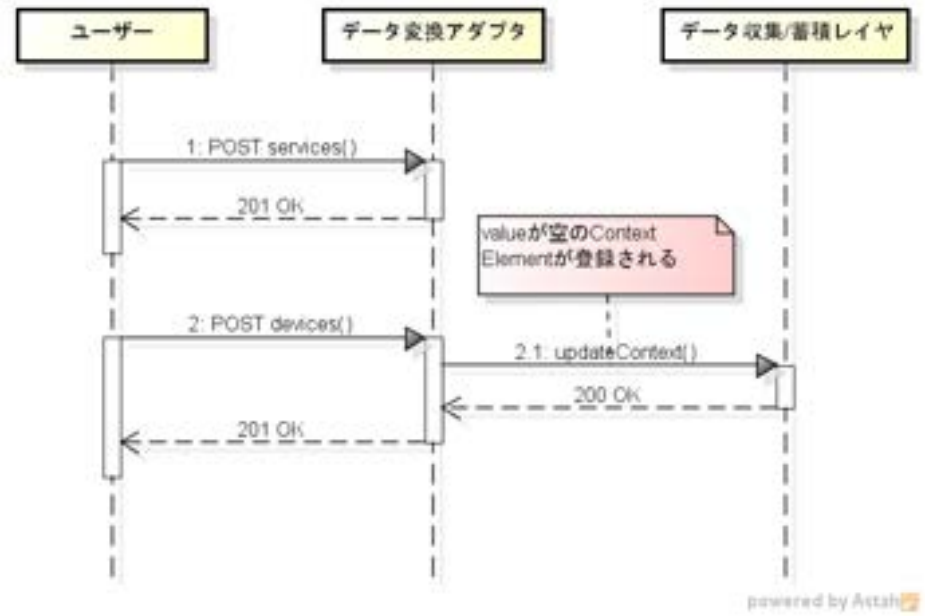

図 7-1 NGSI 非対応デバイス情報登録

まず、POST services でデバイス種別情報を登録し、次に POST devices でデバイス情報を登録します。デバイス登録には、予め以下の情報を決定しておく必要があります。

- デバイス種別情報

  API キー、エンティティタイプ

- デバイス情報

  デバイス ID、エンティティタイプ、エンティティ ID、属性リスト

デバイス種別情報登録とデバイス情報登録は 1 セットで考え、エンティティタイプは同じものを登録します。

7.1.1 デバイス種別情報登録

NGSI 非対応デバイスのデバイス種別情報登録は POST メソッドの services を利用します。services（リクエスト/レスポンス）の一例を以下に記載します。

```
[services リクエスト]
 (curl -k -X POST https://${IP}/iot/v1.0/services -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
  -d @- ) <<EOF
{
      "services": [
            {
                  "apikey": "1234567",
                  "entity_type": "TempSensor",
                  "resource": "/iot/json"
            }
      ]
}
EOF
```

```
[services レスポンス]
 < HTTP/1.1 201 Created
 < Connection: keep-alive
 < Content-Type: application/json; charset=utf-8
 < Fiware-Correlator: e31a7cbd-8273-4c34-b6ea-f8db1a0170df
 < Date: Tue, 13 Feb 2018 10:30:21 GMT
```

7.1.2 デバイス情報登録

NGSI 非対応デバイスのデバイス情報登録は POST メソッドの devices を利用します。devices（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、sensor1 という部屋の温度/湿度を提供する CP（センサ）を想定しています。この CP は MQTT プロトコルでデータを送信するものとします。

[devices リクエスト]

```
(curl -k -X POST https://${IP}/iot/v1.0/devices -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 -d @- ) <<EOF
{
    "devices": [
        {
            "device_id": "sensor1",
            "entity_type": "TempSensor",
            "entity_name": "TempSensor1",
            "endpoint": "http://deviceEndpoint:80",
            "transport": "MQTT",
            "attributes": [
                {
                    "name": "temperature",
                    "type": "float"
                },
                {
                    "name": "humidity",
                    "type": "float"
                }
            ]
        }
    ]
}
EOF
```

[devices レスポンス]

```
< HTTP/1.1 201 Created
< Connection: keep-alive
< Content-Type: application/json; charset=utf-8
< Fiware-Correlator: 12cffa8b-84b3-48cc-be48-89b3f31a35db
< Date: Tue, 13 Feb 2018 10:40:36 GMT
```

7.2 NGSI 非対応デバイス認証設定（必須）

データ利活用基盤サービス（FIWARE）では、NGSI 非対応デバイスを接続する際、データの暗号化（TLS）およびデバイスのパスワード認証が必要です。

暗号化のために、データ利活用基盤サービスで使用するサーバー証明書に対応するルート証

明書を使用して TLS 接続を行ってください。なお、ルート証明書の入手先はシステム運用者に確認してください。TLS は Version1.2 もしくは 1.3 を使用してください。

また、認証のために、システム運用者から入手した「デバイスアカウント」を使用してパスワード認証を行ってください。

7.3 NGSI 非対応デバイス情報削除（任意）

接続するデバイスの情報が不要になった場合はデバイス情報の削除を実行します。デバイス情報の削除には IDAS インタフェースを使用します。詳細は、第 8 章 を参照してください。

まず、DELETE services でデバイス種別情報を削除し、次に DELETE devices でデバイス情報を削除します。削除する際は、以下の NGSI 非対応デバイス情報登録時の情報が必要です。

- デバイス種別情報

  API キー

- デバイス情報

  デバイス ID

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}

7.3.1 デバイス種別情報削除

NGSI 非対応デバイスのデバイス種別情報削除は DELETE メソッドの services を利用します。services（リクエスト/レスポンス）の一例を以下に記載します。

```
[services リクエスト]
 (curl -k -X DELETE https://${IP}/iot/v1.0/services?resource=/iot/json¥&apikey=1234567 ¥
  -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
  --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy")
```

```
[services レスポンス]
  < HTTP/1.1 204 No Content
  < Connection: keep-alive
  < Fiware-Correlator: 18d3543e-4976-44b4-8973-cce80d2afcee
  < Date: Tue, 13 Feb 2018 10:45:52 GMT
```

7.3.2 デバイス情報削除

NGSI 非対応デバイスのデバイス情報削除は DELETE メソッドの devices を利用します。devices（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、sensor1 というデバイス ID の削除を想定しています。

```
[devices リクエスト]
  (curl -k -X DELETE https://${IP}/iot/v1.0/devices/sensor1 -s -S -i ¥
    --header "Authorization: Bearer ${TOKEN}" ¥
    --header "Accept: application/json" ¥
    --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy")
```

```
[devices レスポンス]
  < HTTP/1.1 204 No Content
  < Connection: keep-alive
  < Fiware-Correlator: 1acc88c6-0a77-4801-b697-697d2d4d6378
  < Date: Tue, 13 Feb 2018 10:55:13 GMT
```

7.4 NGSI 非対応デバイスからのデータ更新

NGSI非対応デバイスからデータ変換アダプタにデータを送信する場合はMQTTプロトコルを使用します。このデータは以下の条件に則っている必要があります。

① MQTT のトピック名は規定のフォーマットに従っていること

使用できる MQTT のトピック名は以下になります。

表 7-1 データ更新 MQTT トピック名

No	トピック名	内容

1	/{API キー}/{デバイス ID}/attrs/{属性名}	単一の属性にデータ送信 （NGSI 非対応デバイス→データ変換アダプタ）
2	/{API キー}/{デバイス ID}/attrs	複数の属性にデータ送信 （NGSI 非対応デバイス→データ変換アダプタ）

② JSON データはデータ変換アダプタが解釈可能な形式であること

具体的には下記の条件を満たしている必要があります。

- デバイス登録時に指定した属性のみが含まれること
- Single Level の JSON であること

一例を以下に記載します。

例えば NGSI 非対応デバイス登録時、temperature、humidity の 2 つの属性を登録したとします。

```
[NGSI 非対応デバイス登録時の属性]
 "attributes": [
      {
           "name": "temperature",
           "type": "float"
      },
      {
           "name": "humidity",
           "type": "float"
      }
 ]
```

この場合、NGSI 非対応デバイスは以下のように temperature、humidity のみの JSON データを送る必要があります。

以下の手順で、NGSI 非対応デバイスからデータ収集蓄積レイヤにデータを登録することができます。

1. 下記の JSON メッセージをトピックに対して MQTT プロトコルで publish する

単一の属性にデータ送信する場合

```
[publish 先トピック]
/1234567/sensor1/attrs/temperature

[JSON メッセージ]
 10.0
```

複数の属性にデータ送信する場合

```
[publish 先トピック]
/1234567/sensor1/attrs

[JSON メッセージ]
{
      "temperature": “10.0”,
      "humidity": "20.0"
 }
```

2. NGSI 非対応デバイス情報登録時に設定したエンティティタイプ、エンティティ ID の属性に対して、データが登録される

7.5 NGSI 非対応デバイスからのデータ参照

データ収集蓄積レイヤに蓄積されたデータを NGSI 非対応デバイスから取得する場合は MQTT プロトコルを使用します。このデータは以下の条件に則っている必要があります。

① MQTT のトピック名は規定のフォーマットに従っていること

使用できる MQTT のトピック名は以下になります。

表 7-2 データ参照 MQTT トピック名

No	トピック名	内容
1	/{API キー}/{デバイス ID}/configuration/values	/configuration/commands で指定した属性値を受信 （データ変換アダプタ→NGSI 非対応デバイス）
2	/{API キー}/{デバイス ID}/configuration/commands	/configuration/values で受信する属性名を送信 （NGSI 非対応デバイス→データ変換アダプタ）

② JSON データはデータ変換アダプタが解釈可能な形式であること

具体的には下記の条件を満たしている必要があります。

- デバイス登録時に指定した属性のみが含まれること
- Single Level の JSON であること

使用例を以下に示します。例えば、「7.1.2 デバイス情報登録」で登録した sensor1 の属性に対して、下記リクエストにより warningLevel を追加したものとします。

```
(curl -k -X POST "https://${IP}/orion/v1.0/updateContext" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}"
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "contextElements": [
        {
            "type": "TempSensor",
            "isPattern": "false",
            "id": "TempSensor1",
            "attributes": [
                {
                    "name": "warningLevel",
                    "type": "int",
                    "value": "40"
                }
            ]
        }
    ],
    "updateAction": "APPEND"
}
EOF
```

以下の手順で、上記で設定した warningLevel の値を NGSI 非対応デバイスから取得することができます。

1. 下記のトピックに対して MQTT プロトコルで subscribe する

[subscribe 先トピック] /1234567/sensor1/configuration/values

2. 下記の JSON メッセージをトピックに対して MQTT プロトコルで publish する

type には configuration を設定し、fields にはデータ収集/蓄積レイヤからデータを受け取るエンティティの属性名（配列）を指定します。

```
[publish 先トピック]
/1234567/sensor1/configuration/commands

[JSON メッセージ]
{
      "type": "configuration",
      "fields": ["warningLevel"]
}
```

3. 1.で subscribe したトピックに、2.で publish した属性名のデータが送信される

7.6 NGSI 非対応デバイスへのデータ更新通知

データ収集蓄積レイヤに蓄積されたデータの更新発生時に NGSI 非対応デバイスが通知を受ける場合は MQTT プロトコルを使用します。このデータは以下の条件に則っている必要があります。

① MQTT のトピック名は規定のフォーマットに従っていること

使用できる MQTT のトピック名は以下になります。

表 7-3 データ更新通知 MQTT トピック名

No	トピック名	内容
1	/{API キー}/{デバイス ID}/configuration/values	/configuration/commands で指定した属性値を受信 （データ変換アダプタ→NGSI 非対応デバイス）
2	/{API キー}/{デバイス ID}/configuration/commands	/configuration/values で受信する属性名を送信 （NGSI 非対応デバイス→データ変換アダプタ）

② JSON データはデータ変換アダプタが解釈可能な形式であること

具体的には下記の条件を満たしている必要があります。

- デバイス登録時に指定した属性のみが含まれること
- Single Level の JSON であること

使用例を以下に示します。例えば、「7.1.2 デバイス情報登録」で登録した sensor1 の属性に対して、warningLevel が追加されているものとします。

以下の手順で、warningLevel の値が更新された場合に NGSI 非対応デバイスが通知を受けることができます。

1. 下記のトピックに対して MQTT プロトコルで subscribe する

```
[subscribe 先トピック]
/1234567/sensor1/configuration/values
```

2. 下記の JSON メッセージをトピックに対して MQTT プロトコルで publish する
type には subscription を設定し、fields にはデータ収集/蓄積レイヤから更新通知を受け取るエンティティの属性名（配列）を指定します。

```
[publish 先トピック]
/1234567/sensor1/configuration/commands

[JSON メッセージ]
{
    "type": "subscription",
    "fields": ["warningLevel"]
}
```

3. 2.で publish した属性名の値が更新された場合、1.で subscribe したトピックにデータが送信される

7.7 NGSI 非対応デバイスのコマンド実行

NGSI 非対応デバイス側で時間がかかる処理を実行したい場合、データ収集/蓄積レイヤから「7.1.2 デバイス情報登録」の commands に設定した属性値を更新することで、NGSI 非対応デバイス側の処理を実行することができ、処理の実行状態および実行結果をデータ収集/蓄積レイヤに保持することが可能です。

NGSI 非対応デバイスのコマンドを実行する場合は MQTT プロトコルを使用します。このデータは以下の条件に則っている必要があります。

① MQTT のトピック名は規定のフォーマットに従っていること

使用できる MQTT のトピック名は以下になります。

表 7-4 コマンド実行 MQTT トピック名

No	トピック名	内容
1	/{API キー}/{デバイス ID}/cmd	デバイス情報登録で commands に登録した属性の値を受信 （データ変換アダプタ→NGSI 非対応デバイス）
2	/{API キー}/{デバイス ID}/cmdexe	NGSI 非対応デバイス側処理の実行結果を送信 （NGSI 非対応デバイス→データ変換アダプタ）

② JSON データはデータ変換アダプタが解釈可能な形式であること

具体的には下記の条件を満たしている必要があります。

- デバイス登録時に指定した属性のみが含まれること
- Single Level の JSON であること

コマンド実行を行う場合は「7.1.2 デバイス情報登録」時に commands の属性名を登録することが必要です。デバイス情報の登録には IDAS インタフェースを使用します。詳細は、第 8 章 を参照してください。

使用例を以下に示します。例えば、「7.1.2 デバイス情報登録」で下記リクエストにより commands の name に reset、type に Text を登録したものとします。

[devices リクエスト]

```
(curl -k -X POST https://${IP}/iot/v1.0/devices -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "devices": [
        {
            "device_id": "sensor1",
            "entity_type": "TempSensor",
            "entity_name": "TempSensor1",
            "endpoint": "http://deviceEndpoint:80",
            "transport": "MQTT",
            "attributes": [
                {
                    "name": "temperature",
                    "type": "float"
                }
            ],
            "commands": [
                {
                    "name": "reset",
                    "type": "Text"
                }
            ]
        }
    ]
}
EOF
```

コマンド実行機能を使う場合は必須

この時、データ収集/蓄積レイヤには下記のように commands で登録した属性の他、type が commandStatus の<属性名>_status、type が commandResult の<属性名>_info の属性が登録されます。

[commands 登録による属性（抜粋）]


```
{
    "name": "reset",
    "type": "Text",
    "value": ""
},
{
    "name": "reset_status",
    "type": "commandStatus",
    "value": "",
    "metadatas": []
},
{
    "name": "reset_info",
    "type": "commandResult",
    "value": "",
    "metadatas": []
}
```


以下の手順で、reset 属性値が更新された場合に NGSI 非対応デバイスがコマンド実行できます。

1. 下記のトピックに対して MQTT プロトコルで subscribe する

[subscribe 先トピック]
/1234567/sensor1/cmd

2. NGSI 非対応デバイス側の処理を開始させるため、データ収集/蓄積レイヤの reset 属性値のデータ更新を行う

[updateContext リクエスト]

```
(curl -k -X POST "https://${IP}/orion/v1.0/updateContext" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}"
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "contextElements": [
        {
            "type": "TempSensor",
            "isPattern": "false",
            "id": "TempSensor1",
            "attributes": [
                {
                    "name": "reset",
                    "type": "Text",
                    "value": "5"
                }
            ]
        }
    ],
    "updateAction": "UPDATE"
}
EOF
```

3. 1.で subscribe したトピックに更新された reset 属性のデータが送信される

この時、データ収集/蓄積レイヤには下記のように type が commandStatus の<属性名>_status の属性値が PENDING に更新されます。

[<属性名>_status の属性値（抜粋）]

```
{
    "name": "reset_status",
    "type": "commandStatus",
    "value": "PENDING",
    "metadatas": [
        {
            "name": "TimeInstant",
            "type": "ISO8601",
            "value": "2018-02-13T10:55:39.004Z"
        }
    ]
}
```

4. NGSI 非対応デバイス側は、受信した reset 属性のデータによって処理を実行する

5. NGSI 非対応デバイス側の処理が完了した場合、実行結果を下記のトピックに対して MQTT プロトコルで publish する

```
[publish 先トピック]
/1234567/sensor1/cmdexe

[JSON メッセージ]
{
      "reset": "SUCCESS"
}
```

この時、データ収集/蓄積レイヤには下記のように type が commandStatus の<属性名>_status の属性値が OK に更新され、type が commandResult の<属性名>_info の属性値が上記で publish した値に更新されます。

```
[<属性名>_status、<属性名>_info の属性値（抜粋）]
                        {
                              "name": "reset_status",
                              "type": "commandStatus",
                              "value": "OK",
                              "metadatas": [
                                    {
                                          "name": "TimeInstant",
                                          "type": "ISO8601",
                                          "value": "2018-02-13T10:56:24.929Z"
                                    }
                              ]
                        },
                        {
                              "name": "reset_info",
                              "type": "commandResult",
                              "value": "SUCCESS",
                              "metadatas": [
                                    {
                                          "name": "TimeInstant",
                                          "type": "ISO8601",
                                          "value": "2018-02-13T10:56:24.929Z"
                                    }
                              ]
                        }
```

第8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。

8.1 API 一覧

8.1.1 NGSI v1 インタフェース

NGSI-10 標準 API

No	API 名	HTTP	機能
1	updateContext	POST	コンテキスト情報を作成／更新する
2	queryContext	POST	コンテキスト情報を取得する
3	subscribeContext	POST	コンテキスト情報が変化した時に通知させる
4	unsubscribeContext	POST	コンテキスト情報の変化通知を解除する
5	updateContextSubscription	POST	コンテキスト情報の変化通知設定を変更する
6	notifyContext	POST	コンテキスト情報が変化したことを通知する

NGSI-10 拡張 API

No	API 名	HTTP	機能
1	contextEntities	POST	コンテキストの属性値を追加する
		GET	指定されたコンテキスト・エンティティに関するすべての情報を検索する
2	contextEntities/{エンティティ ID}	POST	コンテキストの属性値を追加する
		GET	指定されたコンテキスト・エンティティに関するすべての情報を検索する
		PUT	コンテキストの属性値を置換する
		DELETE	指定されたエンティティのすべての情報を削除する
3	contextEntities/{エンティティ ID}/attributes	POST	指定されたエンティティに関する属性情報を登録する
		GET	指定されたコンテキスト・エンティティに関する任意の属性情報を検索する
		PUT	コンテキストの属性値を置換する
		DELETE	指定されたエンティティのすべての情報を削除する
4	contextEntities/{エンティティ ID}/attributes/{属性名}	POST	コンテキストの属性値を追加する
		GET	属性値と関連するメタデータを検索する
		DELETE	指定されたすべての属性値を削除する
5	contextEntities/{エンティティ ID}/attributes/{属性名}/{属性 ID}	GET	指定された属性値を検索する
		PUT	属性値を置換する
		DELETE	属性値を削除する
6	contextEntityTypes/{タイプ名}	GET	指定されたタイプのすべてのコンテキスト・エンティティに関するすべての情報を検索する
7	contextEntityTypes/{タイプ名}/attributes	GET	指定されたタイプのすべてのコンテキスト・エンティティに関するすべての属性情報を検索する
8	contextEntityTypes/{タイプ名}/attributes/{属性名}	GET	指定されたタイプのコンテキスト・エンティティのすべての属性値を検索する
9	contextSubscriptions	POST	新しいサブスクリプションを生成する
10	contextSubscriptions/{サブスクリプション ID}	PUT	指定されたサブスクリプションを更新する
		DELETE	指定されたサブスクリプションをキャンセルする

NGSI-9 標準 API

No	API 名	HTTP	機能
1	registerContext	POST	コンテキストの所在を登録する
2	discoverContextAvailability	POST	コンテキストの所在を検索する
3	subscribeContextAvailability	POST	コンテキストの所在更新通知を予約する
4	updateContextAvailabilitySubscription	POST	コンテキストの所在更新通知を変更する
5	unsubscribeContextAvailability	POST	コンテキストの所在更新通知を解除する
6	notifyContextAvailability	POST	コンテキストの所在更新を通知する

NGSI-9 拡張 API

No	API 名	HTTP	機能
1	contextEntities/{エンティティ ID}	POST	指定されたエンティティに関する情報をプロバイダに登録する
		GET	指定されたコンテキスト・エンティティに関する任意の情報をプロバイダ上で検索する
2	contextEntities/{エンティティ ID} /attributes	POST	指定されたエンティティに関する属性情報をプロバイダに登録する
		GET	指定されたコンテキスト・エンティティに関する任意の属性情報をプロバイダ上で検索する
3	contextEntities/{エンティティ ID} /attributes/{属性名}	POST	指定された属性情報をプロバイダに登録する
		GET	指定された属性値の情報をプロバイダ上で検索する
4	contextEntityTypes/{タイプ名}	POST	指定されたタイプのコンテキスト・エンティティに関する情報をプロバイダに登録する
		GET	指定されたタイプのコンテキスト・エンティティに関する任意の情報をプロバイダ上で検索する
5	contextEntityTypes/{タイプ名} /attributes	POST	指定されたタイプのコンテキスト・エンティティに関する属性情報をプロバイダに登録する
		GET	指定されたタイプのコンテキスト・エンティティに関する任意の属性情報をプロバイダ上で検索する
6	contextEntityTypes/{タイプ名} /attributes/{属性名}	POST	指定されたタイプのコンテキスト・エンティティの属性情報をプロバイダに登録する
		GET	指定されたタイプのコンテキスト・エンティティの属性値の情報をプロバイダ上で検索する
7	contextAvailabilitySubscriptions	POST	新しいアベイラビリティ・サブスクリプションを生成する
8	contextAvailabilitySubscriptions /{サブスクリプション ID}	PUT	指定されたサブスクリプションを更新する
		DELETE	指定されたサブスクリプションをキャンセルする

8.1.2 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	subscriptions	GET	全てのサブスクリプションの情報を取得する
		POST	新しいサブスクリプションを生成する
9	subscriptions /{サブスクリプション ID}	GET	指定されたサブスクリプションの情報を取得する
		PATCH	指定されたサブスクリプションの情報を更新する
		DELETE	指定されたサブスクリプションをキャンセルする
10	registrations	GET	登録されているすべてのコンテキストの所在を取得する
		POST	コンテキストの所在を登録する
11	registrations /{レジストレーション ID}	GET	指定されたコンテキストの所在を検索する
		DELETE	指定されたコンテキストの所在を解除する
12	op/update	POST	バッチ更新オペレーションを実行する
13	op/query	POST	バッチ検索オペレーションを実行する

8.1.3 IDAS インタフェース

> MQTT プロトコルのトピック仕様については、第 7 章 および付録 A 参考情報 [10]を参照してください。

services

No	API 名	HTTP	機能
1	services	POST	デバイス種別情報を登録する
		GET	登録しているデバイス種別情報を取得する
		PUT	登録しているデバイス種別情報を更新する
		DELETE	登録しているデバイス種別情報を削除する

devices

No	API 名	HTTP	機能
1	devices	POST	デバイス情報を登録する
		GET	登録しているデバイス情報を全て取得する
2	devices/{デバイス ID}	GET	指定されたデバイス情報を取得する
		PUT	指定されたデバイス情報を更新する
		DELETE	指定されたデバイス情報を削除する

8.2 API 仕様

8.2.1 NGSI v1 インタフェース

NGSI-10 標準 API

1. updateContext

機能	コンテキスト情報を作成／更新する		
＜リクエスト＞			
HTTP メソッド	POST		
URL	https://${IP}/orion/v1.0/updateContext		
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)		
ボディ	タグ名／キー名		説明
	contextElements		コンテキストエレメントリスト
		type	エンティティタイプ
		isPattern	false
		id	エンティティ ID
		attributes	属性リスト。複数指定可
			name	属性名称
			type	属性タイプ。ロケーションによる検索を行う場合、”geo:point”を指定する。
			value	属性値。type に”geo:point”を指定した場合、緯度と経度を CSV 形式で指定する。 （例：”40.418889, -3.691944”）
			metadatas	メタデータリスト
				name	メタデータ名称
				type	メタデータタイプ
				value	メタデータ値
	updateAction		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）

＜レスポンス＞			
ボディ	タグ名／キー名		説明
	contextResponses		コンテキストレスポンス
		contextElement	コンテキストエレメントリスト
		attributes	属性リスト
		metadatas	作成／更新したメタデータリスト
		name	メタデータ名称
		type	メタデータタイプ
		value	メタデータ値
		name	作成／更新した属性名称
		type	作成／更新した属性タイプ
		value	""
		id	作成または更新の対象となったエンティティ ID
		isPattern	false
		type	作成または更新の対象となったエンティティのタイプ
		statusCode	ステータス情報
		code	HTTP ステータスコード
		details	ステータス詳細。必要に応じて返却される
		reasonPhrase	メッセージ

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/updateContext" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "contextElements": [
        {
            "type": "Room",
            "isPattern": "false",
            "id": "Room-1",
            "attributes": [
                {
                    "name": "temperature",
                    "type": "float",
                    "value": "20.5"
                },
                {
                    "name": "humidity",
                    "type": "integer",
                    "value": "50"
                }
            ]
```

```
        }
    ],
    "updateAction": "APPEND"
}
EOF
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": ""
                    },
                    {
                        "name": "humidity",
                        "type": "integer",
                        "value": ""
                    }
                ],
                "id": "Room-1",
                "isPattern": "false",
                "type": "Room"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

2. queryContext

機能	コンテキスト情報を取得する
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/queryContext
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)

ボディ	タグ名／キー名	説明
	entities	取得対象エンティティリスト
	type	エンティティタイプ
	isPattern	id に正規表現を使用するか否かを”true”か”false”で指定
	id	エンティティ ID。正規表現も可。
	attributes	属性名称リスト。複数指定可。省略時、すべての属性が返却される。
	restriction	フィルタリング条件
	scopes	スコープリストを以下のいずれかで指定
	エンティティタイプの有無	
	type	以下のいずれかで指定 "FIWARE::Filter::Existence" :エンティティタイプが存在する "FIWARE::Filter::Not::Existence" :エンティティタイプが存在しない
	value	"entity::type"
	属性値またはメタデータ値の文字列比較	
	type	比較対象により以下のいずれかを指定 "FIWARE::StringQuery" :属性値 "FIWARE::StringQuery::Metadata" :メタデータ値
	value	比較する属性名称またはメタデータ名称と値と式を指定。複数指定時は“;”(セミコロン)で連結させること。なお、属性値/メタデータともに文字列型でなければならないことに注意(数値型は不可)。 例:comment が”1st floor”かつ status が”normal”の場合の指定 “comment==1st floor;status==normal”
	ジオロケーション検索 :属性タイプ”geo:point”で登録されたエンティティが対象	
	type	"FIWARE::Location"
	value	領域を以下のいずれかで指定
	円形指定	
	circle	円情報
	centerLatitude	円の中心の緯度
	centerLongitude	円の中心の経度
	radius	半径
	inverted	円の外部領域を対象にする場合、true を指定。省略時、false となり円の内部領域が対象となる。
	ポリゴン指定	
	polygon	ポリゴン情報
	vertices	ポリゴンを構成する頂点のリスト
	latitude	緯度
	longitude	経度

					inverted	ポリゴンの外部領域を対象にする場合、true を指定。省略時、false となりポリゴンの内部領域が対象となる。
＜レスポンス＞						
ボディ	タグ名／キー名					説明
	contextResponses					コンテキストレスポンスリスト
		contextElement				コンテキストエレメント
			attributes			属性リスト
				name		属性名称
				type		属性タイプ
				value		属性値
				metadatas		メタデータリスト
					name	メタデータ名称
					type	メタデータタイプ
					value	メタデータ値
			id			エンティティタイプ
			isPattern			false
			type			エンティティ ID
		statusCode				ステータス情報
			code			HTTP ステータスコード
			details			ステータス詳細。必要に応じて返却される
			reasonPhrase			メッセージ

curl コマンド実行例：

```
(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": "Room",
            "isPattern": "true",
            "id": ".*"
        }
    ],
    "attributes": [
        "temperature"
    ]
}
EOF
```

＜レスポンス＞


```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": "20.5"
                    }
                ],
                "id": "Room-1",
                "isPattern": "false",
                "type": "Room"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

3. subscribeContext

機能	データ更新通知を予約する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/subscribeContext	
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)	
ボディ	タグ名／キー名	説明
	entities	データ更新通知対象エンティティリスト
	type	エンティティタイプ
	isPattern	id に正規表現を使用するか否かを”true”か”false”で指定
	id	エンティティ ID。正規表現も可。
	attributes	属性リスト。データ更新通知で通知される属性の名称。省略時、すべての属性情報が通知される。
	reference	データ更新通知を送信する通知先
	duration	有効期限を ISO8601 標準フォーマットの期間仕様で指定（省略時 24 時間）。有効期限が切れた後は、通知条件を満たす変更があっても、データ更新通知は発行されない。
	notifyConditions	通知条件
	type	"ONCHANGE"（固定）。condValues で指定された属性に変化があった場合に通知する。
	condValues	監視対象となる属性名称のリスト。省略時、すべての属性が対象となる。
	throttling	最小通知間隔を ISO8601 標準フォーマットの期間仕様で指定。省略可能。
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	subscribeResponse	サブスクライブレスポンス
	duration	設定された有効期限
	subscriptionId	データ更新通知予約の識別子。データ更新通知予約の更新/削除時に使用。
	throttling	設定された最小通知間隔。

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/subscribeContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
```

```
  -d @- | python -mjson.tool) <<EOF
{
      "entities": [
            {
                  "type": "Room",
                  "isPattern": "false",
                  "id": "Room-1"
            }
      ],
      "attributes": [
            "temperature"
      ],
      "reference": "http://example.com:1026/v1/notify",
      "duration": "P1M",
      "notifyConditions": [
            {
                  "type": "ONCHANGE",
                  "condValues": [
                        "temperature"
                  ]
            }
      ],
      "throttling": "PT5S"
}
EOF
```

＜レスポンス＞

```
{
      "subscribeResponse": {
            "duration": "P1M",
            "subscriptionId": "58d76733e46f5af904162a01",
            "throttling": "PT5S"
      }
}
```

4. unsubscribeContext

機能	データ更新通知予約を削除する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/unsubscribeContext	
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)	
ボディ	タグ名／キー名	説明
	subscriptionId	削除対象となるデータ更新通知予約の識別子
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	statusCode	ステータス情報
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ
	subscriptionId	指定したデータ更新通知予約識別子

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/unsubscribeContext" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
    "subscriptionId": "58d76733e46f5af904162a01"
}
EOF
```

＜レスポンス＞

```
{
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    },
    "subscriptionId": "58d76733e46f5af904162a01"
}
```

5. updateContextSubscription

機能	**データ更新通知予約を変更する。 subscribeContext で登録した内容の以下のフィールドを変更することが可能。 ・duration ・notifyConditions ・throttling**	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/updateContextSubscription	
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)	
ボディ	タグ名／キー名	説明
	subscriptionId	変更対象となるデータ更新通知予約の識別子
	duration	有効期限を ISO8601 標準フォーマットの期間仕様で指定
	notifyConditions	通知条件
	type	"ONCHANGE"（固定）。condValues で指定された属性に変化があった場合に通知する。
	condValues	監視対象となる属性名称のリスト。省略時、すべての属性が対象となる。
	throttling	最小通知間隔を ISO8601 標準フォーマットの期間仕様で指定
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	subscribeResponse	サブスクライブレスポンス
	duration	設定された有効期限
	subscriptionId	データ更新通知予約の識別子。データ更新通知予約の更新/削除時に使用。
	throttling	設定された最小通知間隔。

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/updateContextSubscription" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "subscriptionId": "58d76733e46f5af904162a01",
      "duration": "P3M"
}
```

```
EOF
```

＜レスポンス＞

```
{
    "subscribeResponse": {
        "duration": "P3M",
        "subscriptionId": "58d76733e46f5af904162a01"
    }
}
```

6. notifyContext

機能	データ更新通知を送信する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/notifyContext	
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)	
ボディ	タグ名／キー名	説明
	subscriptionId	通知の元となったデータ更新通知予約の識別子
	originator	localhost（固定）
	contextResponses	通知対象となったエンティティと属性 構成は(2)queryContext のレスポンスの contextResponses を参照
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	responseCode	ステータス情報
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/notifyContext" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
  "subscriptionId" : "58d79959dc7d443c8e6ad54d",
  "originator" : "localhost",
  "contextResponses" : [
    {
      "contextElement" : {
        "type" : "Room",
        "isPattern" : "false",
        "id" : "Room-1",
        "attributes" : [
          {
            "name" : "temperature",
            "type" : "float",
            "value" : "20.5"
          }
```

```
          ]
        },
        "statusCode" : {
          "code" : "200",
          "reasonPhrase" : "OK"
        }
      }
    ]
}
```

＜レスポンス＞

```
{
  "responseCode" : {
    "code" : "200",
    "reasonPhrase" : "OK"
  }
}
```

NGSI-10 拡張 API

1. contextEntities

機能	エンティティに対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。	
属性の追加		
エンティティの属性を追加する。		
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/contextEntities	
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)	
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。
	attributes	属性。複数指定可
	name	属性名称
	type	属性タイプ
	value	属性値
	metadatas	メタデータリスト
	name	メタデータ名称
	type	メタデータタイプ
	value	メタデータ値
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	contextResponses	コンテキストレスポンス
	attributes	属性
	name	追加された属性名称
	type	追加された属性のタイプ
	value	-
	metadatas	メタデータリスト
	name	メタデータ名称
	type	メタデータタイプ
	value	メタデータ値
	statusCode	ステータス情報
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ
	id	作成または属性追加の対象となったエンティティ ID

	isPattern	false
	type	エンティティタイプ

curl コマンド実行例:

```
(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": "Room1",
      "type": "Room",
      "attributes": [
            {
                  "name": "pressure",
                  "type": "integer",
                  "value": "720"
            }
      ]
}
EOF
```

＜レスポンス＞

```
{
      "type": "Room",
      "isPattern": "false",
      "id": "Room1",
      "contextResponses": [
        {
          "attributes": [
            {
              "name": "pressure",
              "type": "integer",
              "value": ""
            }
          ],
          "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
          }
        }
      ]
}
```

エンティティの照会
指定されたコンテキスト・エンティティに関するすべての情報を検索する。
＜リクエスト＞

HTTP メソッド	GET
URL	https://${IP}/orion/v1.0/contextEntities
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)
ボディ	なし
＜レスポンス＞	

ボディ	タグ名／キー名	説明
	contextResponses	コンテキストレスポンス
	contextElement	コンテキストエレメント
	attributes	指定したエンティティ ID が持つ属性リスト
	name	属性名称
	type	属性のタイプ
	value	属性値
	metadatas	メタデータリスト
	name	メタデータ名称
	type	メタデータタイプ
	value	メタデータ値
	id	エンティティ ID
	isPattern	false
	type	エンティティタイプ
	statusCode	ステータス情報
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンド実行例：

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

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "pressure",
                        "type": "integer",
                        "value": "720"
                    }
```

```
                ],
                "id": "Room1",
                "isPattern": "false",
                "type": "Room"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        },
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": "26.5"
                    }
                ],
                "id": "Room2",
                "isPattern": "false",
                "type": "Room"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```


2. contextEntities/{エンティティ ID}

機能	エンティティに対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。	
属性の追加		
エンティティの属性を追加する。		
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}	
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)	
ボディ	タグ名／キー名	説明
	attributes	属性。複数指定可

		name		属性名称
		type		属性タイプ
		value		属性値
		metadatas		メタデータリスト
			name	メタデータ名称
			type	メタデータタイプ
			value	メタデータ値
＜レスポンス＞				
ボディ	タグ名／キー名			説明
	contextResponses			コンテキストレスポンス
		attributes		属性
			name	追加された属性名称
			type	追加された属性のタイプ
			value	-
			metadatas	メタデータリスト
				name	メタデータ名称
				type	メタデータタイプ
				value	メタデータ値
		statusCode		ステータス情報
			code	HTTP ステータスコード
			details	ステータス詳細。必要に応じて返却される
			reasonPhrase	メッセージ
	id			作成または属性追加の対象となったエンティティ ID
	isPattern			false
	type			エンティティタイプ

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities/Room1" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
    "attributes": [
        {
            "name": "pressure",
            "type": "integer",
            "value": "720"
        }
    ]
}
EOF
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "attributes": [
                {
                    "name": "pressure",
                    "type": "integer",
                    "value": ""
                }
            ],
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ],
    "id": "Room1",
    "isPattern": "false",
    "type": ""
}
```


エンティティの照会			
指定されたコンテキスト・エンティティに関するすべての情報を検索する。			
＜リクエスト＞			
HTTP メソッド	GET		
URL	https://${IP}/orion/v1.0/contextEntities/{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)		
ボディ	なし		
＜レスポンス＞			
ボディ	タグ名／キー名		説明
	contextElement		コンテキストエレメント
		attributes	指定したエンティティ ID が持つ属性リスト
			name 属性名称
			type 属性のタイプ
			value 属性値
			metadatas メタデータリスト
			name メタデータ名称
			type メタデータタイプ
			value メタデータ値

		id	エンティティ ID
		isPattern	false
		type	エンティティタイプ
	statusCode		ステータス情報
		code	HTTP ステータスコード
		details	ステータス詳細。必要に応じて返却される
		reasonPhrase	メッセージ

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntities/Room1" -s -S ¥
--header "Authorization: Bearer ${TOKEN}" ¥
--header "Accept: application/json" ¥
 | python -mjson.tool)
```

＜レスポンス＞

```
{
    "contextElement": {
        "attributes": [
            {
                "name": "pressure",
                "type": "integer",
                "value": "720"
            }
        ],
        "id": "Room1",
        "isPattern": "false",
        "type": "Room"
    },
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    }
}
```

属性値の置換	
指定されたエンティティの持つ属性値を置換する。	
＜リクエスト＞	
HTTP メソッド	PUT
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}
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)

ボディ	タグ名／キー名	説明
	attributes	置換対象となる属性。複数指定可
	name	属性名称
	type	属性タイプ
	value	置換する属性値
	metadatas	メタデータリスト
	name	メタデータ名称
	type	メタデータタイプ
	value	メタデータ値
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	contextResponses	コンテキストレスポンス
	attributes	置換対象となった属性リスト
	name	属性名称
	type	属性のタイプ
	value	属性値
	metadatas	メタデータリスト
	name	メタデータ名称
	type	メタデータタイプ
	value	メタデータ値
	statusCode	ステータス情報
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンド実行例：

```
(curl -k -X PUT    "https://${IP}/orion/v1.0/contextEntities/Room1 " -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "attributes": [
            {
                  "name": "pressure",
                  "type": "integer",
                  "value": "763"
            }
      ]
}
EOF
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "attributes": [
                {
                    "name": "pressure",
                    "type": "integer",
                    "value": ""
                }
            ],
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

エンティティの削除		
指定されたエンティティのすべての情報を削除する。		
＜リクエスト＞		
HTTP メソッド	DELETE	
URL	https://${IP}/orion/v1.0/contextEntities/{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)	
ボディ	なし	
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンド実行例：

```
 (curl -k -X DELETE "https://${IP}/orion/v1.0/contextEntities/Room1" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
    "code": "200",
    "reasonPhrase": "OK"
```

```
}
```

3. contextEntities/{エンティティ ID}/attributes

機能	エンティティに対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。
属性の追加	
エンティティの属性を追加する。	
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}/attributes
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)
ボディ	(1) contextEntities/{エンティティ ID}を参照
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities/Room1/attributes" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
    "attributes": [
        {
            "name": "pressure",
            "type": "integer",
            "value": "720"
        },
        {
            "name": "temperature",
            "type": "float",
            "value": "26.5",
            "metadatas": [
                {
                    "name": "ID",
                    "type": "string",
                    "value": "ground"
                },
                {
                    "name": "average",
```

```
                    "type": "float",
                    "value": "22.4"
                }
            ]
        },
        {
            "name": "temperature",
            "type": "float",
            "value": "24.5",
            "metadatas": [
                {
                    "name": "ID",
                    "type": "string",
                    "value": "wall"
                },
                {
                    "name": "average",
                    "type": "float",
                    "value": "21.8"
                },
                {
                    "name": "accuracy",
                    "type": "float",
                    "value": "0.9"
                }
            ]
        }
    ]
}
EOF
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "attributes": [
                {
                    "name": "pressure",
                    "type": "integer",
                    "value": ""
                },
                {
                    "metadatas": [
                        {
                            "name": "ID",
                            "type": "string",
                            "value": "ground"
                        },
```

```
                        {
                            "name": "average",
                            "type": "float",
                            "value": "22.4"
                        }
                    ],
                    "name": "temperature",
                    "type": "float",
                    "value": ""
                },
                {
                    "metadatas": [
                        {
                            "name": "ID",
                            "type": "string",
                            "value": "wall"
                        },
                        {
                            "name": "average",
                            "type": "float",
                            "value": "21.8"
                        },
                        {
                            "name": "accuracy",
                            "type": "float",
                            "value": "0.9"
                        }
                    ],
                    "name": "temperature",
                    "type": "float",
                    "value": ""
                }
            ],
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ],
    "id": "Room1",
    "isPattern": "false",
    "type": ""
}
```


属性の照会
指定されたコンテキスト・エンティティの属性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/v1.0/contextEntities/{エンティティ ID}/attributes
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)
ボディ	なし
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntities/Room1/attributes" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
    | python -mjson.tool)
```

＜レスポンス＞

```
{
    "contextElement": {
        "attributes": [
            {
                "name": "pressure",
                "type": "integer",
                "value": "720"
            },
            {
                "metadatas": [
                    {
                        "name": "ID",
                        "type": "string",
                        "value": "ground"
                    },
                    {
                        "name": "average",
                        "type": "float",
                        "value": "22.4"
                    }
                ],
                "name": "temperature",
                "type": "float",
                "value": "26.5"
            },
            {
                "metadatas": [
                    {
```

```
                        "name": "ID",
                        "type": "string",
                        "value": "wall"
                    },
                    {
                        "name": "accuracy",
                        "type": "float",
                        "value": "0.9"
                    },
                    {
                        "name": "average",
                        "type": "float",
                        "value": "21.8"
                    }
                ],
                "name": "temperature",
                "type": "float",
                "value": "24.5"
            }
        ],
        "id": "Room1",
        "isPattern": "false",
        "type": ""
    },
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    }
}
```


属性値の置換	
指定されたエンティティの持つ属性値を置換する。	
＜リクエスト＞	
HTTP メソッド	PUT
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}/attributes
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)
ボディ	(1) contextEntities/{エンティティ ID}を参照
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例:

```
(curl -k -X PUT   "https://${IP}/orion/v1.0/contextEntities/Room1/attributes" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "attributes": [
        {
            "name": "pressure",
            "type": "integer",
            "value": "720"
        },
        {
            "name": "temperature",
            "type": "float",
            "value": "26.5",
            "metadatas": [
                {
                    "name": "ID",
                    "type": "string",
                    "value": "ground"
                },
                {
                    "name": "average",
                    "type": "float",
                    "value": "22.4"
                 },
                {
                    "name": "accuracy",
                    "type": "float",
                    "value": "0.9"
                }
            ]
        },
        {
            "name": "temperature",
            "type": "float",
            "value": "24.5",
            "metadatas": [
                {
                    "name": "ID",
                    "type": "string",
                    "value": "wall"
                },
                {
                    "name": "average",
                    "type": "float",
                    "value": "21.8"
```

```
                }
            ]
        }
    ]
}
EOF
```

＜レスポンス＞

```
{
    "contextResponses": [
        {
            "attributes": [
                {
                    "name": "pressure",
                    "type": "integer",
                    "value": ""
                },
                {
                    "metadatas": [
                        {
                            "name": "ID",
                            "type": "string",
                            "value": "ground"
                        },
                        {
                            "name": "average",
                            "type": "float",
                            "value": "22.4"
                        },
                        {
                            "name": "accuracy",
                            "type": "float",
                            "value": "0.9"
                        }
                    ],
                    "name": "temperature",
                    "type": "float",
                    "value": ""
                },
                {
                    "metadatas": [
                        {
                            "name": "ID",
                            "type": "string",
                            "value": "wall"
                        },
                        {
```

```
                        "name": "average",
                        "type": "float",
                        "value": "21.8"
                    }
                ],
                "name": "temperature",
                "type": "float",
                "value": ""
            }
        ],
        "statusCode": {
            "code": "200",
            "reasonPhrase": "OK"
        }
    }
]
}
```

エンティティの削除	
指定されたエンティティのすべての情報を削除する	
＜リクエスト＞	
HTTP メソッド	DELETE
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}/attributes
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)
ボディ	なし
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例：

```
(curl -k -X DELETE "https://${IP}/orion/v1.0/contextEntities/Room1/attributes" -s -S ¥
--header "Authorization: Bearer ${TOKEN}" ¥
--header "Accept: application/json" ¥
 | python -mjson.tool)
```

＜レスポンス＞

```
{
    "code": "200",
    "reasonPhrase": "OK"
}
```

4. contextEntities/{エンティティ ID}/attributes/{属性名}

機能	属性に対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。

属性の追加		
エンティティの属性を追加する。		
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}/attributes/{属性名}	
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)	
ボディ	タグ名／キー名	説明
	type	属性タイプ
	value	属性値
	metadatas	メタデータリスト
	name	メタデータ名称
	type	メタデータタイプ
	value	メタデータ値
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンド実行例

```
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities/Room1/attributes/temperature" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "type": "float",
    "value": "26.5",
    "metadatas": [
        {
            "name": "max",
            "type": "float",
            "value": "40.2"
        },
        {
            "name": "min",
            "type": "float",
            "value": "-5.0"
        }
    ]
```

```
}
EOF
```

＜レスポンス＞

```
{
    "code": "200",
    "reasonPhrase": "OK"
}
```

属性の照会			
指定されたコンテキスト・エンティティに関する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/v1.0/contextEntities/{エンティティ ID}/attributes/{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)		
ボディ	なし		
＜レスポンス＞			
ボディ	タグ名／キー名		説明
	attributes		指定したエンティティ ID が持つ属性リスト
		name	属性名称
		type	属性のタイプ
		value	属性値
		metadatas	メタデータリスト
			name　メタデータ名称
			type　メタデータタイプ
			value　メタデータ値
	statusCode		ステータス情報
		code	HTTP ステータスコード
		details	ステータス詳細。必要に応じて返却される
		reasonPhrase	メッセージ

curl コマンド実行例

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntities/Room1/attributes/temperature" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
   | python -mjson.tool)
```

＜レスポンス＞

```
{
    "attributes": [
```

```
        {
            "metadatas": [
                {
                    "name": "max",
                    "type": "float",
                    "value": "40.2"
                },
                {
                    "name": "min",
                    "type": "float",
                    "value": "-5.0"
                }
            ],
            "name": "temperature",
            "type": "float",
            "value": "26.5"
        }
    ],
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    }
}
```

属性の削除		
指定されたすべての属性値を削除する。		
＜リクエスト＞		
HTTP メソッド	DELETE	
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}/attributes/{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)	
ボディ	なし	
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンド実行例：

```
(curl -k -X DELETE "https://${IP}/orion/v1.0/contextEntities/Room1/attributes/temperature" -s -S ¥
--header "Authorization: Bearer ${TOKEN}" ¥
--header "Accept: application/json" ¥
```

```
  | python -mjson.tool)
```

＜レスポンス＞

```
{
    "code": "200",
    "reasonPhrase": "OK"
}
```

5. contextEntities/{エンティティ ID}/attributes/{属性名}/{属性 ID}

機能	属性 ID を持つ属性に対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。		
属性値の照会			
指定されたコンテキスト・エンティティに関する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/v1.0/contextEntities/{エンティティ ID}/attributes/{属性名}/{属性 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)		
ボディ	なし		
＜レスポンス＞			
ボディ	タグ名／キー名		説明
	attributes		指定したエンティティ ID が持つ属性リスト
		name	属性名称
		type	属性のタイプ
		value	属性値
		metadatas	メタデータリスト
			name　メタデータ名称
			type　メタデータタイプ
			value　メタデータ値
	statusCode		ステータス情報
		code	HTTP ステータスコード
		details	ステータス詳細。必要に応じて返却される
		reasonPhrase	メッセージ

curl コマンド実行例

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntities/Room1/attributes/temperature/ground" -s
-S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
    | python -mjson.tool)
```

＜レスポンス＞

```
{
    "attributes": [
        {
            "metadatas": [
                {
                    "name": "ID",
                    "type": "string",
                    "value": "ground"
                },
                {
                    "name": "average",
                    "type": "float",
                    "value": "22.4"
                }
            ],
            "name": "temperature",
            "type": "float",
            "value": "26.5"
        }
    ],
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    }
}
```

属性値の置換			
エンティティの属性値を置換する			
＜リクエスト＞			
HTTP メソッド	PUT		
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}/attributes/{属性名}/{属性 ID}		
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)		
ボディ	タグ名／キー名		説明
	type		属性タイプ
	value		属性値
	metadatas		メタデータリスト
		name	メタデータ名称
		type	メタデータタイプ
		value	メタデータ値
＜レスポンス＞			

ボディ	タグ名／キー名	説明
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンドの例は、以下のエンティティがすでに登録されているものとする。

```
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities/Room1" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "attributes": [
        {
            "name": "temperature", "type": "float", "value": "26.5",
            "metadatas": [
                { "name": "ID", "type": "string", "value": "ground" },
                { "name": "average",              "type": "float",   "value": "22.4"   }
            ]
        },
        {
            "name": "temperature", "type": "float", "value": "24.5",
            "metadatas": [
                { "name": "ID", "type": "string", "value": "wall" },
                { "name": "average",              "type": "float",   "value": "21.8" },
                {"name": "accuracy",              "type": "float",   "value": "0.9" }
            ]
        }
    ]
}
EOF
```

curl コマンド実行例

```
(curl -k -X PUT   "https://${IP}/orion/v1.0/contextEntities/Room1/attributes/temperature/wall" -s -S
¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
    "value": "35.0",
    "metadatas": [
        {
            "name": "average",
            "type": "float",
            "value": "23.2"
        }
    ]
}
```

```
EOF
```

＜レスポンス＞

```
{
    "code": "200",
    "reasonPhrase": "OK"
}
```

属性値の削除		
指定された属性値を削除する。		
＜リクエスト＞		
HTTP メソッド	DELETE	
URL	https://${IP}/orion/v1.0/contextEntities/{エンティティ ID}/attributes/{属性名}/{属性 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)	
ボディ	なし	
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンドの例は、以下のエンティティがすでに登録されているものとする。

```
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities/Room1" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
    "attributes": [
        {
            "name": "temperature", "type": "float", "value": "26.5",
            "metadatas": [
                { "name": "ID", "type": "string", "value": "ground" },
                { "name": "average", "type": "float", "value": "22.4"    }
            ]
        },
        {
            "name": "temperature", "type": "float", "value": "35.0",
            "metadatas": [
                { "name": "ID", "type": "string", "value": "wall" },
                { "name": "average", "type": "float", "value": "23.2" },
                {"name": "accuracy", "type": "float", "value": "0.9" }
            ]
```

```
        }
    ]
}
EOF
```

curl コマンド実行例：（上記の四角で囲まれた部分が削除対象）

```
 (curl -k -X DELETE
"https://${IP}/orion/v1.0/contextEntities/Room1/attributes/temperature/ground" ¥
-s -S --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
    "code": "200",
    "reasonPhrase": "OK"
}
```

6. contextEntityTypes /{タイプ名}

機能	指定されたエンティティタイプのコンテキスト・エンティティに関するすべての情報を検索する										
＜リクエスト＞											
HTTP メソッド	GET										
URL	https://${IP}/orion/v1.0/contextEntityTypes/{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)										
ボディ	なし										
＜レスポンス＞											
ボディ	タグ名／キー名										説明
	contextResponses										コンテキストレスポンスリスト
		contextElement									コンテキストエレメント
			attributes								属性リスト
				name							属性名称
				type							属性タイプ
				value							属性値
				metadatas							メタデータリスト
					name						メタデータ名称
					type						メタデータタイプ
					value						メタデータ値
			id								エンティティタイプ
			isPattern								false
			type								エンティティ ID

		statusCode		ステータス情報
			code	HTTP ステータスコード
			details	ステータス詳細。必要に応じて返却される
			reasonPhrase	メッセージ

curl コマンドの例は、以下のエンティティがすでに登録されているものとする。

```
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities/Shop1" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "type": "Favorite",
      "attributes": [
            {       "name": "pressure",                                     "type": "integer",
                    "value": "710"    },
            {       "name": "temperature",          "type": "float",   "value": "21.0"   }
      ]
}
EOF
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities/Shop2" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "type": "Favorite",
      "attributes": [
            {       "name": "temperature",          "type": "float",   "value": "20.5"   }
      ]
}
EOF
(curl -k -X POST "https://${IP}/orion/v1.0/contextEntities/Shop1" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
      "type": "Workplace",
      "attributes": [
            {       "name": "light",  "type": "integer",                  "value": "3"       },
            {       "name": "pressure",                "type": "integer",                  "value": "730"
                    }
      ]
}
EOF
```

curl コマンド実行例

```
(curl -k -X GET "https://${IP}/orion/v1.0/contextEntityTypes/Favorite" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
```

```
  | python -mjson.tool)
```

＜レスポンス＞


```
{
    "contextResponses": [
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "pressure",
                        "type": "integer",
                        "value": "710"
                    },
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": "21.0"
                    }
                ],
                "id": "Shop1",
                "isPattern": "false",
                "type": "Favorite"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        },
        {
            "contextElement": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "float",
                        "value": "20.5"
                    }
                ],
                "id": "Shop2",
                "isPattern": "false",
                "type": "Favorite"
            },
            "statusCode": {
                "code": "200",
                "reasonPhrase": "OK"
            }
        }
    ]
}
```

7. contextEntityTypes /{タイプ名}/attributes

機能	指定されたエンティティタイプのコンテキスト・エンティティに関するすべての情報を検索する
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/orion/v1.0/contextEntityTypes/{タイプ名}/attributes
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)
ボディ	なし
＜レスポンス＞	
ボディ	(6) contextEntityTypes /{タイプ名}を参照

8. contextEntityTypes /{タイプ名}/attributes/{属性名}

機能	指定されたエンティティタイプの指定された属性に関するすべての情報を検索する
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/orion/v1.0/contextEntityTypes/{タイプ名}/attributes/{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)
ボディ	なし
＜レスポンス＞	
ボディ	(6) contextEntityTypes /{タイプ名}を参照

9. contextSubscriptions

機能	データ更新通知を予約する
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/contextSubscriptions
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)

ボディ	NGSI-10 標準 API の(3) subscribeContext を参照
＜レスポンス＞	
ボディ	NGSI-10 標準 API の(3) subscribeContext を参照

10. contextSubscriptions/{サブスクリプション ID}

機能	**データ更新通知予約を変更または削除する**
データ更新通知予約を変更	
＜リクエスト＞	
HTTP メソッド	PUT
URL	https://${IP}/orion/v1.0/contextSubscriptions/{サブスクリプション ID}
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)
ボディ	NGSI-10 標準 API の(5)updateContextSubscription を参照
＜レスポンス＞	
ボディ	NGSI-10 標準 API の(5) updateContextSubscription を参照
データ更新通知予約を削除	
＜リクエスト＞	
HTTP メソッド	DELETE
URL	https://${IP}/orion/v1.0/contextSubscriptions/{サブスクリプション 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)
ボディ	なし
＜レスポンス＞	
ボディ	NGSI-10 標準 API の(4) unsubscribeContext を参照

NGSI-9 標準 API

1. registerContext

機能	コンテキストの所在を登録/変更する。	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/registry/registerContext	
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)	
ボディ	タグ名／キー名	説明
	contextRegistrations	コンテキストレジストレーションリスト
	entities	エンティティ情報リスト
	type	エンティティタイプ
	isPattern	false(固定)
	id	エンティティ ID
	attributes	属性リスト
	name	属性名称
	type	属性タイプ
	providingApplication	コンテキストプロバイダの URL
	duration	有効期限を ISO8601 標準フォーマットの期間仕様で指定（省略時 24 時間）。有効期限が切れた後は discoverContextAvailability での可用性確認でエラーコード 404（"No context element found"）が返却される
	registrationId	登録済みのコンテキスト情報を更新する際に、対象となる登録 ID を指定する。
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	duration	設定された有効期限
	registrationId	登録したコンテキスト情報に対する登録 ID
	errorCode	エラー情報
	code	エラーコード
	details	エラー詳細
	reasonPhrase	エラーメッセージ

curl コマンド実行例：

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

```
{
    "contextRegistrations": [
        {
          "entities": [
            {
              "type": "Room",
              "isPattern": "false",
              "id": "ConferenceRoom"
            },
            {
              "type": "Room",
              "isPattern": "false",
              "id": "OfficeRoom"
            }
          ],
          "attributes": [
            {
              "name": "temperature",
              "type": "degree"
            }
          ],
          "providingApplication": "http://mysensors.com/Rooms"
        }
    ],
    "duration": "P1M"
}
EOF
```

＜レスポンス＞

```
{
     "duration": "P1M",
     "registrationId": "58b92c48fcc6c2a47a01186a"
}
```

2. discoverContextAvailability

機能	コンテキストの所在を検索（有効かどうかを確認）する。	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/registry/discoverContextAvailability	
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)	
ボディ	タグ名／キー名	説明
	entities	取得対象エンティティリスト

		type	エンティティタイプ
		isPattern	id に正規表現を使用するか否かを”true”か”false”で指定
		id	エンティティ ID。正規表現も可。
	attributes		属性名称リスト。複数指定可。省略時、すべての属性が返却される。
＜レスポンス＞			
ボディ	タグ名／キー名		説明
	contextRegistrationResponses		コンテキストレジストレーションレスポンス
		contextRegistration	コンテキストレジストレーションリスト
			entities	エンティティ情報リスト
				type	エンティティタイプ
				isPattern	false(固定)
				id	エンティティ ID
			attributes	属性リスト
				name	属性名称
				type	属性タイプ
			providingApplication	コンテキストプロバイダの URL
		errorCode		エラー情報
			code	エラーコード
			details	エラー詳細
			reasonPhrase	エラーメッセージ

curl コマンド実行例：

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

EOF

＜レスポンス＞

```
{
    "contextRegistrationResponses": [
        {
            "contextRegistration": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": "degree"
                    }
                ],
                "entities": [
                    {
                        "id": "ConferenceRoom",
                        "isPattern": "false",
                        "type": "Room"
                    },
                    {
                        "id": "OfficeRoom",
                        "isPattern": "false",
                        "type": "Room"
                    }
                ],
                "providingApplication": "http://localhost:8060/"
            }
        }
    ]
}
```

3. subscribeContextAvailability

機能	コンテキストの所在更新通知を予約する。	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/registry/subscribeContextAvailability	
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)	
ボディ	タグ名／キー名	説明
	entities	対象エンティティリスト
	type	エンティティタイプ
	isPattern	id に正規表現を使用するか否かを”true”か”false”で指定

		id	エンティティ ID。正規表現も可。
	attributes		属性名称リスト。複数指定可。省略時、すべての属性が対象となる。
	reference		データ更新通知を送信する通知先
	duration		有効期限を ISO8601 標準フォーマットの期間仕様で指定（省略時 24 時間）。
＜レスポンス＞			
ボディ	タグ名／キー名		説明
	subscriptionId		データ更新通知予約識別子
	duration		設定された有効期限
	errorCode		エラー情報
		code	エラーコード
		details	エラー詳細
		reasonPhrase	エラーメッセージ

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/registry/subscribeContextAvailability" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
  "entities": [
    {
      "type": "Room",
      "isPattern": "true",
      "id": ".*Room"
    }
  ],
  "attributes": [
    "temperature",
    "occupancy",
    "lightstatus"
  ],
  "reference": "http://172.19.0.20:8060/ngsi10/notify",
  "duration": "PT5M"
}
EOF
```

＜レスポンス＞

```
{
    "duration": "PT5M",
    "subscriptionId": "58dcc03b75e42de7d5336948"
}
```

4. updateContextAvailabilitySubscription

機能	**データの所在更新通知を変更する。登録内容の以下のフィールドを変更することが可**

	能。 **・entities** **・attributes** **・duration**
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/registry/updateContextAvailabilitySubscription
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)

ボディ	タグ名／キー名		説明
	subscriptionId		変更対象となるデータ更新通知予約の識別子
	entities		対象エンティティリスト
		type	エンティティタイプ
		isPattern	id に正規表現を使用するか否かを”true”か”false”で指定
		id	エンティティ ID。正規表現も可。
	attributes		属性名称リスト。複数指定可。省略時、すべての属性が対象となる。
	duration		有効期限を ISO8601 標準フォーマットの期間仕様で指定（省略時 24 時間）。

＜レスポンス＞

ボディ	タグ名／キー名		説明
	subscriptionId		データ更新通知予約識別子
	duration		設定された有効期限
	errorCode		エラー情報
		code	エラーコード
		details	エラー詳細
		reasonPhrase	エラーメッセージ

curl コマンド実行例：

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

```
    }
  ],
  "attributes": [
    "temperature",
    "pressure"
  ] ,
  "duration": "PT3M",
  "subscriptionId" : "58dcc03b75e42de7d5336948"
}
EOF
```

＜レスポンス＞

```
{
    "duration": "PT3M",
    "subscriptionId": "58dcc03b75e42de7d5336948"
}
```

5. unsubscribeContextAvailability

機能	コンテキストの所在更新通知を解除する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/registry/unsubscribeContextAvailability	
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)	
ボディ	タグ名／キー名	説明
	subscriptionId	削除対象となるデータ更新通知予約の識別子
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	subscriptionId	指定したデータ更新通知予約識別子
	statusCode	ステータス情報
	code	HTTP ステータスコード
	details	ステータス詳細。必要に応じて返却される
	reasonPhrase	メッセージ

curl コマンド実行例：

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

```
{
    "subscriptionId": "58dcc03b75e42de7d5336948"
}
EOF
```

＜レスポンス＞

```
{
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    },
    "subscriptionId": "58dcc03b75e42de7d5336948"
}
```

6. notifyContextAvailability

機能	コンテキストの所在更新を通知する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/registry/notifyContextAvailability	
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)	
ボディ	タグ名／キー名	説明
	subscriptionId	対象となるデータ更新通知予約の識別子
	contextRegistrationResponses	コンテキストレジストレーションレスポンス
	contextRegistrations	コンテキストレジストレーションリスト
	entities	エンティティ情報リスト
	type	エンティティタイプ
	isPattern	false(固定)
	id	エンティティ ID
	attributes	属性リスト
	name	属性名称
	type	属性タイプ
	providingApplication	コンテキストプロバイダの URL
	errorCode	エラー情報
	code	エラーコード
	details	エラー詳細
	reasonPhrase	エラーメッセージ
＜レスポンス＞		
ボディ	タグ名／キー名	説明

	responseCode		ステータス情報
		code	HTTP ステータスコード
		details	ステータス詳細。必要に応じて返却される
		reasonPhrase	メッセージ

curl コマンド実行例:

```
(curl -k -X POST "https://${IP}/orion/v1.0/registry/notifyContextAvailability" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
  "subscriptionId": "51307b66f481db11bf860001",
  "contextRegistrationResponses": [
    {
      "contextRegistration" : {
        "entities": [
          {
            "type": "Room",
            "isPattern": "false",
            "id": "ConferenceRoom"
          }
        ],
        "attributes": [
          {
            "name": "temperature",
            "type": "degree"
          },
          {
            "name": "pressure",
            "type": "clima"
          }
        ],
        "providingApplication": "http://sensor1:1028/providingApplication"
      }
    }
  ]
}
EOF
```

＜レスポンス＞

```
{
    "responseCode": {
        "code": "200",
        "reasonPhrase": "OK"
    }
}
```

NGSI-9 拡張 API

1. contextEntities/{エンティティ ID}

機能	コンテキスト・エンティティの所在に関する情報に対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。	
コンテキスト・エンティティ情報の追加		
指定されたエンティティに関する情報を登録する。		
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/registry/contextEntities/{エンティティ ID}	
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)	
ボディ	タグ名／キー名	説明
	duration	有効期限を ISO8601 標準フォーマットの期間仕様で指定（省略時 24 時間）。
	providingApplication	コンテキストプロバイダの URL
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	duration	設定された有効期限
	registrationId	登録したコンテキスト情報に対する登録 ID
	errorCode	エラー情報
	code	エラーコード
	details	エラー詳細
	reasonPhrase	エラーメッセージ

curl コマンド実行例

```
(curl -k -X POST "https://${IP}/orion/v1.0/registry/contextEntities/Room1" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
  "duration": "PT1H",
  "providingApplication": "http://sensor1:1028/providingApplication"
}
EOF
```

＜レスポンス＞

```
{
```

```
    "duration": "PT1H",
    "registrationId": "58dcdd5975e42de7d5336955"
}
```

コンテキスト・エンティティの所在情報の検索		
指定されたコンテキスト・エンティティの所在に関するすべての情報を検索する。		
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v1.0/registry/contextEntities/{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)	
ボディ	なし	
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	contextRegistrationResponses	コンテキストレジストレーションレスポンス
	contextRegistration	コンテキストレジストレーション
	entities	エンティティ情報リスト
	type	エンティティタイプ
	isPattern	false(固定)
	id	エンティティ ID
	attributes	属性リスト
	name	属性名称
	type	属性タイプ
	providingApplication	コンテキストプロバイダの URL
	errorCode	エラー情報
	code	エラーコード
	details	エラー詳細
	reasonPhrase	エラーメッセージ

curl コマンド実行例

```
(curl -k -X GET "https://${IP}/orion/v1.0/registry/contextEntities/Room1" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
   | python -mjson.tool)
```

＜レスポンス＞

```
{
    "contextRegistrationResponses": [
        {
            "contextRegistration": {
                "entities": [
```

```
                    {
                        "id": "Room1",
                        "isPattern": "false",
                        "type": ""
                    }
                ],
                "providingApplication": "http://sensor1:1028/providingApplication"
            }
        }
    ]
}
```

2. contextEntities/{エンティティ ID}/attributes

機能	指定されたコンテキスト・エンティティの所在に関する属性情報に対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。
コンテキスト・エンティティ情報の追加	
指定されたエンティティに関する情報を登録する。	
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/registry/contextEntities/{エンティティ ID}/attributes
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)
ボディ	(1) contextEntities/{エンティティ ID}を参照
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

```
(curl -k -X POST "https://${IP}/orion/v1.0/registry/contextEntities/Room1/attributes" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
   "duration": "PT1H",
   "providingApplication": "http://sensor1:1028/providingApplication"
}
EOF
```

＜レスポンス＞

```
{
    "duration": "PT1H",
    "registrationId": "58dcdd5975e42de7d5336955"
```

```
}
```

コンテキスト・エンティティの所在情報の照会	
指定されたコンテキスト・エンティティに関するすべての情報を検索する。	
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/orion/v1.0/registry/contextEntities/{エンティティ ID}/attributes
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)
ボディ	なし
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

```
(curl -k -X GET "https://${IP}/orion/v1.0/registry/contextEntities/Room1/attributes" -s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
    | python -mjson.tool)
```

＜レスポンス＞

```
{
    "contextRegistrationResponses": [
        {
            "contextRegistration": {
                "entities": [
                    {
                        "id": "Room1",
                        "isPattern": "false",
                        "type": ""
                    }
                ],
                "providingApplication": "http://sensor1:1028/providingApplication"
            }
        }
    ]
}
```

3. contextEntities/{エンティティ ID}/attributes/{属性名}

機能	指定されたコンテキスト・エンティティの所在に関する属性に対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。
コンテキスト・エンティティ情報の追加	

指定されたコンテキスト・エンティティの属性値を登録する。	
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/registry/contextEntities/{エンティティ ID}/attributes/{属性名}
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)
ボディ	(1) contextEntities/{エンティティ ID}を参照
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

```
(curl -k -X POST "https://${IP}/orion/v1.0/registry/contextEntities/Room1/attributes/temperature"
-s -S ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
   "duration": "PT1H",
   "providingApplication": "http://sensor1:1028/providingApplication"
}
EOF
```

＜レスポンス＞

```
{
      "duration": "PT1H",
      "registrationId": "58dcdd5975e42de7d5336955"
}
```

コンテキスト・エンティティの所在情報の照会	
指定されたコンテキスト・エンティティの属性値に関するすべての情報を検索する。	
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/orion/v1.0/registry/contextEntities/{エンティティ ID}/attributes/{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)
ボディ	なし
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

```
(curl -k -X GET "https://${IP}/orion/v1.0/registry/contextEntities/Room1/attributes/temperature"
-s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
    "contextRegistrationResponses": [
        {
            "contextRegistration": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": ""
                    }
                ],
                "entities": [
                    {
                        "id": "Room3",
                        "isPattern": "false",
                        "type": ""
                    }
                ],
                "providingApplication": "http://sensor1:1028/providingApplication"
            }
        }
    ]
}
```

4. contextEntityTypes /{タイプ名}

機能	指定されたタイプのコンテキスト・エンティティの所在に対する操作を行う。実施可能な操作は、HTTP メソッドにより異なる。
コンテキスト・エンティティ情報の追加	
指定されたタイプのコンテキスト・エンティティに関する情報を登録する。	
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/registry/contextEntityTypes/{タイプ名}
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)

ボディ	(1) contextEntities/{エンティティ ID}を参照
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

```
(curl -k -X POST "https://${IP}/orion/v1.0/registry/contextEntityTypes/Bar" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
  "duration": "PT1H",
  "providingApplication": "http://sensor1:1028/providingApplication"
}
EOF
```

＜レスポンス＞

```
{
    "duration": "PT1H",
    "registrationId": "58dcdd5975e42de7d5336955"
}
```

コンテキスト・エンティティの所在情報の照会	
指定されたタイプのコンテキスト・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/v1.0/registry/contextEntityTypes/{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)
ボディ	なし
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

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

＜レスポンス＞

```
{
    "contextRegistrationResponses": [
        {
```

```
            "contextRegistration": {
                "entities": [
                    {
                        "id": "",
                        "isPattern": "false",
                        "type": "Bar"
                    }
                ],
                "providingApplication": "http://sensor1:1028/providingApplication"
            }
        }
    ]
}
```

5. contextEntityTypes /{タイプ名}/attributes

機能	指定されたタイプのコンテキスト・エンティティの所在に関する属性情報に対して操作を行う。実施可能な操作は、HTTP メソッドにより異なる。
コンテキスト・エンティティ情報の追加	
指定されたタイプのコンテキスト・エンティティに関する属性情報を登録する。	
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/registry/contextEntityTypes/{タイプ名}/attributes
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)
ボディ	(1) contextEntities/{エンティティ ID}を参照
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

```
(curl -k -X POST "https://${IP}/orion/v1.0/registry/contextEntityTypes/Favorite/attributes" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
  "duration": "PT1H",
  "providingApplication": "http://sensor1:1028/providingApplication"
}
EOF
```

＜レスポンス＞

```
{
```

```
    "duration": "PT1H",
    "registrationId": "58dcdd5975e42de7d5336955"
}
```

コンテキスト・エンティティの所在情報の照会	
指定されたタイプのコンテキスト・エンティティに関する属性情報を検索する。	
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/orion/v1.0/registry/contextEntityTypes/{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)
ボディ	なし
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

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

＜レスポンス＞

```
{
    "contextRegistrationResponses": [
        {
            "contextRegistration": {
                "entities": [
                    {
                        "id": "",
                        "isPattern": "false",
                        "type": "Favorite"
                    }
                ],
                "providingApplication": "http://sensor1:1028/providingApplication"
            }
        }
    ]
}
```

6. contextEntityTypes /{タイプ名}/attributes/{属性名}

機能	指定されたタイプのコンテキスト・エンティティの所在に関する属性情報に対して操作を行う。 実施可能な操作は、HTTP メソッドにより異なる。

コンテキスト・エンティティ情報の追加	
指定されたタイプのコンテキスト・エンティティに関する属性情報を登録する。	
＜リクエスト＞	
HTTP メソッド	POST
URL	https://${IP}/orion/v1.0/registry/contextEntityTypes/{タイプ名}/attributes/{属性名}
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)
ボディ	(1) contextEntities/{エンティティ ID}を参照
＜レスポンス＞	
ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

```
(curl -k -X POST ¥
"https://${IP}/orion/v1.0/registry/contextEntityTypes/Workplace/attributes/temperature" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
   "duration": "PT1H",
   "providingApplication": "http://sensor1:1028/providingApplication"
}
EOF
```

＜レスポンス＞

```
{
      "duration": "PT1H",
      "registrationId": "58dcdd5975e42de7d5336955"
}
```

コンテキスト・エンティティの所在情報の照会	
指定されたタイプのコンテキスト・エンティティに関する属性情報を検索する。	
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/orion/v1.0/registry/contextEntityTypes/{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)
ボディ	なし
＜レスポンス＞	

ボディ	(1) contextEntities/{エンティティ ID}を参照

curl コマンド実行例

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

＜レスポンス＞

```
{
    "contextRegistrationResponses": [
        {
            "contextRegistration": {
                "attributes": [
                    {
                        "name": "temperature",
                        "type": ""
                    }
                ],
                "entities": [
                    {
                        "id": "",
                        "isPattern": "false",
                        "type": "Workplace"
                    }
                ],
                "providingApplication": "http://sensor1:1028/providingApplication"
            }
        }
    ]
}
```

7. contextAvailabilitySubscriptions

機能	コンテキストの所在更新通知を予約する。	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v1.0/registry/contextAvailabilitySubscriptions	
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)	
ボディ	タグ名／キー名	説明
	entities	対象エンティティリスト

		type	エンティティタイプ
		isPattern	id に正規表現を使用するか否かを”true”か”false”で指定
		id	エンティティ ID。正規表現も可。
	attributes		属性名称リスト。複数指定可。省略時、すべての属性が対象となる。
	reference		データ更新通知を送信する通知先
	duration		有効期限を ISO8601 標準フォーマットの期間仕様で指定（省略時 24 時間）。
＜レスポンス＞			
ボディ	タグ名／キー名		説明
	subscriptionId		データ更新通知予約識別子
	duration		設定された有効期限
	errorCode		エラー情報
		code	エラーコード
		details	エラー詳細
		reasonPhrase	エラーメッセージ

curl コマンド実行例：

```
(curl -k -X POST "https://${IP}/orion/v1.0/registry/contextAvailabilitySubscriptions" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 -d @- | python -mjson.tool) <<EOF
{
  "entities": [
    {
      "type": "Room",
      "isPattern": "true",
      "id": ".*Room"
    }
  ],
  "attributes": [
    "temperature"
  ],
  "reference": "http://example.com:1026/v1/notify",
  "duration": "PT1M"
}
EOF
```

＜レスポンス＞

```
{
    "duration": "PT1M",
    "subscriptionId": "58ddba3d75e42de7d5336961"
}
```

8. contextAvailabilitySubscriptions /{サブスクリプション ID}

機能	データの所在更新通知予約に関する操作を行う。実施可能な操作は、HTTP メソッドにより異なる。	
データ所在更新通知予約内容の置換		
データの所在更新通知を変更する。登録内容の以下のフィールドを変更することが可能。 ・entities ・attributes ・duration		
＜リクエスト＞		
HTTP メソッド	PUT	
URL	https://${IP}/orion/v1.0/registry/contextAvailabilitySubscriptions/{サブスクリプション ID}	
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)	
ボディ	タグ名／キー名	説明
	subscriptionId	変更対象となるデータ更新通知予約の識別子
	entities	対象エンティティリスト
	type	エンティティタイプ
	isPattern	id に正規表現を使用するか否かを”true”か”false”で指定
	id	エンティティ ID。正規表現も可。
	attributes	属性名称リスト。複数指定可。省略時、すべての属性が対象となる。
	duration	有効期限を ISO8601 標準フォーマットの期間仕様で指定（省略時 24 時間）。
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	subscriptionId	データ更新通知予約識別子
	duration	設定された有効期限
	errorCode	エラー情報
	code	エラーコード
	details	エラー詳細
	reasonPhrase	エラーメッセージ

curl コマンド実行例：

```
(curl -k -X PUT ¥
  "https://${IP}/orion/v1.0/registry/contextAvailabilitySubscriptions/58ddba3d75e42de7d5336961" ¥
-s -S --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- | python -mjson.tool) <<EOF
{
```

```
  "entities": [
    {
      "type": "Room",
      "isPattern": "false",
      "id": "ConferenceRoom"
    }
  ],
  "attributes": [
    "pressure"
  ],
  "duration": "P9Y",
  "subscriptionId" : "58ddba3d75e42de7d5336961"
}
EOF
```

＜レスポンス＞

```
{
    "duration": "P9Y",
    "subscriptionId": "58ddba3d75e42de7d5336961"
}
```

コンテキストの所在更新通知を解除する			
コンテキストの所在更新通知を解除する。			
＜リクエスト＞			
HTTP メソッド	DELETE		
URL	https://${IP}/orion/v1.0/registry/contextAvailabilitySubscriptions/{サブスクリプション 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)		
ボディ	なし		
＜レスポンス＞			
ボディ	タグ名／キー名		説明
	subscriptionId		指定したデータ更新通知予約識別子
	statusCode		ステータス情報
		code	HTTP ステータスコード
		details	ステータス詳細。必要に応じて返却される
		reasonPhrase	メッセージ

curl コマンド実行例：

```
 (curl -k -X DELETE ¥
"https://${IP}/orion/v1.0/registry/contextAvailabilitySubscriptions/58ddba3d75e42de7d5336961" ¥
-s -S --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Accept: application/json" ¥
  | python -mjson.tool)
```

＜レスポンス＞

```
{
    "statusCode": {
        "code": "200",
        "reasonPhrase": "OK"
    },
    "subscriptionId": "58ddba3d75e42de7d5336961"
}
```

8.2.2 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 付録 A 参考情報[9]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 付録 A 参考情報[9]の Simple Query Language 項目を参照のこと
	georal	[任意]座標の位置関係 付録 A 参考情報[9]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 付録 A 参考情報[9]の Geographical Queries 項目を参照のこと
	coords	[任意]座標。WGS-84 に準拠 付録 A 参考情報[9]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 upsert: 指定したエンティティが存在する場合、属性情報を更新
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: /v2/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: text/plain" ¥
  -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. subscriptions

機能	全てのサブスクリプションの情報を取得する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/subscriptions	
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]取得するサブスクリプションの上限数
	offset	[任意]取得するサブスクリプションのオフセット
	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
＜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	サブスクリプション ID
	description	サブスクリプション説明
	subject	サブスクライブ対象情報

		entities	エンティティリスト(配列)
		id	エンティティ ID
		type	エンティティタイプ
		condition	通知トリガ情報
		attrs	属性名称リスト
		expression	表現
		${ExpressionType}	※${ExpressionType}は表現種別 “q”,”mq”,”georel”,”geometry”,”coords”のいずれかとなる
	notification		通知情報
		http	HTTP プロトコル　※httpCustom とは排他
		url	通知先 URL
		httpCustom	HTTP プロトコル　※httpCustom とは排他
		url	通知先 URL
		headers	ヘッダ情報
		${HeaderName}	※${HeaderName}はヘッダフィールド 重複しないヘッダフィールドが複数存在する可能性あり
		qs	クエリ情報
		${Query}	※${Query}はクエリパラメータ 重複しないクエリパラメータが複数存在する可能性あり
		method	HTTP メソッド
		payload	通知のペイロード
		attrsFormat	通知時の属性フォーマット “normalized”(default),”keyValues”,”values”のいずれかとなる
		attrs	通知対象とする属性名称リスト　※exceptAttrs とは排他
		exceptAttrs	通知より除外する属性名称リスト　※attrs とは排他
		metadata	メタデータ名称リスト
		timesSent	送信回数
		lastNotification	最新通知日時(フォーマットは ISO8601 に準拠)
		lastFailure	最終失敗日時(フォーマットは ISO8601 に準拠)
		lastSuccess	最終成功日時(フォーマットは ISO8601 に準拠)
	expires		サブスクリプションの有効期限(フォーマットは ISO8601 に準拠)
	status		サブスクリプションのステータス (active:有効, oneshot:一度だけ有効, inactive:無効, expired:有効期限切れ, failed:内部エラー)
	throttling		最小通知間隔(秒)

curl コマンド実行例：

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

＜レスポンス＞

```
[
  {
    "id": "abcdefg",
    "description": "One subscription to rule them all",
    "subject": {
      "entities": [
        {
          "id": "Bcn-Welt",
          "type": "Room"
        }
      ],
      "condition": {
        "attrs": [
          "temperature "
        ],
        "expression": {
          "q": "temperature>40"
        }
      }
    },
    "notification": {
      "httpCustom": {
        "url": "http://localhost:1234",
        "headers": {
          "X-MyHeader": "foo"
        },
        "qs": {
          "authToken": "bar"
        }
      },
      "attrsFormat": "keyValues",
      "attrs": [
        "temperature",
        "humidity"
      ],
      "timesSent": 12,
      "lastNotification": "2015-10-05T16:00:00.00Z",
      "lastFailure": "2015-10-06T16:00:00.00Z"
    },
    "expires": "2016-04-05T14:00:00.00Z",
    "status": "failed",
    "throttling": 5
  }
]
```


機能	新しいサブスクリプションを生成する

＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v2.0/subscriptions	
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)	
ボディ	タグ名／キー名	説明
	id	サブスクリプション ID
	description	サブスクリプション説明
	subject	サブスクライブ対象情報
	entities	エンティティリスト(配列)
	id	エンティティ ID　※idPattern とは排他
	type	エンティティタイプ　※typePattern とは排他
	idPattern	エンティティ ID の正規表現　※id とは排他
	typePattern	エンティティタイプの正規表現　※type とは排他
	condition	通知トリガ情報
	attrs	属性名称リスト
	expression	表現
	${ExpressionType}	※${ExpressionType}は表現種別 “q”,”mq”,”georel”,”geometry”,”coords”のいずれかとなる
	notification	通知情報
	http	HTTP プロトコル　※httpCustom とは排他
	url	通知先 URL
	httpCustom	HTTP プロトコル　※http とは排他
	url	通知先 URL
	headers	ヘッダ情報
	${HeaderName}	※${HeaderName}はヘッダフィールド 重複しないヘッダフィールドを複数指定可能
	qs	クエリ情報
	${Query}	※${Query}はクエリパラメータ 重複しないクエリパラメータを複数指定可能
	method	HTTP メソッド
	payload	通知のペイロード
	attrsFormat	通知時の属性フォーマット “normalized”(default),”keyValues”,”values”のいずれかとなる
	attrs	通知対象とする属性名称リスト　※exceptAttrs とは排他
	exceptAttrs	通知より除外する属性名称リスト　※attrs とは排他
	metadata	メタデータ名称リスト
	expires	サブスクリプションの有効期限(フォーマットは ISO8601 に準拠)

	status	サブスクリプションのステータス (active:有効, oneshot:一度だけ有効, inactive:無効, expired:有効期限切れ, failed:内部エラー)
	throttling	最小通知間隔
＜レスポンス＞		
ヘッダ	Location: /v2/subscriptions/${id} ※${id}はサブスクリプション ID	
	(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/subscriptions" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" --header "Accept: application/json" ¥
  -d @- ) <<EOF
{
   "description": "One subscription to rule them all",
   "subject": {
      "entities": [
         {
            "idPattern": ".*",
            "type": "Room"
         }
      ],
      "condition": {
         "attrs": [ "temperature" ],
         "expression": {
            "q": "temperature>40"
         }
      }
   },
   "notification": {
      "http": {
         "url": "http://localhost:1234"
      },
      "attrs": ["temperature", "humidity"]
   },
   "expires": "2016-04-05T14:00:00.00Z",
   "throttling": 5
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 201 Created
< Connection: keep-alive
< Location: /v2/subscriptions/5a82be3d093af1b94ac0f730
```

```
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

9. subscriptions/{サブスクリプション ID}

機能	指定されたサブスクリプションの情報を取得する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/subscriptions/{サブスクリプション 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
＜レスポンス＞		
ヘッダ	Content-Type: application/json	
ボディ	タグ名／キー名	説明
	id	サブスクリプション ID
	description	サブスクリプション説明
	subject	サブスクライブ対象情報
	entities	エンティティリスト(配列)
	id	エンティティ ID
	type	エンティティタイプ
	condition	通知トリガ情報
	attrs	属性名称リスト
	expression	表現
	${ExpressionType}	※${ExpressionType}は表現種別 “q”,”mq”,”georel”,”geometry”,”coords”のいずれかとなる
	notification	通知情報
	http	HTTP プロトコル　※httpCustom とは排他
	url	通知先 URL
	httpCustom	HTTP プロトコル　※httpCustom とは排他
	url	通知先 URL
	headers	ヘッダ情報
	${HeaderName}	※${HeaderName}はヘッダフィールド 重複しないヘッダフィールドが複数存在する可能性あり
	qs	クエリ情報
	${Query}	※${Query}はクエリパラメータ 重複しないクエリパラメータが複数存在する可能性あり
	method	HTTP メソッド
	payload	通知のペイロード

		attrsFormat	通知時の属性フォーマット “normalized”(default),”keyValues”,”values”のいずれかとなる
		attrs	通知対象とする属性名称リスト　※exceptAttrs とは排他
		exceptAttrs	通知より除外する属性名称リスト　※attrs とは排他
		metadata	メタデータ名称リスト
		timesSent	送信回数
		lastNotification	最新通知日時(フォーマットは ISO8601 に準拠)
		lastFailure	最終失敗日時(フォーマットは ISO8601 に準拠)
		lastSuccess	最終成功日時(フォーマットは ISO8601 に準拠)
	expires		サブスクリプションの有効期限(フォーマットは ISO8601 に準拠)
	status		サブスクリプションのステータス (active:有効, oneshot:一度だけ有効, inactive:無効, expired:有効期限切れ, failed:内部エラー)
	throttling		最小通知間隔

curl コマンド実行例：

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

＜レスポンス＞

```
{
   "id": "abcdef",
   "description": "One subscription to rule them all",
   "subject": {
      "entities": [
         {
            "idPattern": ".*",
            "type": "Room"
         }
      ],
      "condition": {
         "attrs": [ "temperature" ],
         "expression": {
            "q": "temperature>40"
         }
      }
   },
   "notification": {
      "http": {
         "url": "http://localhost:1234"
      },
      "attrs": ["temperature", "humidity"],
```

```
      "timesSent": 12,
      "lastNotification": "2015-10-05T16:00:00.00Z"
      "lastSuccess": "2015-10-05T16:00:00.00Z"
   },
   "expires": "2016-04-05T14:00:00.00Z",
   "status": "active",
   "throttling": 5
}
```

機能	指定されたサブスクリプションの情報を更新する	
＜リクエスト＞		
HTTP メソッド	PATCH	
URL	https://${IP}/orion/v2.0/subscriptions/{サブスクリプション ID}	
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)	
ボディ	タグ名／キー名	説明
	id	サブスクリプション ID
	description	サブスクリプション説明
	subject	サブスクライブ対象情報
	entities	エンティティリスト(配列)
	id	エンティティ ID
	type	エンティティタイプ
	condition	通知トリガ情報
	attrs	属性名称リスト
	expression	表現
	${ExpressionType}	※${ExpressionType}は表現種別 “q”,”mq”,”georel”,”geometry”,”coords”のいずれかとなる
	notification	通知情報
	http	HTTP プロトコル　※httpCustom とは排他
	url	通知先 URL
	httpCustom	HTTP プロトコル　※httpCustom とは排他
	url	通知先 URL
	headers	ヘッダ情報
	${HeaderName}	※${HeaderName}はヘッダフィールド 重複しないヘッダフィールドを複数指定可能
	qs	クエリ情報
	${Query}	※${Query}はクエリパラメータ 重複しないクエリパラメータを複数指定可能
	method	HTTP メソッド
	payload	通知のペイロード

		attrsFormat	通知時の属性フォーマット “normalized”(default),”keyValues”,”values”のいずれかとなる
		attrs	通知対象とする属性名称リスト　※exceptAttrs とは排他
		exceptAttrs	通知より除外する属性名称リスト　※attrs とは排他
		metadata	メタデータ名称リスト
	expires		サブスクリプションの有効期限(フォーマットは ISO8601 に準拠)
	status		サブスクリプションのステータス (active:有効, oneshot:一度だけ有効, inactive:無効, expired:有効期限切れ, failed:内部エラー)
	throttling		最小通知間隔
＜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/subscriptions/abcdef" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Content-Type: application/json" ¥
  -d @-) <<EOF
{
   "expires": "2016-04-05T14:00:00.00Z"
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: 5c301e3e-59e1-11e9-b35f-02ad23835e3a
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

機能	**指定されたサブスクリプションをキャンセルする**	
＜リクエスト＞		
HTTP メソッド	DELETE	
URL	https://${IP}/orion/v2.0/subscriptions/{サブスクリプション 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
＜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/subscriptions/abcdef" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json")
```

＜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
```

10. registrations

機能	**登録されているすべてのコンテキストの所在を取得する**	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/registrations	
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]取得するコンテキストプロバイダの上限数
	offset	[任意]取得するコンテキストプロバイダのオフセット
	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
＜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	レジストレーション ID
	description	レジストレーション説明
	dataProvided	このレジストレーションより中継されるデータ情報
	entities	エンティティリスト(配列)
	id	エンティティ ID
	type	エンティティタイプ
	attrs	属性名称リスト
	expression	表現
	${ExpressionType}	※${ExpressionType}は表現種別

				”georel”,”geometry”,”coords”のいずれかとなる
	provider			プロバイダ情報
		http		HTTP プロトコル
			url	登録先 URL
		supportedForwardingMode		コンテキストプロバイダがサポートする転送モード “none”,”query”,”update”,”all”(default)のいずれかとなる
	expires			レジストレーションの有効期限(フォーマットは ISO8601 に準拠)
	status			レジストレーションのステータス (active:有効, inactive:無効, expired:有効期限切れ, failed:内部エラー)
	forwardingInformation			転送情報
		timeSent		転送回数
		lastForwarding		最新転送日時(フォーマットは ISO8601 に準拠)
		lastSuccess		最終成功日時(フォーマットは ISO8601 に準拠)
		lastFailure		最終失敗日時(フォーマットは ISO8601 に準拠)

curl コマンド実行例:

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

＜レスポンス＞

```
[
  {
    "id": "abcdefg",
    "description": "Example Context Source",
    "dataProvided": {
      "entities": [
        {
          "id": "Bcn_Welt",
          "type": "Room"
        }
      ],
      "attrs": [
        "temperature"
      ]
    },
    "provider": {
      "http": {
        "url": "http://contextsource.example.org"
      },
      "supportedForwardingMode": "all"
    },
    "expires": "2017-10-31T12:00:00",
```

```
    "status": "active",
    "forwardingInformation": {
      "timesSent": 12,
      "lastForwarding": "2017-10-06T16:00:00.00Z",
      "lastSuccess": "2017-10-06T16:00:00.00Z",
      "lastFailure": "2017-10-05T16:00:00.00Z"
    }
  }
]
```

機能	コンテキストの所在を登録する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/orion/v2.0/registrations	
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)	
ボディ	タグ名／キー名	説明
	description	レジストレーション説明
	dataProvided	このレジストレーションより中継されるデータ情報
	entities	エンティティリスト(配列)
	id	エンティティ ID
	type	エンティティタイプ
	attrs	属性名称リスト
	expression	表現
	${ExpressionType}	※${ExpressionType}は表現種別 ”georel”,”geometry”,”coords”のいずれかとなる
	provider	プロバイダ情報
	http	HTTP プロトコル
	url	登録先 URL
	supportedForwardingMode	コンテキストプロバイダがサポートする転送モード “none”,”query”,”update”,”all”(default)のいずれかとなる
＜レスポンス＞		
ヘッダ	Location: /v2/registrations/${id} ※${id}はレジストレーション ID	
	(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/registrations" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
```

```
 --header "Content-Type: application/json" ¥
 -d @-) <<EOF
{
  "description": "Relative Humidity Context Source",
  "dataProvided": {
    "entities": [
      {
        "id": "room2",
        "type": "Room"
      }
    ],
    "attrs": [
      "relativeHumidity"
    ]
  },
  "provider": {
    "http":{
      "url": "http://localhost:1234"
    },
    "legacyForwarding": true
  }
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 201 Created
< Connection: keep-alive
< Location: /v2/registrations/5a82be3d093af1b94ac0f730
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

11. registrations/{レジストレーション ID}

機能	指定されたコンテキストの所在を検索する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/orion/v2.0/registrations/{レジストレーション 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
＜レスポンス＞		
ヘッダ	Content-Type: application/json	

ボディ	タグ名／キー名	説明
	id	レジストレーション ID
	description	レジストレーション説明
	dataProvided	このレジストレーションより中継されるデータ情報
	entities	エンティティリスト(配列)
	id	エンティティ ID
	type	エンティティタイプ
	attrs	属性名称リスト
	expression	表現
	${ExpressionType}	※${ExpressionType}は表現種別 ”georel”,”geometry”,”coords”のいずれかとなる
	provider	プロバイダ情報
	http	HTTP プロトコル
	url	登録先 URL
	supportedForwardingMode	コンテキストプロバイダがサポートする転送モード “none”,”query”,”update”,”all”(default)のいずれかとなる
	expires	レジストレーションの有効期限(フォーマットは ISO8601 に準拠)
	status	レジストレーションのステータス (active:有効, inactive:無効, expired:有効期限切れ, failed:内部エラー)
	forwardingInformation	転送情報
	timeSent	転送回数
	lastForwarding	最新転送日時(フォーマットは ISO8601 に準拠)
	lastSuccess	最終成功日時(フォーマットは ISO8601 に準拠)
	lastFailure	最終失敗日時(フォーマットは ISO8601 に準拠)

curl コマンド実行例：

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

＜レスポンス＞

```
{
        "id": "abcdefg",
        "description": "Example Context Source",
        "dataProvided": {
          "entities": [
            {
              "id": "Bcn_Welt",
              "type": "Room"
            }
          ],
          "attrs": [
```

```
                "temperature"
            ]
        },
        "provider": {
            "http": {
                "url": "http://contextsource.example.org"
            },
            "supportedForwardingMode": "all"
        },
        "expires": "2017-10-31T12:00:00",
        "status": "failed",
        "forwardingInformation": {
            "timesSent": 12,
            "lastForwarding": "2017-10-06T16:00:00.00Z",
            "lastFailure": "2017-10-06T16:00:00.00Z",
            "lastSuccess": "2017-10-05T18:25:00.00Z",
        }
}
```

機能	指定されたコンテキストの所在を解除する	
＜リクエスト＞		
HTTP メソッド	DELETE	
URL	https://${IP}/orion/v2.0/registrations/{レジストレーション 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
＜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/registrations/abcdefg" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json")
```

＜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
```

12. 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
```

13. 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
	attrs	属性リスト(配列)
	expression	「q」、「mq」、「georel」、「geometry」、「coords」で構成されるクエリ式 ※各構成要素の説明は GET /entities クエリパラメータの項目を参照のこと
	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

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"
    }
  ],
  "attrs": [
    "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"
        }
      }
    }
]
```

8.2.3 IDAS インタフェース

MQTT プロトコルのトピック仕様については、第 7 章 および付録 A 参考情報 [10]を参照してください。

services

1. デバイス種別情報登録

機能	デバイス種別情報を登録する	
＜リクエスト＞		
HTTP メソッド	POST	
URL	https://${IP}/iot/v1.0/services	
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: (任意の文字列) Fiware-ServicePath: (任意の文字列)	
ボディ	タグ名／キー名	説明
	services	デバイス種別情報リスト（配列）
	apikey	API キー文字列
	entity_type	デバイス種別に割り当てるタイプの名前
	resource	HTTP の URI（プロトコルが HTTP の場合、デバイスはこの URI に情報を送信可能）
	trust	[任意] データ収集/蓄積レイヤへのセキュアなアクセスに使用するトラストトークン （オプション、セキュリティで保護されたシナリオでのみ必要）
	cbHost	[任意]データ収集/蓄積レイヤ接続情報 （このオプションを使用して、特定の種類のデバイスのグローバルなものを上書きすることが可能）
	lazy	[任意]デバイスの遅延属性のリスト（配列） 各属性について、名前とタイプを指定する必要あり
	commands	[任意]デバイスのコマンド属性のリスト（配列） 各属性について、名前とタイプを指定する必要あり
	attributes	[任意]デバイスのアクティブな属性のリスト（配列） 各属性について、名前とタイプを指定する必要あり
	static_attributes	[任意]エンティティに追加する静的属性のリスト（配列） データ収集/蓄積レイヤへの全ての要求には、この一連の属性が追加される
＜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}/iot/v1.0/services" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 -d @- ) <<EOF
{
    "services": [
        {
            "apikey": "1234567",
            "entity_type": "TempSensor",
            "resource": "/iot/json"
        }
    ]
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 201 Created
< Connection: keep-alive
< Content-Type: application/json; charset=utf-8
< Fiware-Correlator: e31a7cbd-8273-4c34-b6ea-f8db1a0170df
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

2. デバイス種別情報取得

機能	登録しているデバイス種別情報を取得する	
＜リクエスト＞		
HTTP メソッド	GET	
URL	https://${IP}/iot/v1.0/services	
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: (任意の文字列) Fiware-ServicePath: (任意の文字列)	
クエリパラメータ	パラメータ名	説明
	limit	[任意]取得するデバイス種別情報の上限数（デフォルトは 20、最大で 1000 まで）
	offset	[任意]取得するデバイス種別情報のオフセット
	resource	[任意]指定した resource のデバイス種別のみを取得
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	count	デバイス種別情報数
	services	デバイス種別情報リスト（配列）
	id	デバイス種別を識別するために使用される ID

		service	デバイス種別が属するサービスの名前（Fiware-Service ヘッダ）
		subservice	デバイス種別が属するサブサービスの名前 （Fiware-ServicePath ヘッダ）
		apikey	API キー文字列
		resource	HTTP の URI
		attributes	デバイスのアクティブな属性のリスト（配列）
		lazy	デバイスの遅延属性のリスト（配列）
		commands	デバイスのコマンド属性のリスト（配列）
		entity_type	デバイス種別に割り当てるタイプの名前
		internal_attributes	特定の IoT エージェント設定用のフリーフォーマットの内部属性リスト（配列）
		static_attributes	エンティティに追加する静的属性のリスト（配列）

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/iot/v1.0/services" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 | python -mjson.tool)
```

＜レスポンス＞

```
{
    “_id”, “1234567890abcdef12345678”,
    “subservice”: “/dummy”,
    “service”: “mqttjson”,
    “apikey”: “1234567”,
    “resource”:”/iot/json”,
    “attributes”:[],
    “lazy”:[],
    “commands”:[],
    “entity_type”:”TempSensor”,
    “internal_attributes”:[],
    “static_attributes”:[]
}
```

3. デバイス種別情報更新

機能	登録しているデバイス種別情報を更新
＜リクエスト＞	
HTTP メソッド	PUT
URL	https://${IP}/iot/v1.0/services
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: (任意の文字列) Fiware-ServicePath: (任意の文字列)	
クエリパラメータ	パラメータ名	説明
	apikey	更新するデバイス種別の API キーを指定
	resource	更新するデバイス種別の resource を指定
ボディ	タグ名／キー名	説明
	entity_type	デバイス種別に割り当てるタイプの名前
	trust	データ収集/蓄積レイヤへのセキュアなアクセスに使用するトラストトークン （オプション、セキュリティで保護されたシナリオでのみ必要）
	cbHost	データ収集/蓄積レイヤ接続情報 （このオプションを使用して、特定の種類のデバイスのグローバルなものを上書きすることが可能）
	lazy	デバイスの遅延属性のリスト（配列） 各属性について、名前とタイプを指定する必要あり
	commands	デバイスのコマンド属性のリスト（配列） 各属性について、名前とタイプを指定する必要あり
	attributes	デバイスのアクティブな属性のリスト（配列） 各属性について、名前とタイプを指定する必要あり
	static_attributes	エンティティに追加する静的属性のリスト（配列） データ収集/蓄積レイヤへの全ての要求には、この一連の属性が追加される
＜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}/iot/v1.0/services?resource=/iot/json&apikey=1234567" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 -d @- ) <<EOF
{
  "entity_type": "TempSensor"
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: e31a7cbd-8273-4c34-b6ea-f8db1a0170df
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

4. デバイス種別情報削除

機能	登録しているデバイス種別情報を削除する	
＜リクエスト＞		
HTTP メソッド	DELETE	
URL	https://${IP}/iot/v1.0/services	
ヘッダ	Accept: application/json Authorization: Bearer ${TOKEN}　※${TOKEN}はアクセストークン文字列 Fiware-Service: (任意の文字列) Fiware-ServicePath: (任意の文字列)	
クエリパラメータ	パラメータ名	説明
	apikey	削除するデバイス種別の API キーを指定
	resource	削除するデバイス種別の resource を指定
	device	[任意]指定した service/subservice 内のデバイス種別を削除（Fiware-ServicePath が'/*'または'/#'の場合、無効）
＜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}/iot/v1.0/services?resource=/iot/json&apikey=1234567" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
  --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy")
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: 18d3543e-4976-44b4-8973-cce80d2afcee
< Date: Tue, 13 Feb 2018 10:45:52 GMT
```

devices

1. デバイス情報登録

機能	デバイス情報登録		
＜リクエスト＞			
HTTP メソッド	POST		
URL	https://${IP}/iot/v1.0/devices		
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: (任意の文字列) Fiware-ServicePath: (任意の文字列)		
ボディ	タグ名／キー名		説明
	devices		デバイス情報リスト（配列）
		device_id	デバイスを識別するために使用されるデバイス ID
		entity_name	データ収集/蓄積レイヤ内のデバイスを表すエンティティの名前
		entity_type	データ収集/蓄積レイヤ内のエンティティのタイプ
		timezone	[任意]センサのタイムゾーンがある場合、指定する
		endpoint	[任意]デバイスがコマンドを受信するエンドポイント（存在する場合）
		transport	データ変換アダプタのデバイス転送プロトコルの名前 ※“MQTT”指定必須
		attributes	登録するデバイス情報の属性のリスト（配列）
		name	データ収集/蓄積レイヤ内のエンティティの属性名称
		type	データ収集/蓄積レイヤ内のエンティティの属性タイプ
		object_id	[任意]デバイスの属性名称
		commands	[任意]デバイスのコマンドのリスト（配列）
		name	データ収集/蓄積レイヤ内のエンティティの属性名称
		type	データ収集/蓄積レイヤ内のエンティティの属性タイプ
		object_id	[任意]デバイスの属性名称
		lazy	[任意]デバイスの遅延属性のリスト（配列）
		static_attributes	[任意]エンティティに追加する静的属性のリスト（配列） データ収集/蓄積レイヤへの全ての要求には、この一連の属性が追加される
＜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}/iot/v1.0/devices" -s -S -i ¥
```

```
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 -d @- ) <<EOF
{
    "devices": [
        {
            "device_id": "sensor1",
            "entity_type": "TempSensor",
            "entity_name": "TempSensor1",
            "endpoint": "http://deviceEndpoint:80",
            "transport": "MQTT",
            "attributes": [
                {
                    "name": "temperature",
                    "type": "float"
                },
                {
                    "name": "humidity",
                    "type": "float"
                }
            ],
            "commands": [
                {
                    "name": "reset",
                    "type": "Text"
                }
            ]
        }
    ]
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 201 Created
< Connection: keep-alive
< Content-Type: application/json; charset=utf-8
< Fiware-Correlator: e31a7cbd-8273-4c34-b6ea-f8db1a0170df
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

2. デバイス情報取得

機能	登録しているデバイス情報を全て取得
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/iot/v1.0/devices

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: (任意の文字列) Fiware-ServicePath: (任意の文字列)	
クエリパラメータ	パラメータ名	説明
	limit	[任意]取得するデバイス情報の上限数（デフォルトは 20、最大で 1000 まで）
	offset	[任意]取得するデバイス情報のオフセット
＜レスポンス＞		
ボディ	タグ名／キー名	説明
	count	デバイス情報数
	devices	デバイス情報リスト（配列）
	device_id	デバイスを識別するために使用されるデバイス ID
	service	デバイスが属するサービスの名前（Fiware-Service ヘッダ）
	service_path	デバイスが属するサブサービスの名前 （Fiware-ServicePath ヘッダ）
	entity_name	データ収集/蓄積レイヤ内のデバイスを表すエンティティの名前
	entity_type	データ収集/蓄積レイヤ内のエンティティのタイプ
	endpoint	デバイスがコマンドを受信するエンドポイント（存在する場合）
	transport	データ変換アダプタのデバイス転送プロトコルの名前
	attributes	デバイスのアクティブな属性のリスト（配列）
	name	データ収集/蓄積レイヤ内のエンティティの属性名称
	type	データ収集/蓄積レイヤ内のエンティティの属性タイプ
	object_id	デバイスの属性名称
	commands	デバイスのコマンドのリスト（配列）
	name	データ収集/蓄積レイヤ内のエンティティの属性名称
	type	データ収集/蓄積レイヤ内のエンティティの属性タイプ
	object_id	デバイスの属性名称
	lazy	デバイスの遅延属性のリスト（配列）
	static_attributes	エンティティに追加する静的属性のリスト（配列）

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/iot/v1.0/devices" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 | python -mjson.tool)
```

＜レスポンス＞

```
{
    “count”:1,
    "devices": [
        {
            "device_id": “sensor1",
            “service": "mqttjson",
            “service_path": "mqttjson",
            "entity_type": “TempSensor",
            "entity_name": “TempSensor1",
            "endpoint": “http://deviceEndpoint:80",
            "transport": “MQTT",
            "attributes": [
                {
                    "object_id": "temperature",
                    "name": "temperature",
                    "type": "float"
                },
                {
                    "object_id": "humidity",
                    "name": "humdity",
                    "type": "float"
                }
            ],
            "lazy": [
            ],
            "commands": [
                {
                    "object_id": "reset",
                    "name": "reset",
                    "type": "Text"
                }
            ],
            "static_attributes": [
            ]
        }
    ]
}
```

3. デバイス情報取得（単一）

機能	指定されたデバイス情報を取得する
＜リクエスト＞	
HTTP メソッド	GET
URL	https://${IP}/iot/v1.0/devices/{デバイス ID}
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: (任意の文字列) Fiware-ServicePath: (任意の文字列)
ボディ	なし

＜レスポンス＞

ボディ	タグ名／キー名	説明
	device_id	デバイスを識別するために使用されるデバイス ID
	service	デバイスが属するサービスの名前（Fiware-Service ヘッダ）
	service_path	デバイスが属するサブサービスの名前 （Fiware-ServicePath ヘッダ）
	entity_name	データ収集/蓄積レイヤ内のデバイスを表すエンティティの名前
	entity_type	データ収集/蓄積レイヤ内のエンティティのタイプ
	endpoint	デバイスがコマンドを受信するエンドポイント（存在する場合）
	transport	データ変換アダプタのデバイス転送プロトコルの名前
	attributes	デバイスのアクティブな属性のリスト（配列）
	name	データ収集/蓄積レイヤ内のエンティティの属性名称
	type	データ収集/蓄積レイヤ内のエンティティの属性タイプ
	object_id	デバイスの属性名称
	commands	デバイスのコマンドのリスト（配列）
	name	データ収集/蓄積レイヤ内のエンティティの属性名称
	type	データ収集/蓄積レイヤ内のエンティティの属性タイプ
	object_id	デバイスの属性名称
	lazy	デバイスの遅延属性のリスト（配列）
	static_attributes	エンティティに追加する静的属性のリスト（配列）

curl コマンド実行例：

```
(curl -k -X GET "https://${IP}/iot/v1.0/devices/sensor1" -s -S ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 | python -mjson.tool)
```

＜レスポンス＞

```
{
    "device_id": "sensor3",
    "service": "mqttjson",
    "service_path": "mqttjson",
    "entity_type": "TempSensor",
    "entity_name": "TempSensor1",
    "endpoint": "http://deviceEndpoint:80",
    "transport": "MQTT",
```

```
    "attributes": [
        {
            "object_id": "temperature",
            "name": "temperature",
            "type": "float"
        },
        {
            "object_id": "humidity",
            "name": "humdity",
            "type": "float"
        }
    ],
    "lazy": [
    ],
    "commands": [
        {
            "object_id": "reset",
            "name": "reset",
            "type": "Text"
        }
    ],
    "static_attributes": [
    ]
}
```

4. デバイス情報更新（単一）

機能	登録されたデバイス情報を更新する	
＜リクエスト＞		
HTTP メソッド	PUT	
URL	https://${IP}/iot/v1.0/devices/{デバイス ID}	
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: (任意の文字列) Fiware-ServicePath: (任意の文字列)	
ボディ	タグ名／キー名	説明
	entity_name	データ収集/蓄積レイヤ内のデバイスを表すエンティティの名前
	entity_type	データ収集/蓄積レイヤ内のエンティティのタイプ
	endpoint	デバイスがコマンドを受信するエンドポイント（存在する場合）
	transport	データ変換アダプタのデバイス転送プロトコルの名前
	attributes	デバイスの属性のリスト（配列）
	name	データ収集/蓄積レイヤ内のエンティティの属性名称

		type	データ収集/蓄積レイヤ内のエンティティの属性タイプ
		object_id	デバイスの属性名称
	commands		デバイスのコマンドのリスト（配列）
		name	データ収集/蓄積レイヤ内のエンティティの属性名称
		type	データ収集/蓄積レイヤ内のエンティティの属性タイプ
		object_id	デバイスの属性名称
	lazy		デバイスの遅延属性のリスト（配列）
	static_attributes		エンティティに追加する静的属性のリスト（配列）
＜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}/iot/v1.0/devices/sensor1" -s -S -i ¥
 --header "Authorization: Bearer ${TOKEN}" ¥
 --header "Content-Type: application/json" --header "Accept: application/json" ¥
 --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy" ¥
 -d @- ) <<EOF
{
    "entity_name": "TempSensor1234"
}
EOF
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: 18d3543e-4976-44b4-8973-cce80d2afcee
< Date: Tue, 13 Feb 2018 10:45:52 GMT
```

5. デバイス情報削除（単一）

機能	**指定されたデバイス情報を削除する**
＜リクエスト＞	
HTTP メソッド	DELETE
URL	https://${IP}/iot/v1.0/devices/{デバイス 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: (任意の文字列) Fiware-ServicePath: (任意の文字列)
ボディ	なし
＜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}/iot/v1.0/devices/sensor1" -s -S -i ¥
  --header "Authorization: Bearer ${TOKEN}" ¥
  --header "Accept: application/json" ¥
  --header "fiware-service: mqttjson" --header "fiware-servicepath: /dummy")
```

＜レスポンス＞

```
< HTTP/1.1 204 No Content
< Connection: keep-alive
< Fiware-Correlator: 18d3543e-4976-44b4-8973-cce80d2afcee
< Date: Tue, 13 Feb 2018 10:45:52 GMT
```

8.3 NGSI v1, NGSI v2 API 利用時の仕様・制限等

8.3.1 データ収集/蓄積レイヤ単体

本開発ガイド中の NGSIv1, NGSIv2 の API を実行する URL 中の/orion/v1.0, /orion/v2.0 は、/orion/v1, /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。

8.3.2 データ収集/蓄積レイヤと CKAN の連携

データ収集/蓄積レイヤに対するデータ更新を CKAN に通知するための設定に、NGSIv2 の subscriptions を使用する場合、"attrsFormat"属性に”legacy”を指定する必要があります。
詳しく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（データ分析参照編）を参照してください。

8.3.3 データ収集/蓄積レイヤと STH-Comet の連携

データ収集/蓄積レイヤに対するデータ更新を STH-Comet に通知するための設定に、NGSIv2 の subscriptions を使用する場合、"attrsFormat"属性に”legacy”を指定する必要があります。
詳しく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（データ分析参照編）を参照してください。

8.3.4 データ収集/蓄積レイヤと QuantumLeap の連携

データ収集/蓄積レイヤに対するデータ更新を QuantumLeap に通知するためには、NGSIv2 の subscriptions を使用してください。NGSIv1 の subscribeContext は使用できません。

メモ

QuantumLeap は、Managed 版のみ利用することができます。

付録A　参考情報

項番	タイトル	URL
[1]	Fiware Orion	http://fiware-orion.readthedocs.io/en/2.1.0/
[2]	Fiware NGSI APIv1 Walkthrougn	http://fiware-orion.readthedocs.io/en/1.15.1/user/walkthrough_apiv1/index.html
[3]	Multi tenancy	http://fiware-orion.readthedocs.io/en/2.1.0/user/multitenancy/index.html
[4]	Entity service paths	http://fiware-orion.readthedocs.io/en/2.1.0/user/service_path/index.html
[5]	NGSIv1 filtering	http://fiware-orion.readthedocs.io/en/1.15.1/user/filtering/index.html#ngsiv1-filtering
[6]	HTTP and NGSI response codes	http://fiware-orion.readthedocs.io/en/1.15.1/user/http_and_ngsi_sc/index.html
[7]	FIWARE NGSI APIv2 Walkthrough	http://fiware-orion.readthedocs.io/en/2.1.0/user/walkthrough_apiv2/
[8]	NGSIv2 filtering	http://fiware-orion.readthedocs.io/en/2.1.0/user/filtering/index.html
[9]	Simple Query Language	http://telefonicaid.github.io/fiware-orion/api/v2/stable/
[10]	IoTAgent for JSON	https://fiware-iotagent-json.readthedocs.io/en/stable/
[11]	MQTT	http://mqtt.org/documentation

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。

（2）MQTT プロトコルを用いたデバイスを使用する場合

MQTT プロトコルでの通信時、タイミングによっては送信データがロストすることがあります。詳しくは付録 A 参考情報[11]を参照してください。

（3）MQTT プロトコル通信について

IDAS コンポーネントは冗長化構成を取っていないため、コンポーネント異常による停止時は MQTT プロトコルでの送信データはロストします。高信頼性が必要な場合は NGSI 形式でのデータ登録とし、かつ送信のリトライ処理を AP 側で実装してください。

（4）NGSI 非対応デバイスのデバイス種別情報登録について

デバイス種別情報登録を 2 回以上実施する場合は Fiware-ServicePath を変更するようにしてください。1 つの Fiware-ServicePath に 2 つ以上のデバイス種別情報が登録されている場合、「7.6 NGSI 非対応デバイスへのデータ更新通知」と「7.7 NGSI 非対応デバイスのコマンド実行」が正しく動作しない場合があります。

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

データ利活用基盤-第 17-011 号

2019 年 10 月

日本電気株式会社