FIWARE Orion利用手順書
(1.2.0版)

2024年03月01日

一般社団法人データ社会推進協議会

改版履歴

バージョン	改版内容	公開日
1.0.0	初版（FIWARE Orion 3.6.0 を対象に作成）	2022/07/01
1.1.0	FIWARE Orion 3.7.0 向けに手順書を改版 2-1 に画像データ等の取り扱いに関する説明を追加 2-4 に Fiware-Service/Fiware-ServicePath に関する説明を追加	2023/03/01
1.2.0	FIWARE Orion 3.10.1 向けに手順書を改版	2024/03/01

目次

1.　はじめに

1-1. 概要

本書では、FIWARE Orion（正式名称：Orion Context Broker）（以下、「Orion」）の概要と、その代表的なAPI（Application Programming Interface）の利用手順を説明する。

1-2. 事前準備

本書の説明においては、Orionの実行環境が構築済みであり、Orionが稼働していることを前提とする。Orionの実行環境の構築方法に関しては、別紙の「FIWARE Orionビルド手順書」及び「FIWARE Orion構築手順書」を参照。

1-3. 表記方法

1-3-1. コマンドの表記方法

（例）

```
# source ~/ENV.sh
```

コマンド入力を表す箇所については、上記のように実線で囲んでいる。
行頭の # はプロンプトであり、入力するのはそれ以降の青い背景色の部分である。
コマンドライン中の「| python -mjson.tool」は、JSONデータの整形コマンドであり、コマンド出力結果を分かりやすく表示するためのものである。

1-3-2. コマンド入力結果の表記方法

（例）

```
HTTP/1.1 201 Created
Date: Tue, 07 Feb 2023 01:38:14 GMT
Content-Type: application/json; charset=utf-8
    :
…
```

コマンド入力結果を表す箇所については、上記のように破線で囲み橙色の背景色で表記している。
また、データについても同様の形式で表記している。

2. Orion の概要

本章では、Orion の概要、公式ドキュメント、Orion が扱うデータモデルである「NGSI データモデル」、データのスコープを限定するために利用可能な「Fiware-Service/Fiware-ServicePath」について説明する。

2-1. Orion とは

Orion は、FIWARE のソフトウェアモジュール（Generic Enabler）の１つであり、ブローカーとしての機能を提供するモジュールである。Open Mobile Alliance（モバイル事業者／ベンダ中心の標準化団体）が策定した Next Generation Service Interfaces（以下、NGSI）という国際標準規格を採用している。データの登録、参照、更新、非同期通知（サブスクリプション）、リクエスト転送など、データのライフサイクル全体を管理することができる。オープンソースとして公開されていることから誰でも利用可能であり、欧州を中心として日本国内でも多数の導入実績を持つブローカーである。

図 2-1 に Orion のデータフロー図、表 2-1 に Orion の代表的な機能を記載する。

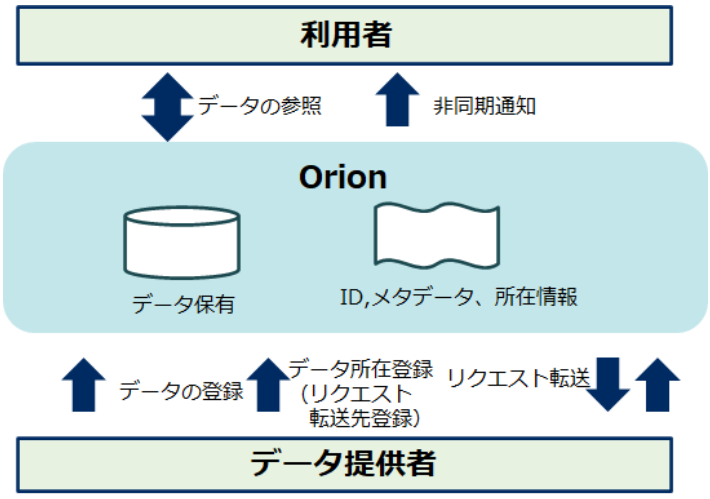

図 2-1　Orion データフロー図

表 2-1　Orion の代表的な機能

機能	概要	本書での説明
データの登録	データを蓄積する機能。	3-1
データの参照	データを参照する機能。REST API を用いて、データの所在を意識することなくデータを参照することができる。	3-2
データの更新	蓄積したデータを更新する機能。	3-3
非同期通知	データが更新された場合にあらかじめ登録された通知先へデータ更新通知を送信する機能。	3-4
リクエスト転送	リクエストを外部のデータ提供元へ転送する機能。Orion は、事前に登録されたデータ所在情報をもとに、データ提供者にデータを問い合わせる。	3-5

参考までに、エリア・データ連携基盤の推奨モジュール「ブローカー（非パーソナル)」の機能要件と Orion の機能との対応関係を表 2-2 に示す。

表 2-2　ブローカー（非パーソナル）の推奨モジュール要件と Orion の機能との対応関係

ブローカー（非パーソナル）の要件						Orion の機能（本書内の記載箇所)
データ連携目的	分類	機能	要件	必須	推奨	
データ利活用	データ参照	データ分散	データ参照の要求を受け付け、外部サービスが保持するデータを返却可能なこと	●		データの参照（3-2)
			データ利用者に対してデータの所在を隠蔽することができること		●	データの参照（3-2)
		データ蓄積	データ参照の要求を受け付け、データストア機能に蓄積されたデータを返却可能なこと		●	データの参照（3-2)
	サービス呼び出し	イベント処理	サービス呼び出しの要求を受け付け、外部サービスの処理を実行し結果を返却可能なこと（例：交通サービスでタクシーを予約)		●	リクエスト転送（3-5)
	API 仕様	API 仕様	データ利活用の利便性を考慮し、標準ルールに沿った API（REST 等）を提供可能なこと	●		共通
データ収集	データ更新	イベント処理	データ提供者からデータを受け付け、必要なサービスへデータを送信できること		●	非同期通知（3-4)
		データ蓄積	データ提供者からデータを受け付け、データストア機能に蓄積可能なこと		●	データの登録（3-1) データの更新（3-3)

なお、Orion で取り扱うデータとしては、テキストベースのデータを想定している。そのため、画像や動画などのバイナリデータは直接は取り扱わず、メタデータ（URL やサイズ、コンテンツタイプなど）のみを連携し、連携された URL をもとに各利用者がデータを取得する流れとなる。

2-2. 公式ドキュメント

本書では、NGSI v2 FIWARE Orion（動作確認バージョン：Orion v3.10.1）を対象に、代表的な機能の利用手順の概要を説明する。詳細については、以下の公式ドキュメントを確認すること。

表 2-3　公式ドキュメント一覧

No.	公式ドキュメントの URL	概要
1	https://github.com/telefonicaid/fiware-orion/blob/3.10.1/	Orion の公式ドキュメント
2	https://github.com/telefonicaid/fiware-orion/blob/3.10.1/doc/manuals.jp/user/walkthrough_apiv2.md	コンテキスト管理 API の利用手順
3	https://github.com/telefonicaid/fiware-orion/blob/3.10.1/doc/manuals.jp/user/context_providers.md	リクエスト転送に関する説明
4	https://telefonicaid.github.io/fiware-orion/archive/api/v2/	NGSIv2 API リファレンス
5	https://github.com/telefonicaid/fiware-orion/tree/3.10.1/doc/manuals.jp/user	Orion: ユーザ & プログラマ・マニュアル
6	https://github.com/telefonicaid/fiware-orion/blob/3.10.1/doc/manuals.jp/orion-api.md#general-syntax-restrictions	禁止されている文字の情報
7	https://github.com/telefonicaid/fiware-orion/blob/3.10.1/doc/manuals.jp/user/known_limitations.md	既知の制限事項
8	https://github.com/telefonicaid/fiware-orion/blob/3.10.1/doc/manuals.jp/orion-api.md#entity-service-path	Fiware-ServicePath（階層スコープ）に関する説明
9	https://github.com/telefonicaid/fiware-orion/blob/3.10.1/doc/manuals.jp/orion-api.md#multi-tenancy	Fiware-Service（マルチテナンシー）に関する説明

2-3. NGSI データモデル

Orion は、前述の通り、NGSI というインタフェースを利用してデータを操作する。NGSI は、図 2-2 に示すように、エンティティ（Entity）、属性（Attribute）、付加情報（Metadata）の 3 つの要素で構成される。NGSI によって、多様なコンテキスト情報（データ）を表すことができる。また、エンティティの定義を統一することで、様々なサービス間でのデータの相互運用が可能となる。

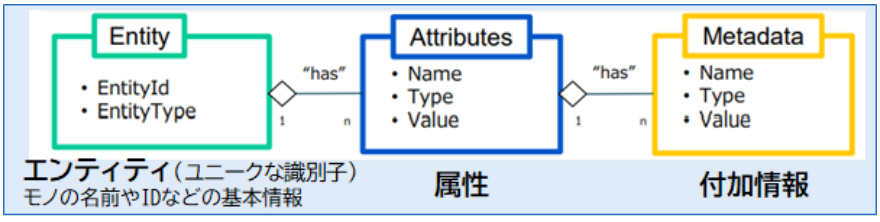

図 2-2　NGSI の構成要素

エンティティ（Entity)、属性（Attribute)、付加情報（Metadata）の概要を表 2-4 に示す。

表 2-4　NGSI のデータモデル要素

要素	概要
エンティティ（Entity）	物理的もしくは論理的なモノ（センサー、人間、部屋など）を表す概念。 EntityId（以下「id」)、EntityType（以下「type」)、属性（Attribute）を持つ。各エンティティは、id と type の組み合わせで一意に識別可能にする必要がある。また、エンティティは、当該エンティティに関連する複数の属性を持つことができる。 ※属性（Attribute）は任意。
属性（Attribute）	エンティティが持つ性質（名称、場所、情報など)。 属性名（name)、属性型（type)、属性値（value）と、付加情報（Metadata）を持つ。 ※付加情報（Metadata）は任意。
付加情報（Metadata）	属性値への付加情報（属性値の計測時刻など)。 メタデータ名（name)、メタデータ型（type)、メタデータ値（value）を持つ。

「気温（temperature)」と「気圧（pressure)」という 2 つの属性を持つ「部屋（Room)」というエンティティを NGSI で表現した場合の例を以下に示す。ここでは、部屋という概念をエンティティタイプ「Room」で、部屋の識別子をエンティティ ID の「Room1」「Room2」で表している。また、部屋のの属性として、小数型（Float）のデータを持つ「気温（temperature)」と、整数型（Integer）のデータを持つ「気圧（pressure)」の 2 つの属性を定義している。

表 2-5　NGSI のデータモデルの例

Entity				
id	type	Attribute		
		name	type	value
Room1	Room	temperature	Float	23.0
		pressure	Integer	1020
Room2	Room	temperature	Float	21.0
		pressure	Integer	1010

2-4. Fiware-Service/Fiware-ServicePath

Fiware-Service 及び Fiware-ServicePath を利用することにより、データのスコープを限定することができる。Fiware-Service はデータをグループ化する概念であり、Fiware-ServicePath は Fiware-Service 内のデータの階層構造を定義する概念である（図 2-3）。

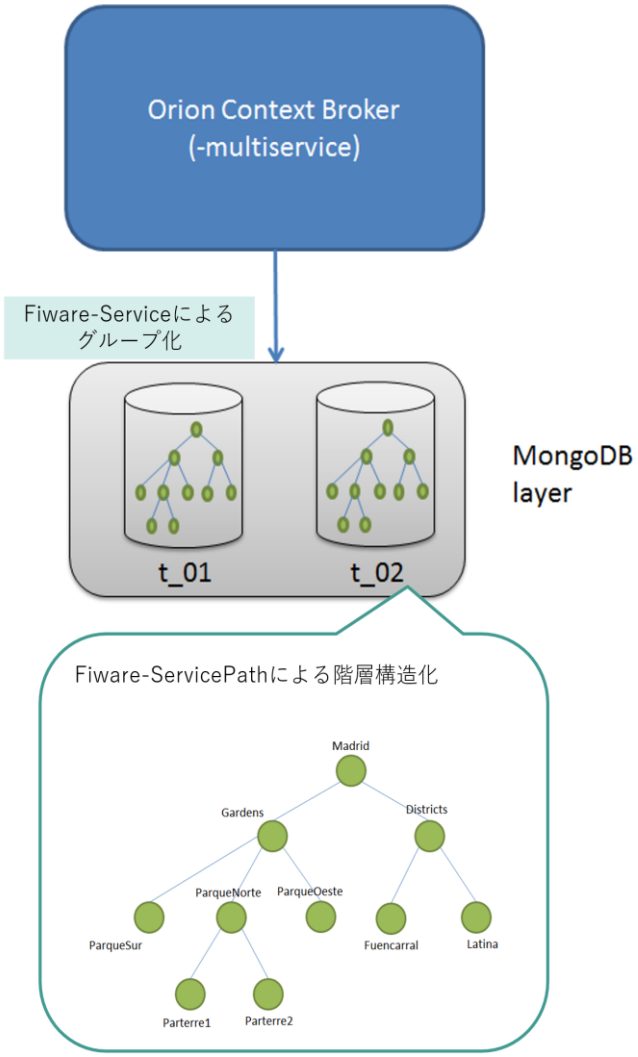

図 2-3　Fiware-Service 及び Fiware-ServicePath

Orion API 呼び出し時における HTTP ヘッダのオプションに、Fiware-Service 及び Fiware-ServicePath を指定することができる（図 2-4）。利用可能文字や制限事項については、表 2-6 を参照のこと。

```
Fiware-Service: t_02
Fiware-ServicePath: /Madrid/Gardens/ParqueNorte/Parterre1
```

図 2-4　Fiware-Service 及び Fiware-ServicePath の指定例

表 2-6　Fiware-Service 及び Fiware-ServicePath の利用可能文字・制限事項

	利用可能文字	制限事項
Fiware-Service	・英数字 ・"_"（アンダースコア）	・最大 50 文字
Fiware-ServicePath		・"/"（スラッシュ）で開始 ・最大 10 階層 ・各階層で 1 文字以上 50 文字以下 ・最大 10 個までカンマ(,)区切りで指定可能 ただし、更新関係の API の場合は、1 個 ・末尾の"/"（スラッシュ）は削除

Fiware-Service 及び Fiware-ServicePath によって、マルチテナンシーを実現することができる。マルチテナンシーとは、複数のサービス/テナント（組織）がクラウド上の同じアプリケーションやリソースを共有するアーキテクチャのことである。Fiware-Service では、あるサービス/テナントのエンティティ/属性/サブスクリプションが、他のサービス/テナントに対して影響しないことを保証する。この機能を利用し、利用するデータをサービス/テナント単位で分離することが可能。また、Fiware-ServicePath で定義した階層構造を利用して、データに対する公開/非公開、認可/不認可を設定することができる。

Fiware-Service 及び Fiware-ServicePath の詳細については、表 2-3 No.8,9 の公式ドキュメントを確認のこと。

3. 利用手順

以下では、Orion の基本的な利用手順として、データの登録、参照、更新、非同期通知、リクエスト転送について説明する。詳細な利用手順、仕様に関しては、「表 2-3　公式ドキュメント一覧」に記載の公式サイトを確認のこと。

3-1. データの登録

以下の Room1 と Room2 に気温と気圧の２つの属性のデータを登録する。

表 3-1　登録するデータ

Entity				
id	type	Attribute		
		name	type	value
Room1	Room	temperature	Float	23.0
		pressure	Integer	1020
Room2	Room	temperature	Float	21.0
		pressure	Integer	1010

以下のコマンドを実行し、Room1 のデータを登録する。

```
# curl localhost:1026/v2/entities -i -S -H 'Content-Type: application/json' ¥
    -d @- <<EOF
{
  "id": "Room1",
  "type": "Room",
  "temperature": {
    "value": 23.0,
    "type": "Float"
  },
  "pressure": {
    "value": 1020,
    "type": "Integer"
  }
}
EOF
```

・ localhost:1026　　：Orion のホスト名：ポート番号

登録に成功すると、以下のレスポンスが返却される。

```
HTTP/1.1 201 Created
Date: Wed, 27 Dec 2023 02:52:46 GMT
Fiware-Correlator: 03b489bc-a463-11ee-956f-0242ac110003
Location: /v2/entities/Room1?type=Room
Content-Length: 0
```

同様な方法で Room2 のデータを登録する。

```
# curl localhost:1026/v2/entities -i -S -H 'Content-Type: application/json' ¥
    -d @- <<EOF
{
  "id": "Room2",
  "type": "Room",
  "temperature": {
    "value": 21.0,
    "type": "Float"
  },
  "pressure": {
    "value": 1010,
    "type": "Integer"
  }
}
EOF
```

- localhost:1026　　：Orion のホスト名：ポート番号

3-2. データの参照

(1) Room1 を参照する

以下のコマンドを実行して Room1 の情報を参照する。

```
# curl localhost:1026/v2/entities/Room1 -s -S -H 'Accept: application/json' ¥
    | python -mjson.tool
```

- localhost:1026　　：Orion のホスト名：ポート番号
- Room1　　：参照するエンティティ

参照に成功すると、以下のレスポンスが返却される。

```
{
    "id": "Room1",
    "pressure": {
        "metadata": {},
        "type": "Integer",
        "value": 1020
    },
    "temperature": {
        "metadata": {},
        "type": "Float",
        "value": 23
    },
    "type": "Room"
}
```

(2) 全データを参照する

以下のコマンドを実行して全ての情報を参照する。

```
# curl localhost:1026/v2/entities -s -S -H 'Accept: application/json' ¥
    | python -mjson.tool
```

・ localhost:1026 ：Orion のホスト名：ポート番号

参照に成功すると、以下のレスポンスが返却される。

```
[
    {
        "id": "Room1",
        "pressure": {
            "metadata": {},
            "type": "Integer",
            "value": 1020
        },
        "temperature": {
            "metadata": {},
            "type": "Float",
            "value": 23
        },
        "type": "Room"
    },
    {
        "id": "Room2",
        "pressure": {
            "metadata": {},
            "type": "Integer",
            "value": 1010
        },
        "temperature": {
            "metadata": {},
            "type": "Float",
            "value": 21
        },
        "type": "Room"
    }
]
```

(3)フィルタしたデータを参照する（idPattern）

idPattern パラメータを使用して、参照するデータをフィルタすることができる。

以下の例では Room2～5 に前方一致するデータを参照する。

```
# curl localhost:1026/v2/entities?idPattern=^Room[2-5] -g -s -S ¥
    -H 'Accept: application/json' | python -mjson.tool
```

- localhost:1026　　：Orion のホスト名：ポート番号
- idPattern=^Room[2-5]　　：フィルタ条件

参照に成功すると、条件に一致する以下のレスポンス（Room2 のみ）が返却される。

```
[
    {
        "id": "Room2",
        "pressure": {
            "metadata": {},
            "type": "Integer",
            "value": 1010
        },
        "temperature": {
            "metadata": {},
            "type": "Float",
            "value": 21
        },
        "type": "Room"
    }
]
```

(4) フィルタしたデータを参照する（q パラメータ）

q パラメータを使用して、参照するデータをフィルタすることができる。

以下の例では気温が 22 度より大きいデータを参照する。

```
# curl 'localhost:1026/v2/entities?q=temperature>22' -s -S ¥
    -H 'Accept: application/json' | python -mjson.tool
```

- **localhost:1026** ：Orion のホスト名：ポート番号
- **temperature>22** ：フィルタ条件

参照に成功すると、条件に一致する以下のレスポンス（Room1 のみ）が返却される。

```
[
    {
        "id": "Room1",
        "pressure": {
            "metadata": {},
            "type": "Integer",
            "value": 1020
        },
        "temperature": {
            "metadata": {},
            "type": "Float",
            "value": 23
        },
        "type": "Room"
    }
]
```

(5)分割してデータを参照する（ページング）

Orion では一度に参照できるデータ件数が限られている。以下のパラメータを使用し、ページ分割して参照することで、大量のデータを参照することが出来る。

- limit　：返却するデータの最大数を指定する（デフォルト：20、最大値：1000）
- offset　：先頭から指定された件数スキップしたデータを返却する（デフォルト：0）
- count　：レスポンスの Fiware-Total-Count ヘッダに総データ数を設定する(オプション)

（使い方の例）

① 先頭ページのデータとデータ件数を取得する

パラメータに limit と count オプションを指定して参照する。

以下の例では先頭から最大 20 件のデータを参照し、さらに、総データ件数を取得する。

```
#curl 'localhost:1026/v2/entities?limit=20&options=count' -i -S ¥
    -H 'Accept: application/json'
```

- **localhost:1026**　：Orion のホスト名：ポート番号
- **limit=20**　：最大取得件数
- **options=count**　：総データ数設定オプション

参照に成功すると、以下のレスポンスが返却される。

```
HTTP/1.1 200 OK
Date: Wed, 27 Dec 2023 03:03:27 GMT
Fiware-Correlator: 8178a2f6-a464-11ee-a54d-0242ac110003
Fiware-Total-Count: 30
Content-Type: application/json
Content-Length: 2884

[data(max20)]]
```

- *[data(max20)]*　：1 件目から最大 20 件のデータが返却される。

Fiware-Total-Count ヘッダに総データ件数が設定される。

② ２ページ目のデータを取得する

パラメータに offset と limit を指定して２ページ目のデータを参照する。

以下の例では 21 件目のデータから最大 20 件のデータを参照する。

```
#curl 'localhost:1026/v2/entities?offset=20&limit=20' -i -S ¥
    -H 'Accept: application/json'
```

- localhost:1026　　：Orion のホスト名：ポート番号
- offset=20　　：先頭から取得をスキップするデータ件数
- limit=20　　：最大取得件数

参照に成功すると、以下のレスポンスが返却される。

```
HTTP/1.1 200 OK
Date: Wed, 27 Dec 2023 03:04:52 GMT
Fiware-Correlator: b47b9122-a464-11ee-a285-0242ac110003
Content-Type: application/json
Content-Length: 1449

[data(max20)]]
```

- [data(max20)]　　：21 件目から最大 20 件のデータが返却される

③ Ｎページ目のデータを取得する

パラメータに offset と limit を指定してＮページ目のデータを参照する。

以下の例では先頭から NN データ目から最大 20 件のデータを参照する。

```
#curl 'localhost:1026/v2/entities?offset=NN&limit=20' -i -S ¥
    -H 'Accept: application/json'
```

- localhost:1026　　：Orion のホスト名：ポート番号
- offset=NN　　：先頭から取得をスキップするデータ件数
- limit=20　　：最大取得件数

参照に成功すると、以下のレスポンスが返却される。

```
HTTP/1.1 200 OK
Date: Wed, 27 Dec 2023 03:07:16 GMT
Fiware-Correlator: 0a264130-a465-11ee-bdbe-0242ac110003
Content-Type: application/json
Content-Length: 2893

[data(max20)]]
```

- [data(max20)]　　：NN+1 件目から最大 20 件のデータが返却される

④ ③を繰り返すことで、20 件ずつ全てのデータを取得することが出来る。

3-3. データの更新

以下のコマンドを実行し、Room1 のデータを更新する。

```
#curl localhost:1026/v2/entities/Room1/attrs -i -S ¥
    -H 'Content-Type: application/json' -X PATCH -d @- <<EOF
{
  "temperature": {
    "value": 26.5,
    "type": "Float"
  },
  "pressure": {
    "value": 1025,
    "type": "Integer"
  }
}
EOF
```

- localhost:1026　　：Orion のホスト名：ポート番号
- Room1/attrs　　：更新するエンティティ／属性

更新に成功すると、以下のレスポンスが返却される。

```
HTTP/1.1 204 No Content
Date: Wed, 27 Dec 2023 03:09:09 GMT
Fiware-Correlator: 4d5f4708-a465-11ee-b28c-0242ac110003
```

3-4. 非同期通知

Orion では、サブスクリプションを利用することで、特定のコンテキストデータのデータ更新の通知を受け取ることが出来る。

例として、Room1 の気温の更新通知を収集アプリ 1 で受け取る場合について説明する。

図 3-1　非同期通知の構成と流れ

①収集アプリ 1 から Orion にサブスクリプションを要求する（Room1 の気温変化時）。

```
# curl -v localhost:1026/v2/subscriptions -i -S ¥
    -H 'Content-Type: application/json' -d @- << EOF
{
  "description": "A subscription to get info about Room1",
  "subject": {
    "entities": [
      {
        "id": "Room1",
        "type": "Room"
      }
    ],
    "condition": {
      "attrs": [
        "temperature"
      ]
    }
  },
  "notification": {
    "http": {
      "url": "http://example.collection.com:1028/app1"
    },
    "attrs": [
      "temperature"
    ]
  },
  "expires": "2038-12-01T09:00:00.00Z"
}
EOF
```

- localhost:1026　　：Orion のホスト名：ポート番号
- http://example.collection.com:1028/app1　　：収集アプリ 1 へ通知するための URL

登録に成功すると、以下のレスポンスが返却される。

```
HTTP/1.1 201 Created
Date: Wed, 27 Dec 2023 03:18:48 GMT
Fiware-Correlator: a6b4f4be-a466-11ee-972d-0242ac110003
Location: /v2/subscriptions/658b97981749d1d5720ec0ea
Content-Length: 0
```

② Room1 センサーアプリから Room1 の気温を更新する。

```
# curl localhost:1026/v2/entities/Room1/attrs -s -S -H 'Content-Type:
application/json' -X PATCH -d @- <<EOF
{
  "temperature": {
    "value": 27.5,
    "type": "Float"
  }
}
EOF
```

- localhost:1026　　：Orion のホスト名：ポート番号

③ 収集アプリ 1 に Room1 の気温が通知される。

Orion から収集アプリ 1 に下記のリクエストが発行され、収集アプリ 1 に Room1 の気温が通知される。

```
POST http://example.collection.com:1028/app1
Content-Length: 141
User-Agent: orion/3.10.1 libcurl/7.74.0
Ngsiv2-Attrsformat: normalized
Host:example.collection.com:1028
Accept: application/json
Content-Type: application/json; charset=utf-8
Fiware-Correlator: 079b55da-a68d-11ed-8da5-0242ac110005; cbnotif=1

{
    "data": [
        {
            "id": "Room1",
            "temperature": {
                "metadata": {},
                "type": "Float",
                "value": 27.5
            },
            "type": "Room"
        }
    ],
    "subscriptionId": "63e1b1c7657f6c75b605100e"
}
```

・ http://example.collection.com:1028/app1　　：収集アプリ 1 へ通知するための URL

3-5. リクエスト転送

Orion は、データの所在を管理することで、Orion 自身がデータを保持していない場合でも、データを保持しているコンテキスト・プロバイダへリクエストを転送し、該当データにアクセスすることができる。

クライアントから Orion にデータの要求があった場合、Orion がコンテキスト・プロバイダからデータを取得してデータをクライアントへ返す。(図 3-2)

このリクエスト転送の仕組みにより、クライアントはデータの所在を意識することなくデータを取得することが出来る。

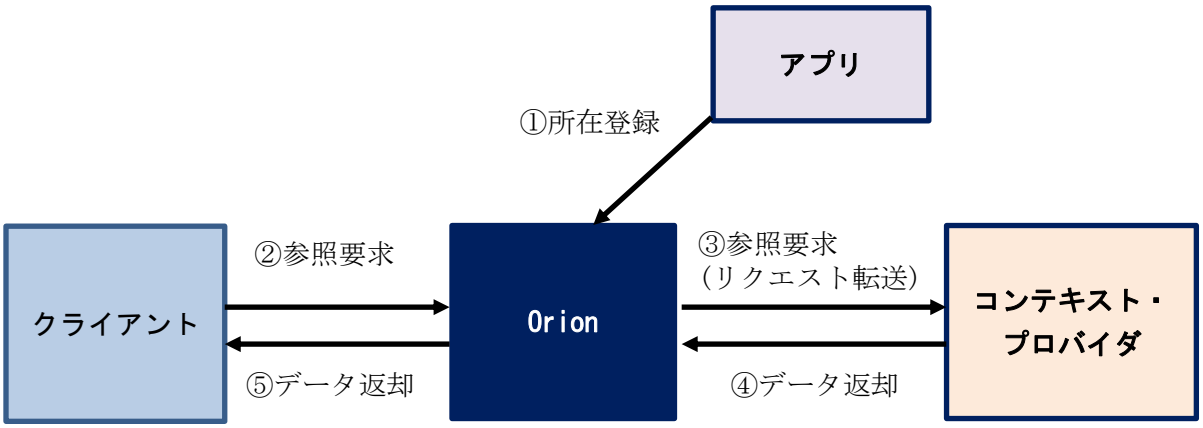

図 3-2　リクエスト転送の流れ

①アプリから Orion にコンテキスト・プロバイダを所在登録する。
　以下の例では、Street 型の Street4 エンティティの気温属性について所在登録する。

```
# curl -v localhost:1026/v2/registrations -i -S -H 'Content-Type: application/json' ¥
        -H 'Accept: application/json' -d @-  <<EOF
{
  "dataProvided": {
    "entities": [
      {
        "id": "Street4",
        "type": "Street"
      }
    ],
    "attrs": [
      "temperature"
    ]
  },
  "provider": {
    "http": {
      "url": "http://sensor.mycity.com/v2"
    }
  }
}
EOF
```

- localhost:1026　　：Orion のホスト名：ポート番号
- http://sensor.mycity.com/v2　　：コンテキスト・プロバイダの URL

②クライアントアプリから Orion にデータ参照要求を出す。
　以下の例では、Street4 エンティティの気温を参照する。

```
# curl localhost:1026/v2/entities/Street4/attrs/temperature?type=Street -s -S ¥
        -H 'Accept: application/json' | python -mjson.tool
```

- localhost:1026　　：Orion のホスト名：ポート番号

　なお、所在登録されたデータ（すなわち、分散管理されたデータ）を参照する場合には、エンティティ ID を直接指定して参照要求を出すことを推奨する。

③Orion からコンテキスト・プロバイダにデータ参照要求の転送が行われる。

```
POST http://sensor.mycity.com/v2/op/query

{
  "entities": [
    {
      "id": "Street4",
      "type": "Street"
    }
  ],
  "attrs": [
    "temperature"
  ]
}
```

・ http://sensor.mycity.com/v2 ：コンテキスト・プロバイダの URL

④ ③の結果、コンテキスト・プロバイダは Orion にデータを返却する。

```
[
  {
    "id": "Street4",
    "type": "Street",
    "temperature": {
      "metadata": {},
      "value": 16,
      "type": "Float"
    }
  }
]
```

⑤ Orion からクライアントアプリにデータを転送する。

```
{
   "metadata": {},
   "value": 16,
   "type": "Float”
}
```

APPENDIX

付録A.　　API 一覧

NGSI v2 FIWARE Orion の API 一覧を表 A-1 に示す。

表 A-1　NGSI v2 FIWARE Orion の API 一覧

No.	API 名	HTTP	説明
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