


Kong GatewayとFIWARE Orionの連携手順書 (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 (Kong Gateway 2.8.0、FIWARE Orion 3.6.0 を対象に作成)	2022/07/01
1.1.0	Kong Gateway 3.1.0、FIWARE Orion 3.7.0 向けに手順書を改版 1-1 に「  1-1 本書の構成」を追加	2023/03/01
1.2.0	Kong Gateway 3.4.0、FIWARE Orion 3.10.1 向けに手順書を改版	2024/03/01

目次

1.	はじめに	4
1-1.	概要	4
1-2.	参考資料	5
1-3.	前提条件	5
1-4.	構成図	5
1-5.	表記方法	6
1-5-1.	コマンドの表記方法	6
1-5-2.	コマンド入力結果の表記方法	6
2.	事前準備	7
3.	設定	10
4.	動作確認	14
4-1.	Orion のエンティティの作成	14
4-2.	Orion のエンティティの取得	15
4-3.	Orion のエンティティの更新	15

1. はじめに

本書は、API ゲートウェイとして利用するソフトウェアの Kong Gateway とブローカー（非パーソナル）として利用するソフトウェアの FIWARE Orion（正式名称 Orion Context Broker）（以下、「Orion」）を連携する手順を記載・説明するものである。

1-1. 概要

本書では以下の手順で Kong Gateway と Orion を連携する。

1. Kong Gateway コンテナから Orion コンテナにアクセスできるようにするため、コンテナを起動しなおす。（2章）
2. Kong Gateway に Orion をサービスとして登録しルーティングの設定を行う。（3章）
3. Kong Gateway 経由で Orion のエンティティが作成できることを確認する。（4-1節）
4. Kong Gateway 経由で Orion のエンティティが取得できることを確認する。（4-2節）
5. Kong Gateway 経由で Orion のエンティティが更新できることを確認する。（4-3節）

2章では、事前準備として上記手順1について説明する。

3～4章では、上記手順2～5について説明する。3～4章の説明内容の関係性を図1-1に示す。

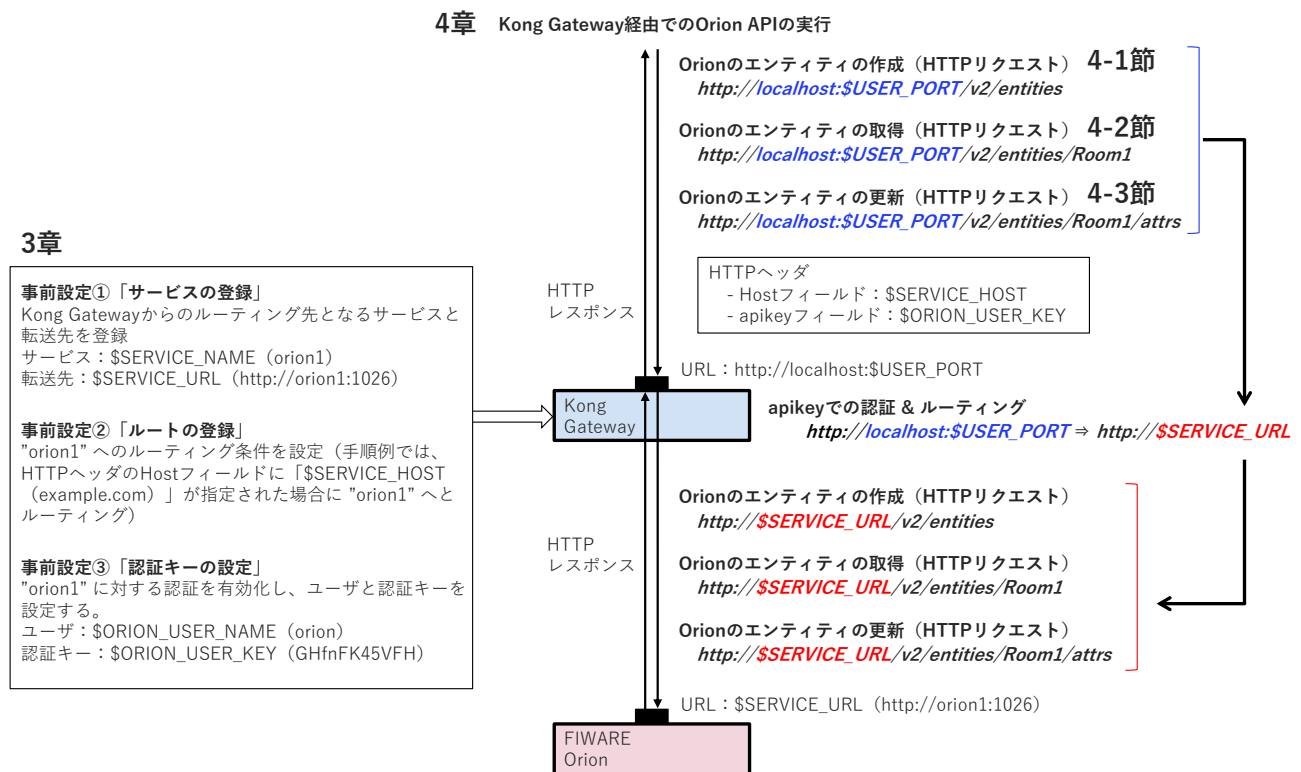


図 1-1 本書の構成 (3章及び4章)

1-2. 参考資料

本書では、以下の参考資料を参照している。

[https://github.com/telefonicaid/fiware-](https://github.com/telefonicaid/fiware-orion/blob/3.10.1/doc/manuals.jp/user/walkthrough_apiv2.md#context-management)

[orion/blob/3.10.1/doc/manuals.jp/user/walkthrough_apiv2.md#context-management](https://github.com/telefonicaid/fiware-orion/blob/3.10.1/doc/manuals.jp/user/walkthrough_apiv2.md#context-management)

<https://docs.konghq.com/gateway/3.4.x/>

1-3. 前提条件

本書を用いて構築する際の前提条件は以下のとおりである。

- Kong Gateway 構築手順書を用いてインストールされていること。
- FIWARE Orion 構築手順書を用いてインストールされていること。
- Kong Gateway と Orion は同じ OS 上にインストールされていること。
- インターネットへアクセス可能であること。
- root ユーザーでログインできること。

1-4. 構成図

本書で連携させる Kong Gateway と Orion の構成図は以下の通りである。(図 1-2)

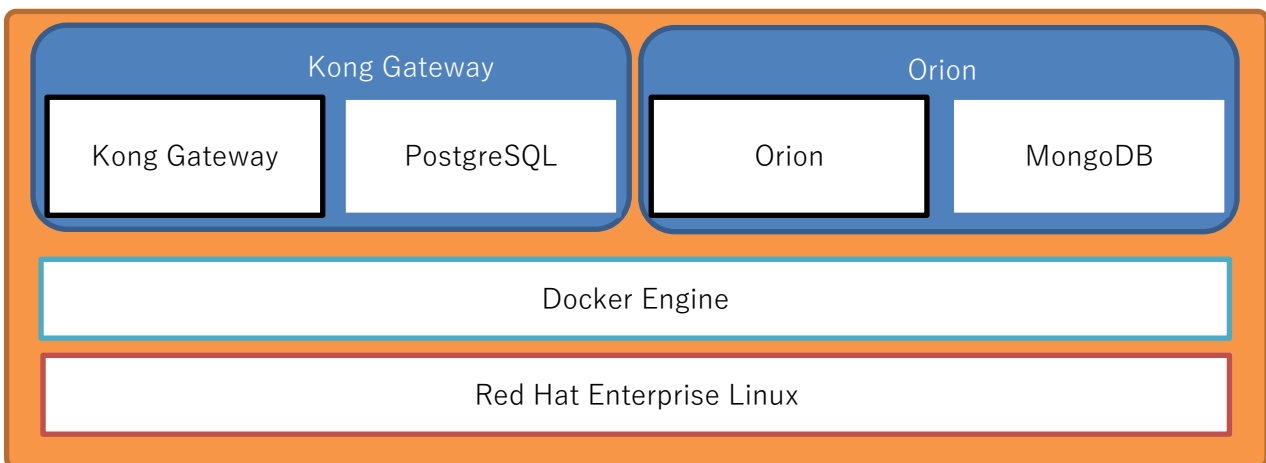


図 1-2 構成図

1-5. 表記方法

1-5-1. 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。

1-5-2. コマンド入力結果の表記方法

(例)

```
HTTP/1.1 201 Created
Date: Wed, 27 Dec 2023 07:45:51 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。

2. 事前準備

本章では Kong Gateway と Orion を連携する設定について記載・説明する。

はじめに連携させるマシン (Red Hat Enterprise Linux 7.9) へ root ユーザでログインしておくこと。

Kong Gateway、FIWARE Orion 各構築手順書での個別の構築ではお互いのコンテナ間で疎通できない状態となっている。そのため、構築・起動済みの場合、いったん関連するコンテナをすべて削除してから作業を行うこととする。

起動中の Docker コンテナを確認する。

```
# docker ps
```

起動中のコンテナの名称 (NAME 列) をメモしておく (以下の例では kong, kong-database, orion, mongodb)。

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS
PORTS				
NAMES				
c9a1a2fac33b	fiware/orion:3.10.1	"/usr/bin/contextBro..."	3 minutes ago	Up 3 minutes
(healthy)		0.0.0.0:1026->1026/tcp,		:::1026->1026/tcp
orion1				
a6f85be436bb	mongo:4.4	"docker-entrypoint.s..."	4 minutes ago	Up 4 minutes
27017/tcp				
mongodb				
b336b940662b	kong:3.4.0	"/docker-entrypoint..."	15 minutes ago	Up 15 minutes
(healthy)	0.0.0.0:8000-8001->8000-8001/tcp, :::8000-8001->8000-8001/tcp, 0.0.0.0:8443-8444->8443-8444/tcp, :::8443-8444->8443-8444/tcp	kong		
b799acc02502	postgres:12.17	"docker-entrypoint.s..."	17 minutes ago	Up 17 minutes
0.0.0.0:5432->5432/tcp,				:::5432->5432/tcp
kong-database				

起動中のコンテナをいったん削除する。起動中のコンテナが存在しなかった場合はスキップする。

```
# docker rm -f <orion の CONTAINER ID>  
# docker rm -f <mongo の CONTAINER ID>  
# docker rm -f <kong の CONTAINER ID>  
# docker rm -f <kong-database の CONTAINER ID>
```

MongoDB を起動する。

```
# docker run --name mongodb -d mongo:4.4
```

Orion を起動する。

```
# docker run -d --name orion1 --link mongodb:mongodb -p 1026:1026 fiware/orion:3.10.1 -dbhost  
mongodb
```

パラメータを準備する。

```
# cat << EOF > ~/ENV.sh  
KONG_HOST=kong  
KONG_VERSION=3.4.0  
USER_PORT=8000  
USER_HTTPS_PORT=8443  
ADMIN_PORT=8001  
ADMIN_HTTPS_PORT=8444  
DB_HOST=kong-database  
DB_VERSION=12.17  
DB_PORT=5432  
DB_USER=kong  
DB_PASSWD=kong  
DB_NAME=kong  
SERVICE_NAME=orion1  
SERVICE_URL=http://orion1:1026  
SERVICE_HOST=example.com  
ORION_USER_NAME=orion  
ORION_USER_KEY=GHfnFK45VFH  
EOF
```

各パラメータの詳細は以下の通りである。

- KONG_HOST: Kong Gateway の Docker コンテナの名前を指定する。
- KONG_VERSION: Kong Gateway の Docker イメージのバージョンを指定する。
- USER_PORT: Kong Gateway の利用者向け API のポートを指定する。
- USER_HTTPS_PORT: Kong Gateway の利用者向け API のポート (SSL) を指定する。
- ADMIN_PORT: Kong Gateway の管理用 API のポートを指定する。
- ADMIN_HTTPS_PORT: Kong Gateway の管理用 API のポート (SSL) を指定する。
- DB_HOST: PostgreSQL の Docker コンテナの名前を指定する。
- DB_VERSION: PostgreSQL の Docker イメージのバージョンを指定する。
- DB_PORT: PostgreSQL のポートを指定する。
- DB_USER: PostgreSQL の接続ユーザ名を指定する。
- DB_PASSWD: PostgreSQL の接続パスワードを指定する。
- DB_NAME: PostgreSQL のデータベース名を指定する。
- SERVICE_NAME: Kong Gateway に関連付ける Orion のサービス名称を指定する。
- SERVICE_URL: Kong Gateway に関連付ける Orion の URL を指定する。
- SERVICE_HOST: Kong Gateway に関連付けた Orion を呼び出す際のホスト名を指定する。
- ORION_USER_NAME: Kong Gateway 認証プラグインで Orion に認証する際のユーザ名を指定する。
- ORION_USER_KEY: Kong Gateway 認証プラグインで Orion に認証する際の認証キーを指定する。

パラメータを読み込む。

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


PostgreSQL を起動する。

```
# docker run -d ¥
  --name $DB_HOST ¥
  -p $DB_PORT:$DB_PORT ¥
  -e "POSTGRES_USER=$DB_USER" ¥
  -e "POSTGRES_PASSWORD=$DB_PASSWD" ¥
  -e "POSTGRES_DB=$DB_NAME" ¥
  postgres:$DB_VERSION
```

一時的な Kong Gateway コンテナを用いて PostgreSQL のマイグレーションを行う。

```
# docker run --rm ¥
  --link $DB_HOST:$DB_HOST ¥
  -e "KONG_DATABASE=postgres" ¥
  -e "KONG_PG_HOST=$DB_HOST" ¥
  -e "KONG_PG_USER=$DB_USER" ¥
  -e "KONG_PG_PASSWORD=$DB_PASSWD" ¥
  -e "KONG_CASSANDRA_CONTACT_POINTS=$DB_HOST" ¥
  $KONG_HOST:$KONG_VERSION kong migrations bootstrap
```

Kong Gateway コンテナを起動する。

```
# docker run -d ¥
  --name $KONG_HOST ¥
  --link $DB_HOST:$DB_HOST ¥
  --link $SERVICE_NAME:$SERVICE_NAME ¥
  -e "KONG_DATABASE=postgres" ¥
  -e "KONG_PG_HOST=$DB_HOST" ¥
  -e "KONG_PG_USER=$DB_USER" ¥
  -e "KONG_PG_PASSWORD=$DB_PASSWD" ¥
  -e "KONG_CASSANDRA_CONTACT_POINTS=$DB_HOST" ¥
  -e "KONG_PROXY_ACCESS_LOG=/dev/stdout" ¥
  -e "KONG_ADMIN_ACCESS_LOG=/dev/stdout" ¥
  -e "KONG_PROXY_ERROR_LOG=/dev/stderr" ¥
  -e "KONG_ADMIN_ERROR_LOG=/dev/stderr" ¥
  -e "KONG_ADMIN_LISTEN=0.0.0.0:$ADMIN_PORT, 0.0.0.0:$ADMIN_HTTPS_PORT ssl" ¥
  -p $USER_PORT:$USER_PORT ¥
  -p $USER_HTTPS_PORT:$USER_HTTPS_PORT ¥
  -p $ADMIN_PORT:$ADMIN_PORT ¥
  -p $ADMIN_HTTPS_PORT:$ADMIN_HTTPS_PORT ¥
  $KONG_HOST:$KONG_VERSION
```

3. 設定

Kong Gateway から Orion の API につなぐため、Kong Gateway にサービスを登録する。

```
# curl -i -X POST ¥  
  --url http://localhost:$ADMIN_PORT/services/ ¥  
  --data "name=$SERVICE_NAME" ¥  
  --data "url=$SERVICE_URL"
```

201 Created が返却されることを確認する。

```
HTTP/1.1 201 Created  
Date: Fri, 29 Dec 2023 05:54:25 GMT  
Content-Type: application/json; charset=utf-8  
:
```

Kong Gateway がアクセスされたときにどのサービスを呼び出すかを定めるため、ルーティングを追加する。

以下の例では、リクエストのホスト名が Orion サービスホスト名と一致した時となる。

```
# curl -i -X POST ¥  
  --url http://localhost:$ADMIN_PORT/services/$SERVICE_NAME/routes ¥  
  --data "hosts[]=$SERVICE_HOST"
```

201 Created が返却されることを確認する。

```
HTTP/1.1 201 Created  
Date: Fri, 29 Dec 2023 05:56:25 GMT  
Content-Type: application/json; charset=utf-8  
:
```

ルーティングの有効性を確認するため、Kong Gateway のサービスにアクセスを行う。

```
# curl -i -X GET ¥  
  --url http://localhost:$USER_PORT/version ¥  
  --header "Host: $SERVICE_HOST"
```

Kong Gateway のサービスにアクセスした際の Orion 情報が返却されることを確認する。

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 745
Connection: keep-alive
Date: Fri, 29 Dec 2023 05:57:47 GMT
Fiware-Correlator: 311eebec-a60f-11ee-8652-0242ac110003
X-Kong-Upstream-Latency: 1
X-Kong-Proxy-Latency: 0
Via: kong/3.4.0

{
  "orion" : {
    "version" : "3.10.1",
    "uptime" : "0 d, 0 h, 25 m, 52 s",
    "git_hash" : "9a80e06abe7f690901cf1586377acec02d40e303",
    "compile_time" : "Mon Jun 12 16:55:20 UTC 2023",
    "compiled_by" : "root",
    "compiled_in" : "buildkitsandbox",
    "release_date" : "Mon Jun 12 16:55:20 UTC 2023",
    "machine" : "x86_64",
    "doc" : "https://fiware-orion.rtdf.io/en/3.10.1/",
    :
```

このままでは無制限にアクセスできてしまうため、認証プラグインを追加する。

```
# curl -i -X POST ¥
--url http://localhost:$ADMIN_PORT/services/$SERVICE_NAME/plugins/ ¥
--data 'name=key-auth'
```

201 Created が返却されることを確認する。

```
HTTP/1.1 201 Created
Date: Fri, 29 Dec 2023 06:00:44 GMT
Content-Type: application/json; charset=utf-8
:
```

認証が有効であることを確認する。

```
# curl -i -X GET ¥  
--url http://localhost:$USER_PORT/version ¥  
--header "Host: $SERVICE_HOST"
```

認証が有効になっているために、Kong Gateway のサービスにアクセスできず「No API key found in request」と表示され、401 認証エラーが返却されることを確認する。

```
HTTP/1.1 401 Unauthorized  
Date: Fri, 29 Dec 2023 06:03:17 GMT  
Content-Type: application/json; charset=utf-8  
:  
{  
  "message": "No API key found in request"  
}
```

認証を通過できるようにするため、コンシューマ（利用者ユーザ）を追加する。

```
# curl -i -X POST ¥  
--url http://localhost:$ADMIN_PORT/consumers/ ¥  
--data "username=$ORION_USER_NAME"
```

201 Created が返却されることを確認する。

```
HTTP/1.1 201 Created  
Date: Fri, 29 Dec 2023 06:04:46 GMT  
Content-Type: application/json; charset=utf-8  
:
```

コンシューマ（利用者ユーザ）に認証キーを設定する。

```
# curl -i -X POST ¥  
--url http://localhost:$ADMIN_PORT/consumers/$ORION_USER_NAME/key-auth/ ¥  
--data "key=$ORION_USER_KEY"
```

201 Created が返却されることを確認する。

```
HTTP/1.1 201 Created  
Date: Fri, 29 Dec 2023 06:09:43 GMT  
Content-Type: application/json; charset=utf-8  
:
```

認証が通過できることを確認する。

```
# curl -i -X GET ¥
--url http://localhost:$USER_PORT/version ¥
--header "Host: $SERVICE_HOST" ¥
--header "apikey: $ORION_USER_KEY"
```

```
HTTP/1.1 200 OK
Content-Type: application/json
Content-Length: 745
Connection: keep-alive
Date: Fri, 29 Dec 2023 06:11:13 GMT
Fiware-Correlator: 116f99ac-a611-11ee-86bb-0242ac110003
X-Kong-Upstream-Latency: 0
X-Kong-Proxy-Latency: 6
Via: kong/3.4.0

{
  "orion" : {
    "version" : "3.10.1",
    "uptime" : "0 d, 0 h, 39 m, 18 s",
    "git_hash" : "9a80e06abe7f690901cf1586377acec02d40e303",
    "compile_time" : "Mon Jun 12 16:55:20 UTC 2023",
    "compiled_by" : "root",
    "compiled_in" : "buildkitsandbox",
    "release_date" : "Mon Jun 12 16:55:20 UTC 2023",
    "machine" : "x86_64",
    "doc" : "https://fiware-orion.rtdf.io/en/3.10.1/",
    :
```

認証キーを指定することで、Kong Gateway のサービスにアクセス可能となり、Orion 情報が返却されることを確認する。

4. 動作確認

本章では、正しく連携ができているか、Kong Gateway 経由で Orion の API を実行し確認を行う。

4-1. Orion のエンティティの作成

ここでは、Room1 という名前のエンティティを作成する。

```
# curl -i -X POST ¥
  --url http://localhost:$USER_PORT/v2/entities ¥
  --header "Content-Type: application/json" ¥
  --header "Host: $SERVICE_HOST" ¥
  --header "apikey: $ORION_USER_KEY" ¥
  --data @- <<EOF
{
  "id": "Room1",
  "type": "Room",
  "temperature": {
    "value": 23,
    "type": "Float"
  },
  "pressure": {
    "value": 720,
    "type": "Integer"
  }
}
EOF
```

201 Created が返却されることを確認する。

```
HTTP/1.1 201 Created
Content-Length: 0
Connection: keep-alive
Date: Fri, 29 Dec 2023 06:15:18 GMT
:
```

4-2. Orion のエンティティの取得

次に、前章で作成したエンティティが確認できるかどうか取得する。

```
# curl -s -X GET ¥
--url http://localhost:$USER_PORT/v2/entities/Room1 ¥
--header "Accept: application/json" ¥
--header "Host: $SERVICE_HOST" ¥
--header "apikey: $ORION_USER_KEY" | python -mjson.tool
```

エンティティの情報が取得できることを確認する。

```
{
  "id": "Room1",
  "pressure": {
    "metadata": {},
    "type": "Integer",
    "value": 720
  },
  "temperature": {
    "metadata": {},
    "type": "Float",
    "value": 23
  },
  "type": "Room"
}
```

4-3. Orion のエンティティの更新

最後に、Room1 エンティティの値を更新する。

```
# curl -i -X PATCH ¥
--url http://localhost:$USER_PORT/v2/entities/Room1/attrs ¥
--header "Content-Type: application/json" ¥
--header "Host: $SERVICE_HOST" ¥
--header "apikey: $ORION_USER_KEY" ¥
--data @- <<EOF
{
  "temperature": {
    "value": 26.5,
    "type": "Float"
  },
  "pressure": {
    "value": 763,
    "type": "Float"
  }
}
EOF
```

204 No Content が返却されることを確認する。

```
HTTP/1.1 204 No Content
Connection: keep-alive
Date: Fri, 29 Dec 2023 06:18:19 GMT
:
```

更新したエンティティを取得して、値が反映されているか確認する。

```
# curl -s -X GET ¥
--url http://localhost:$USER_PORT/v2/entities/Room1 ¥
--header "Accept: application/json" ¥
--header "Host: $SERVICE_HOST" ¥
--header "apikey: $ORION_USER_KEY" | python -mjson.tool
```

エンティティの情報が更新されていることを確認する。

```
{
  "id": "Room1",
  "pressure": {
    "metadata": {},
    "type": "Float",
    "value": 763
  },
  "temperature": {
    "metadata": {},
    "type": "Float",
    "value": 26.5
  },
  "type": "Room"
}
```