アプリケーション開発ガイド（概要）

加古川市　企画部　政策企画課
2023年12月22日

目次

１．データ利活用基盤サービスとは

① データ利活用基盤サービスとは？

データ利活用基盤サービス（ＦＩＷＡＲＥ）とは？

- EUが開発・実装し、欧州で実績のあるスマートシティ向けIoTプラットフォーム「FIWARE」を活用した、産学官の多様な主体がデータ利活用することができるプラットフォームです。

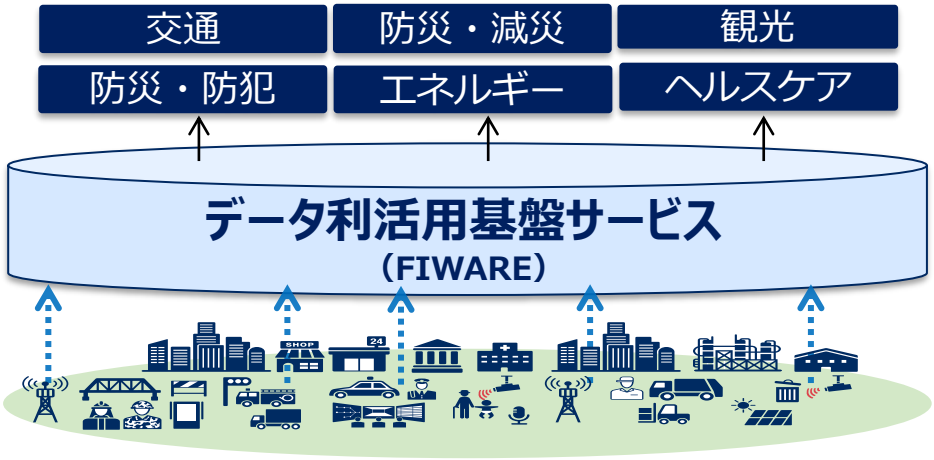
サービスの特長

分野を横断したデータの蓄積

- 様々な分野（防災・観光・防犯・etc..）のデータ相互運用性を考慮したデータモデル

地域課題に応じたサービスの構築

- プラットフォームに蓄積されたデータを利活用し、地域課題に応じた新たなサービスの構築



エリアに関するデータを取得

地域課題に応じたサービスを開発

出典：日本電気株式会社

② 利用イメージ

データ利活用基盤サービスは、地域課題を現状把握し、新たな価値を創出することにより、**魅力的な街づくり**を支援します。

エリアの可視化

・収集したデータを元に地域の課題や価値を顕在化

行政サービスの最適制御

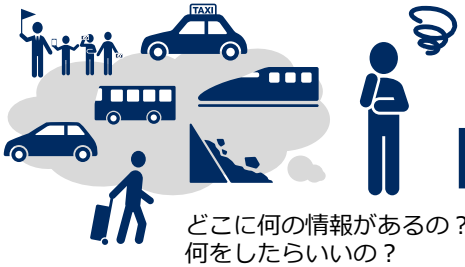
・分析技術を活用して行政サービスを最適化・コスト最適化

知見の共有と民間利活用

・蓄積されたデータを分析し知見を共有
・オープンAPIの活用により新たな価値を創出

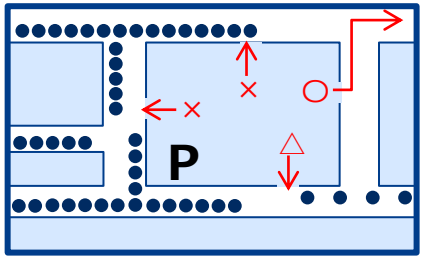
データ利活用基盤サービス（FIWARE）

エリアに関するデータ取得



どこに何の情報があるの？
何をしたらいいの？

データによる現状の把握

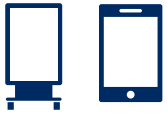


こんな動きなのか！

ステークホルダーの合意形成、行政との連携

行政都市計画などへ

民間（地元企業）でのデータ利活用



新たなサービス創出へ

出典：日本電気株式会社

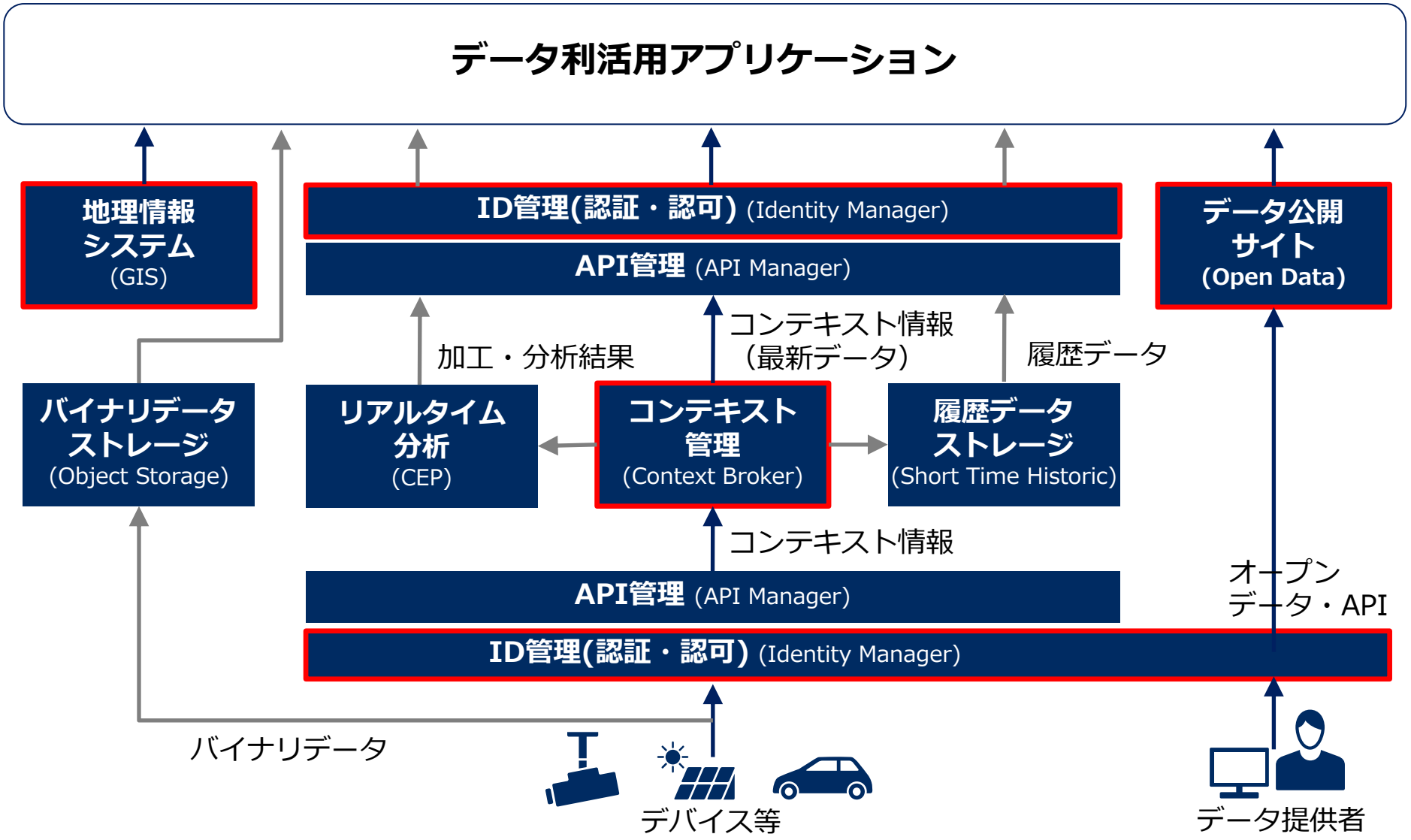
③ 提供機能について

機能名	概要
データ公開サイト	データ利活用者向けに、基盤に収集・蓄積されているデータの種類を一覧化し、そのデータへのアクセス方法を公開するポータルサイト
地理情報システム	アプリケーションへ地理情報を提供する機能
リアルタイム分析	収集データをリアルタイム分析し、結果を出力する機能
コンテキスト管理	都市に存在するモノ・コトをデータ(コンテキスト情報)として統合管理し、データ提供者、データ利用者へオープンAPIを提供する機能
履歴データストレージ	コンテキスト情報の履歴を蓄積・参照する機能
バイナリデータストレージ	画像・動画などのバイナリデータを管理するストレージ機能
API管理	Web APIの管理機能、セキュリティプロキシ機能
ID管理(認証・認可)	管理機能やAPIへのアクセス権限をユーザID単位で制御する認証・認可機能

出典：日本電気株式会社

④ システム構成

本書での説明範囲



出典：日本電気株式会社

⑤ データ利活用アプリケーションの実装例

地理情報システム

アプリケーション上へ地図情報を表示する。

リアルタイム分析

コンテキスト管理が収集した情報をリアルタイムに処理し、アプリケーションへの通知する。

コンテキスト管理

各コンテキストの最新情報（現在の状態）をアイコンや吹き出しで表示する。

バイナリデータストレージ

画像などのバイナリデータをアプリケーションで表示する。

履歴データストレージ

収集したコンテキスト情報の推移をグラフで表示する。

出典：日本電気株式会社

2. 認証・認可

① 認証・認可機能について

データ利活用基盤サービスでは、認証・認可方式として OAuth2.0 を採用しています。

OAuth2.0 の特徴

- 認可サーバ（Authorization Server）によるIDの一元管理
- サードパーティ・アプリケーション（OAuth Client）に認証情報を渡さずに認証・認可機能が利用可能
- Webアプリケーションだけでなく、クライアントアプリケーション等からも利用可能（OAuth1.0からの改善）
- 多くのサービスで採用されている実績のある認証・認可方式

データ利活用基盤サービスの各APIは、OAuth2.0で取得したアクセストークンを利用します。

② 認証方式の一覧

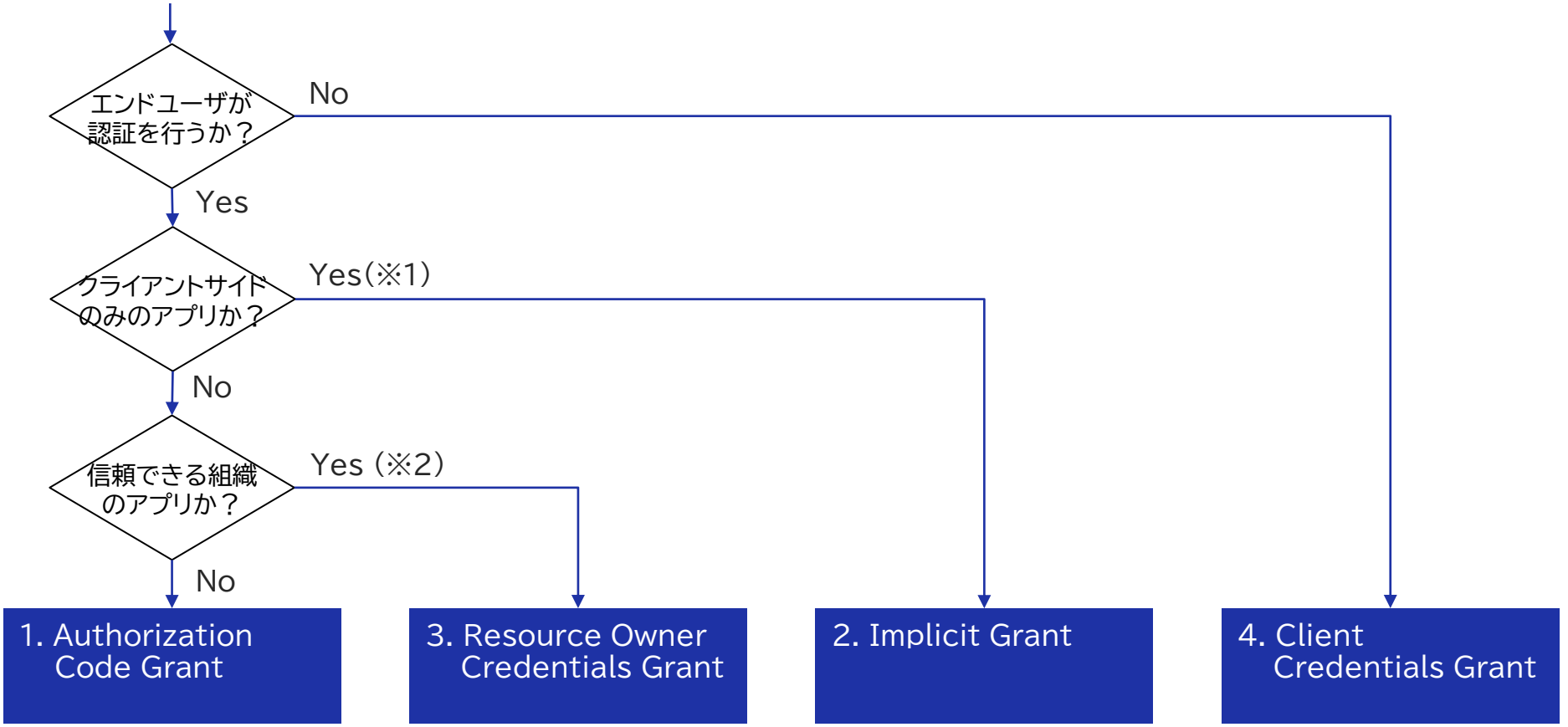
APIの利用を制御するためOAuth 2.0による認証機能が利用可能です。
OAuth 2.0は、4通りの認証方法があり、利用者はアプリケーションが必要とするセキュリティレベルや実装方式等により認証方式を選択できます。

No	認証方式	説明
1	Authorization Code Grant	信頼関係にないWebアプリケーションに有効な認証方式。 アプリケーションには認証情報を渡さず、利用者自身が認可サーバの認証を受ける。Consumer Key, Consumer Secretを利用する。 エンドユーザがID/PWを使ってログインするWebアプリケーションを構築する場合に選択します。 Webアプリケーションを経由してID/PWのやり取りを行わないないため、よりセキュアに認証することができます。 エンドユーザがID/PWを使ってログインする場合、通常はこの認証方式を選択してください。
2	Implicit Grant	JavaScriptのような、資格情報を秘密にできないプログラムに有効な認証方式。Consumer Secretを利用せず、Consumer Keyのみで認証する。 現在、セキュリティレベルが低いため非推奨となっており、一般公開するようなWebアプリケーションでは利用しないでください。 利用者を限定したツールやテスト等での一時的な利用のみとしてください。
3	Resource Owner Credentials Grant	信頼関係(同一ドメイン内など)のあるWebアプリケーションに有効な認証方式。アプリケーションに対し利用者が認証情報(ID, パスワード)を提供する必要がある。 エンドユーザがID/PWを使ってログインするWebアプリケーションを構築する場合に選択します。 Webアプリケーションを経由してID/PWをやり取りするため、No.1よりセキュリティレベルが低くなります。 運営事業者公認のWebアプリケーションなどの信頼関係のある場合のみ選択してください。
4	Client Credentials Grant	プログラム(バイナリ)に有効な認証方式。 利用者の認証情報(ID, パスワード)は利用せず、アプリケーションの認証情報(ConsumerKey, ConsumerSecret)を利用して認証する。 バッチ処理やエンドユーザの認証なしでバックグラウンドでアクセスする場合に選択します。

※ 上記の4通りの認証方法に加えて、No.1、3で取得したアクセストークンの有効期限を延長、再発行を行うためのRefresh Token Grantが利用可能です。
※ 各認証の詳細についてはアプリケーション開発ガイドを参照。
※ 本章では、「Authorization Code Grant」での認証を例に説明します。

② 認証方式の一覧

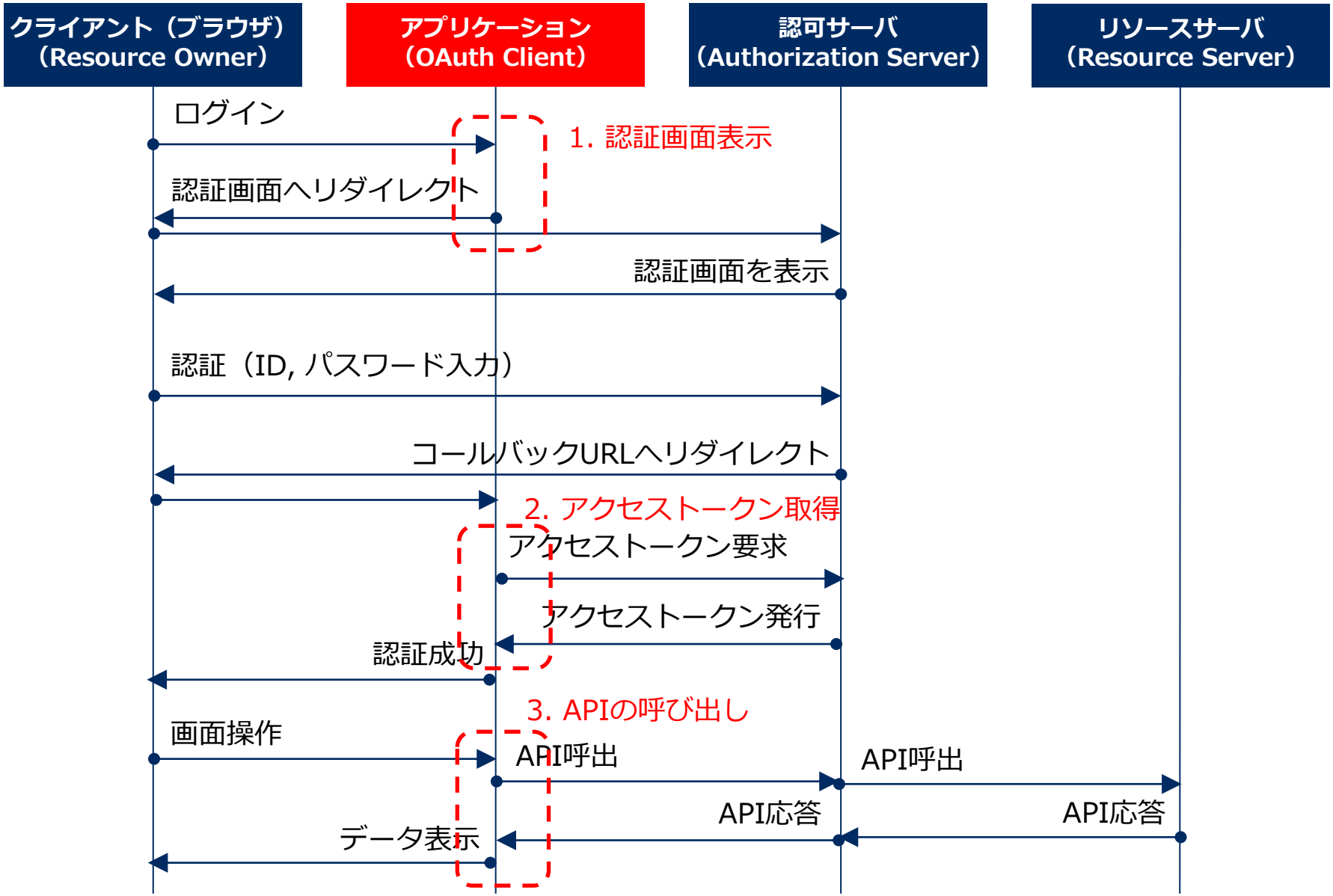
原則として、認証方式はAuthorization Code Grant を利用する （最もセキュア）。
Authorization Code Grantが利用できないケースで他方式の利用を検討する。



(※1)セキュリティレベルが低いため、利用者を限定したツールやテスト等での一時的な利用のみとしてください
(※2) Authorization Code Grant で不都合が無いのであれば Authorization Code Grant を推奨
（Authorization Code Grant のほうがよりセキュアであるため）

③-1. 認証シーケンス（Authorization Code Grant）

アプリケーションで
実装が必要な箇所



③-2. 認証シーケンス（Authorization Code Grant）

1．認証画面表示

- エンドポイント：
  https://api.opendata-api-kakogawa.jp/oauth2/authorize
- HTTPメソッド：
  GET
- リクエストパラメータ：

パラメータ名	説明	例
scope	“default”を指定してください。	default
response_type	Authorization Code Grantの場合は、“code” を指定してください。	code
redirect_uri	アプリケーション登録時に指定したコールバックURLを指定してください。（※）	https://example.com/oauth/callback
client_id	アプリケーション登録時に発行されたConsumer Key を指定してください。	ejwiao23tr4

（※）利用者が認証画面で認証した後、redirect_uriで指定したURLへ認可コードが渡されます。
この認可コードを「２．アクセストークン取得」で利用します。
例）https://example.com/oauth/callback?code=abcdef

認可コード

③-3. 認証シーケンス（Authorization Code Grant）

2．アクセストークン取得

- エンドポイント：
  https://api.opendata-api-kakogawa.jp/oauth2/token
- HTTPメソッド：
  POST
- Content-Type：
  application/x-www-form-urlencoded
- リクエストパラメータ：

パラメータ名	説明	例
code	「１．認証画面表示」で取得した認可コードを指定してください。	abcdef
grant_type	Authorization Code Grant の場合は、“authorization_code”を指定してください。	code
client_secret	アプリケーション登録時に発行された Consumer Secret を指定してください	poijf0we9aijkewahe89
redirect_uri	「１．認証画面表示」と同様	https://example.com/oauth/callback
client_id	「１．認証画面表示」と同様	ejwiao23tr4

③-4. 認証シーケンス（Authorization Code Grant）

Kakogawa City 加古川市 いつまでも住み続けたい ウェルネス都市 加古川

- レスポンスボディ
  JSON形

パラメータ名	説明	例
scope	「１．認証画面表示」で指定したscopeの値が返却されます。	default
token_type	"Bearer"が返却されます。	Bearer
expires_in	アクセストークンの有効期限（秒）が返却されます。	2413
refresh_token	リフレッシュトークンが返却されます。あアクセストークンの有効期限延長に使用します（※）	e156236ef50596b80d44adbb1c2773b0
access_token	アクセストークンが返却されます。API呼び出しに使用します。	4e88f99fa193bafbeb41c528b9b9e070

例）
{"scope":"default","token_type":"Bearer","expires_in":2413,"refresh_token":"e156236ef50596b80d44adbb1c2773b0","access_token":"4e88f99fa193bafbeb41c528b9b9e070"}

（※）アクセストークンには有効期限があるため、継続して使用するためにはリフレッシュトークンを使用してアクセストークンの有効期限を延長する必要があります。詳細は下記URLを参照。
https://is.docs.wso2.com/en/5.10.0/learn/refresh-token-grant/

③-5. 認証シーケンス（Authorization Code Grant）

3．APIの呼び出し

各APIを呼び出す際に、HTTPヘッダに「2．アクセストークン取得」で取得したアクセストークンを指定します。

- HTTPヘッダ

ヘッダフィールド	値
Authorization	Bearer 【OAuth 2.0のアクセストークン文字列】

例）curlコマンドで、コンテキスト管理機能のデータ参照（GETメソッド entities）のAPIを呼び出す場合

```
(curl -k -v -X GET “https:// api.opendata-api-kakogawa.jp /orion/v2.0/
entities?type=Room&idPattern=Room.*&attrs=temperature” -s –S ¥
 --header ”Authorization: Bearer 4e88f99fa193bafbeb41c528b9b9e070“ ¥
 --header ”Accept: application/json” ¥
 --header “Fiware-Service: TenantA” ¥
 --header ”Fiware-ServicePath: /” ¥
 -d @- | python -mjson.tool) <<EOF
```

3．コンテキスト管理

①-1. コンテキスト管理機能について

Kakogawa City 加古川市 いつまでも住み続けたい ウェルネス都市 加古川

加古川市に存在するモノ・コトをデータ(コンテキスト情報)として統合管理し、データ提供者、データ利用者へオープンAPIを提供する機能です。

コンテキスト管理の機能（NGSI-9/10）

- コンテキストの最新状態の管理（登録・参照）
- 他コンポーネントへのデータ転送機能
- コンテキストの分散管理、所在検索

①-2. FIWARE　NGSI の API バージョンについて

データ利活用基盤サービスは、NGSI v2に対応しています。

- NGSI v2（バージョン2）
  - NGSI v1を改良、単純化された、開発者フレンドリーなオープンAPI
  - v1との機能差分
    - HTTPメソッドやHTTPレスポンスを利用した、よりRESTfulな呼び出しが可能
    - 強化された機能（ジオロケーション、フィルタリングなど）が利用可能
    - ネイティブなJSONデータ型をサポートしております

参考：
FIWARE NGSI APIv2 Walkthrough
https://fiware-orion.readthedocs.io/en/1.14.0/user/walkthrough_apiv2/

②-1. NGSIv2：データ参照　リクエスト内容

データ参照（entities）

- エンドポイント：
  https://api.opendata-api-kakogawa.jp/orion/v2.0/entities
- HTTPメソッド：
  GET
- HTTPヘッダ：
  Content-Type：application/json
  Accept：application/json
  Authorization：Bearer ${アクセストークン}
- クエリパラメータ：
  ※次ページ参照
- リクエスト例：

https://api.opendata-api-kakogawa.jp/orion/v2.0/entities?type=Room&idPattern=.*&attrs=temperature&limit=1000

取得位置の指定。
1000件取得する例。

エンティティ検索条件の指定。
Typeが”Room”、の任意のIDのデータを取得する例。

取得対象の属性絞り込みの指定。
“temperature”の属性のみを取得する例。

②-2. NGSIv2：データ参照　パラメータ

Kakogawa City 加古川市 いつまでも住み続けたい ウェルネス都市 加古川

●クエリパラメータ

パラメータ	説明	例
id	エンティティID ※ idPattern パラメータと排他関係	Room-1
type	エンティティタイプ ※ typePattern パラメータと排他関係	Room
idPattern	エンティティID の正規表現	.*
typePattern	エンティティタイプの正規表現	.*
q	属性値のクエリ式	dateModified>2018-11-20T00:00:00.00
mq	メタデータのクエリ式	temperature.unit==℃
georal	座標の位置関係	near;maxDistance:100
geometry	座標が示すオブジェクトの形状	point
coords	座標。WGS-84 に準拠	34.339014,134.052283

パラメータ	説明	例
limit	取得するエンティティの上限数	20
offset	取得するエンティティのオフセット	0
attrs	属性名称 ※カンマ区切りで複数指定可	temperature
metadata	メタデータ名称 ※カンマ区切りで複数指定可	unit
orderBy	ソート条件	
options	オプション情報 ※カンマ区切りで複数指定可	count, keyValues, values, unique

※ SIMPLE QUERY LANGUAGE参照

※ GEOGRAPHICAL QUERIES参照

参考：FIWARE NGSI APIv2 Walkthrough
https://fiware-orion.readthedocs.io/en/1.14.0/user/walkthrough_apiv2/
参考：SIMPLE QUERY LANGUAGE
https://fiware-orion.readthedocs.io/en/1.14.0/user/filtering/index.html#simple-query-language
参考：GEOGRAPHICAL QUERIES
https://fiware-orion.readthedocs.io/en/1.14.0/user/filtering/index.html#geographical-queries

②-3. NGSIv2：データ参照　レスポンス内容

- ステータスコード：200 OK
- レスポンスボディ：json形式で下記のコンテキスト情報が返却される。

パラメータ名	型	説明	例
id	String	エンティティID	Room-1
type	String	エンティティタイプ	Room
${AttributeName}	Object	属性の名称	temperature
type	String	属性の型	Number
value	${type}	属性の値	20.5
metadata	Object	メタデータ	
${MetadataName}	Object	メタデータの名称	uint
type	String	メタデータの型	Text
value	${type}	メタデータの値	℃

- レスポンスボディ(例)

```
[
  {
    "id": "Room-1",
    "type": "Room",
    "temperature":{
      "type": "Number",
      "value": 20.5,
      "metadata":{
        "uint":{
          "type": "Text",
          "value": "℃"
} } } }
]
```

${type}はJSONデータ型の文字列や数値に変わる。

検索条件に合致したコンテキスト情報が返却される。出力されるのはlimitで指定した件数が上限。

③-1. NGSIv2：データ登録　リクエスト内容

データ登録（entities）

- エンドポイント：
  https://api.opendata-api-kakogawa.jp/orion/v2.0/entities
- HTTPメソッド：
  POST　：新規作成
- HTTPヘッダ：
  Content-Type：application/json
  Accept：application/json
  Authorization：Bearer ${アクセストークン}

- リクエスト例：

https://api.opendata-api-kakogawa.jp/orion/v2.0/entities

エンドポイントに、エンティティIDや、属性名、属性値を、
HTTPメソッドに、更新(PATCH)や削除(DELETE)を組み合わせることで、
データの更新や削除が可能となります。

参考：UPDATE ACTION TYPES
https://fiware-orion.readthedocs.io/en/1.14.0/user/update_action_types/index.html

③-2. NGSIv2：データ登録　パラメータ

●リクエストボディ：json形式で下記の内容を指定する。

パラメータ名	型	説明	例
id	String	エンティティID	Room-1
type	String	エンティティタイプ	Room
${AttributeName}	Object	属性の名称	temperature
type	String	属性の型	Number
value	${type}	属性の値	20.5
metadata	Object	メタデータの名称	
${MetadataName}	String	メタデータの名称	uint
type	String	メタデータの型	Text
value	${type}	メタデータの値	℃

●リクエストボディ(例)

```
{
 "id": "Room-1",
 "type": "Room",
 "temperature": {
  "value": 20.5,
  "type": "Number“,
  "metadata" : {
   "uint" : {
    "value": "℃",
    "type": "Text"
   }
  }
}
}
```

登録するコンテキスト情報の指定。

登録する属性情報の指定。

登録するメタデータの指定。

③-3. NGSIv2：データ登録　レスポンス内容

- ステータスコード：201 Created
- レスポンスボディ：なし
- レスポンスヘッダ(例)

```
< HTTP/1.1 201 Created
< Connection: Keep-Alive
< Content-Length: 0
< Fiware-Correlator: e4f0f334-10a8-11e8-ab6e-000c29173617
< Location: /v2/entities/Room-1?type=Room
< Date: Tue, 13 Feb 2018 10:30:21 GMT
```

entitiesを利用したデータ登録処理は1つのエンティティにのみ対応しています。
バッチオペレーション(/op/update)を利用することで、複数のエンティティの登録/更新が可能となります。

参考：BATCH OPEARATION
https://fiware-orion.readthedocs.io/en/1.14.0/user/walkthrough_apiv2/#batch-operations

④ その他

禁則文字

- 下記の文字はURLエンコードを実施した上で登録してください。（参照時は逆変換）

禁則文字	エンコード後	備考
<	%3C	不等号（より小）（0x3c）
>	%3E	不等号（より大）（0x3e）
“	%22	ダブルクォート（0x22）
'	%27	シングルクォート（0x27）
=	%3D	イコール（0x3d）
;	%3B	セミコロン（0x3b）
(	%28	左小括弧（0x28）
)	%29	右小括弧（0x29）
%	%25	パーセント（0x25） ※パーセントは禁則文字ではありませんが、他の変換と競合するため同様にURLエンコードを実施してください

データモデル

- コンテキスト管理では任意のデータを登録できますが、標準化規格にあわせたデータモデルにすることで、他の利用者が使いやすくなります。下記を参考にデータモデルを定義することをお勧めします。
  - 政府相互運用性フレームワーク（Government Interoperability Framework）：
    https://github.com/JDA-DM/GIF/

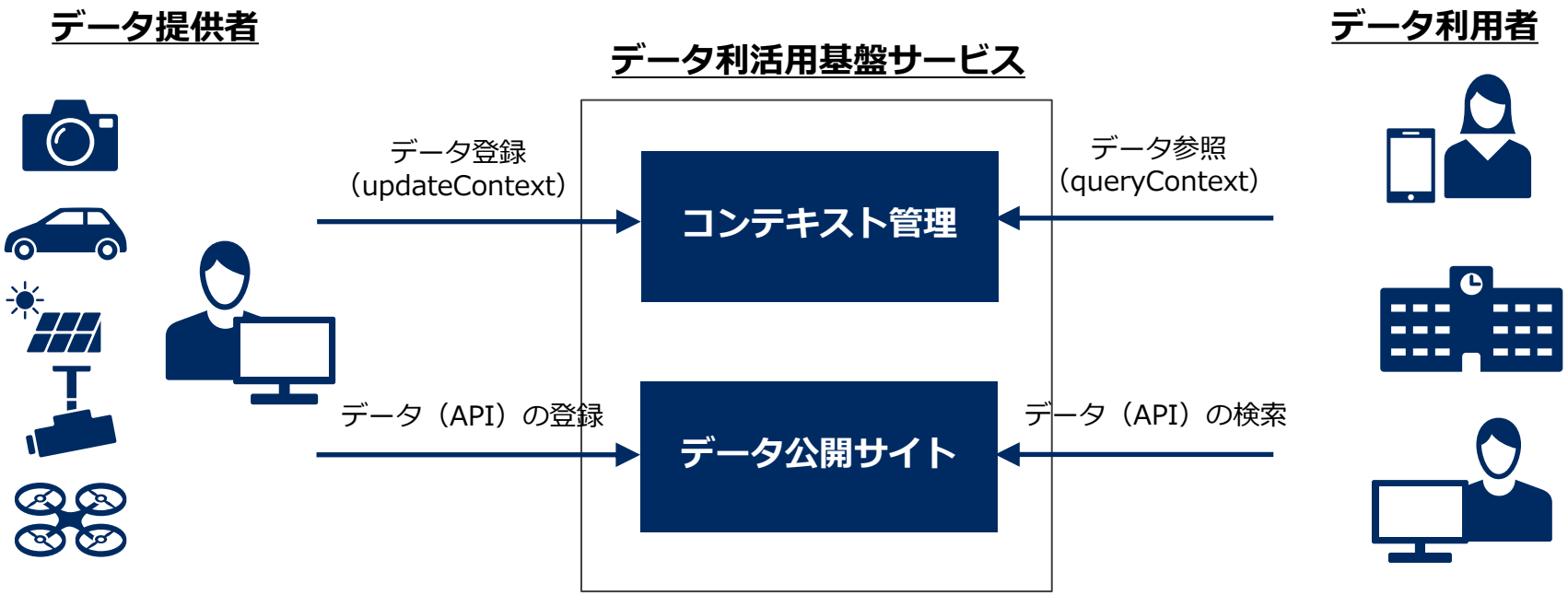
４．データ公開サイト

① データ公開サイトについて

データ利活用基盤サービスに登録したデータを、オープンデータとして公開する機能です。

- CSVやPDFなど、ファイル形式で公開
- オープンAPIとして公開
  ★プログラムから直接扱えるデータとして公開することができます★

活用イメージ



② オープンデータ・APIの検索

データ公開サイトURL

https://opendata-api-kakogawa.jp/odp/

キーワードでのオープンデータの検索機能

検索

データセットの一覧を表示

選択

トップページ

新着データ、人気のデータなどおススメ情報を表示

データセット一覧

５．地理情報システム

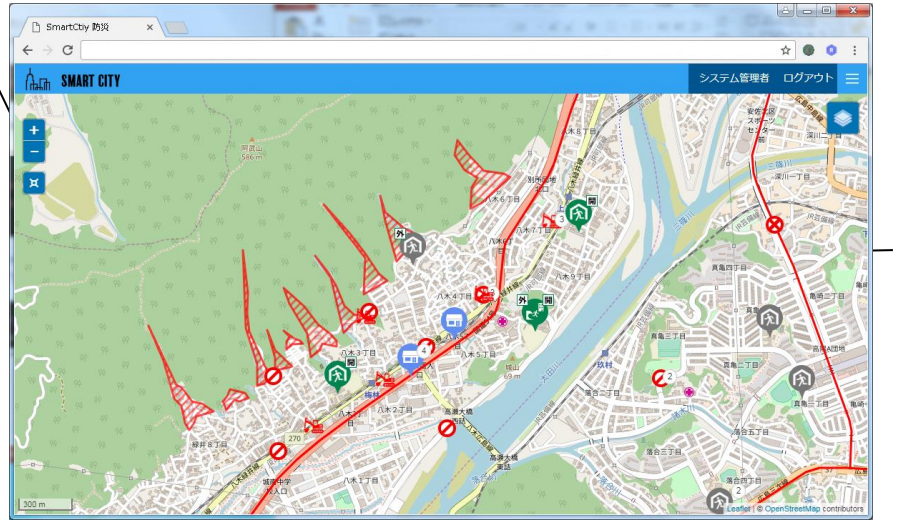
① 地理情報システムについて

アプリケーションへ地理情報を提供する機能です。

- 収集した情報を地図上にマッピングすることで可視性の向上
- GeoJSON形式で、既存ライブラリとの親和性

利便性

既存ライブラリを活用することで、地図表示に必要な各機能を容易に実装可能



可視性

収集した情報を地図上にマッピングすることで、地域全体の情報の可視性向上

注意事項

データ利活用基盤サービス（FIWARE）は、OpenStreetMapの地図を利用しています。本機能を利用する際はOpenStreetMapのクレジット表記を行ってください。

参考：https://www.openstreetmap.org/copyright

② 地図の利用方法

地図表示のサンプルコード（Leaflet.jsを使用）

```
<html>
<head>
<meta http-equiv="X-UA-Compatible" content="IE=EmulateIE10">
<link rel="stylesheet" href="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.css"/>
<script src="http://cdn.leafletjs.com/leaflet-0.7.3/leaflet.js"></script>
<script type="text/javascript">

function init () {

  // GIS Data Providerレイヤの定義
  var gisdataproviderLayer = L.tileLayer(
        'http://api.opendata-api-
kakogawa.jp/gis/geoserver/gwc/service/tms/1.0.0/mn2gt@EPSG:900913@png/{z}/{x}/{y}.png', {
          tms: true,
          attribution: '&copy; <a href="http://osm.org/copyright" target="_blank">OpenStreetMap</a> contributors'
        }
  );
  // 地図の初期表示設定
  mymap = L.map(
    'mapid', {
      center: [35.571841, 139.66577],
      zoom: 16,
      layers: [gisdataproviderLayer]
    }
  );
}
</script>
</head>
<body onload="init()">
   <div id="mapid" style="width:100%; height:100%;"></div>
</body>
</html>
```

Leaflet.jsの読み込み

レイヤの定義
・本システムのGISを指定
・OSMのコピーライトを指定

地図の表示処理
・座標
・ズームレベル

地図の表示エリア

Leaflet.jsのその他機能は下記を参照ください。
http://leafletjs.com/

サンプルコードの出力結果