データ利活用基盤サービス（FIWARE）
アプリケーション開発ガイド

1.1 版
2026 年3月 26 日

加古川市　企画部　デジタル改革推進課

目次

第1章 はじめに

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（以降、本ガイド）は、データ利活用基盤サービス（FIWARE）（以降、本サービス）と連携しデータを登録/参照するアプリケーションの基本的な開発手順を記載しています。本ガイドはアプリケーション開発者をターゲットとしています。本章では本サービスの概要について記載します。

1.1 用語の定義

本ガイド内で使用する用語の定義について記載します。
用語の定義については、表 1-1-1 のとおりです。

表 1-1-1 用語の定義

用語	概要
アプリケーション	本サービスを利用して開発するアプリケーション。
利用者	アプリケーション開発者。
データ提供者	本サービスへデータを提供する個人または法人。
システム管理者	本サービスを利用するために各コンポーネントに必要なユーザの作成や設定を行うユーザ。
外部システム	外部から本サービスに NGSI 形式でデータ連携をするシステム。
センサ/IoT デバイス（NGSI 非対応）	外部から本サービスと連携するセンサや IoT デバイス。
外部システム（NGSI 非対応）	外部から本サービスに NGSI 形式以外でデータ連携をするシステム。
コンポーネント	本サービスの機能を構成するソフトウェアの名称。
WSO2 Identity Server（IS）	本サービスの「ID 管理（認証・認可）」機能で提供するコンポーネントであり、画面上でユーザの管理や制御を行う機能。 ※3 章以降で記載している手順で、IS に対しての設定の場合は IS を記載しています。
WSO2 API Manager（AM）	本サービスの「API 管理」で提供するコンポーネントであり、画面上で、本サービスで提供する Web API の管理を行う機能。 ※3 章以降で記載している手順で、AM に対しての設定の場合は AM を記載しています。
サブテナント	1つの環境内で親団体、子団体毎が独立して管理できるように分けた仮想領域

1.2 サービス構成および機能との関係性

本サービスの構成と、各機能の関係性について記載します。

1.2.1 サービスの構成

本サービスは複数の機能から成り立ちます。利用者は、用途/目的に応じて使用する機能を選択します。
各機能に対応する利用用途について、以下の図に示します。

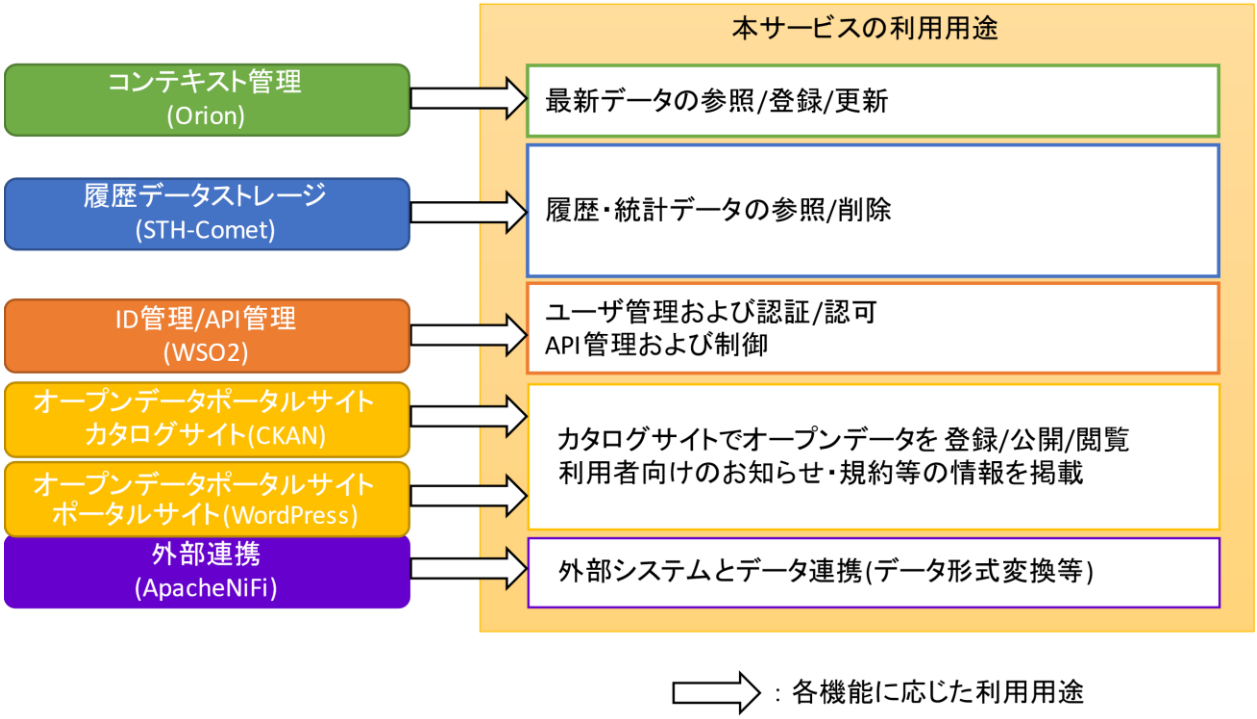

図 1-2-1 機能と対応する利用用途

本サービスの機能を 2 つに分けると、Web API を通して利用する API コンポーネントと、画面操作によって利用する UI コンポーネントに分類されます。本ガイドでは主に API コンポーネントについて記載します。

本書で説明する機能は表 1-2-1 のとおりです。
また、各機能と構成するコンポーネントおよび分類について以下の表に示します。

表 1-2-1 機能一覧

機能	コンポーネント	概要	分類	
			API	UI
コンテキスト管理	Orion	最新のデータを管理する API を提供する。また、過去データを履歴や分析を行うデータストレージへ転送する。	○	
履歴データストレージ	STH-Comet	コンテキスト情報の履歴を蓄積・参照する。	○	
ID 管理(認証・認可)	WSO2	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・認可を行う。	○	○
API 管理		アプリケーション作成やサブスクライブを行い、Web API を管理する。		
オープンデータポータルサイト	CKAN	オープンデータを登録/公開するためのカタログサイト。		○
	WordPress	ポータルサイト。		
外部連携	ApacheNiFi	Web ブラウザでデータフローの設定、制御、監視を行うことにより、外部システムのデータを Orion へ登録する。		

1.2.2 各コンポーネントの関係性

各コンポーネントの関係性について、以下の図に示します。

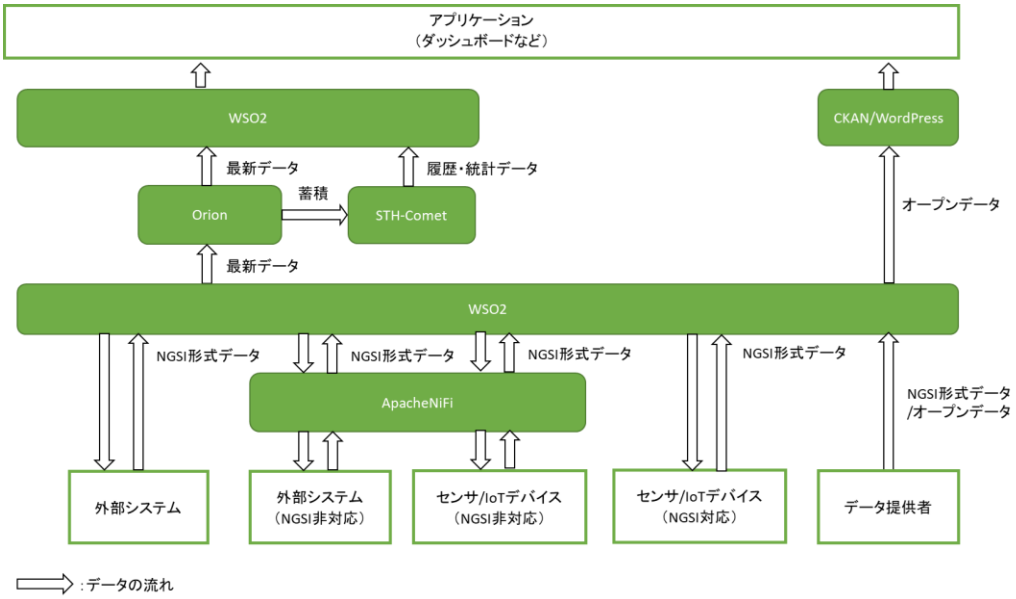

図 1-2-2 コンポーネントの関係性

（補足）NGSI

NGSI 形式データは主に、3 つの要素から成り立ちます。
NGSI 形式データを構成する要素は以下の表のとおりです。

表 1-2-2 NGSI 形式データの要素

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。 エンティティ ID（id）、エンティティタイプ（type）、属性（Attribute）を持つ。各エンティティはエンティティ ID とエンティティタイプの組み合わせで一意に識別可能にする必要がある。また、データとして管理したい複数の属性をグループ化するように設計する。 ※属性（Attribute）は任意。 **※エンティティ ID（id）、エンティティタイプ（type）の値にドットを連続して使用した場合、STH-Comet で利用ができなくなります。** **例） 使用可能：「sample.12345」** **使用不可：「sample..12345」**
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、メタデータ（Metadata）を持つ。 ※メタデータ（Metadata）は任意
メタデータ（Metadata）	属性値への付加情報（属性値の計測時刻など）。属性値に付加情報を持たせる必要がない場合は定義不要。 メタデータ名（name）、メタデータ型（type）、メタデータ値（value）を持つ。

※サンプル

```
{
    ”id”: ”<IP>.Station.0001”,
    ”type”: ”Station”,
    ”name”: {
        ”value”: ”東京駅”,
        ”type” : ”Text”,
        ”metadata” : {
            ”fullNameInKana” : {
                ”value” : ”とうきょうえき”,
                ”type”  : ”Text”
            }
        }
    },
    ”address”: {
        ”value”: ”東京都千代田区丸の内１丁目９”,
        ”type” : ”Text”
    },
    ”location”: {
        ”value”: ”35.6812362,139.7671248”,
```

```
        "type": "geo:point"
    },
    "organization": {
        "value": "東日本旅客鉄道株式会社",
        "type" : "Text"
    },
    "telephoneNumber": {
        "value": "050-2016-1601",
        "type" : "Text"
    },
    "referencedObject": {
        "value":
"https://www.jreast.co.jp/estation/stations/1039.html",
        "type": "URL"
    },
    "uploadDate": {
        "value": "2019-03-08T 12:12:00+09:00",
        "type": "DateTime"
    }
 }
```

（補足）Fiware-Service/FiwareServicePath

Fiware-Service および Fiware-ServicePath はマルチテナント/マルチサービスを提供するための機能です。Fiware-Service によりNGSI形式データを論理的にグループ化し、隔離します。また、Fiware-ServicePath は、同一 Fiware-Service に所属する NGSI 形式データを階層構造化する仕組みです。Fiware-Service および Fiware-ServicePath は API 呼び出し時の HTTP ヘッダに指定することで利用できます（任意）。

Fiware-Service および Fiware-ServicePath の指定ありで登録した NGSI 形式データを参照する場合は、登録時と同じ Fiware-Service および Fiware-ServicePath を指定することで、データを参照できます（論理的に隔離されているため）。

以下に Fiware-Service および Fiware-ServicePath の一例と命名規則を示します。

表 1-2-3 Fiware Sevice 命名規則

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 1-2-4 Fiware SevicePath 命名規則

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(/) ・階層はスラッシュ(/)で表現 例)/A/B/C ・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 個までコンマ(,)区切りで指定可能。(更新系の API 呼び出し時の場合は 1 個のみ) ・末尾にスラッシュ(/)を指定した場合は無視(破棄)

なお、コンテキスト情報の履歴を蓄積するためには、エンティティID(id)、エンティティタイプ(type)、Fiware Service、Fiware ServicePath の文字数の合計が 59byte 以内になるようにそれぞれを命名してください。

※Fiware ServicePath の「/」は「x002f」にエンコードされるので 5byte として計算します

※Fiware Service が未指定の場合は 7byte(default)で計算します

※Fiware ServicePath が未指定の場合は 5byte(/)で計算します

※サンプル

以下は階層構造の例より Fiware-Service および Fiware-ServicePath を指定した場合の例です。

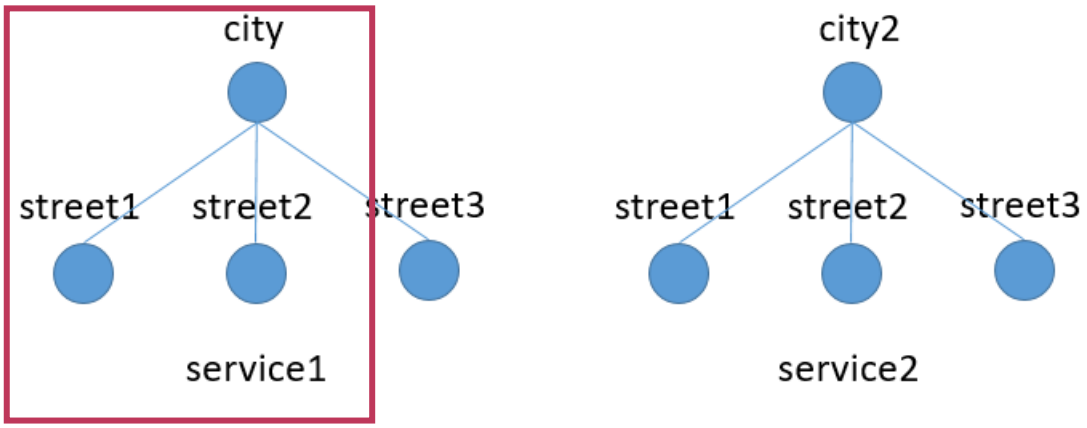

図 1-2-3 Fiware-ServicePath の階層例

以下を指定した場合、Fiware-ServicePath に/city/street1 および/city/street2 を指定した NGSI 形式データを取得できます。(図 1-2-3 赤枠)

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

表 1-2-5 FiwareSevice/FiwareSevicePath 利用可否

コンポーネント	利用可否	FiwareSevice	FiwareSevicePath
Orion	利用可	任意 ※1	任意 ※1
STH-Comet	利用可	必須	必須
WSO2	利用不可	–	–
CKAN	利用不可	–	–
WordPress	利用不可	–	–
ApacheNiFi	利用可	任意	任意

※1 他機能と連携して利用する場合は必須

第2章 API コンポーネント

本章では本サービスが提供する API コンポーネントの機能について記載します。

各 API コンポーネントが提供する詳細な API 仕様は API DevPortal より確認してください。

※「3.4.2 API DevPortal 実行例」の手順参照

2.1 Orion

Orion はデータを管理する API を提供します。Orion の機能について表 2-1-1 に示します。

表 2-1-1 Orion の API の種類

用途	概要
データ管理	データの登録、更新、削除、参照をする API
所在管理	データの所在(URL)を管理する API 登録されているデータに更新/参照リクエストがあった場合、外部システムへリクエストの転送を行う
サブスクリプション管理	データ更新通知を管理する API 更新通知登録されているデータに更新があった場合、外部システムに対してデータの更新通知を行う

2.1.1 データ管理

データ管理について以下の図に示します。
データ管理の API を利用することで、データの登録や更新、削除、参照が可能となります。

図 2-1-1 Orion のデータ管理について

2.1.2 所在管理

所在管理について以下の図に示します。
所在管理の API を利用することで、本サービスにデータを保存することなく、外部システムからリアルタイムにデータを更新/取得できます。

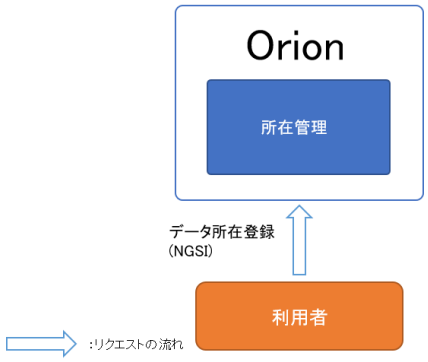

図 2-1-2 Orion の所在管理について

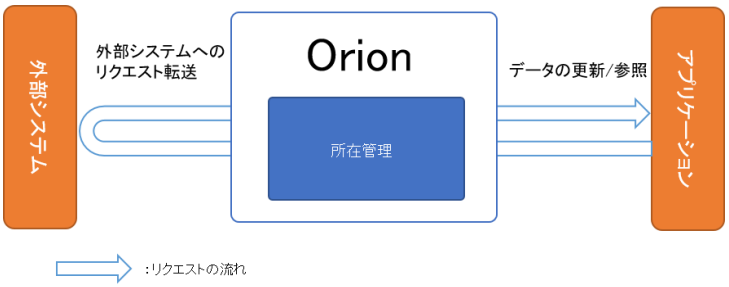

図 2-1-3 Orion に所在登録されているデータ参照の流れ

2.1.3 サブスクリプション管理

サブスクリプション管理について以下の図に示します。
サブスクリプション管理の API を利用することで、データ更新時に他システムにデータの更新通知を行うことができます。

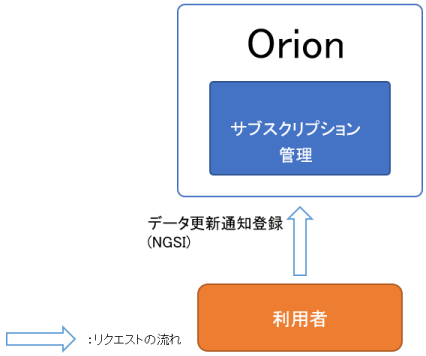

図 2-1-4 Orion のサブスクリプション管理について

図 2-1-5 Orion の通知登録後の処理について

2.1.4 インタフェース構成

Orion のインタフェース構成を、表 2-1-2 に示します。

表 2-1-2 Orion

用途	概要	API 名	HTTP メソッド
データ管理	データリストの参照	entities	GET
	データの登録	entities	POST
	バッチ更新オペレーションの実行	op/update	POST
	バッチクエリオペレーションの実行	op/query	POST
	データの参照(ID 指定)	entities/{entityId}	GET
	データの削除(ID 指定)	entities/{entityId}	DELETE
	データの属性情報の参照 (ID 指定，属性指定)	entities/{entityId}/attrs	GET
	データの属性情報の登録 (ID 指定，属性指定)	entities/{entityId}/attrs	POST
	データの属性情報の更新 (ID 指定，属性指定)	entities/{entityId}/attrs	PUT
	データの属性情報の更新 (ID 指定，属性指定)	entities/{entityId}/attrs	PATCH
	データの属性情報の参照 (ID 指定，属性名称指定)	entities/{entityId}/attrs/{attrName}	GET
	データの属性情報の更新 (ID 指定，属性名称指定)	entities/{entityId}/attrs/{attrName}	PUT
	データの属性情報の削除 (ID 指定，属性名称指定)	entities/{entityId}/attrs/{attrName}	DELETE
	データの属性値の検索 (ID 指定，属性名称指定)	entities/{entityId}/attrs/{attrName}/value	GET
	データの属性値の更新 (ID 指定，属性名称指定)	entities/{entityId}/attrs/{attrName}/value	PUT
	タイプリストの参照	types	GET
	タイプ情報の参照 (タイプ指定)	types/{entityType}	GET
所在管理	データの所在参照	registrations	GET
	データの所在登録	registrations	POST
	データの所在削除	registrations/{registrationId}	DELETE
	データの所在参照(ID 指定)	registrations/{registrationId}	GET
サブスクリプション管理	データ更新通知予約の参照	subscriptions	GET
	データ更新通知予約登録	subscriptions	POST
	データ更新通知予約更新 (ID 指定)	subscriptions/{subscriptionId}	PATCH
	データ更新通知予約の参照 (ID 指定)	subscriptions/{subscriptionId}	GET
	データ更新通知予約削除(ID 指定)	subscriptions/{subscriptionId}	DELETE

2.1.5 注意、制限事項

① 参照 API でタイムアウトエラーが発生する場合

参照により取得されるデータ件数、またはデータサイズが大きい場合はエラーが発生する場合があります。HTTP レスポンスコードが 504 の場合、またはレスポンスコードが 500 で以下のレスポンスメッセージの場合は、取得するデータ件数、データサイズが小さくなるよう条件を追加して再度 API を実行してください。

```
<am:fault xmlns:am=”http://wso2.org/apimanager”>
<am:code>101508</am:code>
<am:type>Status report</am:type>
<am:message>Runtime Error</am:message>
<am:description>Error in Sender</am:description>
</am:fault>
```

② 性能最適化について

本項では、検索性能を向上させるための最適化方法について記載します。
Orion の性能を最大限に発揮するには、以下を考慮して設計/利用する必要があります。

(1).Fiware-Service 分割

Fiware-Service を分割してデータ蓄積することを推奨しています。
データの取得において、インデックスを活用できない場合、Fiware-Service を分割してデータ蓄積することで性能劣化を防ぎます。

(1).インデックスが有効な検索方法での利用

●id 指定/id の前方一致、type 検索

データの取得において、検索条件に「id 指定/id の前方一致、type 検索」を指定することを推奨しています。
検索条件に「id 指定/id の前方一致、type 検索」を指定することでインデックスを利用できるため、性能の向上が見込めます。

・id 指定/id の前方一致検索例

```
例） /orion/v2.0/entities?id=weather-000XX
     /orion/v2.0/entities/weather-000XX
     /orion/v2.0/entities?idPattern=weather-0000.+
```

・type 指定検索例

```
例） /orion/v2.0/entities?type=weather
```

●地理情報を持つデータの検索

データの取得において、検索条件に地理情報で用いることを推奨しています。
検索条件に地理情報を用いることでインデックスを活用できるため、性能の向上が見込めます。
地理情報タイプ：geo:point, geo:line, geo:box, geo:polygon

・地理情報データ例

```
{
  ”id”: ”Hospital1”,
  ”type”: ”Hospital”,
  ...
  ”location”: {
    ”value”: {
      ”type”: ”Point”,
      ”coordinates”: [ -3.68666,
40.48108 ]
    },
    ”type”: ”geo:json”
  }
}
```

・地理情報検索例

```
例）/orion/v2.0/entities?georel=near;maxDistance:1000
                &geometry=point&coords=40.48108,-3.68666
```

2.2 STH-Comet

STH-Comet は履歴・統計情報管理の API を利用することで、履歴・統計データを蓄積し、その蓄積したデータを参照するための API を提供します。データを蓄積するためには Orion のサブスクリプション管理機能の通知登録を実施しておく必要があります。通知登録時の通知先に STH-Comet を指定することで、蓄積が可能となります。STH-Comet への蓄積は Orion からの通知登録によってのみ可能となります。直接 STH-Comet へ履歴を蓄積する事はできません。
Orion の通知登録については「2.1 Orion」の「2.1.3 サブスクリプション管理」を、通知先として STH-Comet を指定する方法については API DevPortal より「2.1.4 インタフェース構成」の「データ更新通知予約登録 API」の仕様を参照してください。STH-Comet の機能について表 2-2-1 に示します。

表 2-2-1 STH-Comet の API の種類

用途	概要
履歴・統計データ管理	Orion から仲介して蓄積された履歴・統計(合計、最大値)データを管理する API

2.2.1 履歴・統計データ管理

履歴・統計データ管理について以下の図に示します。

図 2-2-1 STH-Comet の機能について

2.2.2 インタフェース構成

STH-Comet のインタフェース構成を表 2-2-2 に示します。

表 2-2-2 STH-Comet

概要	API 名	HTTP メソッド
履歴・統計データの参照	contextEntities/type/{entityType}/id/{entityId}/attributes/{attributeName}	GET
履歴・統計データの削除	contextEntities	DELETE
履歴・統計データの削除 (ID 指定，属性指定)	contextEntities/type/{entityType}/id/{entityId}	DELETE
履歴・統計データの削除 (ID 指定，属性，属性名称指定)	contextEntities/type/{entityType}/id/{entityId}/attributes/{attributeName}	DELETE

2.2.3 注意、制限事項

① Orion に対するデータ更新を STH-Comet に通知するための設定の注意点

STH-Comet は NGSIv1 の notify による通知にしか対応していないため、Orion 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”を指定し、NGSIv1 形式の通知が送信されるようにする必要があります。

② 蓄積対象のエンティティ ID、エンティティタイプの注意点

STH-Comet では、蓄積対象のエンティティ ID、エンティティタイプの値として連続したドットを使用することができません。たとえば以下のようなエンティティ ID は使用できません。

例） 使用可能:「sample.12345」

使用不可:「sample..12345」

③ 蓄積するデータの注意点

STH-Comet では、蓄積するデータのエンティティに含まれる属性値に複数のドットと数字のみを組み合わせた文字列を使用することができません。また、先頭がハイフンで、複数のドットと数字のみを組み合わせた文字列も属性値として使用することができません。たとえば以下のような文字列は属性値として使用できません。

例） 使用可能:「sample.1.2.3.4」

使用不可:「1.2.3.4」

使用不可:「-1.2.3.4」

2.3 ApacheNiFi

ApacheNiFi はさまざまなデータを加工/変換し送信/受信ができる機能を有しています。
本サービスでは、さまざまなデータ形式を持つ外部システムから本サービスへデータを送信するといった利用を想定しております。
本ガイドではサービス既定で用意している API の呼び出し方法について記載します。
詳細な仕様につきましてはシステム管理者にご確認ください。

表 2-3-1 ApacheNiFi の API の種類

用途	概要
データ管理	CSV データを ORION へ登録する API

2.3.1 データ管理

データ管理について以下の図に示します。
データ管理の API を利用することで、外部システムから Orion のデータを更新できます。

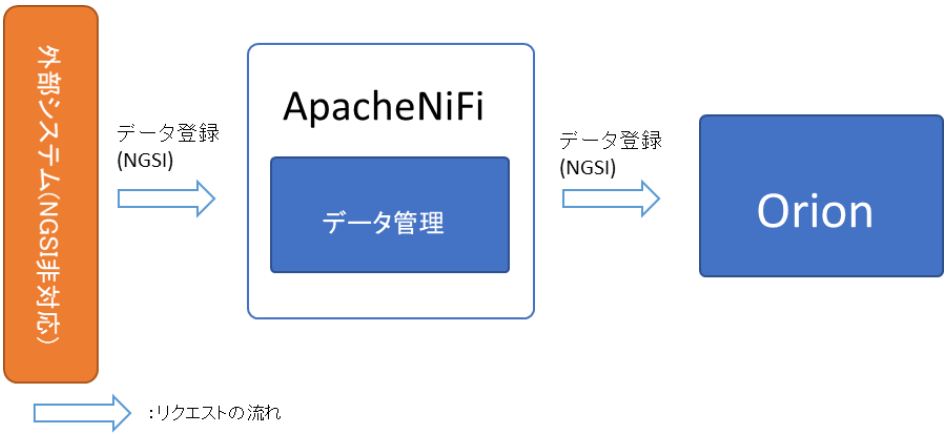

図 2-3-1 ApacheNiFi の機能について

2.3.2 インタフェース構成

ApacheNiFi のインタフェース構成を、表 2-7-2 に示します。

表 2-3-2 ApacheNiFi

概要	API 名	HTTP メソッド
CSV 形式のデータの登録	api/{API 番号}/v1.0/entities	POST

※API 番号：プロセスグループ別に割り当てられた API 番号

2.4 WSO2

本サービスにおいて WSO2 は各機能へアクセスするための情報を取得する API と、ユーザ情報を管理する API を提供します。WSO2 の機能について表 2-4-1 に示します。

表 2-4-1 WSO2 の API の種類

用途	概要
認証	認証・認可とアクセストークン取得をする API
ユーザ管理	ユーザ情報を取得する API

2.4.1 認証

認証について以下の図に示します。
認証の API を利用することで、各 API コンポーネントへアクセスするためのトークンが取得できます。認証方式など詳細は「3.3 API 認証(アクセストークンの取得)」を参照してください。

図 2-4-1 WSO2 の認証について

2.4.2 ユーザ管理

ユーザ管理について以下の図に示します。
ユーザ管理の API を利用することで、ログインユーザ情報を参照できます。

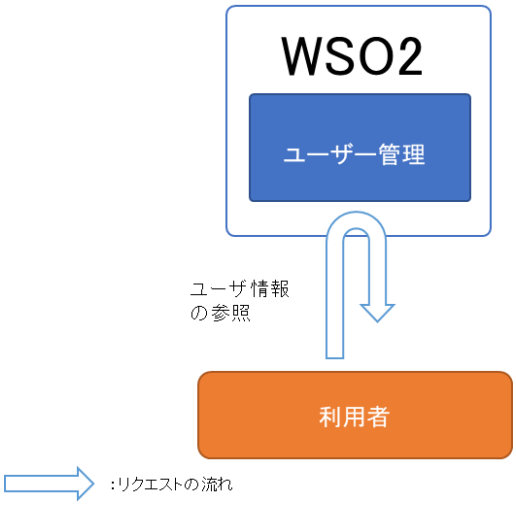

図 2-4-2 WSO2 のユーザ管理について

2.4.3 インタフェース構成

WSO2 のインタフェース構成を、表 2-8-2 に示します。

表 2-4-2 WSO2

用途	概要	API 名	HTTP メソッド
認証	認証・認可	oauth2/authorize	GET
	アクセストークンの取得	oauth2/token	POST
ユーザ管理	ユーザプロフィールの取得	wso2/scim/users/me	GET

※上表の API を使用する際は、Identity Server 用ドメイン名を使用してください。

※サンプル
curl コマンド実行例(ユーザプロフィールの取得):

```
(curl -s -S -k -X GET "https://{Identity Server 用ドメイン名}/wso2/scim/Users/me" ¥
   -H "Authorization: Bearer {3.3 で取得したアクセストークン}" ¥
   -H "Accept: application/json" ¥
 | python -mjson.tool)
 {
      "emails": [
          "aaa@bbb.local"
      ],
      "groups": [
          {
              "display": "Internal/subscriber"
          },
```

```
            {
                "display": "ContextProducer"
            },
            {
                "display": "Internal/creator"
            },
            {
                "display": "Internal/publisher"
            },
            {
                "display":
    "Application/testuser001_DefaultApplication_PRODUCTION"
            },
            {
                "display": "ContextConsumer"
            },
            {
                "display": "Publisher"
            }
        ],
        "id": "718c41e7-9337-4648-a41d-4eea562403a0",
        "ims": [
            "IM"
        ],
        "meta": {
            "created": "2018-01-11T07:14:00",
            "lastModified": "2018-01-17T11:52:57"
        },
        "name": {
            "familyName": "testuser001",
            "givenName": "testuser001"
        },
        "schemas": [
            "urn:scim:schemas:core:1.0"
        ],
        "username": "testuser001"
}
```

第3章 API 利用手順

本章ではアプリケーションから API を利用する手順を記載します。
ここでのアプリケーションとは本サービスを利用するために開発するアプリケーションのことです。
そのアプリケーションから API 呼び出すまでの流れを以下の図に示します。

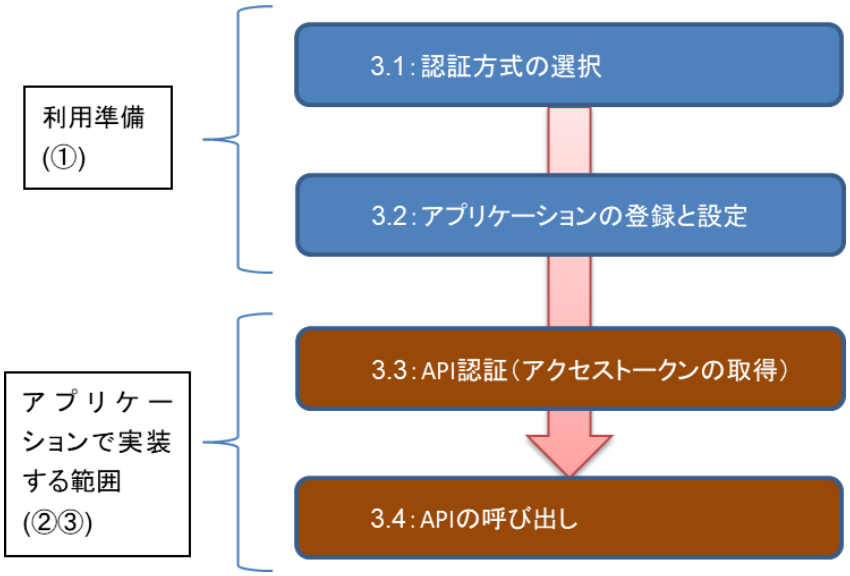

図 3-1-1 利用準備から API 呼び出しまでの流れ

以下、図 3-1-1 の①～③について記載します。

① 利用準備

アプリケーションが API を利用するにあたっての利用準備について記載します。
※「3.1 認証方式の選択」、「3.2 アプリケーションの登録と設定」参照

② アクセストークンの取得

アプリケーションへアクセス時やログイン時の OAuth 2.0 認証の利用方法について記載します。
※「3.3 API 認証（アクセストークンの取得）」参照

③ API の呼び出し

API の呼び出し方法について記載します。
※「3.4 API の呼び出し」参照

また、記載の中で、各認証や API の呼び出し時に記載がある URL のドメインについては、[ドメイン名]、[Identity Server 用ドメイン名]、[API Manager 用ドメイン名]が存在します。手順に記載のドメインと異なるドメインで URL アクセスした場合接続できないため、接続に失敗した場合はドメインが正しいことを確認してください。

3.1 認証方式の選択

本サービスでは、OAuth 2.0 による認証方式を利用可能です。

OAuth2.0 の認証方式のうち、以下アプリケーションの特性(利用シーンや実装方式など)に応じて 5 通りから選択してください。また、認証により払い出されたアクセストークンを利用して、各 API を実行できます。

表 3-1-1 OAuth 2.0 認証の種類

認証方式	概要	利用シーン	事前に必要なパラメータ
Authorization Code Grant	アプリケーションのアクセス要求に対し、利用者が認可サーバーの認証を受けて認可コードを取得する。アプリケーションがその認可コードを用いて、認可サーバーからアクセストークンを受け取る方式。認可サーバーがログイン画面を表示する。	信頼関係にない Web アプリケーションの認可。	client id (利用者キー) client secret (利用者秘密鍵)
Implicit Grant	アプリケーションのアクセス要求に対し、利用者が認可サーバーの認証を受けて、アクセストークンを取得する方式。 利用者の Web ブラウザへ通知されるリダイレクト URL(URI)にアクセストークンが含まれるため、セキュリティ強度が低い。	JavaScript など、パブリックプログラムの認可。	client id
Resource Owner Credentials Grant	アプリケーションに対し利用者が認証情報を提供し、アプリケーションが認可サーバーの認証を受けてアクセストークンを受け取る方式。 アプリケーションがログイン画面を表示する。	信頼関係(同一ドメイン内など)のある Web アプリケーションの認可。	client id client secret user id password
Client Credentials Grant	アプリケーション自身が認証情報を保持し認可サーバーの認証を受ける方式。利用者は認証情報(ユーザ ID やパスワード)を提供しない。	プログラム(バイナリ)の認可。	client id client secret
Reflesh Token Grant	Reflesh Token Grant を用いてアクセストークンを取得する方式。利用者は再度、認証情報(ユーザ ID やパスワード)を提供しない。	Authorization Code Grant、Resource Owner Credentials Grant で取得したアクセストークンが期限切れ、またはアクセストークンの更新が必要な場合。	Reflesh Token

※各認証で取得したアクセストークンの有効期限は 3600 秒が既定値。

3.2 アプリケーションの登録と設定

アプリケーションが本サービスの API の利用をする場合には、アプリケーションの登録と設定が必要となります。
登録準備のインプットとして以下情報を用意してください。

- アプリケーション名
- アプリケーションの説明
- 認証方式
- アプリケーションのコールバック URL(認証方式によっては必須)

アプリケーション登録後、アウトプットとして以下情報を提供します。以下情報は「3.3 API 認証(アクセストークンの取得)」で利用します。

- アプリケーションの利用者キー（client id）
- アプリケーションの利用者秘密鍵（client secret）

また、認証方式によってはロールが付与されている以下アカウント情報も別途必要な場合もあります。

- 利用者のユーザ ID（user id）
- 利用者のパスワード（password）

3.2.1 アプリケーションの登録

アプリケーションの登録方法を以下に記載します。

【手順】

1. ブラウザから、下記 URL にアクセスします。 AM

   https://[API Manager 用ドメイン名]/devportal/

2.画面右上の[SIGN-IN]をクリックします。

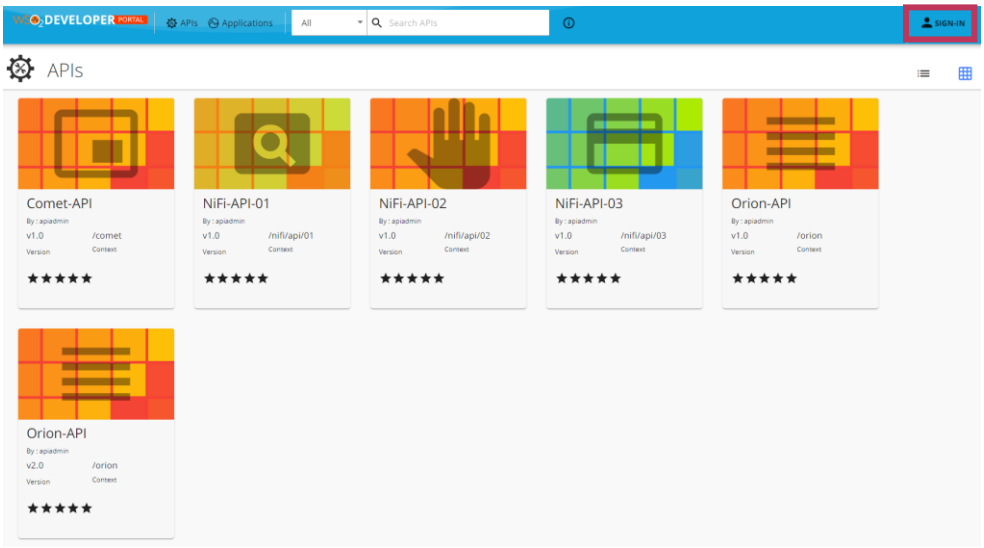

図 3-2-1 API DevPortal アクセス時

3.アプリケーションを管理するユーザでサインインします。
※管理者ユーザ名およびパスワードは別途入手してください。
※サブテナント管理者の場合は、ユーザ名の末尾に「@<テナント ID>」を付与したものを Username に入力してください。
例：unyo@city.pref

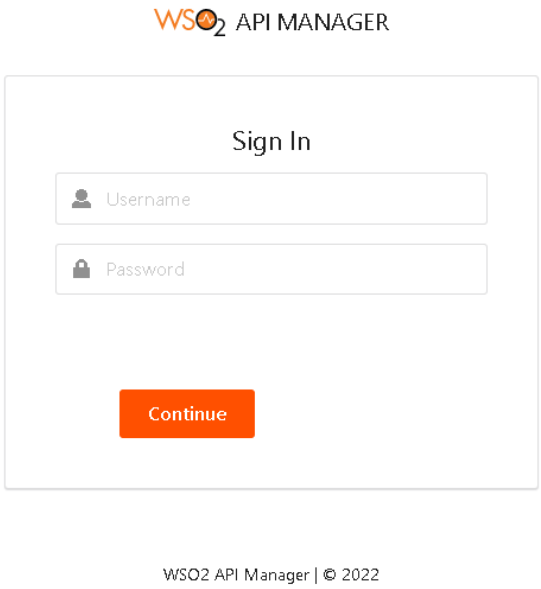

図 3-2-2 サインイン

4.メニューから「Applications」を選択します。

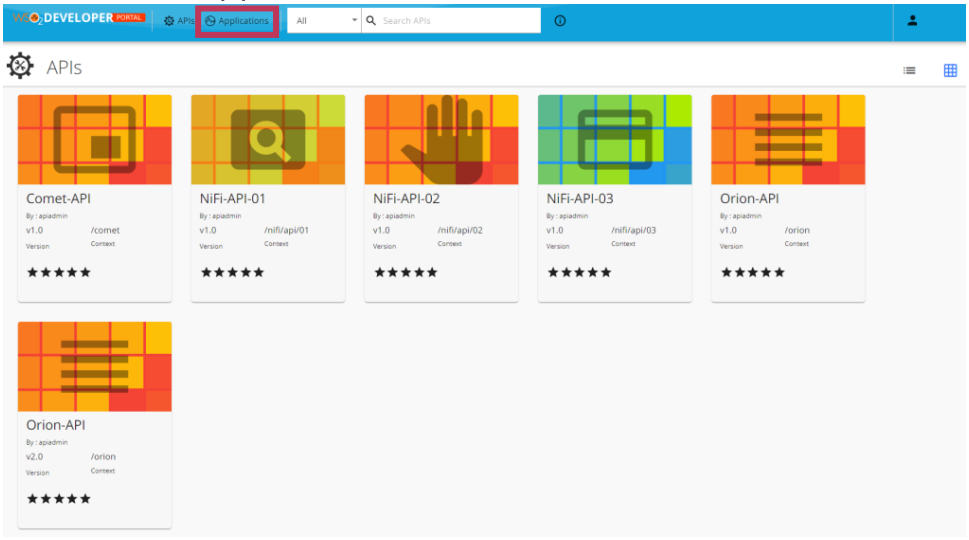

図 3-2-3 [アプリケーション]の選択

5.「ADD NEW APPLICATION」をクリックします。

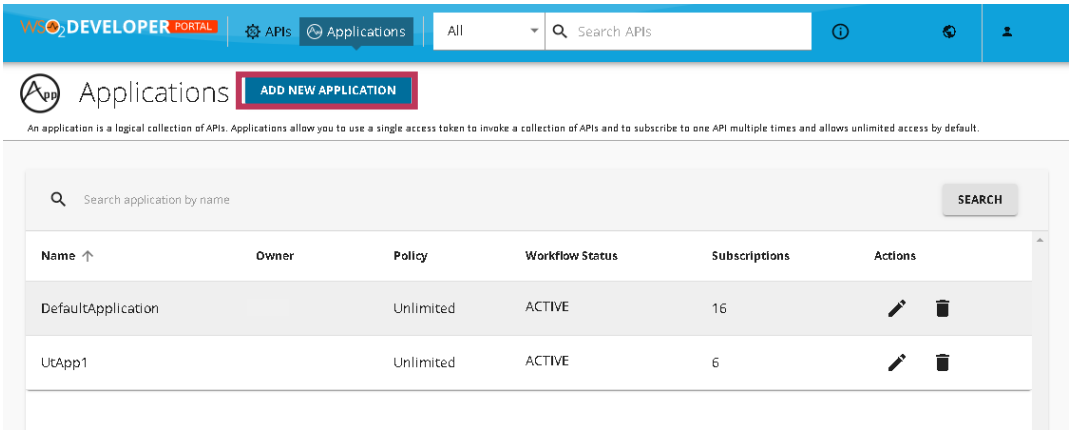

図 3-2-4 [ADD NEW APPLICATION]のクリック

6. 「Application Name」に追加するアプリケーション名を入力、「Per Token Quota.」でトークンごとに許容するリクエスト数を設定して、「SAVE」を選択します。

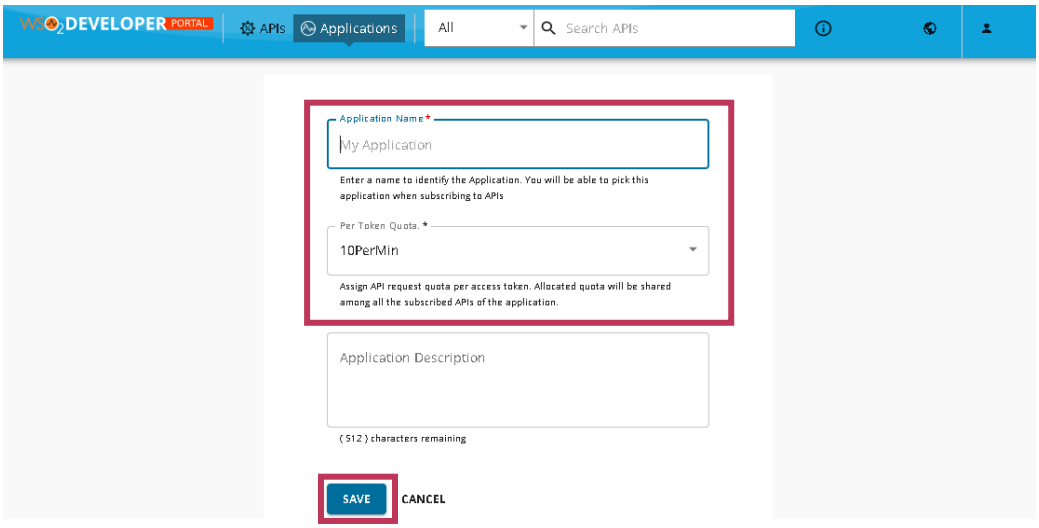

図 3-2-5 アプリケーションの登録

7.登録アプリケーションが表示されます。

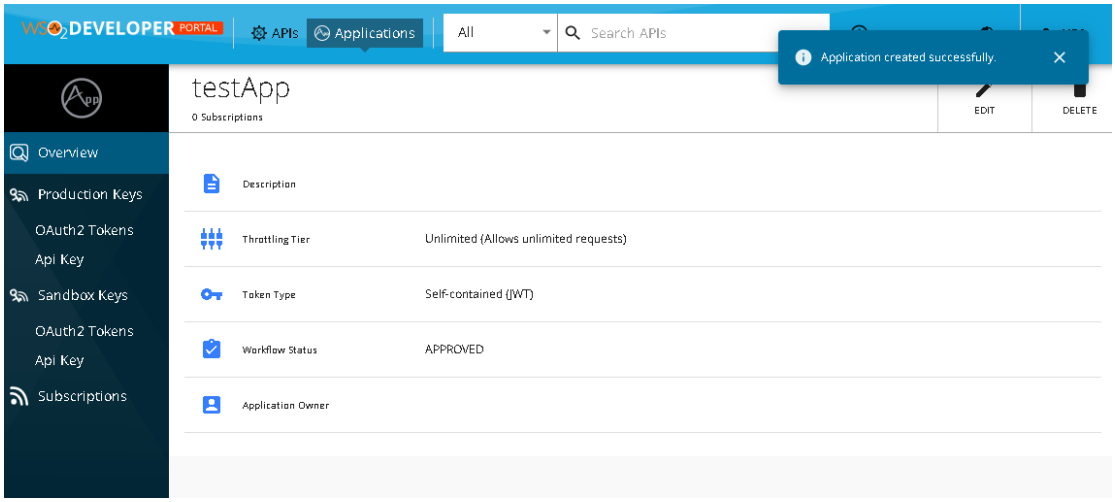

図 3-2-6 アプリケーションの登録確認

3.2.2 アプリケーションのアクセストークン有効期限変更

登録したアプリケーションのアクセストークンの有効期限を変更します。有効期限はデフォルトで 3600 秒(1 時間)です。変更が不要である場合は、本手順は実施不要です。

【手順】

1. ブラウザから、下記 URL にアクセスします。 IS

   https://[Identity Server 用ドメイン名]/ carbon/

2.アプリケーションを管理するユーザの「Username」と「Password」を入力して[Sign-in]をクリックします。
※サブテナントの場合はユーザ名の末尾に「@<テナント ID>」を付与したものを Username に入力してください。

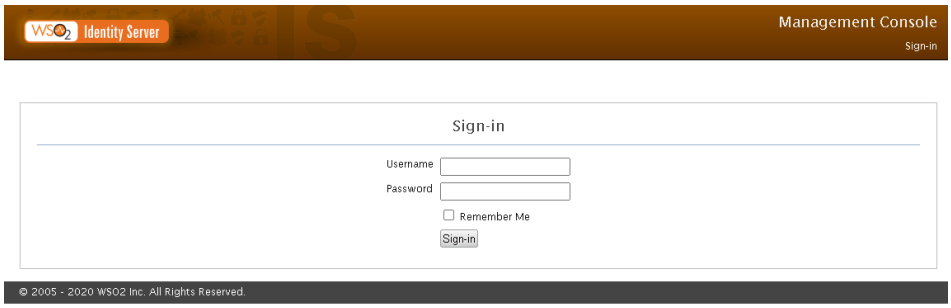

図 3-2-7 サインイン

3.「Service Providers」の[List]をクリックします。

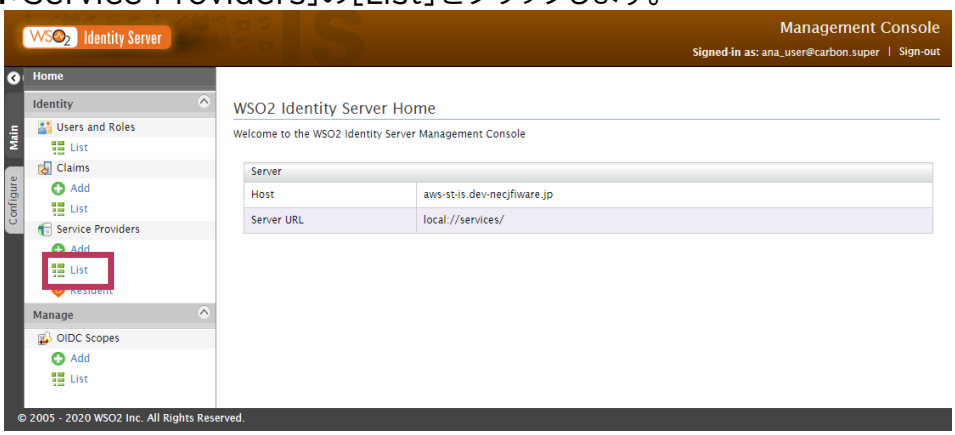

図 3-2-8 [List]のクリック

4.[<ユーザ名>_<作成したアプリケーション名>_PRODUCTION]の[Edit]をクリックします。

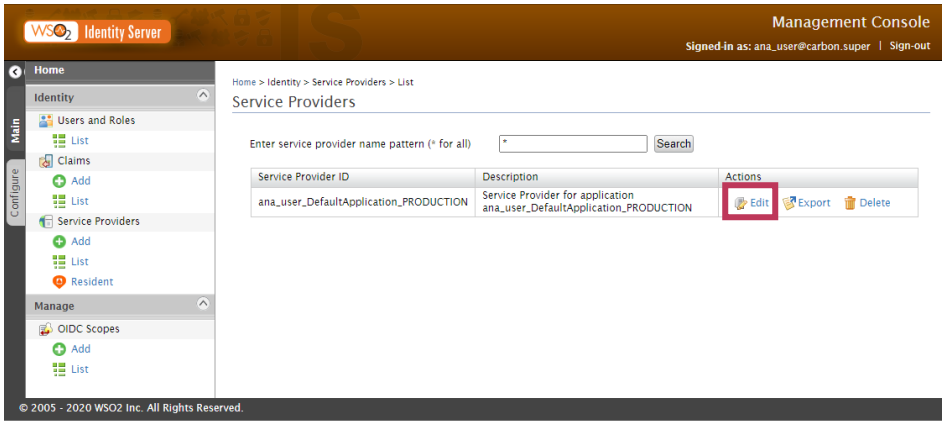

図 3-2-9[Edit]のクリック

5. [Inbound Authentication Configuration]-[OAuth/OpenID Connect Configuration]の[Edit]をクリックします。

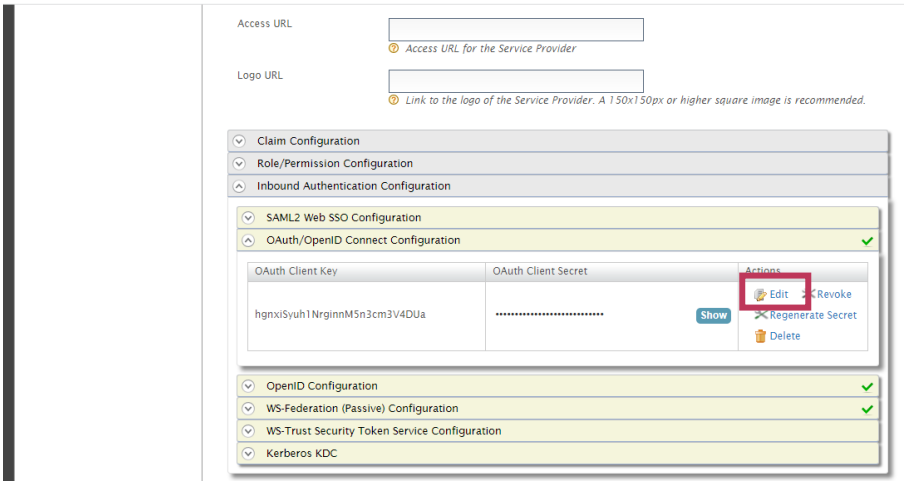

図 3-2-10[Edit]のクリック

6.「Application Access Token Expiry Time」の値を「2147483647」(second)に設定し、画面下部の「Update」をクリックします。

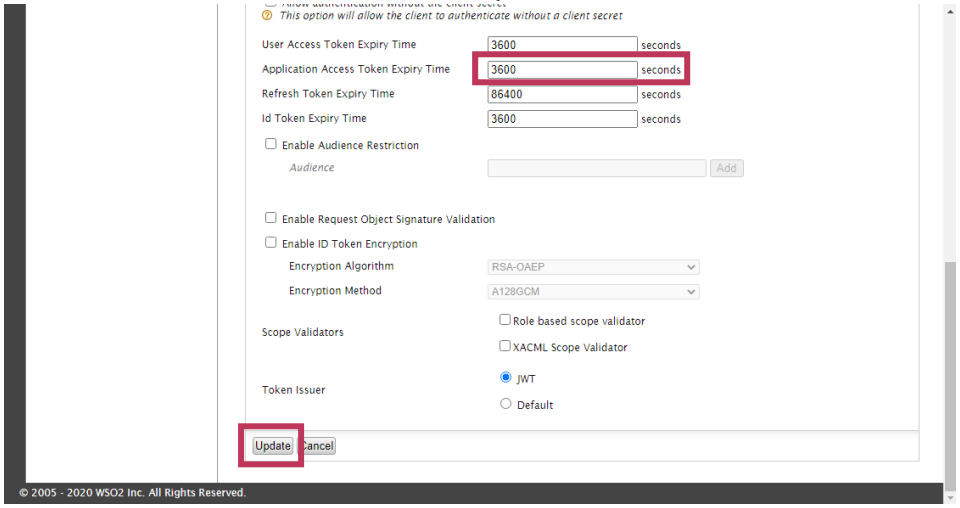

図 3-2-11 有効期限の変更

7. [OK]をクリックします。

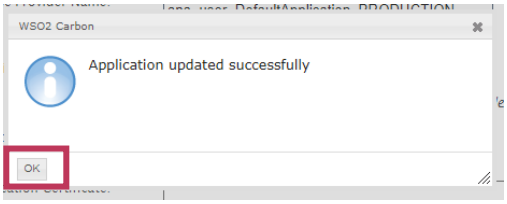

図 3-2-12 [OK]のクリック

8. 画面下部の[Update]をクリックします。

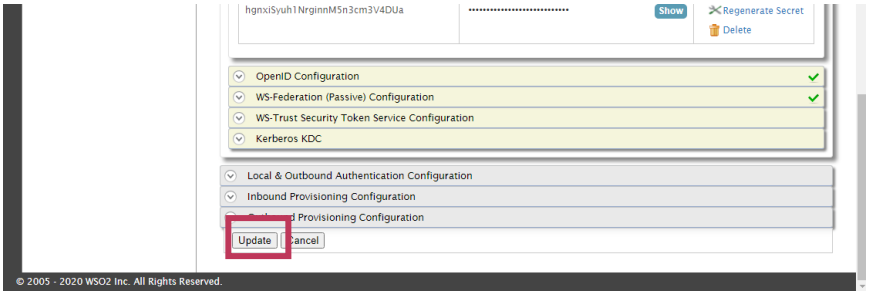

図 3-2-13 [Update]のクリック

3.2.3 アプリケーションの API 設定（Subscribe）

登録したアプリケーションに対して、アプリケーションが実行できる API を設定（Subscribe）します。Subscribe できる API の一覧を以下に示します。

表 3-2-1 API 一覧

表示名	説明
Orion-API	Orion を利用するための API
Comet-API	STH-Comet を利用するための API
NiFi-API-01	ApacheNiFi に登録されているプロセスグループ 01 を利用するための API
NiFi-API-02	ApacheNiFi に登録されているプロセスグループ 02 を利用するための API
NiFi-API-03	ApacheNiFi に登録されているプロセスグループ 03 を利用するための API

【手順】

1. 「3.2.1 アプリケーションの登録」の手順 1～4 を行います。 AM

2. 画面上から Subscribe する API のアイコンをクリックします。

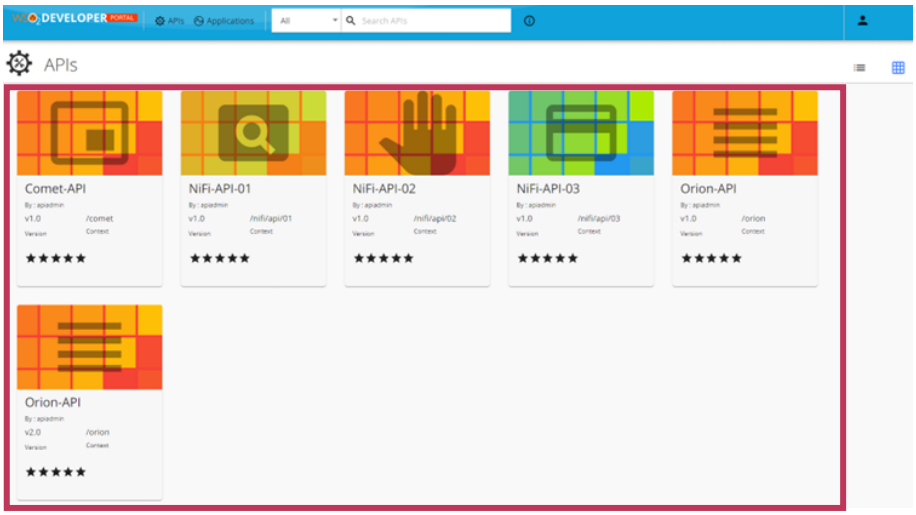

図 3-2-14 Subscribe する API のアイコンのクリック

3. 「Subscriptions」タブをクリックし、「Application」のプルダウンメニューから「3.2.1 アプリケーションの登録」で登録したアプリケーションを選択し、「Throttling Policy」で許容するリクエスト数を設定し、「SUBSCRIBE」をクリックします。

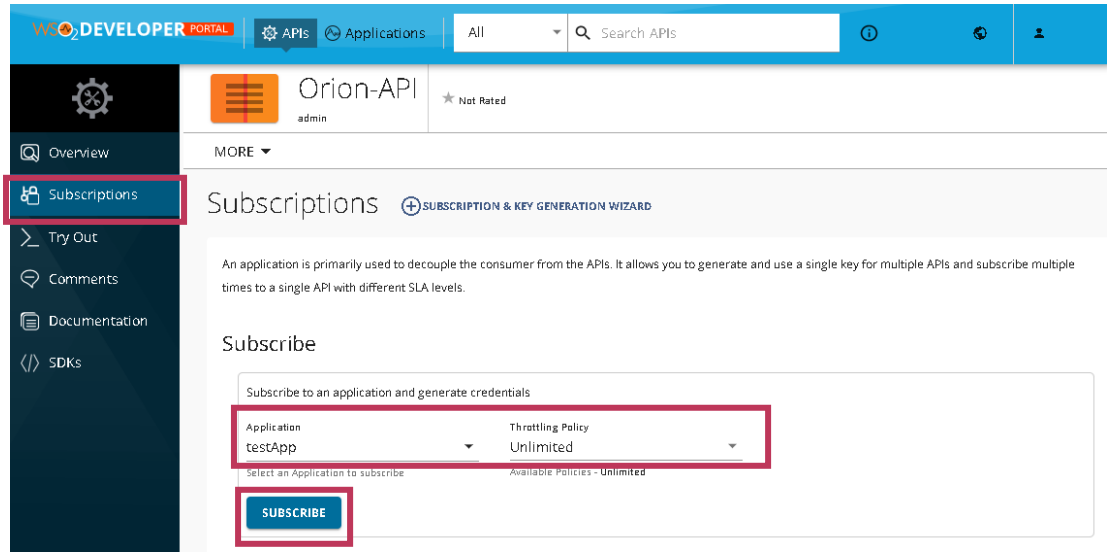

図 3-2-15 アプリケーションの選択

4. [Applications]をクリックします。

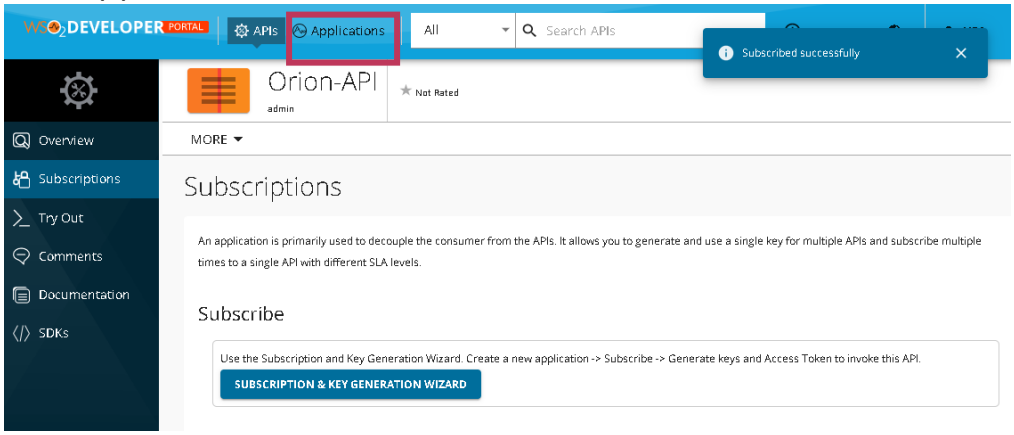

図 3-2-16 [Applications]のクリック

5.「3.2.1 アプリケーションの登録」で登録したアプリケーションを選択します。

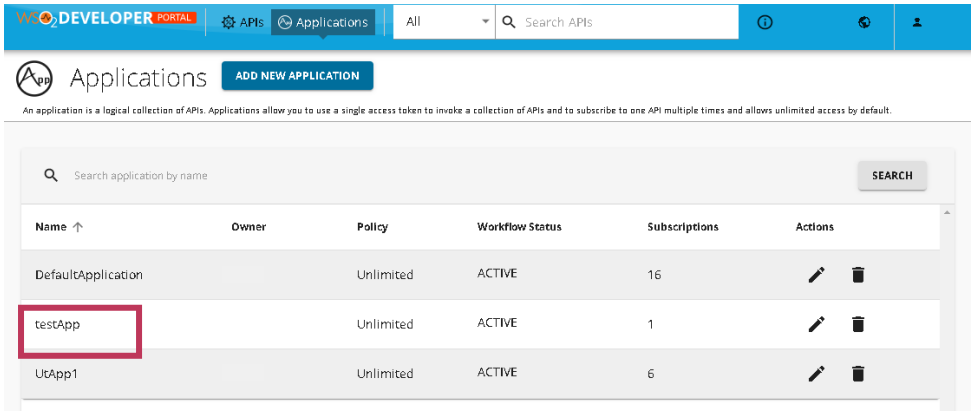

図 3-2-17 アプリケーションの選択

6.「Subscriptions」タブをクリックし、Subscribe した API があることを確認します。

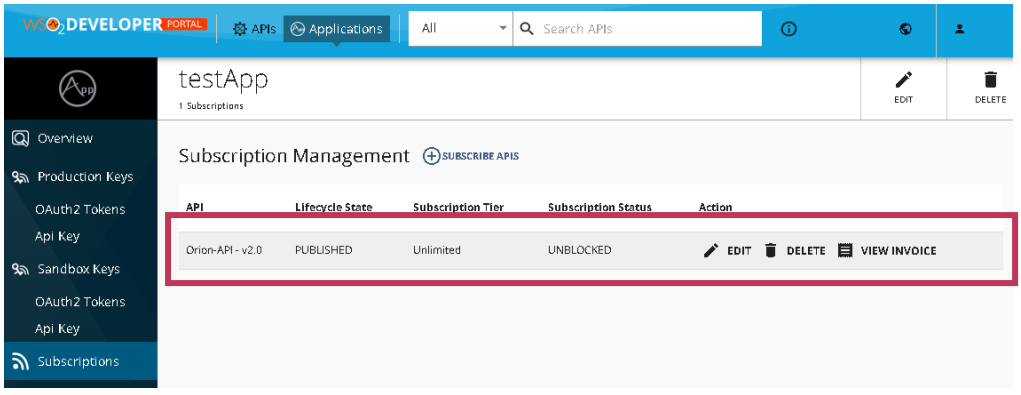

図 3-2-18 Subscribe した API の確認

7.他に Subscribe する API がない場合は、画面右上のアイコンから[Logout]をクリックして完了します。

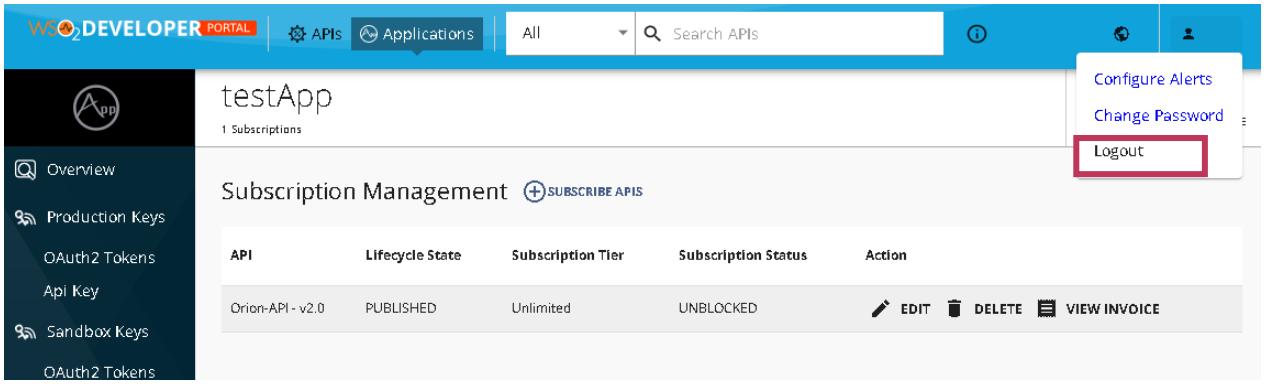

図 3-2-19 ログアウト

3.2.4 アプリケーションアクセス用キー／秘密鍵の生成

登録したアプリケーションへのアクセスに必要な利用者キーと利用者秘密鍵を生成します。

【手順】

1.「3.2.1 アプリケーションの登録」の手順 1～4 を行います。 AM

2.メニューから[Applications]をクリックします。

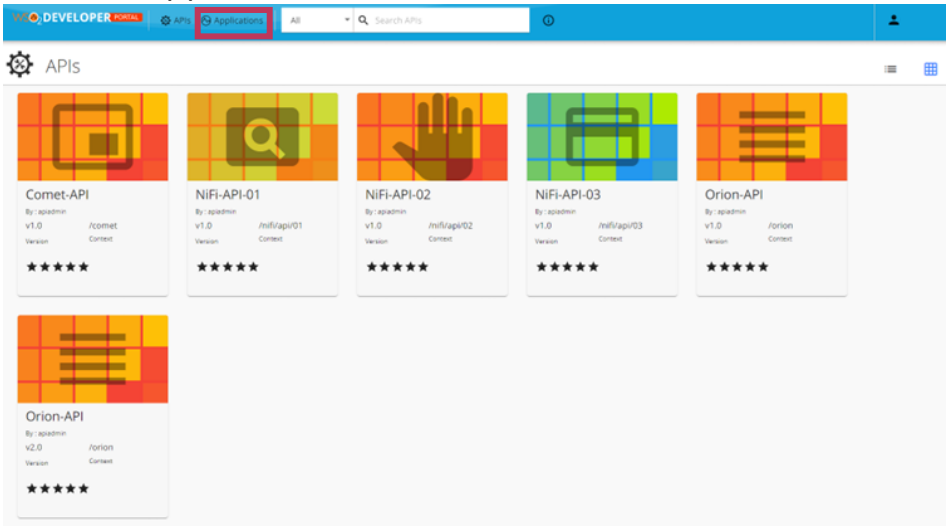

図 3-2-20 アプリケーションの選択

3.アプリケーション一覧から、登録したアプリケーションを選択します。

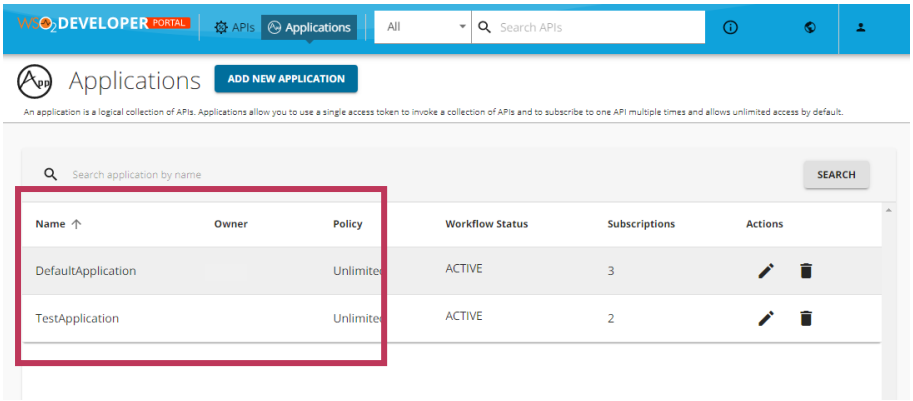

図 3-2-21 アプリケーションの選択

4.「Production Keys」タブをクリックする

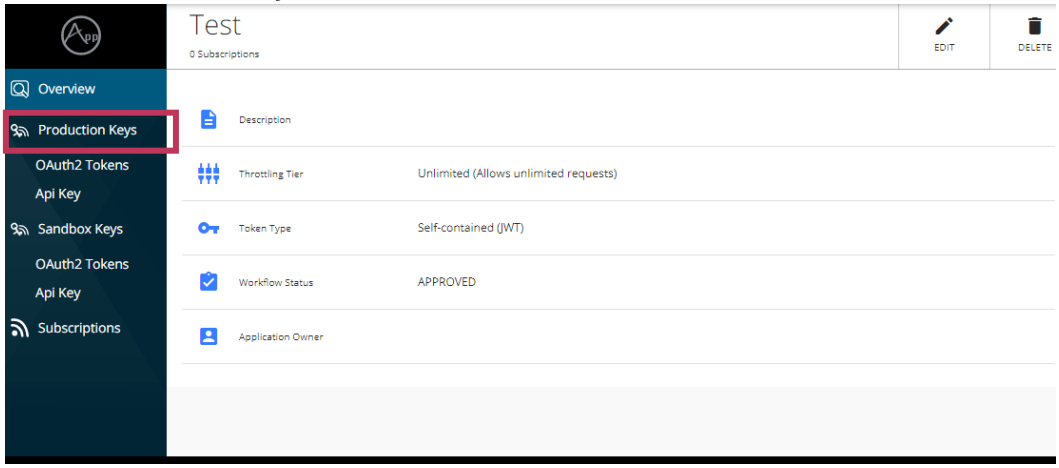

図 3-2-22 「Production Keys」のクリック

5.Grant Types の「SAML2」「IWA-NTLM」「JWT」のチェックを外し、表 3-2-2 からアプリケーションに必要な補助タイプのチェックを選択し、[GENERATE KEYS]をクリックすると、「アクセストークン(Access Token)」が生成されます。

重要

「Client Credential」の補助タイプを外した状態で[GENERATE KEYS]をクリックした場合は、「アクセストークン」が生成されません。この場合、「利用者キー」「利用者秘密鍵」は生成されていますので、「利用者キー」「利用者秘密鍵」を確認してください。

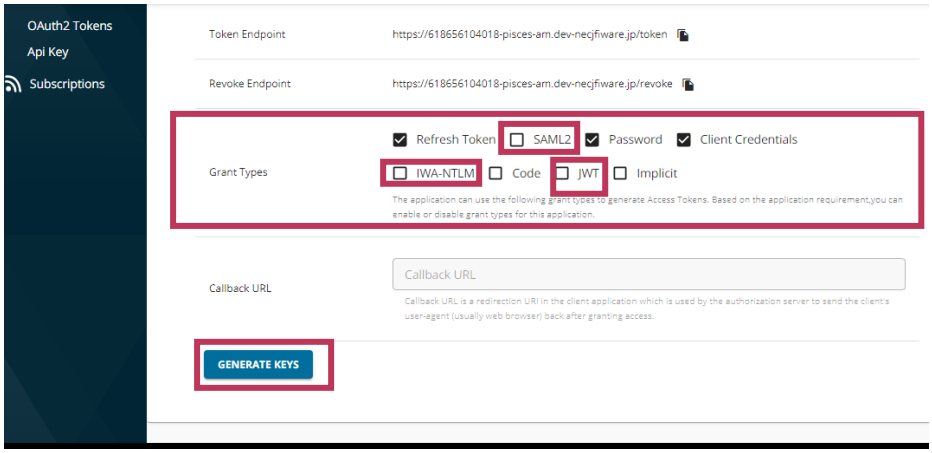

図 3-2-23 アクセストークンの生成

表 3-2-2 補助タイプ詳細

補助タイプ名	対応する Oauth2.0 認証タイプ *1	備考
Code	Authorization Code Grant	コールバック URL 必須
Implicit	Implicit Grant	コールバック URL 必須
Password	Resource Owner Credentials Grant	-
Client Credential	Client Credentials Grant	-
Refresh Token	Refresh Token Grant	-

*1 Oauth2.0 認証タイプについては、「表 3-1-1 OAuth 2.0 認証の種類」を参照。

6. 「アクセストークン」取得する場合は取得し、右下の[close]をクリックすると、「利用者キー（Consumer Key）」、「利用者秘密鍵（Consumer Secret）」が生成されます。

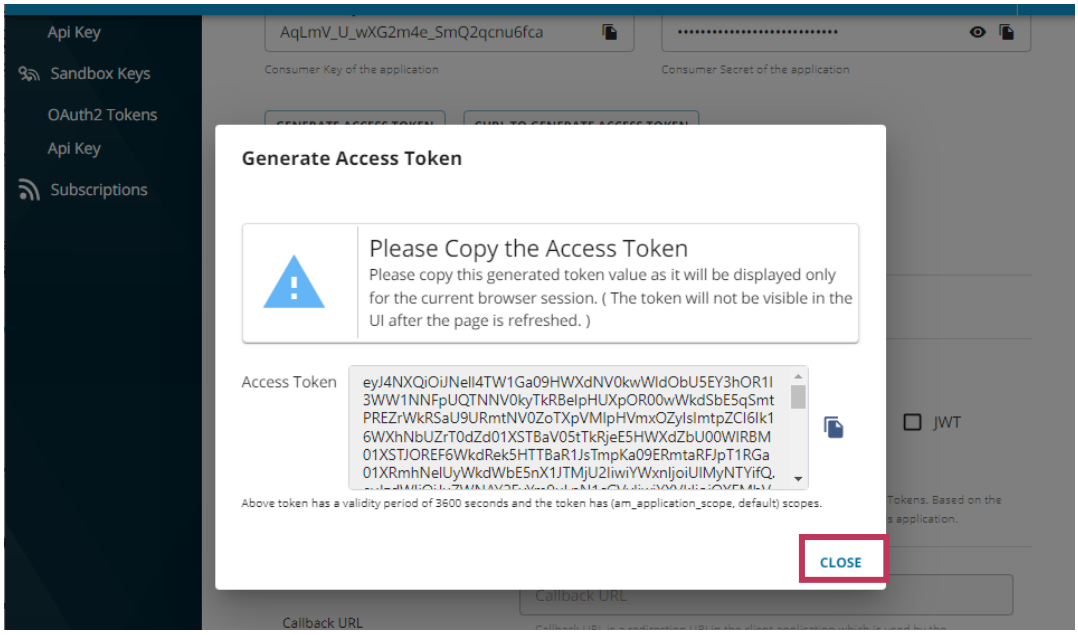

図 3-2-24 利用者キー/利用者秘密鍵の生成

7.「利用者キー(Consumer Key)」、「利用者秘密鍵(Consumer Secret)」の右にあるアイコンをそれぞれクリックし、文字列をクリップボードにコピーします。払い出した「利用者キー」、「利用者秘密鍵」を用いて API の呼び出しに必要なアクセストークンを取得します。

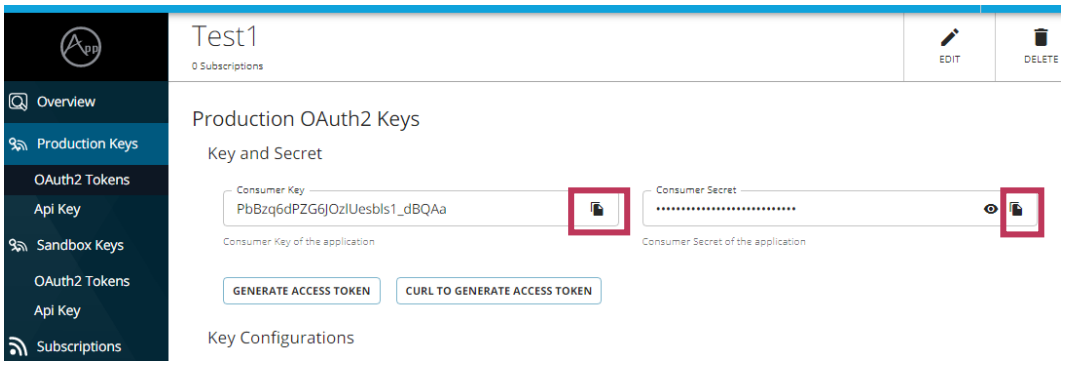

図 3-2-25 利用者キー/利用者秘密鍵の確認

8.生成後は画面右上のアイコンから[Logout]をクリックして完了します。

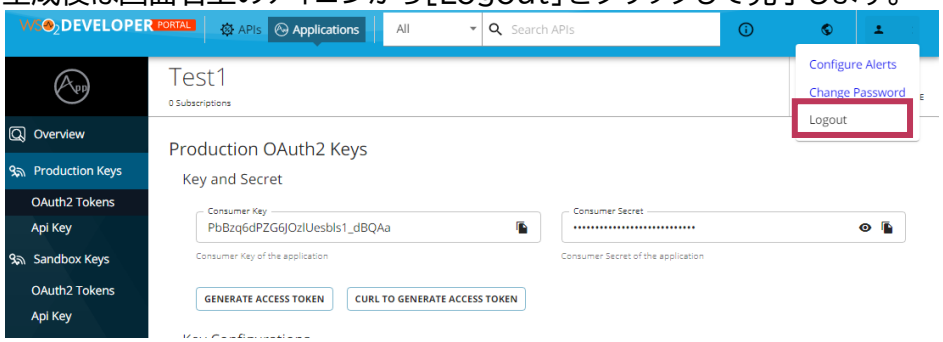

図 3-2-26 ログアウト

3.2.5 アプリケーションの補助タイプ変更

アプリケーションの補助タイプは、「3.2.4 アプリケーションアクセス用キー／秘密鍵の生成」でトークンを生成した後でも変更可能です。

例として、「3.2.1 アプリケーションの登録」で登録したアプリケーションの補助タイプを変更する手順を記載します。

【手順】

1.「3.2.4 アプリケーションアクセス用キー／秘密鍵の生成」の手順 1～4 を行います。

AM

2.表 3-2-2 の補助タイプから変更する補助タイプのチェックを選択し、[Update]をクリックします。

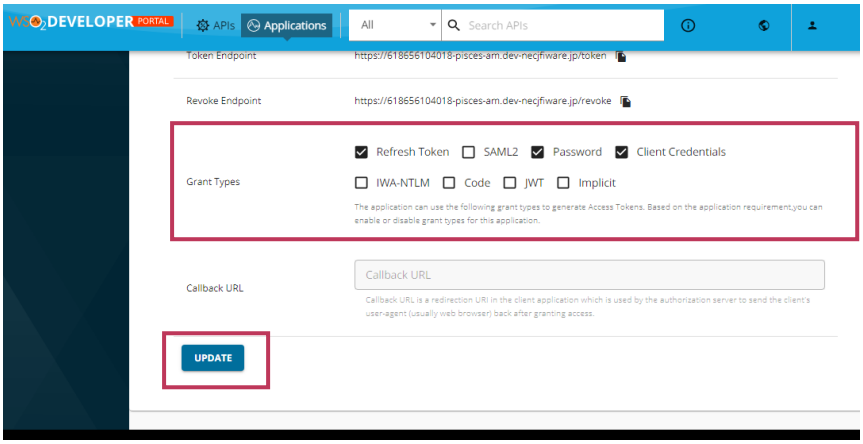

図 3-2-27 補助タイプの変更/更新

3.更新後は画面右上のアイコンから[Logout]をクリックして完了します。

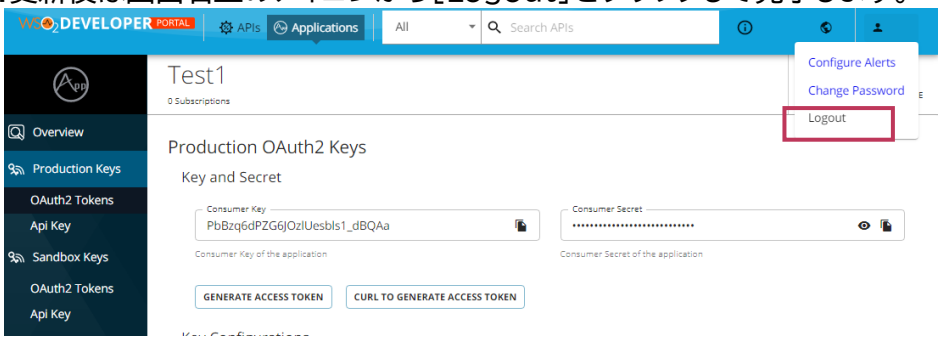

図 3-2-28 ログアウト

重要

実際の認証において、補助タイプが反映されるまで 15 分程度必要です。補助タイプの更新後、アクセストークンの取得は 15 分間の時間を置いてから実施してください。

3.2.6 アプリケーションの削除

アプリケーションを削除する方法を以下に記載します。

【手順】

1.「3.2.1 アプリケーションの登録」の手順 1～4 を行います。 AM

2.メニューから「Applications」を選択します。

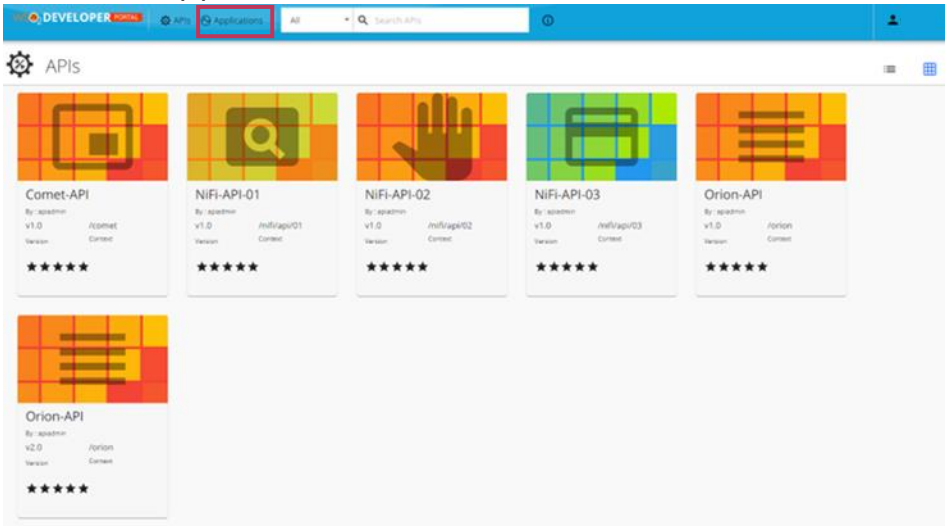

図 3-2-29 [Applications]の選択

3.削除するアプリケーションの行で[Delete]をクリックし、ダイアログで[DELETE]をクリックします。 AM

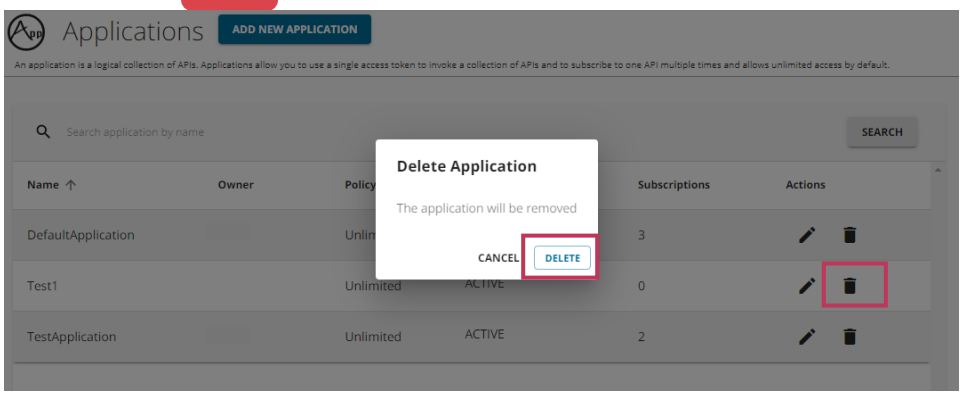

図 3-2-30 アプリケーションの削除

3.3 API 認証(アクセストークンの取得)

各 OAuth 2.0 認証の認証方式でのアクセストークンの取得方法について記載します。

3.3.1 Authorization Code Grant

Authorization Code Grant の実行例について記載します。

アプリケーションは以下の URL 呼び出し処理と、呼び出し後のリダイレクト先となるコールバック URL の受付け処理と、アクセストークン取得処理を実装してください。

<リクエスト>
URL:

https://{API Manager 用ドメイン名}/oauth2/authorize
Method:　GET

表 3-3-1 Authorization Code Grant の認証・認可のリクエストクエリパラメータ

パラメータ名	説明
scope	リクエストで要求するアクセス権のスコープを指定する。 ※"default" を指定。
response_type	Authorization Code Grant の場合は、"code" を指定する。
redirect_uri	「3.2 アプリケーションの登録と設定」で指定したアプリケーションのコールバック URL(URI)を指定する。
client_id	払い出されたアプリケーションの Consumer Key(client id)を指定する。

<レスポンス>
ブラウザで上記 URL を呼び出した際、認証画面が表示されます。
以下の手順でその画面から認証・認可を実施してください。

①認証・認可

認証・認可実施の処理は本サービス側で用意しているため、アプリケーション開発者が実装する必要はありません。
認証・認可の実行手順を以下に示します。

1. 認証画面が表示されるため、Username、Password を入力して、「Continue」をクリックします。

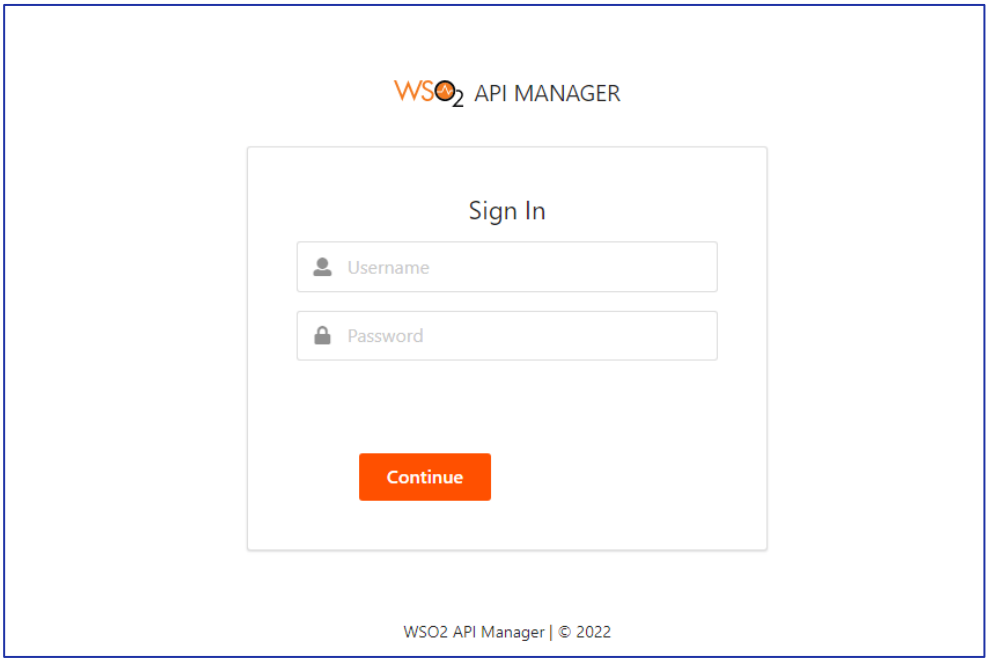

図 3-3-1 WSO2 が用意する認証画面

2. 承認画面が表示されるため、「Continue」をクリックします。
※「Approve Always」を選択した場合 2 回目以降は表示されません。

図 3-3-2 WSO2 が用意する承認画面

3. 承認後は、アプリケーションのコールバック URL へリダイレクトします。

また、コールバック URL のクエリパラメータには認可コードが付与されます。
その認可コードを用いて認証・認可処理後に次のトークン取得処理を実装してください。

＜レスポンス＞

表 3-3-2 Authorization Code Grant の認証・認可のリクエストクエリパラメータ

パラメータ名	説明
code	認可コードを返却する。

※認可コードの有効期限は 600 秒固定

※サンプル

http://{コールバック URL}?code=853312a5-9fd0-3b6e-99d7-f361b77db647

②アクセストークン取得

API 呼び出し時に必要なアクセストークンを取得します。
認証・認可処理後に呼び出されるアプリケーションのコールバック URL で以下の処理を実施してください。
アクセストークン取得のためのアクセス先 URL、リクエスト、レスポンス例を以下に示します。

①認証・認可で取得した認可コードを使用して、アクセストークンを取得します。

＜リクエスト＞

URL:

https://{API Manager 用ドメイン名}/oauth2/token

リクエストヘッダ：

Content-Type: application/x-www-form-urlencoded

Method: POST

表 3-3-3 Authorization Code Grant のトークン取得のリクエストクエリパラメータ

パラメータ名	説明
code	取得した認可コードを指定する。
grant_type	Authorization Code Grant の場合は、”authorization_code”を指定する。
client_secret	払い出されたアプリケーションの利用者秘密鍵（client secret）を指定する。
redirect_uri	アプリケーションのコールバック URL（URI）を指定する。
client_id	払い出されたアプリケーションの利用者キー（client id）を指定する。

<レスポンス>
レスポンスヘッダ:
Content-Type: application/json

表 3-3-4 Authorization Code Grant のトークン取得のレスポンスパラメータ

パラメータ名	説明
scope	リクエストパラメータで指定したスコープの値を返却する。
token_type	”Bearer”を返却する。
expires_in	トークンの有効期限(秒)を返却する。
refresh_token	リフレッシュトークンを返却する。
access_token	アクセストークンを返却する。

※サンプル

```
(curl -k -v -X POST https://{API Manager 用ドメイン
  名}/oauth2/token -d”code=cd73401b-8769-35d7-b480-
  130f76240b70&grant_type=authorization_code&client_sec
  ret=86RUHPsmhkJXTy3reh3p12_4ajoa&redirect_uri={コー
  ルバック
  URL}&client_id=NPp17brQhfDAEQdU4rfDLV9M3Swa”|
  python -mjson.tool)
{
  ”access_token”:”59fb16ac-e5ac-3633-981a-
c375a499adf8”,
  ”refresh_token”:”a73daa41-0015-3968-b2bd-
84165efb18e1”,
  ”scope”:”default”,
  ”token_type”:”Bearer”,
  ”expires_in”:3600
}
```

3.3.2 Implicit Grant

Implicit Grant の実行例について記載します。
開発するアプリケーションで認証が必要な場合、以下の URL にリダイレクトする処理を実装してください。
＜リクエスト＞
URL:
https://{API Manager 用ドメイン名}/oauth2/authorize
Method:　GET

表 3-3-5 Implicit Grant のトークン取得のリクエストクエリパラメータ

パラメータ名	説明
scope	リクエストで要求するアクセス権のスコープを指定する。 ※”default” を指定。
response_type	Authorization Code Grant の場合は、”token” を指定する。
redirect_uri	「3.2 アプリケーションの登録と設定」で指定したアプリケーションのコールバック URL(URI)を指定する。
client_id	払い出されたアプリケーションの Consumer Key(client id)を指定する。

＜レスポンス＞
ブラウザに認証画面が表示されます。
以降の手順でその画面から認証・認可を実施してください。

①認証・認可とアクセストークン取得

認証画面より、認証・認可を実施します。
「3.3.1 Authorization Code Grant」の認証・認可と同様の操作を実施してください。
ただし、Implicit Grant の場合はコールバック URL のクエリパラメータに直接アクセストークンが付与されて返却されます。

＜レスポンス＞

表 3-3-6 Implicit Grant のトークン取得のレスポンスクエリパラメータ

パラメータ名	説明
scope	リクエストパラメータで指定したスコープの値を返却する。
token_type	”Bearer”を返却する。
expires_in	トークンの有効期限(秒)を返却する。
access_token	アクセストークンを返却する。

※サンプル

```
http://{コールバック URL} #access_token=4e3f38e6-e2e8-3e12-b2eb-8564e3592141&token_type=Bearer&expires_in=3600&scope=default
```

3.3.3 Resource Owner Credentials Grant

Resource Owner Credentials Grant の実行例について記載します。

アクセストークン取得のための、アクセス先 URL、リクエスト、レスポンス例を以下に示します。

＜リクエスト＞

URL:

https://{API Manager 用ドメイン名}/oauth2/token

リクエストヘッダ:

Content-Type: application/x-www-form-urlencoded

Method:　POST

表 3-3-7 Resource Owner Credentials Grant のリクエストパラメータ

パラメータ名	説明
grant_type	Resource Owner Credentials Grant の場合は、”password”を指定する。
username	ユーザ名を指定する。
password	パスワードを指定する。
client_secret	払い出されたアプリケーションの利用者秘密鍵（client secret）を指定する。
client_id	払い出されたアプリケーションの利用者キー（client id）を指定する。

＜レスポンス＞

レスポンスヘッダ:

Content-Type: application/json

表 3-3-8 Resource Owner Credentials Grant のレスポンスパラメータ

パラメータ名	説明
scope	リクエストパラメータで指定したスコープの値を返却する。
token_type	”Bearer”を返却する。
expires_in	トークンの有効期限（秒）を返却する。
refresh_token	リフレッシュトークンを返却する。
access_token	アクセストークンを返却する。

※サンプル

```
(curl -k -v -X POST https://{API Manager 用ドメイン
  名}/oauth2/token -
  d"grant_type=password&username=tutorial&password=mW8
  Hatgj&client_id=NPp17brQhfDAEQdU4rfDLV9M3Swa&client_s
  ecret=86RUHPsmhkJXTy3reh3p12_4ajoa" | python -
  mjson.tool)
{
  "access_token":"e2235bbb-7981-35ab-a06e-f50f7ee99034",
  "refresh_token":"c8fc826c-c3da-3e8b-80ad-2d5e8d380c37",
  "scope":"default",
  "token_type":"Bearer",
  "expires_in":3600
}
```

3.3.4 Client Credentials Grant

Client Credentials Grant の実行例について記載します。
アクセストークン取得のための、アクセス先 URL、リクエスト、レスポンス例を以下に示します。

＜リクエスト＞

URL:

https://{API Manager 用ドメイン名}/oauth2/token

リクエストヘッダ：

Content-Type: application/x-www-form-urlencoded

Method:　POST

表 3-3-9 Client Credentials Grant のリクエストパラメータ

パラメータ名	説明
scope	リクエストで要求するアクセス権のスコープを指定する。 ※"default" を指定。
grant_type	Client Credentials Grant の場合は、"client_credentials" を指定する。
client_secret	払い出されたアプリケーションの利用者秘密鍵(client secret)を指定する。
client_id	払い出されたアプリケーションの利用者キー(client id)を指定する。

＜レスポンス＞

レスポンスヘッダ：

Content-Type: application/json

表 3-3-10 Client Credentials Grant のレスポンスパラメータ

パラメータ名	説明
scope	リクエストパラメータで指定したスコープの値を返却する。
token_type	”Bearer”を返却する。
expires_in	トークンの有効期限(秒)を返却する。
access_token	アクセストークンを返却する。

※サンプル

```
(curl -k -v -X POST ”https://{API Manager 用ドメイン
  名}/oauth2/token” -
  d”grant_type=client_credentials&client_secret=86RUHPsmhkJ
  XTy3reh3p12_4ajoa&client_id=NPp17brQhfDAEQdU4rfDLV9M
  3Swa&scope=default” | python -mjson.tool)
{
  ”access_token”:”58ef153d-cc64-3fd4-8b22-e169783cbc3f”,
  ”scope”:”am_application_scope default”,
”token_type”:”Bearer”,
  ”expires_in”:3600
}
```

3.3.5 Reflesh Token Grant

Reflesh Token Grant の実行例について記載します。
払い出したアクセストークンの有効期限を延長・再発行します。
「3.3.1 Authorization Code Grant」、「3.3.3 Resource Owner Credentials Grant」でのみ利用可能です。

アクセストークン取得のための、アクセス先 URL、リクエスト、レスポンス例を以下に示します。

＜リクエスト＞
URL:
https://{API Manager 用ドメイン名}/oauth2/token

リクエストヘッダ:
Content-Type: application/x-www-form-urlencoded
0Method:　POST

表 3-3-11 Client Credentials Grant のレスポンスパラメータ

パラメータ名	説明
grant_type	Reflesh Token Grant の場合は、”refresh_token”を指定する。
client_secret	払い出されたアプリケーションの利用者秘密鍵(client secret)を指定する。
refresh_token	「3.3.1 Authorization Code Grant」または「3.3.2 Implicit Grant」の認証時に払い出された refresh_token の値を指定する。
client_id	払い出されたアプリケーションの利用者キー(client id)を指定する。

<レスポンス>

レスポンスヘッダ:

Content-Type: application/json

表 3-3-12 Client Credentials Grant のレスポンスパラメータ

パラメータ名	説明
scope	リクエストパラメータで指定したスコープの値を返却する。
token_type	”Bearer”を返却する。
expires_in	トークンの有効期限(秒)を返却する。
refresh_token	リフレッシュトークンを返却する。
access_token	アクセストークンを返却する。

※サンプル

```
(curl -v -k -H”Content--Type: application/x--www--form--
  urlencoded” -XPOST ”https://{API Manager 用ドメイン
  名}/oauth2/token” -
  d”grant_type=refresh_token&refresh_token=f751718b-e58f-
  3ebb-b0ac-
  75f17b701e5d&client_secret=86RUHPsmhkJXTy3reh3p12_4ajoa
  &client_id=NPp17brQhfDAEQdU4rfDLV9M3Swa”｜python -
  mjson.tool)
{
  ”access_token”:”59fb16ac-e5ac-3633-981a-c375a499adf8”,
  ”refresh_token”:”a73daa41-0015-3968-b2bd-84165efb18e1”,
  ”scope”:”default”,
  ”token_type”:”Bearer”,
  ”expires_in”:3600
}
```

3.4 API の呼び出し

いずれかの認証方式で正しく認証すると、アクセストークン文字列が返却されます。
例)4e88f99fa193bafbeb41c528b9b9e070

API 呼び出し時に、リクエストヘッダにその取得したアクセストークン文字列を付与してください。

表 3-4-1 API の呼び出し方法

HTTP メソッド	利用したい API の HTTP メソッド ※GET、POST、PUT など
URL	利用したい API の URL 例）https://{API Manager 用ドメイン名}/orion/v2.0/entities
ヘッダ	Content-Type: application/json Authorization: Bearer {3.3 で取得したアクセストークン} or X-Auth-Token: {3.3 で取得したアクセストークン}

3.4.1 curl 実行例

curl コマンドを利用した API 呼び出し方法を以下に示します。

※サンプル

```
(curl -k -X GET "https://{API Manager 用ドメイン名}/orion/v2.0/entities"
  -s -S ¥
 --header "Accept: application/json" ¥
 --header "Authorization: Bearer {3.3 で取得したアクセストークン}" ¥
 | python -mjson.tool)
[
  {
    "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"
        }
      }
    }
  }
]
```

3.4.2 API DevPortal 実行例

API DevPortal を利用した API 呼び出し方法を以下に示します。

1. ブラウザから、以下 URL にアクセスします。 AM

   https://{API Manager 用ドメイン名}/devportal/

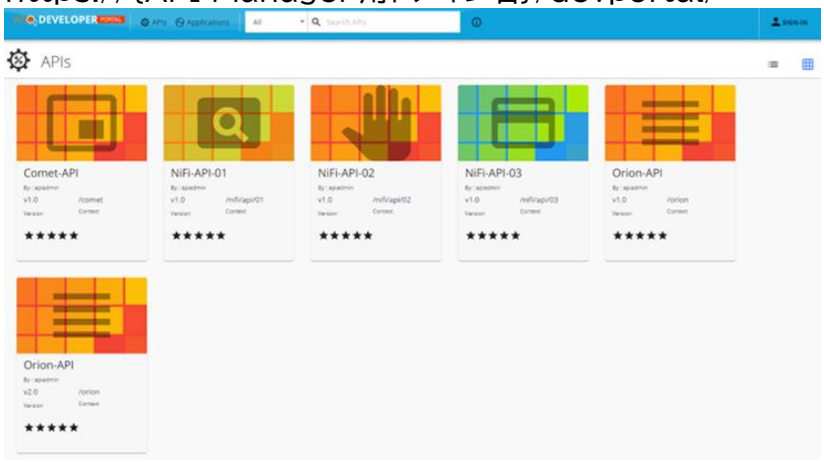

図 3-4-1 API DevPortal アクセス時

2.実行したい API コンポーネントを選択します。

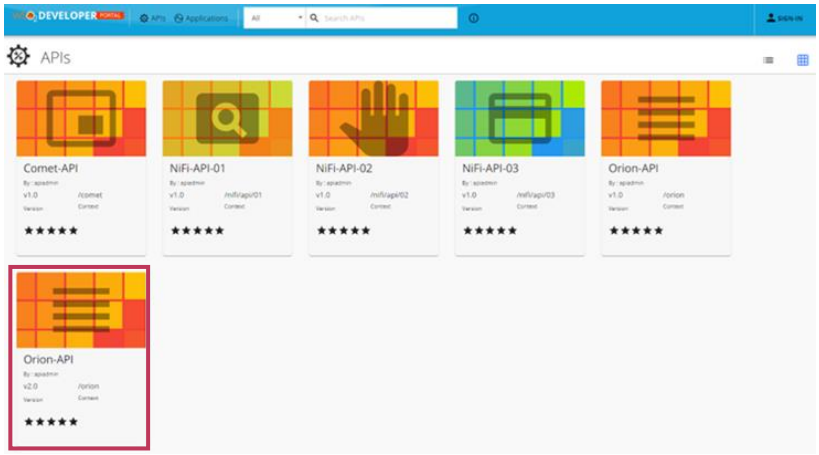

図 3-4-2 API コンポーネントの選択

3.「Try Out」を選択します。

選択すると、API コンポーネントの API 一覧が表示されます。

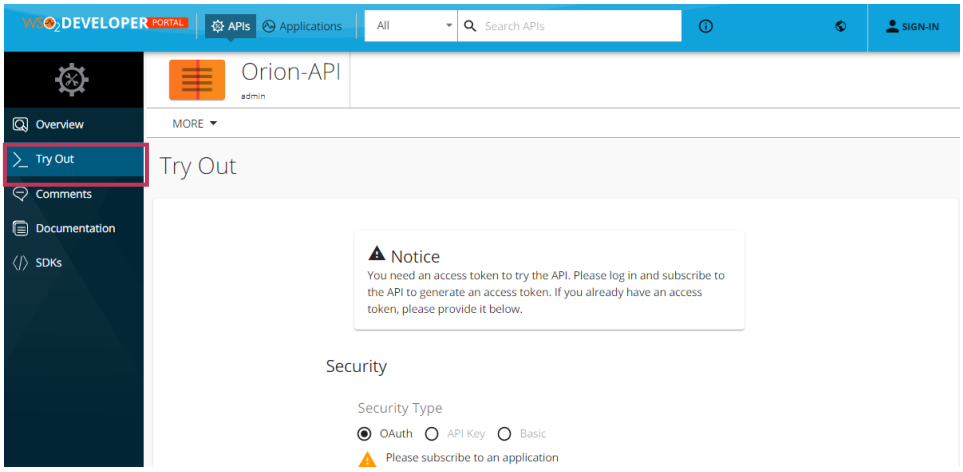

図 3-4-3 API コンソールの指定

4.実行する API を選択します。

選択すると各パラメータなどの詳細情報を表示します。

図 3-4-4 API 仕様確認

表 3-4-2 API DevPortal で指定するパラメータ

パラメータ名	説明
Name	パラメータ情報を表示します。
Description	パラメータに指定する値の説明を表示します。 設定する値が配列など特殊な形式の場合、形式について詳細情報の記載もあります。その場合「Model」を選択すると形式の説明を表示し、「Example Value」を選択するとパラメータ例を表示します。

5. 3.3 で取得したアクセストークンを指定します。

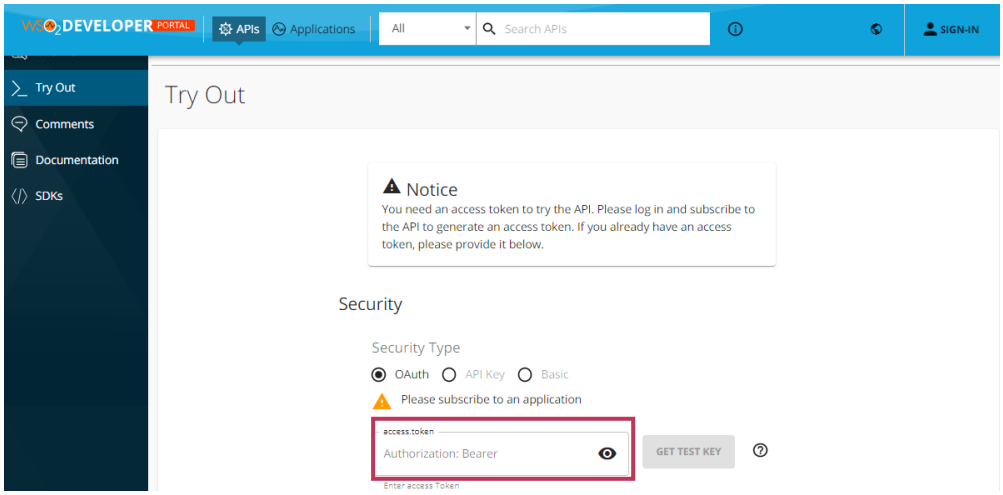

図 3-4-5 アクセストークンの指定

6.入力欄に必要なパラメータの値を指定して「Execute」をクリックすると API を実行します。

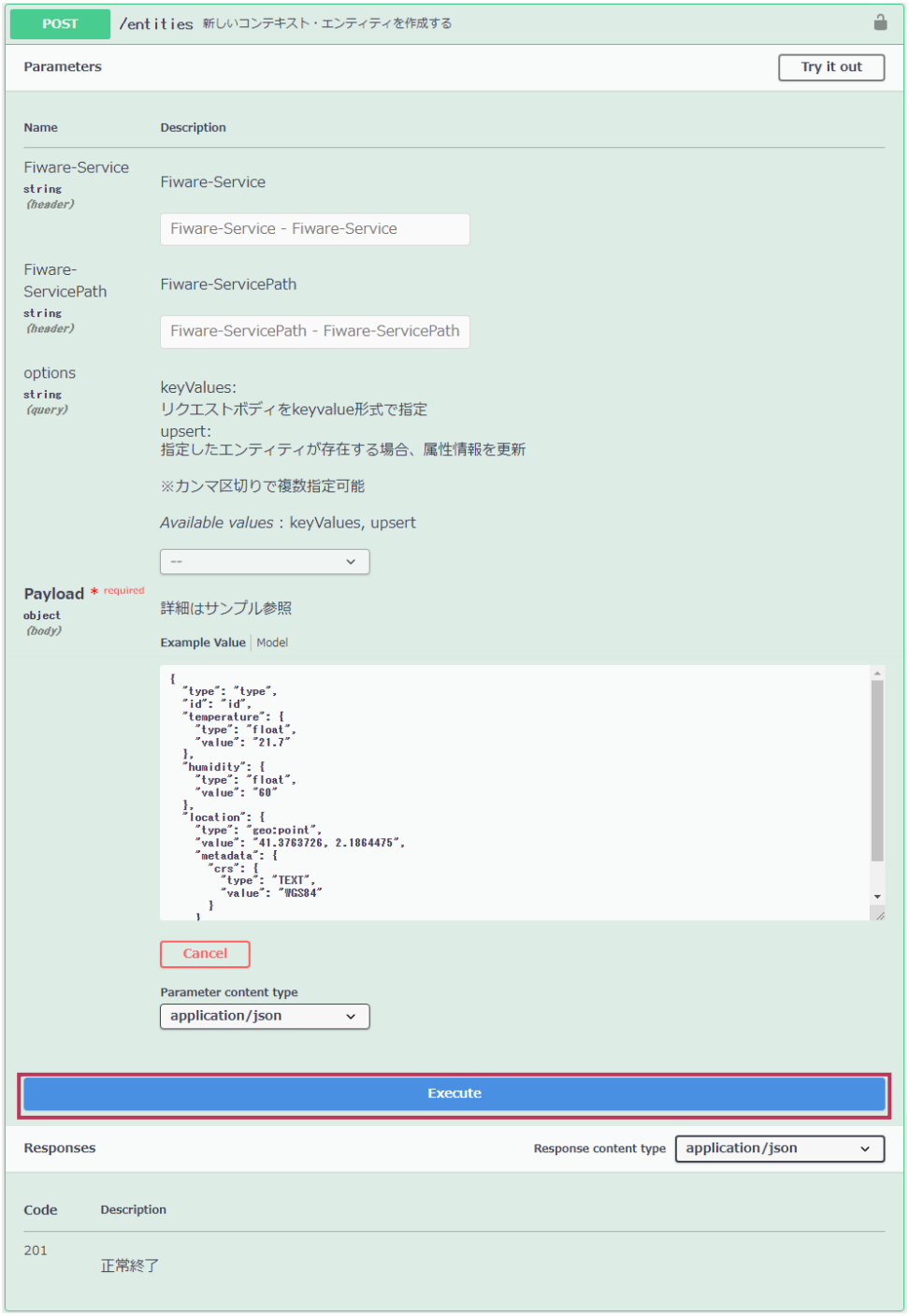

図 3-4-6 API 実行

※実行すると以下のように実行結果が返却されます。

Response Code が 20x であれば API の実行は成功しています。Response Body の返却値を確認してください。

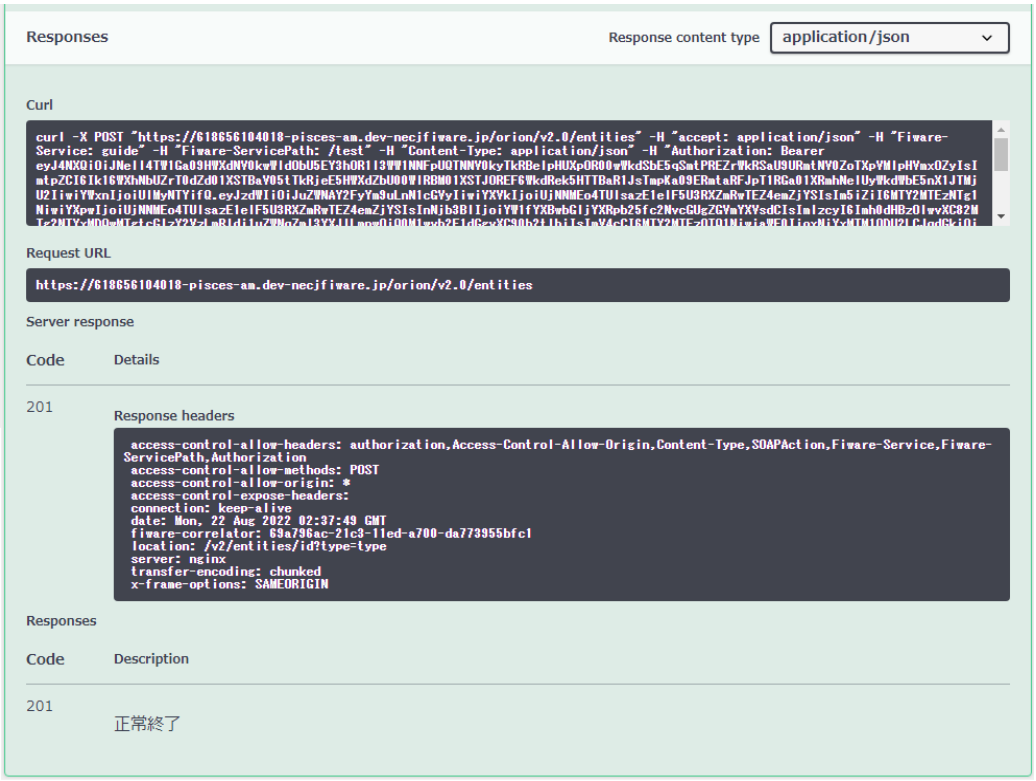

図 3-4-7 API の実行結果