嬬恋スマートシティ

アプリケーション開発ガイド
認証認可編

嬬恋村

2021 年 3 月

目　　次

1. はじめに

1.1. 本書の位置づけ

嬬恋スマートシティは、嬬恋村に関するデータを利活用することで住民、別荘居住者、観光客の満足度を向上させることを目的として、以下の 2 点を実現するサービスです。

- 嬬恋村に関するデータを嬬恋村統合データベース上に集約します。
- ホームページ、SNS、API を通じて職員、住民、事業者で利活用できるようにします。

本書は、嬬恋スマートシティの API サービスの利用者に、読んでいただくドキュメントになります。本書では、嬬恋村スマートシティにおけるアプリケーション開発ガイドの「認証認可」について説明したものです。「認証認可」には以下の 2 つの機能があります。

1. OAuth 2.0 認証の利用。アプリケーションへアクセス時やログイン時の OAuth 2.0 認証の利用方法を説明します。
2. API の呼び出し。公開する API の呼び出しかたを説明します。

1.2. 嬬恋スマートシティのサービスの概要

サービスの概要については、「嬬恋スマートシティ スタートアップガイド」をご参照ください。

2. API の利用準備

アプリケーションが嬬恋スマートシティの API の利用をする場合には、事前準備が必要になります。 利用準備のインプットとして下記情報を用意してください。詳細はシステム管理者に問い合わせをしてください。

なお、API の利用準備の手順は『嬬恋村スマートシティ サービス利用ガイド』に記載されています。

- アプリケーション名
- アプリケーションのコールバック URL(OAuth 2.0 を使用する場合は必須)
- アプリケーションの説明
- アプリケーションで使用する API コンポーネント

API の利用準備のアウトプットとして下記情報が提供されます。

下記情報は 3.2 の手順を実施するときに必要です。

- 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、匿名ユーザーでアクセスする場合には下記情報が併せて提供されます。下記情報は 3.3 の手順を実施するときに必要です。

- 匿名アクセスユーザーのユーザー名
- 匿名アクセスユーザーのパスワード

3. OAuth 2.0 認証と API 呼び出し

3.1.OAuth 2.0 認証の概要

嬬恋スマートシティでは、OAuth 2.0 による認証を利用可能です。

OAuth 2.0 は、4 通りの認証方法があります。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で認証方式を選択できます。各認証方式の「サービス利用ガイド」の「表 2-4 OAuth 2.0 認証の種類」を参照してください。

本書では、Authorization Code Grant の認証方法、および Resource Owner Credentials Grant の認証方法（匿名アクセス方式）について説明します。

Authorization Code Grant による認証の流れを図 3-2 に示します。

Authorization Code Grant による認証を利用するためには、アプリケーションで、図 3-2 中の ①～③の処理が必要です。

① OAuth 2.0 認証呼び出し

② アクセストークン取得

③ API の呼び出し

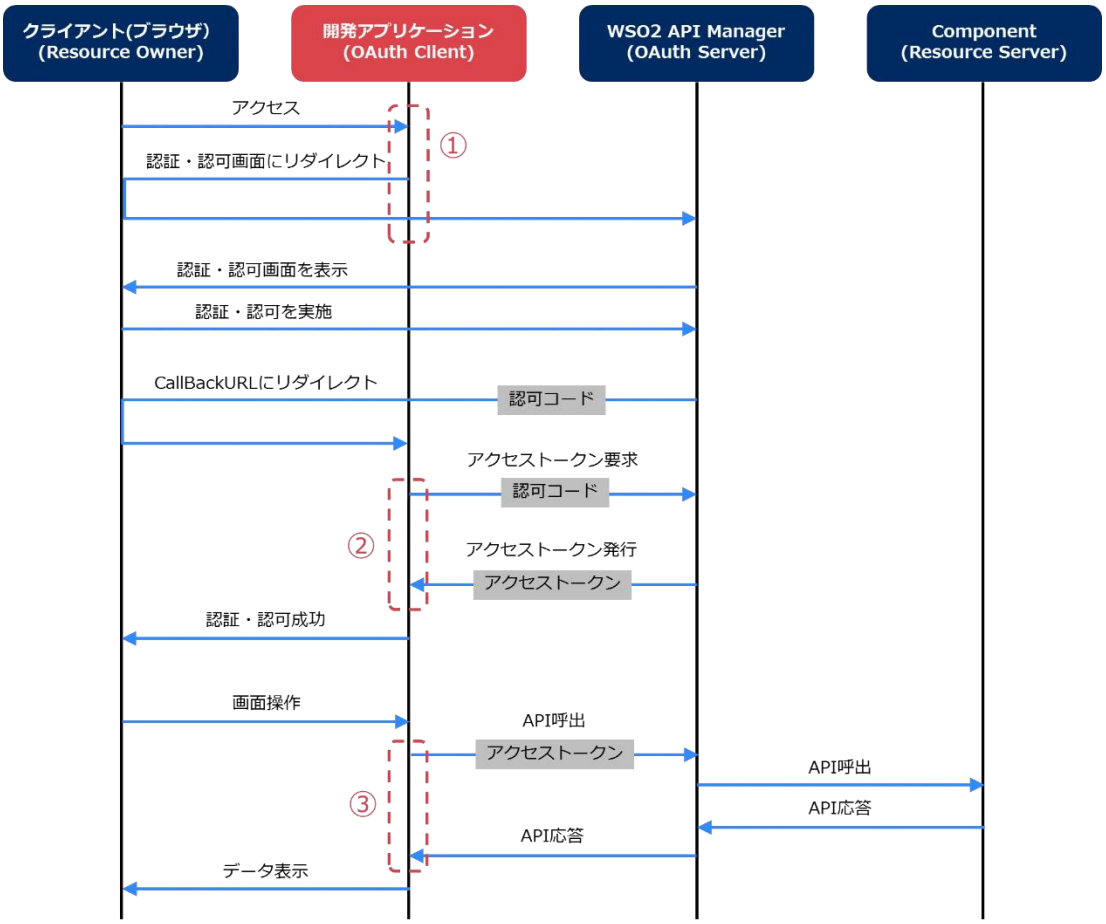

図 3-1 OAuth 2.0 認証のシーケンス（Authorization Code Grant）

Resource Owner Credentials Grant（匿名アクセス方式）による認証の流れを図 3-2 に示します。

Resource Owner Credentials Grant（匿名アクセス方式）による認証を利用するためには、アプリケーションで、図 3-2 中の①～②の処理が必要です。

② OAuth 2.0 認証・アクセストークン取得

② API の呼び出し

Resource Owner Credentials Grant(匿名アクセス方式)による認証の流れを図 3-2 に示します。

Resource Owner Credentials Grant(匿名アクセス方式)による認証を利用するためには、アプリケーションで、図 3-2 中の①～②の処理が必要です。

① OAuth 2.0 認証・アクセストークン取得

② API の呼び出し

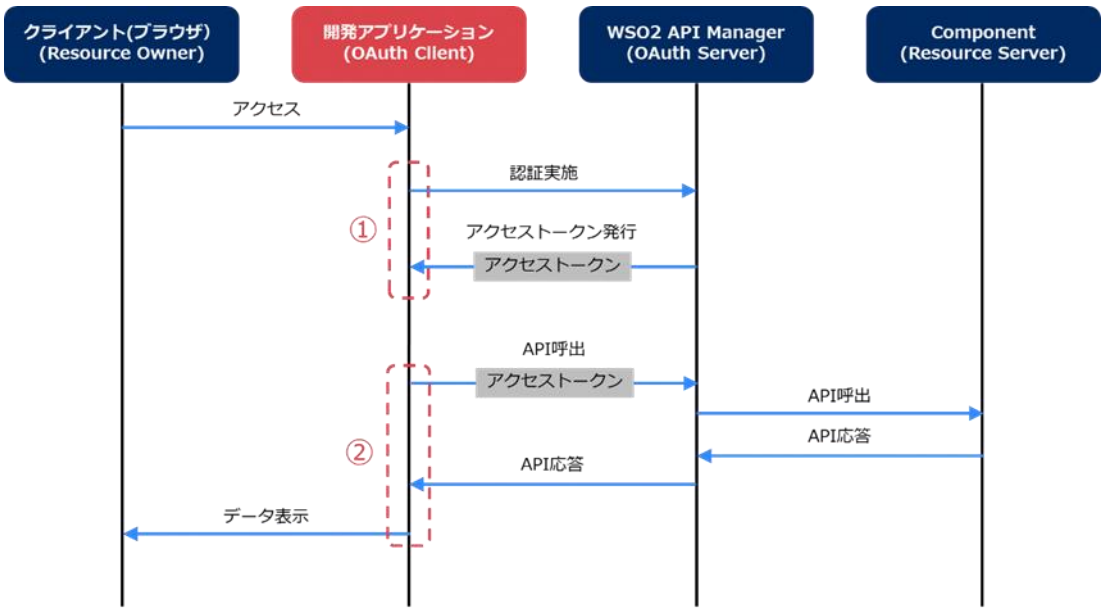

図 3-2 OAuth 2.0 認証のシーケンス
(Resource Owner Credentials Grant (匿名アクセス方式))

3.2. アプリケーションの実装

3.2.1. OAuth 2.0 の認証呼び出し

アプリケーションへアクセス時や、ログイン時に OAuth 2.0 認証の呼び出しをします。
以降では Authorization Code Grant の例を記載しています。

開発するアプリケーションで認証が必要な場合、下記の 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://www.smartcity-net.com/wso2am/oauth2/authorize

※ドメイン名が取得できていない場合は、www.smartcity-net.com に払い出し時の外部 IP アドレスを記載してください。

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します。

	″default″ を指定してください。
response_type	Authorization Code Grant の場合は、″code″ を指定します。
redirect_uri	「第 2 章 API の利用準備」で指定したアプリケーションのコールバック URL を指定します。
client_id	払い出されたアプリケーションの Consumer Key (client id) を指定します。

＜レスポンス＞
ブラウザに認証画面が表示されます。

3.2.2. 認証、認可実施

認証、認可実施の処理は認可サーバにより行われるため、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、認可の実行結果を以下に示します。

1.認証画面が表示されるため、ユーザー名、パスワードを入力して、 [サインイン]をクリックします。

図 3-3 WSO2 が用意する認証画面

2. 承認画面が表示されるため、[許可]をクリックします。

図 3-4 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。

コールバック URL のパラメータには認可コードが付与されるため、その認可コードを用いて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「3.2.3 アクセストークン取得」以降の処理を実装してください。

3.2.3. 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。

- コールバック URL のリクエストパラメータから、認可コードを取得します。
  パラメータ名：code 例:
  【アプリケーション CallbackUrl 1】?code=[認可コード]

- 認可コードを使用して、アクセストークンを取得します。
  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://www.smartcity-net.com/wso2am/oauth2/token

※ドメイン名が取得できていない場合は、www.smartcity-net.com に払い出し時の外部 IP アドレスを記載してください。

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

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

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

レスポンスボディ

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が返却されます。
token_type	"Bearer"が返却されます。
expires_in	トークンの有効期限（秒）が返却されます。
refresh_token	リフレッシュトークンが返却されます。
access_token	アクセストークンが返却されます。

例：
{"scope":"default","token_type":"Bearer","expires_in":2413,"refresh_token":"e156236ef50
596b80d44adbb1c2773b0","access_token":"4e88f99fa193bafbeb41c528b9b9e070"}

・ 取得したアクセストークンは、API の呼び出しをするために保持する必要があります。セッション等を用いて、取得したアクセストークンを保持してください。

アクセストークンの有効期限を延長・再発行する場合はリフレッシュトークンを使用します。
詳細は下記の手順に従ってください。
https://docs.wso2.com/display/IS530/Refresh+Token+Grant

3.2.4. API の呼び出し

認証処理を通すため、API 呼び出し時に、リクエストヘッダに OAuth 2.0 のアクセストークン文字列を付与してください。
リクエスト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}

Orion の API（付録 A の Orion-API）を呼び出す場合のリクエスト例を以下に示します。
Orion の API（付録 A の Orion-API）を呼び出す場合のリクエスト例を以下に示します。

機能	Orion-API を呼び出す	
HTTP メソッド	POST	
URL	https://www.smartcity-net.com/orion/v1.0/queryContext	
ヘッダ	Authorization: Bearer 【OAuth 2.0 のアクセストークン文字列】 Content-Type: application/json Accept: application/json	
ボディ	タグ名／キー名	説明
	Entities	エントリ
	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

```
#!/bin/sh
(curl -k -v -X POST "https://www.smartcity-net.com/orion/v1.0/queryContext" -s -S --header
"Authorization: Bearer【OAuth 2.0 のアクセストークン文字列】" --header 'Content-Type: application/json' --header 'Accept: application/json' -d @- | python mjson.tool) <<EOF
{
    "entities": [
        {
            "type": "Street",                "isPattern": "false",

            "id": "Street4"
        }
    ],
    "attributes": [
        {
            "name": "temperature"
        }
    ]
}
EOF
```

※ドメイン名が取得できていない場合は、www.smartcity-net.com に払い出し時の外部 IP アドレスを記載してください。

3.3.アプリケーションの実装（匿名アクセス方式）

3.3.1.　　OAuth 2.0 認証・アクセストークン取得

開発するアプリケーションで匿名アクセスが必要な場合、匿名アクセスユーザーで OAuth 2.0 認証を実施し、API 呼び出し時に必要なアクセストークンを取得します。
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://www.smartcity-net.com/wso2am/oauth2/token

※ドメイン名が取得できていない場合は、www.smartcity-net.com に払い出し時の外部 IP アドレスを記載してください。

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

パラメータ名	説明
grant_type	Resource Owner Credentials Grant の場合は、"password&username=【匿名アクセスユーザーのユーザー名】&password=【匿名アクセスユーザーのパスワード】"を指定します。
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）を指定します
client_id	払い出されたアプリケーションの利用者キー（client id）を指定します

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

レスポンスボディ

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が返却されます。
token_type	"Bearer"が返却されます。

expires_in	トークンの有効期限（秒）が返却されます。
refresh_token	リフレッシュトークンが返却されます。
access_token	アクセストークンが返却されます。

例：
{"access_token":"74595476-f3d3-3098-9e48-dd66b56d7441","refresh_token":"78d1ddd1d5c8-376c-a0afbb8307a95344","scope":"default","token_type":"Bearer","expires_in":3221}

取得したアクセストークンは、API の呼び出しをするために保持する必要があります。セッション等を用いて、取得したアクセストークンを保持してください。

アクセストークンの有効期限を延長・再発行する場合はリフレッシュトークンを使用します。詳細は下記の手順に従ってください。
https://docs.wso2.com/display/IS530/Refresh+Token+Grant

3.3.2. API の呼び出し

「3.2.4 API の呼び出し」と同様の処理を実装してください。
OAuth 2.0 のアクセストークン文字列は、「3.3.1 OAuth 2.0 認証・アクセストークン取得」で取得したトークンを使用してください。

4. ユーザー管理 API

4.1. ユーザープロフィール取得

ユーザー名やロール情報など、ログインユーザーのプロフィールを取得するには SCIM API を利用します。

SCIM API 呼び出し時にはアクセストークンが必要となるため、リクエストヘッダに OAuth 2.0 のアクセストークン文字列を付与してください。

機能	ユーザープロフィール情報取得	
HTTP メソッド	GET	
URL	https://www.smartcity-net.com/wso2is/wso2/scim/Users/me	
ヘッダ	Authorization: Bearer【OAuth 2.0 のアクセストークン文字列】 Content-Type: application/json Accept: application/json	
ボディ	タグ名／キー名	説明
	userName	トークンを持つユーザーの名前
	groups	アクセス制御用ロール情報
	display	ロール名
	id	SCIM リソース ID
	schemas	スキーマ
	urn	スキーマ名
	meta	メタデータ
	created	作成日時
	lastModified	更新日時
※他の項目についてはスキーマの設定により可変です。		

アクセストークン指定

curl -v -H "Authorization: Bearer a4ac73b6-dc75-39e7-9e99-8f8cd71c27d9" https://dev-necjfiware.jp/wso2is/wso2/scim/Users/me

＜レスポンス例＞

```
{
    "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"
}
```

付録 A　API を公開するコンポーネント一覧

嬬恋村スマートシティで公開する API は以下 3 つのコンポーネントの API です。

- Orion-API
- Comet-API

コンポーネントの API にアクセスをする場合の URL を表 A-1 に記載します。
コンポーネントにアクセスすると、コンポーネントに認証認可のチェックが働きます。コンポーネントの認可に関しては付録 B にて説明します。

表 A-1 コンポーネントにアクセスする場合の URL

コンポーネント	アクセス URL
Orion-API NGSIv1	https://www.smartcity-net.com (*1)/orion/v1.0/【コンポーネントのメソッド、パラメータ】
Orion-API NGSIv2	https://www.smartcity-net.com (*1)/orion/v2.0/【コンポーネントのメソッド、パラメータ】
Comet-API	https://www.smartcity-net.com (*1)/comet/v1.0/【コンポーネントのメソッド、パラメータ】

【コンポーネントのメソッド、パラメータ】、コンポーネントに必要なリクエストヘッダ情報、パラメータ等の送信情報については、『嬬恋村スマートシティ アプリケーション開発ガイド
(データ収集蓄積編)』および『嬬恋村スマートシティ 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。

付録 B　コンポーネントの認可機能

WSO2 API Manager の Gateway 機能を経由することで、認可機能によって各コンポーネントの API 呼び出しが制限されます。

各 API は、使用可/使用不可なロールが機能ごとに定義されています。認可判定に利用するロールは、表 B-1 に記載します。

表 B-1 認可判定に利用するロール一覧

Role 名	Role の持つ権限
ContextAdministrator	Orion-API コンポーネントの全 API を実行する権限を持つ
ContextProducer	Orion-API コンポーネントに対して Context Element, Context Availability を提供するロールであり、Orion API の内、登録/更新に関する API を実行する権限を持つ(削除は除く)
ContextConsumer	Orion に格納された Contex Element, Context Availability を参照する

各 API と使用可能なロールの一覧をそれぞれ、表 B-2 から表 B-7 に記載します。
認可判定の記号は下記を表します。
○：対象のロールで使用することができるリソース
×：対象のロールで使用することができないリソース

表 B-2 Orion-API (NGSI-9) の認可判定

種別	Resource	HTTP メソッド	認可判定			
			Context Producer	Context Consumer	Context Administrator	その他
標準 API	/registry/registerContext	POST	○	×	○	×
	/registry/discoverContextAvailability	POST	×	○	○	×
	/registry/subscribeContextAvailability	POST	×	○	○	×
	/registry/unsubscribeContextAvailability	POST	×	○	○	×
	/registry/updateContextAvailabilitySubscription	POST	×	○	○	×
	/registry/notifyContextAvailability	POST	○	×	○	×
	/registry/contextEntities/{entityId}	GET	×	○	○	×

拡張 API		POST	○	×	○	×
	/registry/contextEntities/{entityId}/attributes	GET	×	○	○	×
		POST	○	×	○	×
	/registry/contextEntities/{entityId}/attributes/{attributeName}	GET	×	○	○	×
		POST	○	×	○	×
	/registry/contextEntityTypes/{typeName}	GET	×	○	○	×
		POST	○	×	○	×
	/registry/contextEntityTypes/{typeName}/attributes	GET	×	○	○	×
		POST	○	×	○	×
	/registry/contextEntityTypes/{typeName}/attributes/{attributeName}	GET	×	○	○	×
		POST	○	×	○	×
	/registry/contextAvailabilitySubscriptions	POST	○	×	○	×
	/registry/contextAvailabilitySubscriptions/{SubscriptionId}	PUT	○	×	○	×
	/registry/contextAvailabilitySubscriptions/{SubscriptionId}	DELETE	×	×	○	×

表 B-3 Orion-API (NGSI-10) の認可判定

種別	Resource	HTTP メソッド	認可判定			
			Context Producer	Context Consumer	Context Administrator	その他
標準 API	/updateContext	POST	○	×	○	×
	/queryContext	POST	×	○	○	×
	/subscribeContext	POST	×	○	○	×
	/unsubscribeContext	POST	×	○	○	×
	/updateContextSubscription	POST	×	○	○	×
	/notifyContext	POST	○	×	○	×
拡張 API	/contextEntities	GET	×	○	○	×
		POST	○	×	○	×
	/contextEntities/{entityId}	GET	×	○	○	×
		PUT	○	×	○	×
		POST	○	×	○	×
		DELETE	×	×	○	×
	/contextEntities/{entityId}/attributes	GET	×	○	○	×
		PUT	○	×	○	×
		POST	○	×	○	×
		DELETE	×	×	○	×
	/contextEntities/{entityId}/attributes/{attributeName}	GET	×	○	○	×
		POST	○	×	○	×
		DELETE	×	×	○	×
	/contextEntities/{entityId}/attributes/{attributeName}/{attributeId}	GET	×	○	○	×
		PUT	○	×	○	×
		DELETE	×	×	○	×
	/contextEntityTypes/{typeName}	GET	×	○	○	×
	/contextEntityTypes/{typeName}/attributes	GET	×	○	○	×
	/contextEntityTypes/{typeName}/attributes/{attributeName}	GET	×	○	○	×
	/contextSubscriptions	POST	○	×	○	×
	/contextSubscriptions/{SubscriptionId}	PUT	○	×	○	×

		DELETE	×	×	○	×

表 B-5 Comet-API の認可判定

種別	Resource	HTTP メソッド	認可判定		
			Comet User	Comet Administrator	その他
	/contextEntities/type/{entityType}/id/{ entityId}/attributes/{attributeName}	GET	○	○	×
	/contextEntities	DELETE	×	○	×
	/contextEntities/type/{entityType}/id/{ entityId}	DELETE	×	○	×
	/contextEntities/type/{entityType}/id/{ entityId}/attributes/{attributeName}	DELETE	×	○	×

種別	Resource	HTTP メソッド	認可判定			
			Context Producer	Context Consumer	Context Administrator	その他
	/entities	POST	○	×	○	×
		GET	×	○	○	×
	/entities/{entityId}	GET	×	○	○	×
		DELETE	×	×	○	×
	/entities/{entityId}/attrs	GET	×	○	○	×
		POST	○	×	○	×
		PATCH	○	×	○	×
		PUT	○	×	○	×
	/entities/{entityId}/attrs/{attrName}	GET	×	○	○	×
		PUT	○	×	○	×
		DELETE	×	×	○	×
	/entities/{entityId}/attrs/{attrName}/value	GET	×	○	○	×
		PUT	○	×	○	×
	/types	GET	×	○	○	×
	/types/{entityType}	GET	×	○	○	×
	/subscriptions	GET	×	○	○	×
		POST	×	○	○	×
	/subscriptions/{subscriptionId}	GET	×	○	○	×
		PATCH	×	○	○	×
		DELETE	×	○	○	×
	/registrations	GET	×	○	○	×
		POST	○	×	○	×
	/registrations/{registrationId}	GET	×	○	○	×
		PATCH	○	×	○	×
		DELETE	×	×	○	×
	/op/update	POST	○	×	○	×
	/op/query	POST	×	○	○	×