SCIM サーバー仕様書

1. はじめに

本 SCIM サーバーは OPTiM StoreのSCIM クライアントからのみ、処理要求があることを前提に記載しています。

2. 注意事項

システムを利用する上で注意が必要な点について説明します。

3. 機能と特徴

この章では SCIM サーバーの概要と特徴を説明します。

3-1. システム要件

SCIM APIのシステム要件を説明します。

3-1-1. 動作環境

HTTP/HTTPSサーバー通信ができるアプリケーションまたはプログラムが必要となります。

3-1-2. 利用権限

SCIM サーバーはAPIにアクセスするための認可フローが必要です。

4. エンドポイント

本章では、SCIM サーバーとして実装して頂く必要があるエンドポイントについて説明します。 なお、エンドポイントまでのパスは任意で構いません。

4-1. エンドポイント

OPTiM Store からアクセスされるエンドポイントは下表のとおりです。

リソース	エンドポイント名	行う操作	説明
利用者	/Users	GET,POST,PUT,DELETE	ユーザーの取得、作成、変更、削除を行います。
認証・認可	/token	POST	認証を行い、アクセストークンを返却します。

5. 処理フロー

OPTiM Store からの送信されるリクエストとレスポンスのフローは以下のとおりです。

図 5-1: 処理フロー

6. API仕様の詳細

本章では OPTiM Store から SCIM サーバーへ接続するための仕様について詳細を説明します。 OPTiM Storeとのプロビジョニングを行うにあたっては、本章に登場するAPIを実装して頂く必要があります。

6-1. 認証・認可フロー

SCIM サーバーの認証・認可では OAuth のクライアントクレデンシャルフローを使用するため、OAuth 2.0のAuthorization Serverを実装して頂く必要があります。 ライブラリをご利用頂いても問題ありません。

6-1-1. OAuth

- OAuth を利用するには「クライアント ID」「クライアントシークレット」が必要となります。
- OAuthではアクセストークンが有効な間、何度も SCIM サーバーにリクエストできます。アクセストークンの有効期限が切れた場合は再度アクセストークンの取得要求がリクエストされます。

6-2. OAuth リクエスト

- OAuthで認可コード、アクセストークンを取得するリクエストについて記載します。
- ここではクライアントID、クライアントシークレットの検証を行い、正しい場合にアクセストークンなどを返却してください。
- クライアントID、クライアントシークレットが不正な場合はエラーを返却してください。

6-2-1. エンドポイント

OAuthのエンドポイントは下表のとおりです。

リクエスト	エンドポイント
アクセストークン新規取得リクエスト	https://[ホスト名]/ecosystem/oauth/v1/token

6-2-2. アクセストークン新規取得リクエスト

1. 処理内容

下記の流れに従って、アクセストークンを発行してください。

1. 送られたクライアントID(client_id)、クライアントシークレット(client_secret)が正しいか検証します。
2. アクセストークン、有効期限を生成し、保持します。
3. JSON形式でレスポンスを作成し、返却します。

2. リクエスト

- アクセストークン新規取得リクエストのbodyは下記のような形式となります。

```
grant_type=client_credentials&client_id=[クライアントID] &client_secret=[クラ
```

- 各パラメータの説明は、下表の通りとなります。

パラメーター	必須	説明
grant_type	○	固定値”client_credentials”が指定されます。
client_id	○	OPTiM からプロビジョニング時に指定されたクライアントIDが指定されます。
client_secret	○	OPTiM からプロビジョニング時に指定されたクライアントシークレットが指定されます。

- 下記は、アクセストークン新規取得リクエストの例です。

```
POST /ecosystem/oauth/v1/token
Host: server.example.com
Content-Type: application/x-www-form-urlencoded;charset=UTF-8

grant_type=client_credentials&client_id=s6BhdRkqt3&client_secret=7Fjfp0
```

3. レスポンス

- アクセストークン新規取得レスポンスは、JSON形式で返却してください。

```
{
  "access_token": [生成したアクセストークン],
  "token_type":"bearer",
  "expires_in": [有効期限(秒)],
}
```

- 各パラメータの説明は、下表の通りとなります。

オブジェクト	説明
access_token	アクセストークンです。
token_type	アクセストークンの種類です。固定値で”bearer”です。
expires_in	アクセストークンの有効期間を秒で表します。

- 下記は、アクセストークン新規取得レスポンスの例です。

```
HTTP/1.1 200 OK
```

```
Content-Type: application/json;charset=UTF-8
Cache-Control: no-store
Pragma: no-cache
{
  "access_token":"2YotnFZFEjr1zCsicMWpAA",
  "token_type":"bearer",
  "expires_in":3600,
}
```

4. エラー

- アクセストークン新規取得リクエストの主なエラー例は、下表のとおりです。

原因	エラー内容
パラメーター不正 (指定がない/未知のパラメーターが指定された)	HTTP 400 Bad Request と共に、レスポンスボディにエラーメッセージを含めて返却してください。 `{"error":"invalid_request","error_description":"authorization header is invalid"}`
パラメーター不正	HTTP 400 Bad Request と共に、レスポンスボディにエラーメッセージを含めて返却してください。 `{"error":"invalid_grant","error_description":"client_id or code or redirect_uri is invalid"}} 以下のケースを含みます。 ・client_id、client_secretが不正`

- 下記は、エラーが発生した場合のレスポンスの例です。

```
HTTP/1.1 401 Unauthorized
Date: Thu, 27 Mar 2014 04:38:00 GMT
Content-Length: 69

{"error":"invalid_client","error_description":"client_id is invalid"}
```

6-3. SCIM API

SCIMサーバーで実装して頂く必要のある、SCIM APIについて記載します。

6-3-1. エンドポイント

- SCIM APIのエンドポイントは下表のとおりです。

エンドポイント	内容	操作
https://[hostname]/ecosystem/v1/Users	利用者情報	POST(追加)・GET(検索)・PUT(更新)・DELETE(削除)

6-4-1. 利用者スキーマ情報

- SCIMで利用する、利用者のスキーマ情報は下表のとおりです。

属性名	スキーマ	必須	更新	返却	一意性	備考
id	SCIM標準	○	参照のみ	常時	有	追加時に独自に生成してください。システムで一意となる値にする必要があります。
bizGuid	拡張	○	不可	常時	有	-

- 返却情報のうち、OPTiM Storeが使用する属性は、idのみとなります。

6-4-2. 利用者情報検索操作(GETメソッド)

- 利用者情報の検索処理を行ってください。
- クエリパラメーターで、検索フィルター・返却属性・ソートキー・ソート順・ページあたりの件数・ページ番号の指定を出来るようにする必要があります(OPTIONAL)。
- 上記のうち、対象者の検索のため、最低限下記の検索フィルターを実装する必要があります。
  - **「bizIdtokenClaimsSubject eq [値] and bizBizIdentityCode eq [値]」**
- 詳細は、SCIM 仕様を参照してください。

1. 処理内容

1. アクセストークンが正しい値であるか確認します。
2. アクセストークンから対象のテナントを特定します。
3. 対象のテナント内から指定された検索フィルターに従い、情報を取得します。
4. 情報をJSON形式に変換し、返却します。id属性は必ず返却してください。その他の属

性は省略しても構いません。

2. リクエスト

- 各パラメータの説明は、下表の通りとなります。

パラメーター	説明
Authorization	OAuth 2.0 Bearer として token リクエストで取得したアクセストークンを設定します。

- 下記は、利用者情報検索操作リクエスト例の例です。

```
GET /ecosystem/v1/Users/?filter=bizUserId%20eq%20user001%20and%20bizIde
Authorization: Bearer MDNhNTlhN2ItYmE1NC00OWQ3LWFkMzQtODliYjhhN2Q0Mjlh
Host: server.example.com
Accept: application/scim+json;charset=UTF-8

HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/scim+json;charset=UTF-8
Content-Length: 2508
Date: Wed, 22 Oct 2014 12:12:12 GMT
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:ListResponse"
    ],
    "totalResults": 1,
    "Resources": [
        {
            "schemas": [
                "urn:x-optim:scim:schemas:extention:cim:1.0:User"
            ],
            "id": "b3603063-2801-4c8c-b602-a142efe7ad6a",
            ※属性を記載する
            "meta": {
                "resourceType": "",
                "created": "2014-09-02T05:38:35.114Z",
                "lastModified": "2014-10-15T09:12:40.519Z",
                "location": "https://server.example.com/ecosystem/v1/User
            }
        },
    ]
}
```

3. レスポンス

- 下記は、利用者情報検索操作レスポンスの例です。

```
{
    "schemas": [
        "urn:ietf:params:scim:api:messages:2.0:ListResponse"
    ],
    "totalResults": [返却数],
    "Resources": [
        {
            (schemas内は固定です)
            "schemas": [
                "urn:x-optim:scim:schemas:extention:cim:1.0:User"
            ],
            (以下から利用者の情報を記載していきます)
            ※属性を記載する
            "meta": {
                "resourceType": "User",
                "created": [作成日時をYYYY-MM-DD”T”HH:MI:SS.sssZのUTCで記載],
                "lastModified": [更新日時をYYYY-MM-DD”T”HH:MI:SS.sssZのUTCで
                "location": [直接アクセスができるhttp://[サーバー]/[パス]/Users/[id
            }
        },
        (複数返却時は以下のように続けて返却します)
        {
            "schemas": [
                "urn:x-optim:scim:schemas:extention:cim:1.0:User"
            ],
            ※属性を記載する
            "meta": {
                "resourceType": "User",
                "created": [作成日時をYYYY-MM-DD”T”HH:MI:SS.sssZのUTCで記載],
                "lastModified": [更新日時をYYYY-MM-DD”T”HH:MI:SS.sssZのUTCで
                "location": [直接アクセスができるhttp://[サーバー]/[パス]/Users/[id
            }
        },
    ]
}
```

4. エラー

6-4-3. 利用者情報追加操作(POSTメソッド)

- 指定された情報を元に、利用者情報の追加処理を行ってください。
- SCIM仕様上、一意に該当利用者を特定するためのidの生成が必要となります。
- SCIM仕様上、レスポンスにUTCの作成日時(created)を生成する必要がありますが、本仕様上は使用していないため、省略しても構いません。

- 詳細は、SCIM 仕様を参照してください。

1. 処理内容

1. アクセストークンが正しい値であるか確認します。
2. アクセストークンから対象のテナントを特定します。
3. システム内で一意に該当利用者を特定するidを生成します。idはUUID4などで生成してください。
4. 指定された属性、値をシステムへ反映します。
5. 反映した情報をJSON形式に変換し、返却します。id属性は必ず返却してください。その他の属性は省略しても構いません。

2. リクエスト

- 各パラメータの説明は、下表の通りとなります。

パラメーター	説明
Authorization	OAuth 2.0 Bearer として token リクエストで取得したアクセストークンを設定します。

- 下記は、利用者情報情報追加操作リクエストの例です。

```
POST /ecosystem/v1/Users
Authorization: Bearer MDNhNTlhN2ItYmE1NC00OWQ3LWFkMzQtODliYjhhN2Q0Mjlh
Host: server.example.com
Content-Type: application/scim+json;charset=UTF-8
Accept: application/scim+json;charset=UTF-8
{
    "schemas": [
        "urn:x-optim:scim:schemas:extention:cim:1.0:User"
    ],
    ※属性を記載する
}
```

3. レスポンス

- 処理に成功した場合は、201 Created の HTTP Status を返します。
- レスポンスbodyは下記のような形式となります。

```
{
    (schemas内は固定です)
    "schemas": [
        "urn:x-optim:scim:schemas:extention:cim:1.0:User"
```

```
    ],
    ※属性を記載する
    "meta": {
           "resourceType": "Users",
           "created": [作成日時をYYYY-MM-DD”T”HH:MI:SS.sssZのUTCで記載],
           "location": [直接アクセスができるhttp://[サーバー]/[パス]/Users/[id]のU
    }
}
```

- 下記は、利用者情報追加操作レスポンスの例です。

```
HTTP/1.1 201 Created
Server: Apache-Coyote/1.1
Content-Type: application/scim+json;charset=UTF-8
Content-Length: 1701
Date: Wed, 22 Oct 2014 12:12:12 GMT
Location: https://example.com/CIMCloudAdminSCIMAPI/v1/Users/8f848134-3f

{
    "schemas": [
        "urn:x-optim:scim:schemas:extention:cim:1.0:User"
    ],
    ※属性を記載する
    "meta": {
           "resourceType": "Users",
           "created": "2014-10-15T17:50:51.316Z",
           "location": "https://server.example.com/ecosystem/v1/Users/8
    }
}
```

6-4-4. 利用者情報更新操作(PUTメソッド)

- 利用者情報の更新処理を行なってください。
- PUT リクエストでは、指定がない属性は削除するようにしてください。
- システム上必要な項目が送信されない場合は、エラーとしてください。
- 利用者の削除を行わず、ライセンス状態のみ変更する場合があります。 ライセンス状態の有無をシステム上管理していない場合は、該当属性から判断し、利用者の削除、追加などの処理に対応付けた実装を行う必要があります。
- SCIMの仕様上、レスポンスにUTCの更新日時(lastModified)を含める必要がありますが、本仕様上は使用していないため、省略しても構いません。
- 詳細は、SCIM 仕様を参照してください。

1. 処理内容

1. アクセストークンが正しい値であるか確認します。
2. アクセストークンから対象のテナントを特定します。
3. テナント内の情報からidを元に対象の利用者を取得します。
4. 指定された属性、値をシステムへ反映します。この時、指定がない属性は削除してください。
5. 反映した情報をJSON形式に変換し、返却します。id属性は必ず返却してください。その他の属性は省略しても構いません。

2. リクエスト

- 各パラメータの説明は、下表の通りとなります。

パラメーター	説明
Authorization	OAuth 2.0 Bearer として token リクエストで取得したアクセストークンを設定します。

- 下記は、利用者情報情報更新操作リクエストの例です。

```
PUT /ecosystem/v1/Users/8f848134-3f6f-4432-9db7-5599a5263696
Authorization: Bearer MDNhNTlhN2ItYmE1NC00OWQ3LWFkMzQtODliYjhhN2Q0Mjlh
Host: server.example.com
Content-Type: application/scim+json;charset=UTF-8
Accept: application/scim+json;charset=UTF-8

{
    "schemas": [
        "urn:x-optim:scim:schemas:extention:cim:1.0:User"
    ],
    ※属性を記載する
}
```

3. レスポンス

- 処理に成功した場合は、200 OK の HTTP Status を返します。
- レスポンスbodyは下記のような形式となります。

```
{
    (schemas内は固定です)
    "schemas": [
        "urn:x-optim:scim:schemas:extention:cim:1.0:User"
    ],
```

```
    ※属性を記載する
    "meta": {
           "resourceType": "Users",
           "created": [作成日時をYYYY-MM-DD”T”HH:MI:SS.sssZのUTCで記載],
           "lastModified": [更新日時をYYYY-MM-DD”T”HH:MI:SS.sssZのUTCで記載]
           "location": [直接アクセスができるhttp://[サーバー]/[パス]/Users/[id]のU
    }
}
```

- 下記は、利用者情報更新操作レスポンスの例です。

```
HTTP/1.1 200 OK
Server: Apache-Coyote/1.1
Content-Type: application/scim+json;charset=UTF-8
Content-Length: 1701
Date: Wed, 22 Oct 2014 12:12:12 GMT
Location: https://server.example.com/ecosystem/v1/Users/8f848134-3f6f-4

{
    "schemas": [
        "urn:x-optim:scim:schemas:extention:cim:1.0:User"
    ],
    ※属性を記載する
    "meta": {
           "resourceType": "Users",
           "created": "2014-10-15T17:50:51.316Z",
           "lastModified ": "2014-10-16T18:10:52.132Z",
           "location": "https://server.example.com/ecosystem/v1/Users/8
    }
}
```

6-4-5. 利用者情報削除操作(DELETEメソッド)

- 利用者情報の削除処理を行ってください。
- 詳細は、SCIM 仕様を参照してください。

1. 処理内容

1. アクセストークンが正しい値であるか確認します。
2. アクセストークンから対象のテナントを特定します。
3. テナント内の情報からidを元に対象の利用者を特定し、削除処理をします。
4. 処理結果のレスポンスを返します。

2. リクエスト

- 各パラメータの説明は、下表の通りとなります。

パラメーター	説明
Authorization	OAuth 2.0 Bearer として token リクエストで取得したアクセストークンを設定します。

- 下記は、利用者情報削除操作のリクエストの例です。

```
DELETE /ecosystem/v1/Users/8f848134-3f6f-4432-9db7-5599a5263696
Authorization: Bearer MDNhNTlhN2ItYmE1NC00OWQ3LWFkMzQtODliYjhhN2Q0Mjlh
Host: server.example.com
Accept: application/scim+json;charset=UTF-8
利用者情報削除操作レスポンス例
HTTP/1.1 204 Not Content
Server: Apache-Coyote/1.1
Date: Wed, 22 Oct 2014 12:12:12 GMT
```

3. レスポンス

- 処理に成功した場合は、204 No Content の HTTP Status を返します。
- bodyは空となります。