Red Hat 3scale 2-saas

開発者ポータルでの API の提供

デベロッパーポータルを適切に設定することが、API 管理機能の向上につながります。

Last Updated: 2024-02-07

Red Hat 3scale 2-saas 開発者ポータルでの API の提供

デベロッパーポータルを適切に設定することが、API 管理機能の向上につながります。

法律上の通知

概要

本ガイドでは、Red Hat 3scale 2-saas でのデベロッパーポータルの使用について説明します。

目次

前書き

API を定義する OpenAPI ドキュメントは、デベロッパーポータルの基盤です。

多様性を受け入れるオープンソースの強化

Red Hat では、コード、ドキュメント、Web プロパティーにおける配慮に欠ける用語の置き換えに取り組んでいます。まずは、マスター (master)、スレーブ (slave)、ブラックリスト (blacklist)、ホワイトリスト (whitelist) の 4 つの用語の置き換えから始めます。この取り組みは膨大な作業を要するため、今後の複数のリリースで段階的に用語の置き換えを実施して参ります。詳細は、CTO である Chris Wright のメッセージ をご覧ください。

パート I. OPENAPI SPECIFICATION (OAS)

第1章 OPENAPI SPECIFICATION の概要

Red Hat 3scale API Management では、OpenAPI Specification (OAS) は OpenAPI ドキュメントを最適に管理するのに役立ちます。OpenAPI Specification (OAS) は、既存のサービスを更新したり、新しいサービスを作成したりするためのツールを提供します。

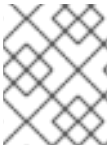
以下は、3scale の OAS に関する特別な留意事項です。

- 3scale toolbox で OpenAPI Specification (OpenAPI ドキュメント) をインポートすることもできます。OpenAPI 定義のインポート を参照してください。
- 3scale 2.8 では、OAS 3.0 の変更が加えられました。詳細は、「3scale での OpenAPI Specification 3.0 の使用」 を参照してください。

前提条件

- API を定義する OpenAPI ドキュメント。
- 3scale 2-saas インスタンステナントのクレデンシャル (**token** または **provider_key**)。

OAS を使用すると、3scale では以下の機能を使用することができます。



注記

OpenAPI ドキュメントのインポート時に、ActiveDocs を作成または更新します。3scale 仕様として使用する OpenAPI ドキュメントの作成方法 を参照してください。

- 3scale サービスの **system_name** をオプションのパラメーターとして渡す機能 (デフォルトは OAS からの **info.title** フィールド)。
- メソッドは、OpenAPI Specification で定義された各オペレーションに対して作成されます。
  - **メソッド** 名は **operation.operationId** フィールドから取得されます。
- 既存の **マッピングルール** は、新規 API 定義をインポートする前にすべて削除されます。
  - コマンドの実行前から存在するメソッドは削除されません。
- マッピングルールは、OpenAPI Specification で定義された各オペレーションに対して作成されます。
- 以下のチャネルの 1 つによって、OpenAPI 定義リソースが提供されます。
  - 利用可能なパスの **ファイル名**
  - **URL** フォーマット (toolbox は所定のアドレスからダウンロードを試みます)
  - **stdin** 標準入力ストリームからの読み込み

1.1. 3SCALE で OPENAPI ドキュメントをインポートするためのコマンドラインオプション

3scale コマンドラインインターフェイス (CLI) は、3scale で管理する API を定義する OpenAPI ドキュメントをインポートするための複数のオプションを提供します。以下は、**openapi** オプションのヘルプ情報です。

```
NAME
   openapi - Import API definition in OpenAPI specification

USAGE
   3scale import openapi [opts] -d <dst> <spec>

DESCRIPTION
   Using an API definition format like OpenAPI, import to your 3scale API

OPTIONS
      -d --destination=<value>                  3scale target instance.
                                    Format: "http[s]://<authentication>@3scale_domain"

 -t --target_system_name=<value>           Target system name

OPTIONS FOR IMPORT
   -c --config-file=<value>                  3scale toolbox
                                    configuration file
                                    (default:
                                    $HOME/.3scalerc.yaml)
   -h --help                          show help for this command
   -k --insecure                       Proceed and operate even
                                    for server connections
                                    otherwise considered
                                    insecure
   -v --version                       Prints the version of this
                                    command
```

1.2. API 仕様をインポートするためのさまざまなソース

API 仕様のインポートに関して、3scale の管理者はさまざまなソースを利用することができます。それらの概要を以下の表に示します。ここでは、ファイル名のパス、URL、および **stdin** から OpenAPI 定義を検出するための使用状況オプションを説明しています。

表1.1 OpenAPI 定義の検出

説明	フォーマット	コマンドラインの使用
ファイル名のパスからの OpenAPI 定義の検出:フォーマットは、ファイル名の拡張子から自動的に検出されます。	**json** および **yaml**	`$ 3scale import openapi -d <destination> /path/to/your/spec/file.[json|yaml|yml]`
URL からの OpenAPI 定義の検出:フォーマットは、URL のパスの拡張子から自動的に検出されます。	**json** および **yaml**	`$ 3scale import openapi -d <destination> http[s]://domain/resource/path.[json|yaml|yml]`

説明	フォーマット	コマンドラインの使用
stdin からの OpenAPI 定義の検出:OpenAPI リソースのコマンドラインパラメーターは **-** です。フォーマットはパーサーにより内部的に自動検出されます。	**json** および **yaml**	$ tool_to_read_openapi_from_source | 3scale import openapi -d <destination> -

第2章 OPENAPI SPECIFICATION の設定方法

OpenAPI Specification が 3scale と連携するには、使用するバージョン用に正しく設定する必要があります。

前提条件

- API を定義する OpenAPI ドキュメント。
- 3scale 2-saas インスタンステナントのクレデンシャル (**token** または **provider_key**)。

2.1. 3SCALE での OPENAPI SPECIFICATION 3.0 の使用

3scale では、OAS 3.0 を使用するための以下のサポートが提供されます。

- 開発者ポータルで **Swagger-ui** が更新され、OAS 3.0 がサポートされるようになりました。
- 今回のリリースでは、**swagger-ui** は webpack アセット (**node_modules**) として組み込まれています。以前は、コンテンツ配信ネットワーク (CDN) から追加されていました。
- 管理ポータルでは、**swagger-ui** が提供する機能を使用して、新しい OAS 3.0 ドキュメントが自動的に識別され、これに応じて処理されます。この機能には、開発者ポータルでの設定が必要になることに注意してください。

OAS 3.0 仕様を ActiveDocs に追加し、開発者ポータルで表示するには、以下の点を考慮してください。

- テンプレートは手動でアップグレードする必要があります。
- ActiveDoc には、要求の試行時の認証情報インジェクションや、サービス名などの実際のデータを使用した自動補完などの追加機能がありません。

2.1.1. OAS 3.0 を使用する開発者ポータルの設定

このスニペットには、新しいバージョンの **swagger-ui** が含まれ、最初に利用可能な ActiveDoc をレンダリングします。OAS 2.0 もレンダリングされますが、通常の ActiveDocs 機能はない点に注意してください。

OAS 3.0 仕様のサポートには、デフォルトのドキュメントページで以下の内容が必要です。

```
{% content_for javascripts %}
 {{ 'active_docs.js' | javascript_include_tag }}
{% endcontent_for %}

{% assign spec = provider.api_specs.first %}

<h1>Documentation</h1>

<div class="swagger-section">
 <div id="message-bar" class="swagger-ui-wrap"></div>
 <div id="swagger-ui-container" class="swagger-ui-wrap"></div>
</div>

<script type="text/javascript">
 (function () {
```

```
    var url = "{{spec.url}}";
    var serviceEndpoint = "{{spec.api_product_production_public_base_url}}"
    SwaggerUI({ url: url, dom_id: "#swagger-ui-container" }, serviceEndpoint);
  }());
</script>
```

OAS 3.0 が設定された開発者ポータルの更新

3scale 2.8 で OAS 3.0 を設定していて、引き続き OAS 3.0 を使用する場合は、テンプレートを更新する必要があります。

設定するテンプレートを以下に示します。

```
{% content_for javascripts %}
    {{ 'active_docs.js' | javascript_include_tag }}
{% endcontent_for %}

<h1>Documentation</h1>

<div class="swagger-section">
  <div id="message-bar" class="swagger-ui-wrap"> </div>
  <div id="swagger-ui-container" class="swagger-ui-wrap"></div>
</div>

<script type="text/javascript">
  (function () {
    var url = "{{provider.api_specs.first.url}}";

    SwaggerUI({ url: url, dom_id: "#swagger-ui-container" });
  }());
</script>
```

テンプレートを更新するには、デフォルトのドキュメンテーションページを 「OAS 3.0 を使用する開発者ポータルの設定」 に含まれるスニペットに置き換えます。

2.2. 3SCALE での OPENAPI SPECIFICATION 2.0 の使用

OAS 2.0 仕様を ActiveDocs に追加し、デベロッパーポータルで表示するには、以下の点を考慮してください。

- テンプレートは手動でアップグレードする必要があります。
- ActiveDoc には、要求の試行時の認証情報インジェクションや、サービス名などの実際のデータを使用した自動補完などの追加機能がありません。

OAS 2.0 仕様のサポートには、デフォルトのドキュメントページで以下の内容が必要です。

```
<h1>Documentation</h1>
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.js %}
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.css %}

{% include 'shared/swagger_ui' %}

<script type="text/javascript">
  $(function () {
```

```
    window.swaggerUi.options['url'] = "{{provider.api_specs.first.url}}";
    window.swaggerUi.load();
  });
</script>
```

2.3. SWAGGER ユーザーインターフェイスの 2.1.3 から 2.2.10 へのアップグレード

Swagger UI 2.1.3 を含む 3scale のバージョンを使用している場合は、Swagger UI バージョン 2.2.10 にアップグレードできます。

3scale 開発者ポータルで以前実装されていた Swagger UI 2.1.3 は、**Documentation** ページに 1 つの **{% active_docs version: "2.0" %}** Liquid タグがあることが条件です。3scale で Swagger 2.2.10 がサポートされたことに伴い、実装メソッドは複数の Liquid タグ **cdn_asset** と **include** に変わっています。

注記

Swagger UI 2.1.3 以前のバージョンについては、3scale では、引き続き古い **active_docs** Liquid タグメソッドを使用して UI を呼び出します。

前提条件

- 管理者アクセスを持つ 3scale インスタンス。
- Swagger UI 2.1.3 が含まれる 3scale インスタンス。

手順

1. 3scale 管理ポータルにログインします。
2. **Developer Portal → Documentation** ページに移動するか、Swagger UI 実装を更新するページに移動します。
3. コードペインの **Draft** タブで、**{% active_docs version: "2.0" %}** Liquid タグを **cdn_asset** Liquid タグおよび 新しいパーシャル **shared/swagger_ui** に置き換えます。

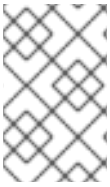
```
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.js %}
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.css %}

{% include 'shared/swagger_ui' %}
```

4. (オプション) デフォルトでは、Swagger UI は **APIs > ActiveDocs** に公開された ActiveDocs 仕様を読み込みます。別の仕様を読み込むには、**window.swaggerUi.load();** の行の前に以下の **window.swaggerUi.options** の行を追加します。ここで、**<SPEC_SYSTEM_NAME>** は読み込む仕様のシステム名です。

```
window.swaggerUi.options['url'] = "{{provider.api_specs.<SPEC_SYSTEM_NAME>.url}}";
```

パート II. 開発者ポータルの API ドキュメント

第3章 ACTIVEDOCS 2.0 への更新



注記

本セクションに記載の情報は、参照用途としてのみ提供されています。このオプションはサポート対象外になったため、できるだけ早期にこの設定から移行することを検討する必要があります。

本チュートリアルでは、ActiveDocs 設定をバージョン 2.0 に正常にアップグレードするのに必要な変更点について説明します。

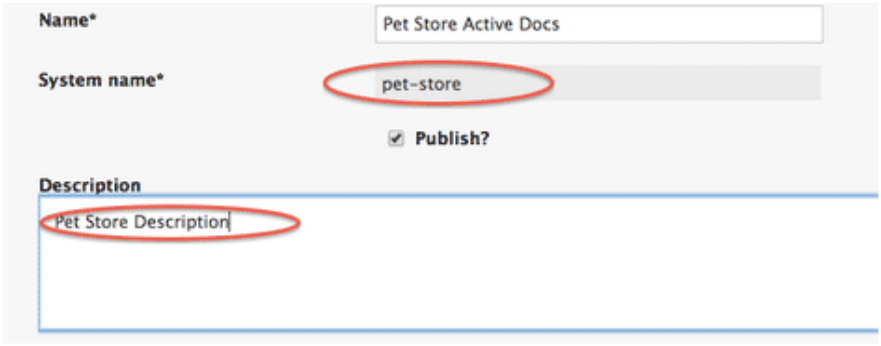
ActiveDocs 2.0 の設定に適用される手順については、Adding Specifications to 3scale を参照してください。仕様関連の詳細な相違点は、公式な Swagger 1.2 から 2.0 への移行ガイド を参照してください。このアーティクルでは、ActiveDocs 2.0 にアップグレードするための追加手順だけを説明します。

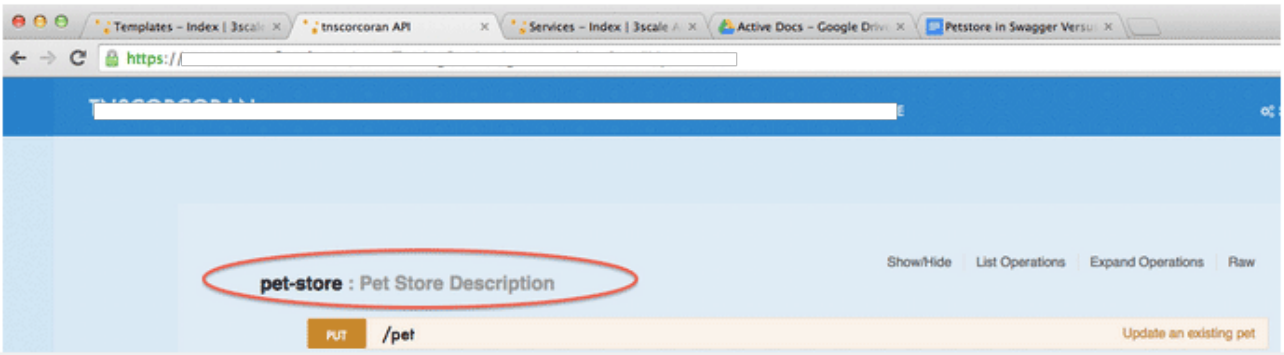
注記

ActiveDocs 仕様がバージョン 1.0 にある場合は、最初にバージョン 1.2 に変換してください。

3.1. ステップ 1: 仕様への適切な命名規則の適用

管理ポータルで **API → ActiveDocs** タブに移動します。これにより、サービス仕様のリストが表示されます。サービス仕様がすでに追加されているはずです (Create a service specification を参照)。





デベロッパーポータルで希望する効果を得るためには、適切な名前を適用する必要があります。ActiveDocs API リストの見出しは System name: Description として表示されます。システム名が読み取り専用であるため、JSON 仕様およびその他のフィールドを単純にコピーして仕様を再度作成する必要がある場合があります。

3.2. ステップ 2: サービス仕様の変更

ActiveDocs 2.0 の仕様には、バージョン 1.2 と比較してバージョンに重要な変更がいくつか加えられています。詳細は、Swagger 1.2 から 2.0 への移行ガイド を参照してください。最も重要な変更点は以下のとおりです。

- **"swaggerVersion": "1.2"** ルート要素は "swagger": "2.0" になり、必須フィールドになりました。
- info オブジェクトは必須になります。
- **"apiVersion": "1.0"** は必須になり、"info" オブジェクトに含まれるようになりました: **"info": { "version": "1.0", ... }**
- info オブジェクトの説明は必須ではありません。
- license オブジェクトが存在する場合は、ライセンス名フィールドが必要になります。
- **"basePath": "https://example.com/api"** フィールドは、**"host": "example.com"**、**"basePath": "/api"**、および **"schemes": ["http"]** の 3 つのフィールドに分割されます。これらのフィールドはいずれも必須ではありません。

3.3. ステップ 3: CMS ページへの JAVASCRIPT および HTML コンテンツの追加

以下のコードスニペットを CMS ページに追加します。ここで、SERVICE_NAME はサービス仕様のシステム名になります。

```
<h1>Documentation</h1>
<p>Use our live documentation to learn about Echo API </p>
{% active_docs version: "2.0" services: "SERVICE_NAME" %}
<script type="text/javascript">
  $(function () {
    {% comment %}
    // you have access to swaggerUi.options object to customize its behaviour
    // such as setting a different docExpansion mode
    window.swaggerUi.options['docExpansion'] = 'none';
    // or even getting the swagger specification loaded from a different url
    window.swaggerUi.options['url'] = "http://petstore.swagger.io/v2/swagger.json";
    {% endcomment %}
  window.swaggerUi.load();
});

</script>
```

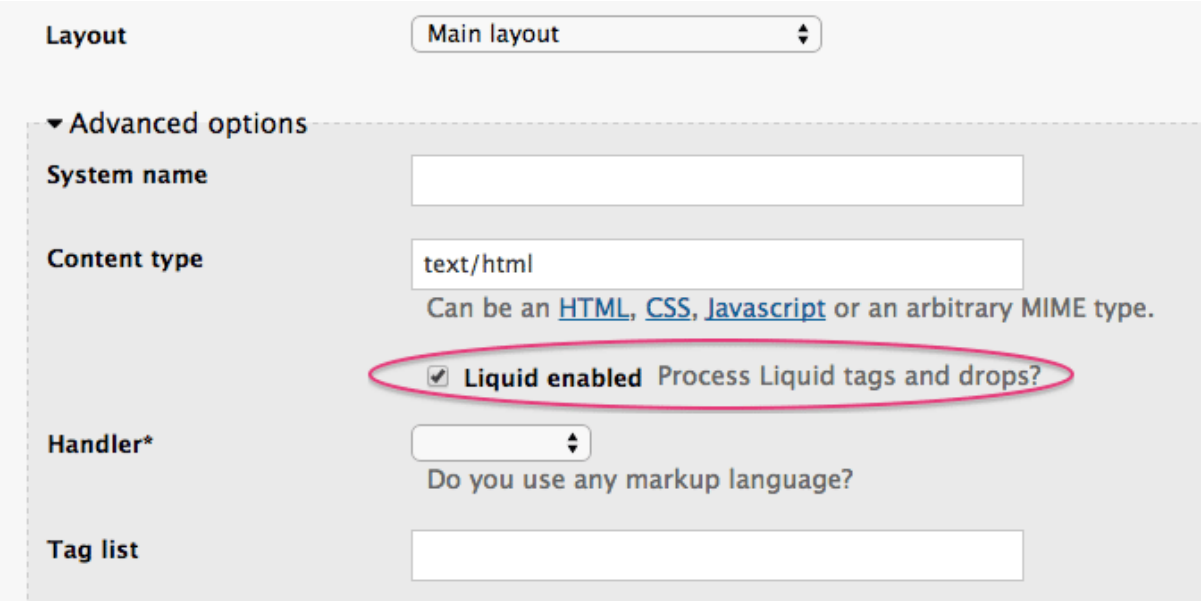
1 つのページに複数の Swagger 仕様を含める場合は、以下のカスタマイズされたスニペットを使用できます。

```
{% active_docs version: "2.0" services: "oauth" %}



<script type="text/javascript">
  $(function () {
    window.swaggerUi.load(); // <-- loads first swagger-ui
     // do second swagger-ui
     var url = "/swagger/spec/sentiment.json";
```

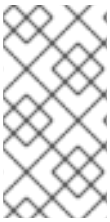
```
    window.anotherSwaggerUi = new SwaggerUi({
      url: url,
      dom_id: "another-swagger-ui-container",
      supportedSubmitMethods: ['get', 'post', 'put', 'delete', 'patch'],
      onComplete: function(swaggerApi, swaggerUi) {
        $('#another-swagger-ui-container pre code').each(function(i, e) {hljs.highlightBlock(e)});
      },
      onFailure: function(data) {
        log("Unable to Load Sentiment-SwaggerUI");
      },
      docExpansion: "list",
      transport: function(httpClient, obj) {
        log("[swagger-ui]>>> custom transport.");
        return ApiDocsProxy.execute(httpClient, obj);
      }
    });
     window.anotherSwaggerUi.load();
  });
</script>
```

3.4. ステップ 4: ACTIVEDOCS 1.2 を使用した API のテスト

- CMS 設定ページで Liquid タグを有効にするのを忘れないようにしてください。



- 最後に、プレビューモードでは、右側の垂直サイドバーを閉じ、ActiveDocs 2.0 を表示します。



注記

新しいスタイルは、新しい Swagger 仕様 (2.0) に準拠しています。外観と操作感を変更するには、スタイルを上書きする必要があります。Swagger の CSS は HTML と共に含まれているため、スタイルをより高次の特異性または !important タグで定義する必要があります。

第4章 3SCALE への ACTIVEDOCS の追加

3scale は、API のインタラクティブドキュメントを作成するためのフレームワークを提供します。

OpenAPI Specification (OAS) を使用することで、API に機能的なドキュメントを設定することができます。これは、開発者が API を調べ、テストを行い、統合するのに役立ちます。

4.1. 3SCALE での ACTIVEDOCS の設定

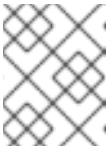
3scale ユーザーインターフェイスで ActiveDocs を API に追加すると、API のインタラクティブドキュメントを作成するためのフレームワークが得られます。

前提条件

- API を定義する OpenAPI ドキュメント。
- 3scale 2-saas インスタンステナントのクレデンシャル (**token** または **provider_key**)。

手順

1. 管理ポータルで **[your_API_name] → ActiveDocs** の順に移動します。3scale では、API のサービス仕様のリストが表示されます。これは最初は空です。
   サービス仕様は、必要なだけ追加することができます。通常、各サービス仕様は API のいずれか 1 つに対応しています。たとえば、3scale では、それぞれの 3scale API に仕様があります (Service Management、Account Management、Analytics、および Billing)。
2. **Create a new spec** をクリックします。
   新しいサービス仕様を追加する場合は、以下を指定します。
   - 名前。
   - システム名。システム名は、開発者ポータルからサービス仕様を参照するために必要です。
   - 仕様を公開するかどうかの選択。公開しない場合は、開発者ポータルで新しい仕様は利用できません。

     注記

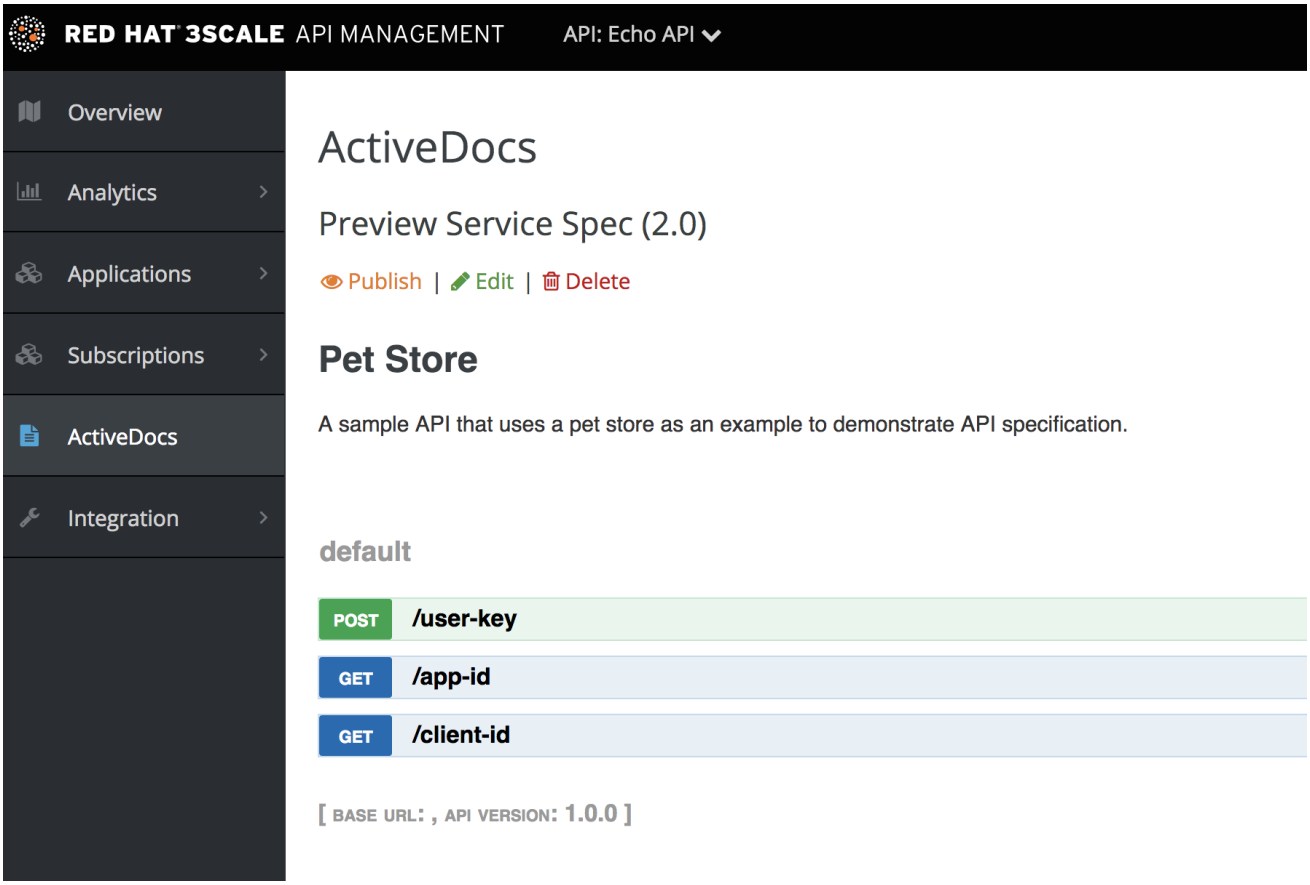
     新しい仕様を作成しても公開しない場合は、後で希望する時期に公開することができます。
   - 使用するのは自分のみという説明の追加。
   - API JSON 仕様の追加。
     API の仕様は、OpenAPI Specification (OAS) が提案する仕様に従って生成してください。このチュートリアルでは、API 用に有効な OpenAPI Specification (OAS) 準拠の仕様がすでに用意されていることを前提としています。

最初の ActiveDoc に関する操作

最初の ActiveDoc を追加すると、これが **[your_API_name] → ActiveDocs** のリストに表示されます。必要に応じて編集、削除、または公開から非公開への切り替えが可能です。また、API から切り離したり、他の API にアタッチしたりできます。**Audience → Developer Portal → ActiveDocs** の順に移動して、API にアタッチされているかどうかに関わらず、すべての ActiveDocs を確認できます。

サービス仕様に指定した名前 (例: Pet Store) をクリックして、ActiveDocs がどのように表示されるかをプレビューできます。まだ仕様を公開していない場合でも、この操作を実行できます。

ActiveDoc は、以下のように表示されます。



第5章 3SCALE OPENAPI 仕様として使用する OPENAPI ドキュメントの作成方法

コードを読むだけであれば、すべての例は OAS Petstore のソースコード例 に記載されています。

3scale ActiveDocs は、Swagger と呼ばれる RESTful Web サービスの仕様をベースにしています (Wordnik より)。この例は、拡張された OpenAPI Specification の Petstore の例 をベースにしており、すべての仕様データを OpenAPI Specification 2.0 の仕様ドキュメント から引用しています。

前提条件

- 開発者ポータルで ActiveDocs を動作させるには、REST API に対する OpenAPI Specification (OAS) 準拠の仕様が必要である。

OAS は単なる仕様ではありません。あらゆる機能のフレームワークも提供します。

- 複数の言語 (NodeJS、Scala、その他) によるリソース仕様のサーバー。
- 仕様ファイルを使用して開発者を引き付ける UI を生成する、さまざまな HTML/CSS/Javascripts アセット。
- Swagger 準拠サーバーからクライアントライブラリーを自動的に生成することのできる、OAS codegen プロジェクト。数多くの最新言語によるクライアント側のライブラリー作成をサポートします。

5.1. 3SCALE ACTIVEDOCS および OAS の設定

ActiveDocs とは OAS のインスタンスのことです。ActiveDocs を使用する場合、独自の OAS サーバーを実行したり、インタラクティブドキュメントのユーザーインターフェイスコンポーネントを扱ったりする必要はありません。インタラクティブドキュメントは、3scale の開発者ポータルから提供され、レンダリングされます。

3scale 2.8 では、OAS 3.0 が導入されましたが、ActiveDocs でのサポートは限定されていました。つまり、自動補完などの ActiveDocs を使用する機能の一部が完全に統合されていないため、新しいアカウントの作成時にデフォルトの 3scale は OAS 2.0 に設定されます。OAS 3.0 および ActiveDocs の詳細は、「3scale での OpenAPI Specification 3.0 の使用」 を参照してください。

前提条件

- 開発者ポータルで使用されるテンプレートが、管理者ポータルで指定されているものと同じ OAS バージョンを実装していることを確認している。

手順

1. OAS に準拠する API 仕様を構築します。
2. 管理ポータルに仕様を追加します。

結果

API 用のインタラクティブなドキュメントが利用できるようになりました。API 利用者は、開発者ポータルを通じて API にリクエストを送信することができます。

すでに OAS 準拠の API 仕様がある場合は、それを開発者ポータルに追加できます。ActiveDocs の設定に関するチュートリアル を参照してください。

3scale では OAS 仕様をさまざまな方法で拡張し、開発者ポータルでのインタラクティブな API ドキュメントに必要な特定の機能に対応しています。以下に例を示します。

- API キーの自動入力
- CORS 非対応 API への呼び出しを許可する OAS プロキシー

5.2. OPENAPI ドキュメントの例: PETSTORE API

オリジナルのソースから仕様を読み取るには、OpenAPI Specification を参照してください。

OAS サイトには、API を定義する OpenAPI ドキュメントの例が複数あります。例を使用して学習したい場合は、OAS API チームが作成した Petstore API の例を参照してください。

Petstore API は非常にシンプルな API です。これは、学習を目的とするものであって、実稼働用ではありません。

Petstore API のメソッド

Petstore API は、4 つのメソッドで設定されています。

- **GET /api/pets**: システムからすべてのペットを返す
- **POST /api/pets**: ストアに新しいペットを作成する
- **GET /api/pets/{id}**: ID に基づき 1 つのペットを返す
- **DELETE /api/pets/{id}**: ID に基づき 1 つのペットを削除する

Petstore API は 3scale と統合されるため、認証用に別のパラメーターを追加する必要があります。たとえば、ユーザーキー認証方法では、API 利用者はユーザーキーパラメーターを各リクエストのヘッダーに配置する必要があります。その他の認証方法の詳細は、API ゲートウェイの管理の 認証パターン を参照してください。

ユーザーキーパラメーター

user_key: {user_key}

user_key は、API 利用者により API へのリクエストで送信されます。API 利用者は、3scale 管理者の開発者ポータルでこれらのキーを取得します。キーの受信時に、3scale 管理者は Service Management API を使用して 3scale に対する承認チェックを行う必要があります。

OpenAPI Specification の詳細

API 利用者にとって、cURL 呼び出しで表される API のドキュメントは、以下のようになります。

```
curl -X GET "http://example.com/api/pets?tags=TAGS&limit=LIMIT" -H "user_key: {user_key}"
curl -X POST "http://example.com/api/pets" -H "user_key: {user_key}" -d "{ "name": "NAME", "tag": "TAG", "id": ID }"
curl -X GET "http://example.com/api/pets/{id}" -H "user_key: {user_key}"
curl -X DELETE "http://example.com/api/pets/{id}" -H "user_key: {user_key}"
```

5.3. OAS 仕様に関する補足情報

ドキュメントを OAS Petstore のドキュメント のようにする場合は、関連の Petstore **swagger.json** ファイルのような Swagger 準拠の仕様を作成する必要があります。この仕様をそのまま使用して、

ActiveDocs をテストすることができます。ただし、これは実際の API ではないことに注意してください。

OAS は、JSON でエンコードされたハッシュにマッピングするリソース宣言に依存します。Petstore **swagger.json** ファイルをサンプルとして使用し、各オブジェクトについて説明します。

OAS オブジェクト

これは API 仕様のルートドキュメントオブジェクトです。最上位レベルのフィールドをすべてリスト表示します。

警告

ホストは、IP アドレスではなくドメインでなければなりません。3scale は、デベロッパーポータルに対して行われたリクエストをプロキシーとしてホストに中継し、結果をレンダリングします。そのためには、セキュリティー上の理由から、**ホスト** および **basePath** エンドポイントが 3scale により承認される必要があります。宣言できるのは、ご自分のホストだけです。API プロバイダーが自身に属さないドメインを中継していることが確認された場合、3scale は API プロバイダーのアカウントを終了する権利を有します。つまり、ローカルホストまたはその他のワイルドカードドメインは機能しません。

info オブジェクト

info オブジェクトは API に関するメタデータを提供します。このコンテンツは ActiveDocs ページに提示されます。

paths オブジェクト

paths オブジェクトは、個々のエンドポイントへの相対パスを保持します。パスが basePath に追加され、完全な URL となります。アクセス制御リスト (ACL) の制約により、**paths** は空になる可能性があります。

オブジェクトではないパラメーターは、プリミティブデータタイプを使用します。Swagger では、プリミティブデータ型は JSON スキーマドラフト 4 でサポートされるタイプに基づきます。プリミティブデータ型には追加の **file** もありますが、3scale では API エンドポイントで CORS が有効な場合にのみ使用されます。CORS が有効になっていると、アップロードは api-docs ゲートウェイを経由せず、拒否されます。

現在、OAS は以下の **dataTypes** をサポートしています。

- 整数型フォーマット: int32 および int64。どちらのフォーマットも署名されています。
- 数値型フォーマット: float および double
- プレーン文字列
- 使用できるフォーマットの文字列: バイト、日付、日時、パスワード、およびバイナリー
- boolean

関連情報

- OpenAPI オブジェクト
- Info オブジェクト
- Paths オブジェクト
- API サーバーおよびベース URL

5.4. OAS の設計および編集ツール

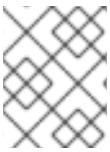
API を定義する OpenAPI Specification を設計および編集する際には、以下のツールが役立ちます。

- オープンソースの Apicurio Studio を使用すると、Web ベースのアプリケーションで OpenAPI ベースの API を設計および編集することができます。Apicurio Studio では設計ビューを利用することができるため、OpenAPI Specification に関する詳細な知識は必要ありません。習熟したユーザーは、ソースビューを使用して直接 YAML または JSON で編集することができます。詳細は、Getting Started with Apicurio Studio を参照してください。
  Red Hat では、Apicurio Studio の軽量バージョンである API Designer も提供しています。このツールは、Fuse Online on OpenShift に含まれています。詳細は、Developing and Deploying API Provider Integrations を参照してください。
- JSON 表記を十分に理解している場合は、JSON Editor Online が便利です。JSON を縮小化する整形フォーマットを使用することができ、JSON オブジェクトブラウザーが提供されます。
- Swagger Editor を使用すると、YAML で書かれた OAS API 仕様をブラウザーで作成および編集し、リアルタイムでプレビュー表示することができます。有効な JSON 仕様を生成することもでき、これを後から 3scale 管理ポータルにアップロードすることができます。また、機能が限定された ライブデモ バージョンを使用することや、独自の OAS エディターをデプロイすることができます。

5.5. ACTIVEDOCS による API 認証情報の自動入力

API 認証情報の自動入力は、3scale ActiveDocs の OAS に対する便利なエクステンションです。**x-data-threescale-name** フィールドには、API 認証モードに応じて以下の値を定義できます。

- **user_keys**: API キー認証のみを使用するサービスのアプリケーションのユーザーキーを返します。
- **app_ids**: アプリケーション ID/アプリケーションキーを使用するサービスのアプリケーションの ID を返します。後方互換のために、OAuth と OpenID Connect もサポートされています。
- **app_keys**: アプリケーション ID/アプリケーションキーを使用するサービスのアプリケーションのキーを返します。後方互換のために、OAuth と OpenID Connect もサポートされています。



注記

x-data-threescale-name フィールドは OAS エクステンションで、ActiveDocs 以外のユースケースでは無視されます。

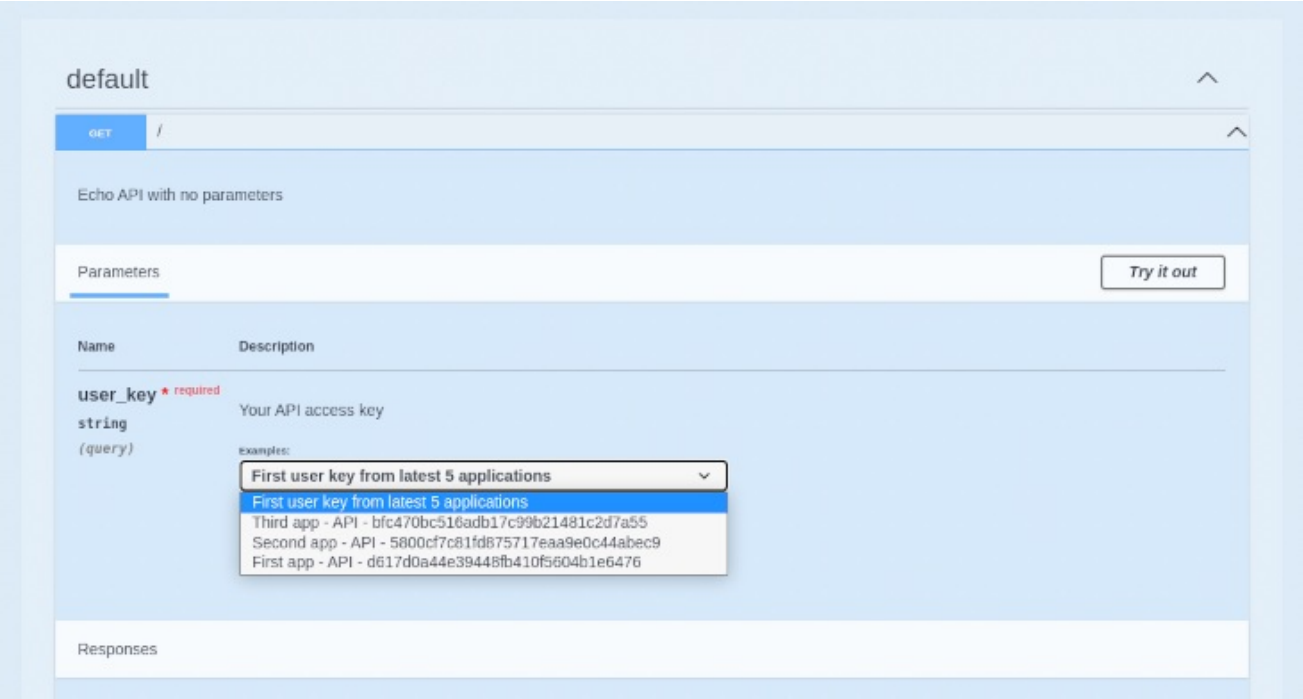
API キー認証の例

API キー認証のみの場合に **"x-data-threescale-name": "user_keys"** を使用する例を以下に示します。

```
"parameters": [
  {
```

```
    "name": "user_key",
    "in": "query",
    "description": "Your API access key",
    "required": true,
    "schema": {
      "type": "string"
    },
    "x-data-threescale-name": "user_keys"
  }
]
```

x-data-threescale-name で宣言されたパラメーターの場合、開発者ポータルにログインすると、仕様で設定された値に従って、最新の 5 つのキー、ユーザーキー、アプリ ID またはアプリキーを含むドロップダウンリストが表示されます。したがって、値をコピーして貼り付けることなく、入力を自動入力できます。



第6章 ACTIVEDOCS と OAUTH

ActiveDocs により、ユーザーは 1 つの設定画面から OAuth 対応の API をテストして呼び出すことができます。

前提条件

- Red Hat Single Sign-On インスタンスおよび OpenID Connect インテグレーションを設定しておく必要がある。設定方法の詳細は、OpenID Connect インテグレーション に関するドキュメントを参照してください。
- また、ActiveDocs の設定方法を十分理解する必要もあります。3scale への ActiveDocs の追加 および OpenAPI Specification の作成 を参照してください。

6.1. 3SCALE 仕様でのクライアント認証情報およびリソースオーナーフローの例

この最初の例は、3scale 仕様の OAuth 2.0 クライアント認証情報フローを使用する API に関するものです。この API は任意のパスを受け入れ、リクエストに関する情報 (パス、リクエストパラメーター、ヘッダーなど) を返します。Echo API には、有効なアクセストークンを使用する場合に限りアクセスできます。API のユーザーは、認証情報 (**client_id** および **client_secret**) を交換してアクセストークンを取得するまで、API を呼び出すことができません。

ユーザーが ActiveDocs から API を呼び出せるようにするには、アクセストークンを要求する必要があります。これは単なる OAuth 承認サーバーへの呼び出しなので、OAuth トークンエンドポイント用の ActiveDocs 仕様を作成できます。これにより、ActiveDocs 内からこのエンドポイントへの呼び出しが可能になります。この例のクライアント認証情報フローの場合、Swagger JSON 仕様は以下のようになります。

```
{
  "swagger": "2.0",
  "info": {
    "version": "v1",
    "title": "OAuth for Echo API",
    "description": "OAuth2.0 Client Credentails Flow for authentication of our Echo API.",
    "contact": {
      "name": "API Support",
      "url": "http://www.swagger.io/support",
      "email": "support@swagger.io"
    }
  },
  "host": "red-hat-sso-instance.example.com",
  "basePath": "/auth/realms/realm-example/protocol/openid-connect",
  "schemes": [
    "http"
  ],
  "paths": {
    "/token": {
      "post": {
        "description": "This operation returns the access token for the API. You must call this before
calling any other endpoints.",
        "operationId": "oauth",
        "parameters": [
          {
```

```
            "name": "client_id",
            "description": "Your client id",
            "type": "string",
            "in": "query",
            "required": true
          },
          {
            "name": "client_secret",
            "description": "Your client secret",
            "type": "string",
            "in": "query",
            "required": true
          },
          {
            "name": "grant_type",
            "description": "OAuth2 Grant Type",
            "type": "string",
            "default": "client_credentials",
            "required": true,
            "in": "query",
            "enum": [
                "client_credentials",
                "authorization_code",
                "refresh_token",
                "password"
            ]
          }
        ]
      }
    }
  }
}
```

リソースオーナー OAuth フローでは、ユーザー名とパスワードのパラメーターおよびアクセストークンを発行するために必要なその他のパラメーターを追加する必要があります。このクライアント認証情報フローの例では、**client_id** と **client_secret** (サインイン済みユーザーの 3scale の値を代入可能)、ならびに grant_type を送信しています。

次に、Echo API の ActiveDocs 仕様で、**client_id** と **client_secret** の代わりに **access_token** パラメーターを追加します。

```
{
  "swagger": "2.0",
  "info": {
    "version": "v1",
    "title": "Echo API",
    "description": "A simple API that accepts any path and returns information about the request",
    "contact": {
      "name": "API Support",
      "url": "http://www.swagger.io/support",
      "email": "support@swagger.io"
    }
  },
  "host": "echo-api.3scale.net",
  "basePath": "/v1/words",
  "schemes": [
```

```
    "http"
  ],
  "produces": [
    "application/json"
  ],
  "paths": {
    "/{word}.json": {
      "get": {
        "description": "This operation returns information about the request (path, request parameters,
headers, etc.),
        "operationId": "wordsGet",
        "summary": "Returns the path of a given word",
        "parameters": [
          {
            "name": "word",
            "description": "The word related to the path",
            "type": "string",
            "in": "path",
            "required": true
          },
          {
            "name": "access_token",
            "description": "Your access token",
            "type": "string",
            "in": "query",
            "required": true
          }
        ]
      }
    }
  }
}
```

その後は、通常どおり開発者ポータルに ActiveDocs を含めることができます。この場合、OAuth エンドポイントが最初に表示されるよう順番を指定する必要があるため、以下のようになります。

```
{% active_docs version: "2.0" services: "oauth" %}




<script type="text/javascript">
  $(function () {
    window.swaggerUi.load(); // <-- loads first swagger-ui

    // do second swagger-ui

    var url = "/swagger/spec/echo-api.json";
    window.anotherSwaggerUi = new SwaggerUi({
      url: url,
      dom_id: "another-swagger-ui-container",
      supportedSubmitMethods: ['get', 'post', 'put', 'delete', 'patch'],
      onComplete: function(swaggerApi, swaggerUi) {
        $('#another-swagger-ui-container pre code').each(function(i, e) {hljs.highlightBlock(e)});
      },
```

```
    onFailure: function(data) {
      log("Unable to Load Echo-API-SwaggerUI");
    },
    docExpansion: "list",
    transport: function(httpClient, obj) {
      log("[swagger-ui]>>> custom transport.");
      return ApiDocsProxy.execute(httpClient, obj);
    }
  });

  window.anotherSwaggerUi.load();

});
</script>
```

6.2. 開発者ポータルでの ACTIVEDOCS の公開

このチュートリアルでは、開発者ポータルで ActiveDocs を公開すると共に、API ドキュメントを自動化する方法について説明します。

前提条件

- 開発者ポータルで ActiveDocs を動作させるには、REST API に対する OpenAPI Specification (OAS) 準拠の仕様が必要である。

手順

- 以下のスニペットを、開発者ポータルの任意のページのコンテンツに追加します。3scale 管理ポータルからこの操作を行う必要があります。

注記

SERVICE_NAME はサービス仕様のシステム名 (この例では **pet_store**) です。

OAS 3.0 を使用した開発者ポータルの設定

```
{% content_for javascripts %}
 {{ 'active_docs.js' | javascript_include_tag }}
{% endcontent_for %}

{% assign spec = provider.api_specs.first %}

<h1>Documentation</h1>

<div class="swagger-section">
 <div id="message-bar" class="swagger-ui-wrap"></div>
 <div id="swagger-ui-container" class="swagger-ui-wrap"></div>
</div>

<script type="text/javascript">
 (function () {
   var url = "{{spec.url}}";
   var serviceEndpoint = "{{spec.api_product_production_public_base_url}}"
```

```
  SwaggerUI({ url: url, dom_id: "#swagger-ui-container" }, serviceEndpoint);
}());
</script>
```

OAS 2.0 を使用した開発者ポータルの設定

```
<h1>Documentation</h1>
<p>Use our live documentation to learn about Echo API</p>
{% active_docs version: "2.0" services: "SERVICE_NAME" %}
{% cdn_asset /swagger-ui/2.2.10/swagger-ui.js %} {% cdn_asset /swagger-ui/2.2.10/swagger-ui.css %} {% include 'shared/swagger_ui' %}
<script type="text/javascript">
  $(function () {
    {% comment %}
      // you have access to swaggerUi.options object to customize its behavior
      // such as setting a different docExpansion mode
      window.swaggerUi.options['docExpansion'] = 'none';
      // or even getting the swagger specification loaded from a different url
      window.swaggerUi.options['url'] = "http://petstore.swagger.io/v2/swagger.json";
    {% endcomment %}
    window.swaggerUi.load();
  });
</script>
```

開発者ポータルで ActiveDocs を公開する場合の、他の考慮事項は以下のとおりです。

- 1 ページに指定できるサービスは 1 つだけです。複数の仕様を表示する場合は、別々のページで表示することが最善の方法となります。
- このスニペットには jQuery が必要ですが、これはデフォルトで開発者ポータルのメインレイアウトに含まれています。jQuery 依存関係をメインレイアウトから削除する場合は、ActiveDocs を含むページでこの依存関係を追加する必要があります。
- 管理ポータルで Liquid タグが有効になっていることを確認します。
- OAS 2.0 の Liquid タグで使用されるバージョン **{{ '{% active_docs version: "2.0" ' }}%}** は、Swagger 仕様のバージョンに対応している必要があります。

外部ソースから仕様を取得する場合は、以下のように JavaScript コードを変更します。

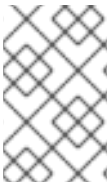
```
$(function () {
 window.swaggerUi.options['url'] = "SWAGGER_JSON_URL";
 window.swaggerUi.load();
});
```

仕様のソースを含む行 (**window.swaggerUi.options['url'] = "SWAGGER_JSON_URL";**) はコメントブロック外であることに注意してください。

検証手順

OpenAPI 仕様 を作成し、3scale に追加 したら、この仕様を公開して、デベロッパーポータルにリンクして、API 開発者が使用できるようにします。

第7章 SELF-MANAGED APICAST (旧バージョン) および OAUTH 2.0



注記

本セクションに記載の情報は、参照用途としてのみ提供されています。このオプションはサポート対象外になったため、できるだけ早期にこの設定から移行することを検討する必要があります。

すべての OAuth 呼び出しには SSL の使用が必須です。

本書では、Nginx ダウンロード可能設定ファイルで設定される旧バージョンの APIcast の OAuth について説明します。このバージョンは 2017 年 5 月以降、新規のお客様は GUI で利用できなくなりました。APIcast の最新バージョンでの OAuth サポートに関する詳細は、こちら を参照してください。

本チュートリアルでは、APIcast が OAuth 2.0 プロバイダーとして動作するよう 3scale の OAuth 拡張機能で Self-managed APIcast を設定するのに必要な手順について説明します。

現在、APIcast では認可コード (サーバー側の) 付与フローのみが利用可能です。ただし、この GitHub リポジトリー で、その他のすべてのフローの設定テンプレートを確認することができます。

7.1. 前提条件

3scale は認証するユーザーに関する情報を保持しないため、OAuth 2.0 を使用して 3scale と統合するためには、お客様側でユーザー認証を処理する必要があります。そのためには、APIcast がアプリケーションの承認のためにユーザーを送信できるページの URL を提供する必要があります。ユーザーを正しく識別し、認証できるように、このページはログインの背後になければなりません。ユーザーが認証され、アプリケーションが承認されたら、ユーザーからの承認付与の結果により APIcast にリダイレクトする必要があります。

APIcast がユーザーを承認 URL にリダイレクトすると、リクエストと共に以下のパラメーターが送信されます。

- **scope:** アプリケーションが属するプランの ID。アプリケーションプランは、3scale でのスコープを定義します。
- **state:** リクエストを特定し、その信頼性を確保するために APIcast と API の間で共有されるハッシュ値
- **tok:** アプリケーションが承認された場合にユーザーに付与されるアクセストークンの値。トークンは、承認コードと交換される場合のみ発行されます。承認コードが交換されない場合、アクセストークンは 10 分後に有効期限が切れます。

ユーザーが認証に成功してアプリケーションを承認すると、承認ページは APIcast のエンドポイントにリダイレクトするはずです。デフォルトでは、これは /callback にありますが、必要に応じて APIcast 設定ファイル内で簡単に変更することができます。

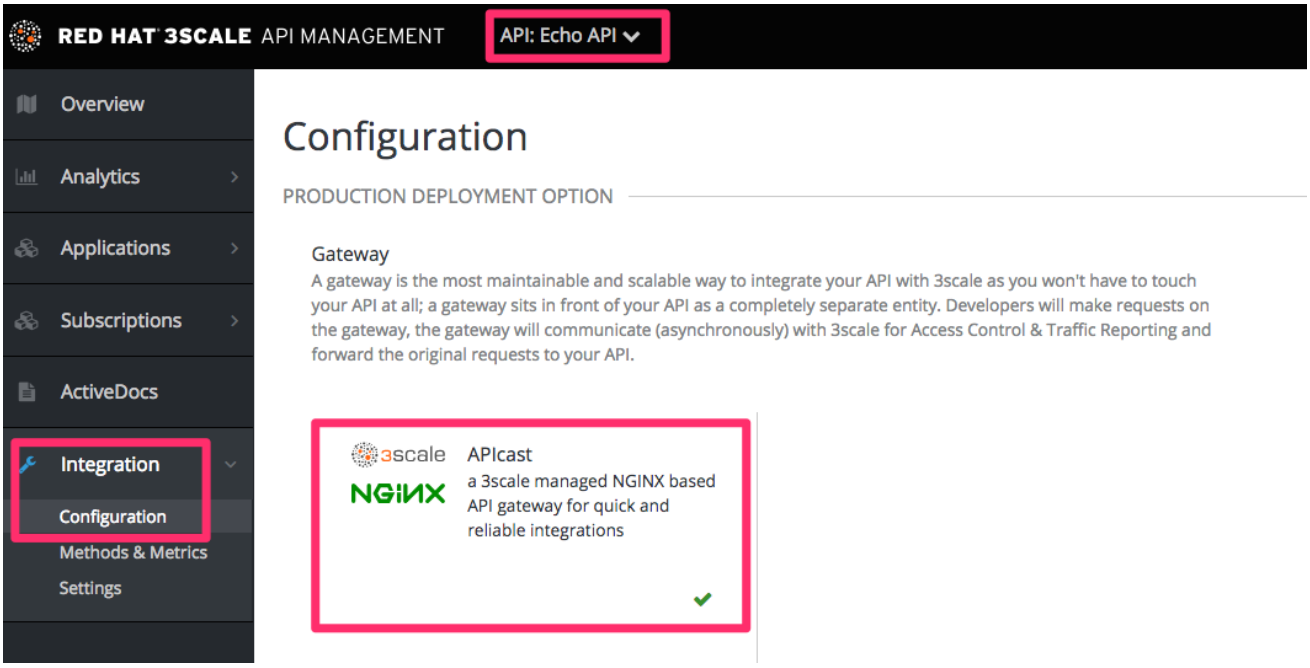
この設定方法を確認します。

7.2. OAUTH の設定

インストールを続行するには、API を設定しメソッドおよびエンドポイントを定義する基本的な APIcast インテグレーションとほとんど同じ手順に従う必要があります。これらの手順は、Installing APIcast を参照してください。また、以下の手順に従う必要があります。

7.2.1. ステップ 1: インテグレーション設定の編集

OAuth は現在 Hosted APIcast では利用できないため、以下のスクリーンショットのとおり、Production Deployment Option を自己管理型ゲートウェイに設定する必要があります。



同じページで、Authentication を OAuth 2.0 に設定する必要もあります。

7.2.2. ステップ 2: OAuth 承認エンドポイントの宣言

これは、ユーザーがお客様のサービスにログインして認証してコンテンツを提供しなければならない際に、ユーザーに提示される URL です。

APIcast OAuth 拡張機能により、APIcast は OAuth プロバイダーとして機能することができます。ただし、ユーザーを認証し、サードパーティーアプリケーションのアクセスを承認/拒否するには、承認エンドポイントをユーザーに提供する必要があります。ユーザーを識別および認証できるように、この承認エンドポイントはログインの背後でなければなりません。承認が完了したら、残りのワークフローに対応できるように、ログインしたユーザーを APIcast のコールバックエンドポイントにリダイレクトする必要があります。

7.2.3. ステップ 3: APIcast 設定ファイルのダウンロード

Integration のページに入力したデータに基づいて、3scale は APIcast を API ゲートウェイおよび OAuth プロバイダーとして使用するのに必要なすべてのファイルを自動的に生成します。必要な情報をすべて入力したら、これらのファイルをダウンロードし、自分の APIcast インスタンスにインストールできます。ダウンロードした zip ファイルには、定義した各サービスの個別 ***.lua** ファイルに加えて、OAuth ハンドシェイクをサポートするための lua ファイルおよびサービス間で共有される **nginx_*.conf** ファイルが含まれます。

複数のサービスが定義されている場合、設定ファイルをダウンロードするユーザーが受け取る zip ファイルの Lua ファイルには、アクセス権限のあるサービスに対応する サービストークン だけが含まれます。これにより、プロバイダーキー は完全な管理者にのみ秘密に保持されます。

7.3. SELF-MANAGED APICAST インスタンスの実行 (実稼働環境)

NGINX に精通している場合は、APIcast をローカルで稼働するのに時間がかかりません。NGINX インストールには Lua プラグインがなければなりません。また、OAuth 2.0 付与タイプによっては、Redis もサーバーにインストールされている必要があります。

NGINX に精通していない場合には、OpenResty をインストールすることを推奨します。これは基本的に標準 NGINX コアと組み込む必要のあるほとんどすべてのサードパーティー NGINX モジュールがバンドルされた素晴らしい Web アプリケーションです。

7.3.1. ステップ 1: 依存関係のインストール (Ubuntu 用)

Debian/Ubuntu linux ディストリビューションの場合は、apt-get を使用して以下のパッケージをインストールする必要があります。

```
sudo apt-get install libreadline-dev libncurses5-dev libpcre3 libpcre3-dev libssl-dev perl
sudo apt-get build-dep nginx
```

異なるシステムについては、OpenResty のドキュメントを参照してください。

7.3.2. ステップ 2: OpenResty のコンパイルおよびインストール

コードをダウンロードしてコンパイルします。VERSION を希望のバージョンに変更します (通常は、最新の安定バージョンを実行することを推奨します)。

```
wget http://agentzh.org/misc/nginx/ngx_openresty-VERSION.tar.gz
tar -zxvf ngx_openresty-VERSION.tar.gz
cd ngx_openresty-VERSION/

./configure --prefix=/opt/openresty --with-luajit --with-http_iconv_module -j2

make
make install
```

この時点で、OpenResty バンドルを使用して NGINX と Lua がインストールされています。

7.3.3. ステップ 3: Redis のインストール

APIcast サーバーに Redis をダウンロードしてインストールします (常に最新の安定バージョンを使用することを推奨します)。

```
tar zxvf  redis-VERSION.tar.gz
cd redis-VERSION
make
sudo make install
```

Redis サーバーをインストールして実行するには、以下のコマンドを実行し、すべてのデフォルト値を受け入れます。

```
sudo ./utils/install_server.sh
```

7.3.4. ステップ 4: 3scale からの APIcast 設定のダウンロード

Integration ページからダウンロードする場合、現在認可コード (サーバー側の) 付与フロー設定のみが利用可能である点に留意してください。ただし、この GitHub リポジトリー で、その他のすべてのフローの設定テンプレートを確認することができます。

Download ボタンをクリックして、3scale から APIcast 設定ファイルをダウンロードします。これにより、内部に 6 つのファイルが含まれる zip ファイルが得られます。

- **authorize.lua**: このファイルには、クライアントの承認、エンドユーザーの OAuth ログインページへのリダイレクト、アクセストークンの生成、戻りの URL が API 利用者が指定した URL と一致するかの確認、に関するロジックが含まれます。これは、/authorize エンドポイントがヒットすると実行されます。

- **authorized_callback.lua**: このファイルには、API エンドユーザーを API 利用者のリダイレクト URL に戻すロジックが含まれます。ユーザーが正常にログインし、API 利用者が要求したアクセスを承認すると、API プロバイダーとしてこのエンドポイントを呼び出す必要があります。このファイルは、/callback エンドポイントが Web アプリケーションによって呼び出されると実行されます。

- **get_token.lua**: このファイルには、client_id によって識別されるクライアントのアクセストークンを返すロジックが含まれます。これは、/oauth/token エンドポイントが呼び出されると実行されます。

- **nginx_*.conf**: .conf は一般的な NGINX 設定ファイルです。NGINX をすでに実行している場合は、自由にこのファイルを編集したり、既存の .conf にコピー/ペーストしたりしてください。

- **nginx_*.lua**: このファイルには、さまざまなメトリックおよびメソッドの使用を追跡するために Web インターフェイスで定義したロジックが含まれます。

- **threescale_utils.lua**

7.3.5. ステップ 5: APIcast の起動および停止

後は、APIcast を起動するだけです。これには複数の方法がありますが、最も分かり易い方法は以下のとおりです。

```
sudo /opt/openresty/nginx/sbin/nginx -p /opt/openresty/nginx/ -c /opt/openresty/nginx/conf/YOUR-CONFIG-FILE.conf
```

この例では、APIcast の作業ディレクトリーが、**--prefix=/opt/openresty** を設定するためにインストールする際に渡したパス **/opt/openresty/nginx** であることを前提としています。このディレクトリーは変更できますが、ユーザー権限に注意してください。

この例では、3scale によって生成された .conf が **/opt/openresty/nginx/conf/** に置かれていることも前提としています。当然ながら、実際の実稼働環境に最適な場所にファイルとディレクトリーを配置し、直接バイナリーを実行するのではなく、システムデーモンとしてプロセスを起動および停止する必要があります。

実行中の APIcast インスタンスを停止するには、以下のコマンドを実行します。

```
sudo /opt/openresty/nginx/sbin/nginx -p /opt/openresty/nginx/ -c /opt/openresty/nginx/conf/YOUR-CONFIG-FILE.conf -s stop
```

-s オプションにより、nginx にシグナルを渡します。停止するプロセスは、.pid が **/opt/openresty/nginx/logs/nginx.pid** に保存されているプロセスです。

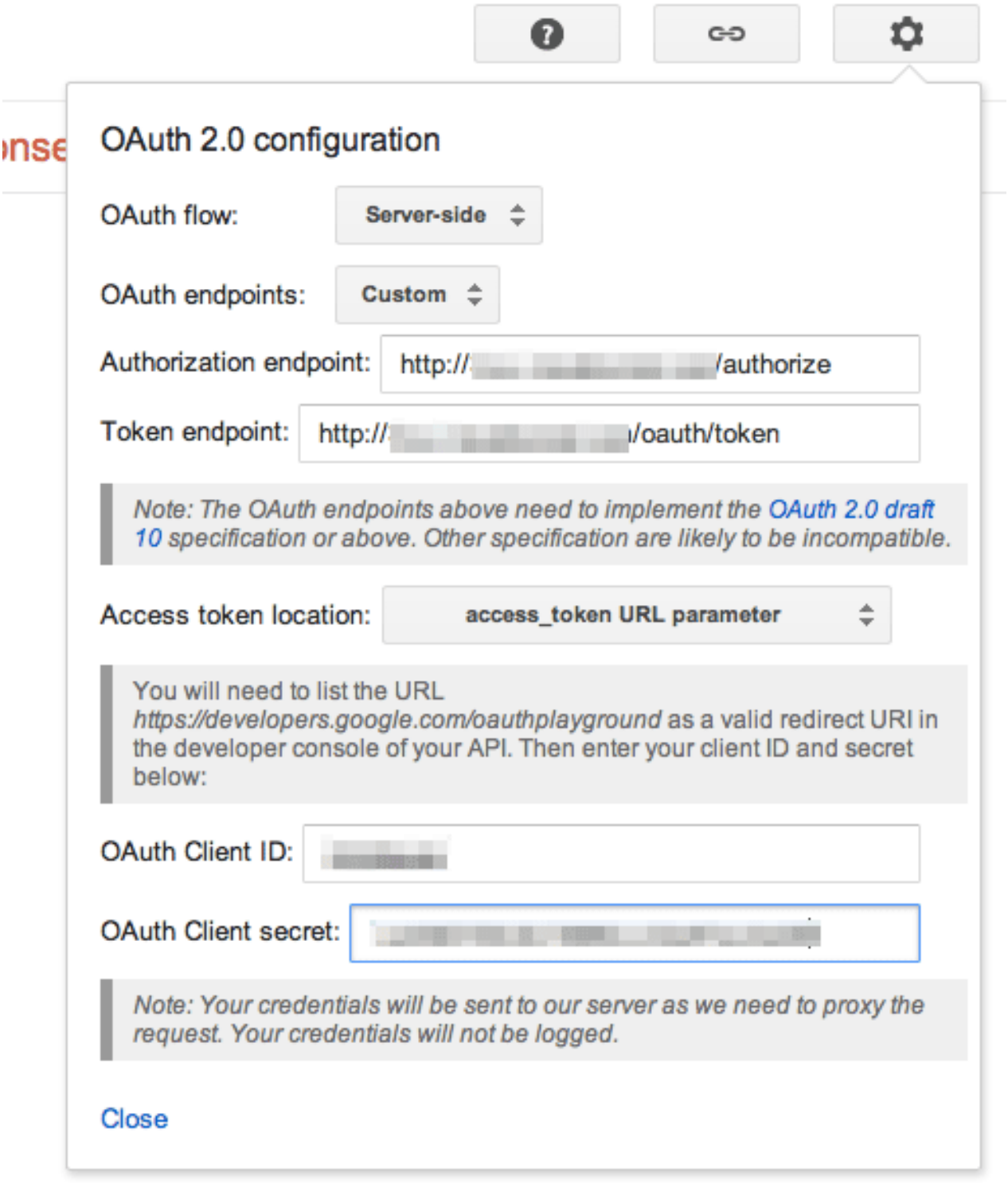
APIcast のログは、デフォルトで同じディレクトリー **/opt/openresty/nginx/logs/** にあります。プロセス全体を設定する場合は、error.log を確認してください。

7.3.6. ステップ 6: OAuth フローのテスト

API が OAuth をサポートしていることをテストする最善の方法は、Google の OAuth Playground (**https://developers.google.com/oauthplayground**) を使用することです。

このテストに使用するアプリケーションのリダイレクト URL を、Google OAuth Playground の URL **https://developers.google.com/oauthplayground** に設定する必要があります。

これで、以下のスクリーンショットに示すように、設定を入力することができます。



承認およびトークンエンドポイントの URL は、APIcast インスタンスからの URL です。スコープに、アプリケーションのアプリケーションプラン名を追加します (例: デフォルト)。

Authorize API をクリックすると、ログイン URL にリダイレクトされます。次に、アプリケーションのユーザーアカウントにログインし、アプリケーションを承認します。完了すると、認可コードと共に Google OAuth Playground にリダイレクトされます。これをアクセストークンと交換します。これで、API の保護されたエンドポイントを呼び出すアクセストークンを取得しました。

これで API にリクエストを行うことができます。ここで、API バックエンドのホスト名 (この例では echo-api.3scale.net) を APIcast インスタンスのホスト名で置き換え、access_token パラメーターを追加します。以下に例を示します。

```
curl -X GET "http://YOUR_APICAST_HOST/read?access_token=YOUR_ACCESS_TOKEN"
```

これで API が 3scale と統合されました。