Red Hat 3scale API Management 2.9

デベロッパーポータルでの 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: 2020-11-01

Red Hat 3scale API Management 2.9 デベロッパーポータルでの 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 API Management 2.9 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

本ガイドでは、デベロッパーポータルを強化する機能について説明します。

パート I. OPENAPI SPECIFICATION (OAS)

第1章 OAS をベースとした新しいサービスの作成

1.1. はじめに

本章では、Red Hat 3scale API Management 2.9 での OpenAPI Specification (OAS) 機能の概要、および既存サービスの更新や新規サービスの作成を行う手順について説明します。

以下は、3scale の OAS に関する特別な留意事項です。

- 3scale toolbox で OpenAPI 仕様をインポートすることもできます。詳細は、『3scale の操作』の「OpenAPI 定義のインポート」を参照してください。
- 3scale 2.8 では、OAS3.0 の変更が複数加えられました。詳細は、「3scale での OAS 3.0 の使用」を参照してください。

1.2. 前提条件

- OpenAPI Specification (OAS)
- 3scale 2.9 インスタンステナントのクレデンシャル (**token** または **provider_key**)

1.3. OPENAPI SPECIFICATION の機能

注記

ActiveDocs は、OpenAPI (OAS) のインポート時に作成、更新されます。

- 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ことができ、そのデフォルトは OAS からの **info.title** フィールドです。
- メソッドは OAS からの各操作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、OAS からの各操作に対して作成されます。
- OpenAPI 定義リソースは、以下のチャンネルのいずれかで提供することができます。
  - 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読み込む

1.4. OPENAPI SPECIFICATION の使用

```
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
```

1.4.1. ファイル名のパスからの OpenAPI 定義の検出

使用できるフォーマットは **json** と **yaml** です。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。

```
$ 3scale import openapi -d <destination> /path/to/your/spec/file.[json|yaml|yml]
```

1.4.2. URL からの OpenAPI 定義の検出

使用できるフォーマットは **json** と **yaml** です。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。

```
$ 3scale import openapi -d <destination> http[s]://domain/resource/path.[json|yaml|yml]
```

1.4.3. 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。

使用できるフォーマットは **json** と **yaml** です。形式はパーサーにより内部的に自動検出されます。

```
$ tool_to_read_openapi_from_source | 3scale import openapi -d <destination> -
```

第2章 OAS の設定

本章では、Red Hat 3scale API Management 2.9 における OpenAPI Specification (OAS) の設定手順の概要について説明します。

2.1. 3SCALE での OAS 3.0 の使用

本セクションでは、3scale で OpenAPI Specification 3.0 (OAS 3.0) を使用する方法について説明します。

限定的なサポートにより、3scale と OAS 3.0 の組み合わせを設定することができます。例を以下に示します。

- デベロッパーポータルで **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、デベロッパーポータル側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、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、要求の試行時のクレデンシャル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 を使用するデベロッパーポータルの設定

デベロッパーポータルで OAS 3.0 を設定するには、新しいページを追加するか、デフォルトのドキュメンテーションページを以下のスニペットに置き換えます。

```
{% 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>
```

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。

2.1.2. OAS 3.0 が設定されたデベロッパーポータルの更新

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。

以下のテンプレートを設定しているはずです。

```
{% 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 を使用するデベロッパーポータルの設定」に含まれるスニペットに置き換えます。

2.2. 3SCALE での OAS 2.0 の使用

本セクションでは、3scale で OpenAPI Specification 2.0 (OAS 2.0) を使用する方法について説明します。

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、要求の試行時のクレデンシャル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。

デベロッパーポータルで OAS 2.0 を設定するには、新しいページを追加するか、デフォルトのドキュメンテーションページを以下のスニペットに置き換えます。

```
<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 UI 2.1.3 から 2.2.10 へのアップグレード

使用している 3scale が Swagger UI 2.1.3 の組み込まれているバージョンである場合、Swagger UI バージョン 2.2.10 にアップグレードできます。

3scale デベロッパー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

3scale の以前のバージョンの Swagger UI は、引き続き、従来の **active_docs** Liquid タグメソッドを使用して呼び出されます。

Swagger UI 2.1.3 を 2.2.10 にアップグレードするには、以下の手順を実施します。

1. 3scale AMP 管理ポータルにログインします。
2. **Developer Portal → Documentation** ページの順に移動するか、Swagger UI 実装を更新するページに移動します。
3. コードペインで、**{% active_docs version: "2.0" %}** Liquid タグを **cdn_asset** Liquid タグの以下のアセットと新しいパーシャル **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. 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. API ドキュメント

第3章 3SCALE への仕様の追加

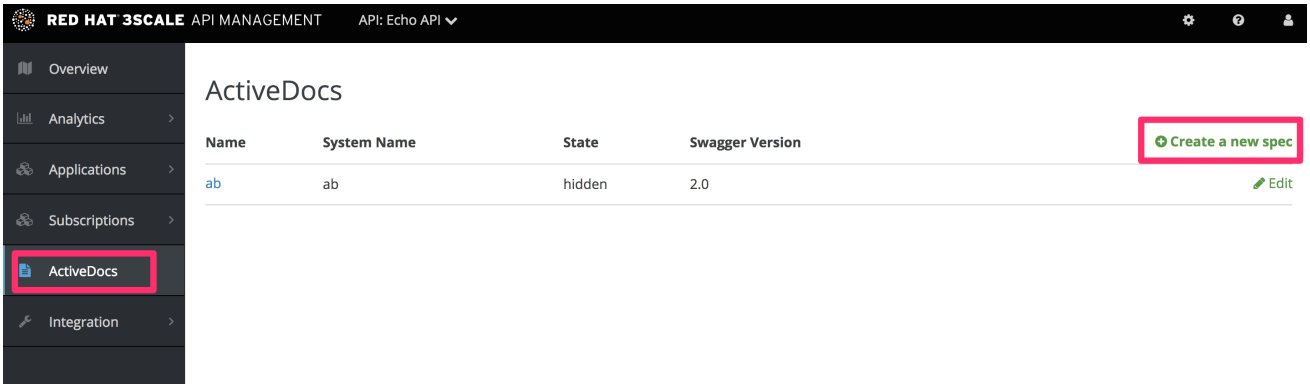
本章では、API に ActiveDocs を設定する手順について説明します。

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。

3.1. ACTIVEDOCS のサービス仕様への移動

管理ポータルで **[your_API_name] → ActiveDocs** の順に移動します。これにより、API のサービス仕様のリストが表示されます (初期段階は空)。



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)。

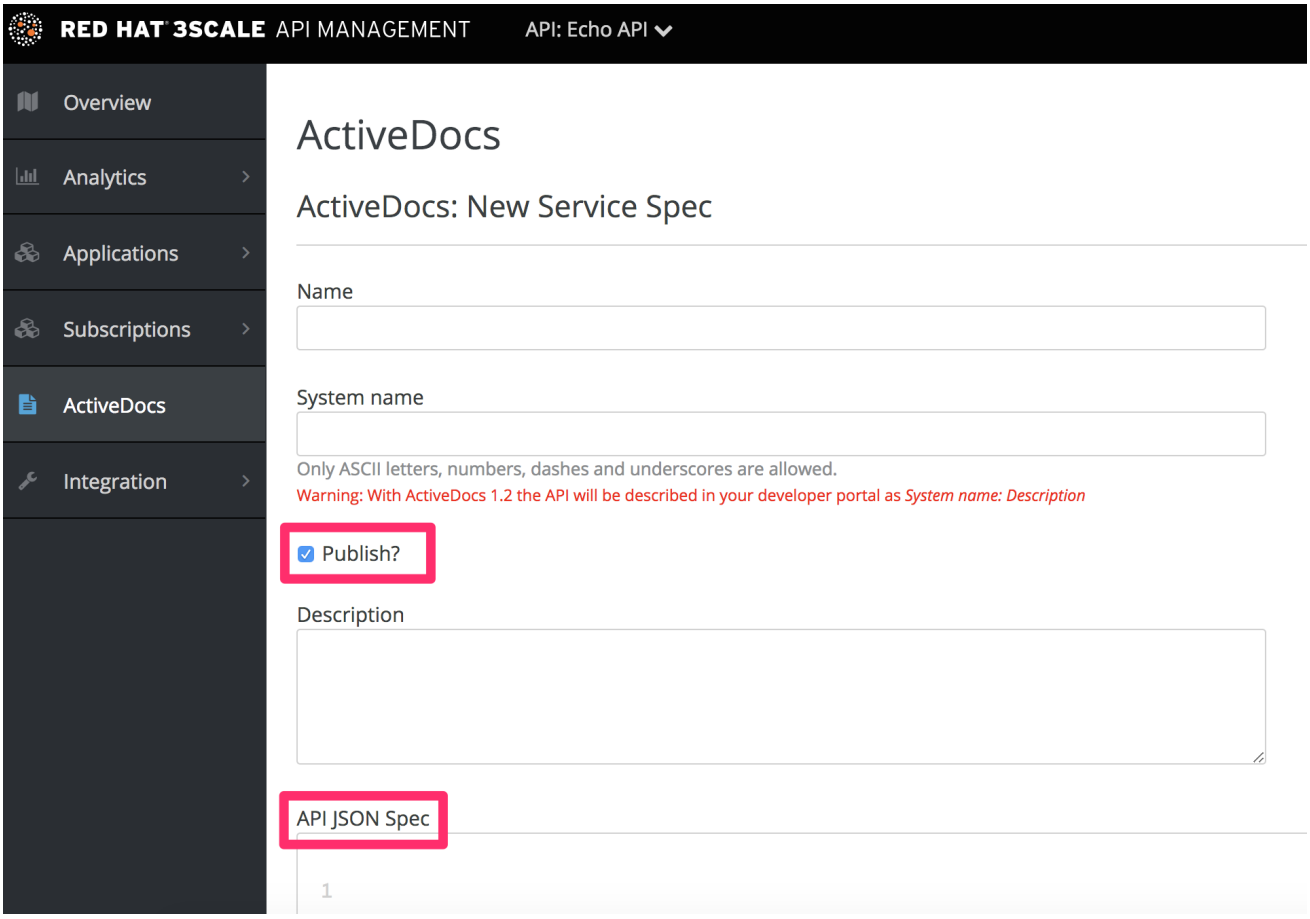
3.2. サービス仕様の作成

新しいサービス仕様を追加する場合は、以下の項目を指定する必要があります。

- 名前
- システム名 (デベロッパーポータルからサービス仕様を参照するために必要)
- 仕様を公開するかどうか
- 説明 (内部的にのみ使用される)
- API JSON 仕様 (以下の図を参照)

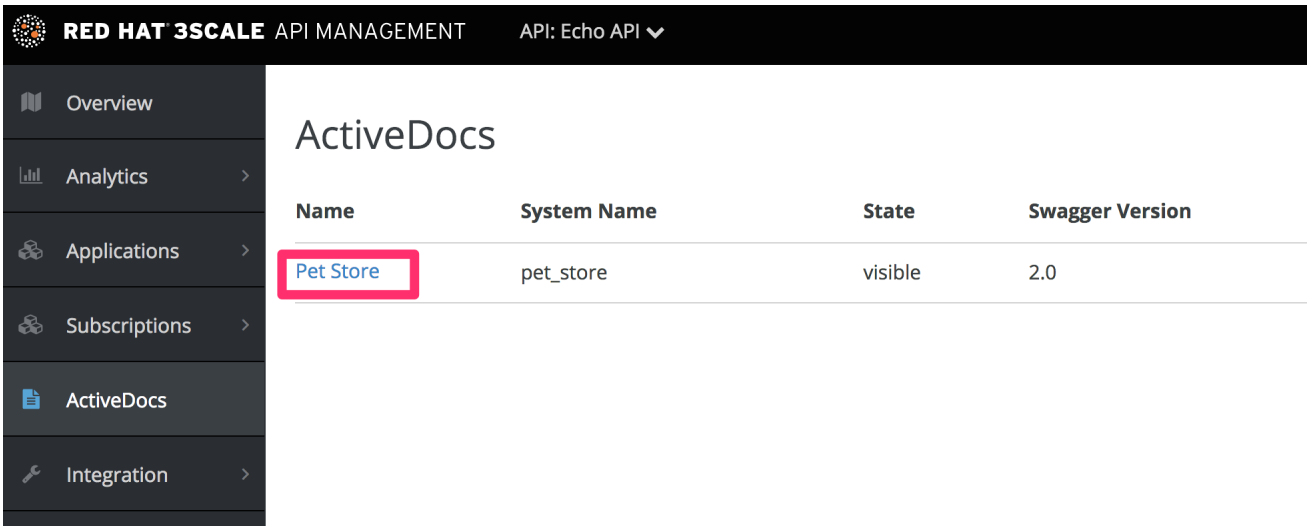
API JSON 仕様は、ActiveDocs の「秘密の成分」です。

API の仕様は、OpenAPI Specification (OAS) が提案する仕様に従って生成する必要があります。本チュートリアルは、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。



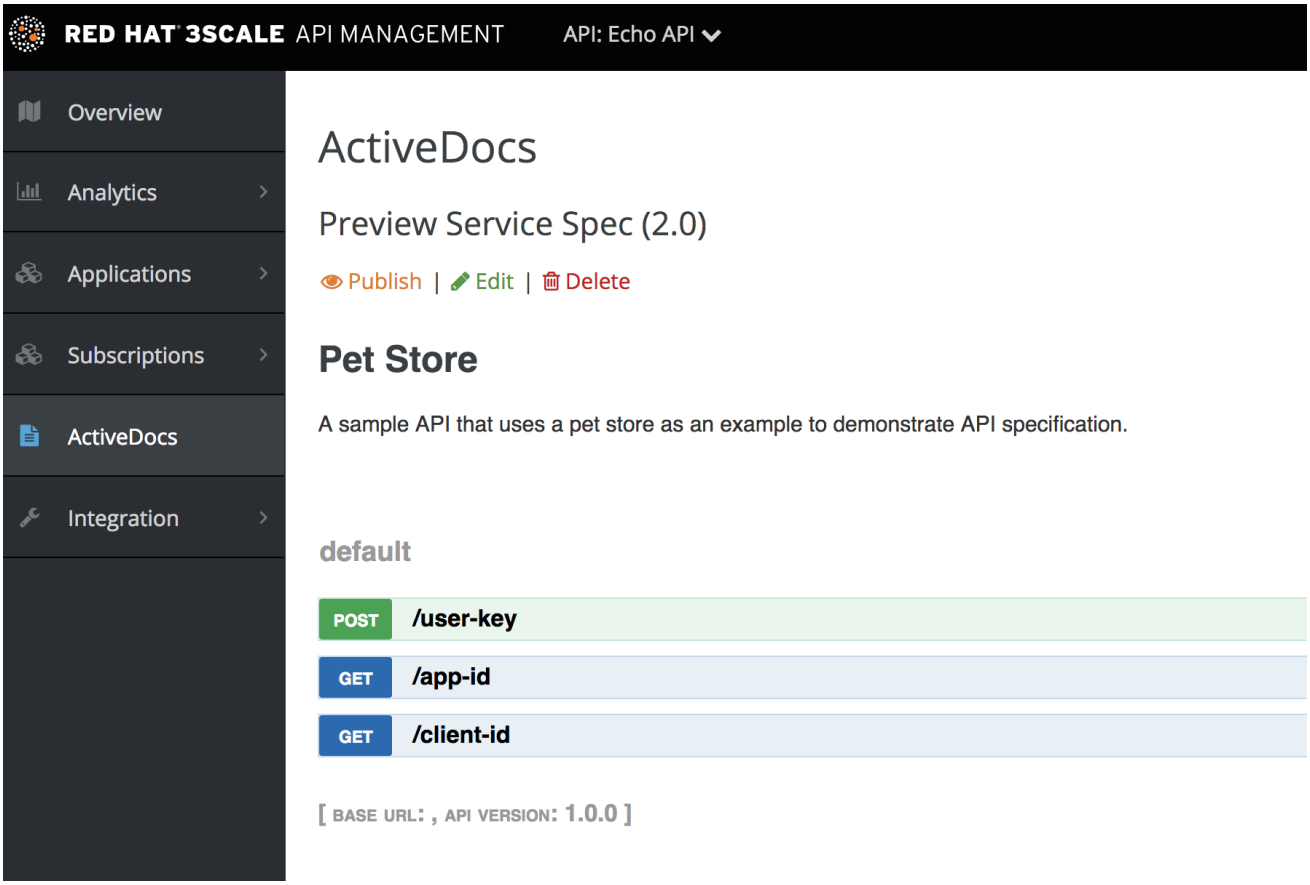
3.3. 最初の ACTIVEDOC に関する操作

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、これを 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。



また、サービス仕様に指定した名前 (この例では Pet Store) をクリックして、ActiveDocs がどう見え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。

以下は、ActiveDoc の見え方の例です。


RED HAT 3SCALE API MANAGEMENT
API: Echo API
Overview
Analytics
Applications
Subscriptions
ActiveDocs
Integration
ActiveDocs
Preview Service Spec (2.0)
Publish | Edit | Delete
Pet Store
A sample API that uses a pet store as an example to demonstrate API specification.
default
POST /user-key
GET /app-id
GET /client-id
[BASE URL: , API VERSION: 1.0.0]

第4章 OAS 仕様の作成

本章は、REST API 用に OpenAPI Specification (OAS) 準拠の仕様を作成するのに役立ちます。この仕様は、デベロッパーポータルで ActiveDocs を動作させるのに必要です。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。

4.1. OPENAPI SPECIFICATION (OAS) について

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。

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

1. 複数の言語 (NodeJS、Scala、その他) によるリソース仕様のサーバー
2. 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
3. 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。数多くの現代的な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。

4.2. 3SCALE ACTIVEDOCS と OAS

ActiveDocs とは OAS のインスタンスのことです。ActiveDocs を使用する場合に、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ースコンポーネントを扱ったりする必要がありません。インタラクティブドキュメントは、3scale のデベロッパー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、ActiveDocs での OAS 3.0 のサポートに限りがあります。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 OAS 3.0 の使用」を参照してください。

前提条件

- デベロッパーポータルで使用するテンプレートが、管理ポータルで指定した OAS バージョンと同じものであることを確認します。

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。

以下の手順により、インタラクティブドキュメントが利用可能になります。開発者は、デベロッパーポータルを通じて API に対するリクエストを行うことができます。

すでに OAS 準拠の API 仕様がある場合は、それをデベロッパー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)。

3scale では OAS 仕様をさまざまな方法で拡張し、独自のインタラクティブな API ドキュメント作成で必要となった特定の機能に対応しました。

- 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

4.3. API 仕様の作成

まず、オリジナルのソースから仕様を確認することを推奨します。OAS 仕様 を参照してください。

OAS サイトには、仕様の例が複数あります。例を使って学習したい場合は、OAS API チームが作成した Petstore API の例に従うことができます。

4.3.1. Petstore API の例を使った学習

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 は、4 つのメソッドで構成されています。

- **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、User Key 認証方法を使用すると、パラメーターがヘッダーに送信されます。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。

以下のパラメーターを追加する必要があります。

user_key: {user_key}

user_key は、開発者により API へのリクエストで送信されます。開発者は、デベロッパーポータルでこれらのキーを取得します。キーの受信時に、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。

開発者にとって、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}"
```

ただし、ドキュメントを OAS Petstore のドキュメント のようにしたい場合は、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。詳細は、次のセクションで説明します。

4.3.2. OAS 仕様の詳細について

OAS 仕様は、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** ファイルを例にして、ステップごとに手順を説明します。

4.3.2.1. 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一覧表示します。

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** エンドポイントがホワイトリストに登録されている必要があります。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。

4.3.2.2. 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。これは ActiveDocs ページに提示されます。

4.3.2.3. 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。ACL の制約により、パスは空である場合があります。

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、これらは JSON スキーマドラフト 4 でサポートされるデータタイプに基づきます。プリミティブデータタイプには「file」もありますが、これが機能するのは、API エンドポイントで CORS が有効な場合 (したがって、アップロードは api-docs ゲートウェイを経由しない) のみです。そうでない場合は、ゲートウェイレベルで停止します。

サポートされるデータタイプ

現在、OAS は以下の **データタイプ** 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符号付きです。
- 数値型フォーマット: float および double
- プレーン文字列
- 使用できる形式の文字列: バイト、日付、日時、パスワード、およびバイナリー
- boolean

4.3.3. OAS の設計および編集ツール

OpenAPI 仕様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ので、OpenAPI 仕様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ます。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。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。

4.3.4. ActiveDoc の API キー自動入力

API キーの自動入力は、3scale ActiveDocs の OAS 仕様に対する有用な拡張機能です。parameters セクションで、API 認証モードに応じて、**x-data-threescale-name** フィールドに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)。

API キー認証の例

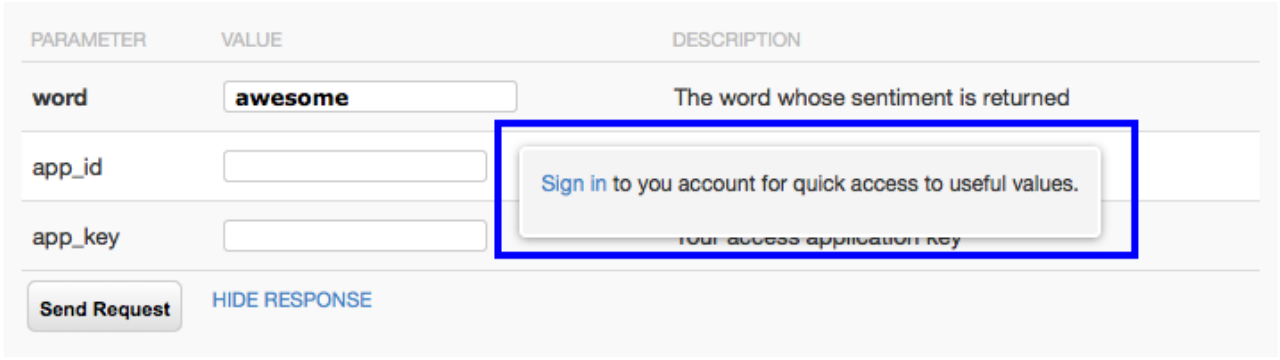
API キー認証のみの場合に **"x-data-threescale-name": "user_keys"** を使用する例を以下に示します。

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

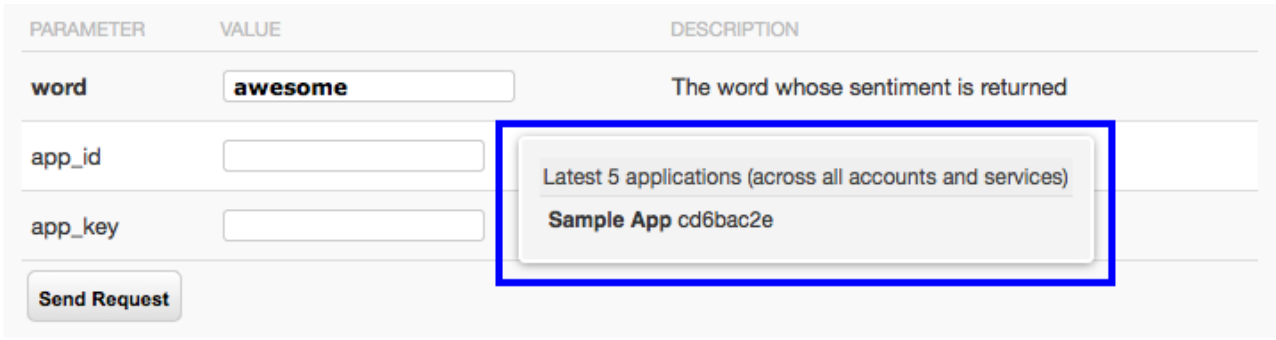
アプリケーション ID/アプリケーションキー認証の例

アプリケーション ID/アプリケーションキー認証モードの場合には、アプリケーション ID を表すパラメーターには **"x-data-threescale-name": "app_ids"** を指定し、アプリケーションキーを表すパラメーターには **"x-data-threescale-name": "app_keys"** を指定します。

パラメーターを宣言していると、ActiveDocs は自動的に、ActiveDocs ユーザーにデベロッパーポータルにログインしてキーを取得するよう求めます (以下のスクリーンショットを参照)。



ユーザーがすでにログインしている場合は、ActiveDocs は該当する可能性のある直近 5 つのキーを表示します。これにより、キーをコピー/ペーストせずにすぐテストすることができます。



注記

x-data-threescale-name フィールドは OAS 仕様の拡張機能で、ActiveDocs 以外のユースケースでは無視されます。

第5章 ACTIVEDOCS と OAUTH

本チュートリアルでは、ユーザーが OAuth 対応の API をテストして呼び出せるように、一元的にさまざまな ActiveDocs を設定する方法について説明します。

5.1. 前提条件

- Red Hat Single Sign-On インスタンスおよび OpenID Connect の統合を設定しておく必要があります。設定方法の詳細は、OpenID Connect の統合 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 への仕様の追加」および「OAS 仕様の作成」を参照してください。

5.2. クライアントクレデンシャルフローとリソースオーナーフロー

この最初の例は、OAuth 2.0 クライアントクレデンシャルフローを使用する 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、ヘッダー等) を返します。Echo API には、有効なアクセストークンがなければアクセスできません。API のユーザーは、クレデンシャル (**client_id** と **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リクエスト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を呼び出すことができます。この例のクライアントクレデンシャルフローの場合、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に加え、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も追加する必要があります。このクライアントクレデンシャルフローの例では、client_id と client_secret (3scale からのサインイン済みユーザーの値を代入可能)、ならびに grant_type を送信するだけです。

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する必要があります。

```
{
  "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、通常どおりデベロッパーポータルに 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>
```

第6章 デベロッパーポータルでの ACTIVEDOCS の公開

本チュートリアルでは、デベロッパーポータルで ActiveDocs を公開する方法について説明します。

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。

以下のスニペットを、デベロッパーポータルの任意のページのコンテンツに追加します。この操作は、デベロッパーポータルの CMS で行わなければなりません。SERVICE_NAME はサービス仕様のシステム名 (この例では **pet_store**) であることに注意してください。

```
<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 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>
```

デベロッパーポータルで 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のが最善の方法です。
- このスニペットには jQuery が必要ですが、デフォルトでデベロッパーポータルの main layout に含まれています。JQuery 依存関係を main layout 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。
- CMS ページで Liquid タグが有効になっていることを確認します。
- 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。