



# Red Hat 3scale API Management 2.10

## 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。



## Red Hat 3scale API Management 2.10 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。

Enter your first name here. Enter your surname here.

Enter your organisation's name here. Enter your organisational division here.

Enter your email address here.

## 法律上の通知

Copyright © 2021 | You need to change the HOLDER entity in the en-US/Providing\_APis\_in\_the\_Developer\_Portal.ent file |.

The text of and illustrations in this document are licensed by Red Hat under a Creative Commons Attribution–Share Alike 3.0 Unported license ("CC-BY-SA"). An explanation of CC-BY-SA is available at

<http://creativecommons.org/licenses/by-sa/3.0/>

. In accordance with CC-BY-SA, if you distribute this document or an adaptation of it, you must provide the URL for the original version.

Red Hat, as the licensor of this document, waives the right to enforce, and agrees not to assert, Section 4d of CC-BY-SA to the fullest extent permitted by applicable law.

Red Hat, Red Hat Enterprise Linux, the Shadowman logo, the Red Hat logo, JBoss, OpenShift, Fedora, the Infinity logo, and RHCE are trademarks of Red Hat, Inc., registered in the United States and other countries.

Linux<sup>®</sup> is the registered trademark of Linus Torvalds in the United States and other countries.

Java<sup>®</sup> is a registered trademark of Oracle and/or its affiliates.

XFS<sup>®</sup> is a trademark of Silicon Graphics International Corp. or its subsidiaries in the United States and/or other countries.

MySQL<sup>®</sup> is a registered trademark of MySQL AB in the United States, the European Union and other countries.

Node.js<sup>®</sup> is an official trademark of Joyent. Red Hat is not formally related to or endorsed by the official Joyent Node.js open source or commercial project.

The OpenStack<sup>®</sup> Word Mark and OpenStack logo are either registered trademarks/service marks or trademarks/service marks of the OpenStack Foundation, in the United States and other countries and are used with the OpenStack Foundation's permission. We are not affiliated with, endorsed or sponsored by the OpenStack Foundation, or the OpenStack community.

All other trademarks are the property of their respective owners.

## 概要

本ガイドでは、Red Hat 3scale API Management 2.10 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 .....	3
多様性を受け入れるオープンソースの強化 .....	4
パート I. OPENAPI SPECIFICATION (OAS) .....	5
第1章 OPENAPI 仕様の概要 .....	6
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	6
1.2. API 仕様をインポートするためのさまざまなソース	7
第2章 OPENAPI SPECIFICATION の設定方法 .....	9
2.1. 3SCALE での OPENAPI SPECIFICATION 3.0 の使用	9
2.1.1. OAS 3.0 を使用するデベロッパーポータルの設定	9
2.2. 3SCALE での OPENAPI SPECIFICATION 2.0 の使用	10
2.3. SWAGGER ユーザーインターフェースの 2.1.3 から 2.2.10 へのアップグレード	11
パート II. デベロッパーポータルの API ドキュメント .....	12
第3章 3SCALE への ACTIVEDOCS の追加 .....	13
3.1. 3SCALE での ACTIVEDOCS の設定	13
第4章 3SCALE OPENAPI 仕様として使用する OPENAPI ドキュメントの作成方法 .....	15
4.1. 3SCALE ACTIVEDOCS および OAS の設定	15
4.2. OPENAPI ドキュメントの例: PETSTORE API	16
4.3. OAS 仕様に関する補足情報	16
4.4. OAS の設計および編集ツール	17
4.5. ACTIVEDOC の API キー自動入力	18
第5章 ACTIVEDOCS と OAUTH .....	20
5.1. 3SCALE 仕様でのクライアントクレデンシャルおよびリソースオーナーフローの例	20
5.2. デベロッパーポータルでの ACTIVEDOCS の公開	23



---

## 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 仕様の概要

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するか、新しいサービスを作成する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 仕様 (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 2.10 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。[How to write an OpenAPI document for use as a 3scale specification](#) 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 (デフォルトは OAS からの **info.title** フィールド) として渡す機能。
- メソッドは、OpenAPI 仕様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 仕様で定義された各オペレーションに対して作成されます。
- 以下のチャンネルの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読み込む

### 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ース (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
<p>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。形式はパーサーにより内部的に自動検出されます。</p>	<p>json および yaml</p>	<pre>\$ tool_to_read_openapi_from _source   3scale import openapi -d &lt;destination&gt; -</pre>

## 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 2.10 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。

- 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

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。

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">&nbsp;</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 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。

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ースの 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

Swagger UI 2.1.3 以前のバージョンでは、3scale では引き続き従来の `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. デベロッパーポータルの API ドキュメント



## 第3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。

### 3.1. 3SCALE での ACTIVEDOCS の設定

3scale ユーザーインターフェースで 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 2.10 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、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です。
  - 仕様を公開するかどうかを選択します。公開しない場合には、デベロッパーポータルで新しい仕様は利用できません。



#### 注記

新しい仕様を作成するものの、公開しない場合には、今後希望の時期に公開するために利用可能な状態を維持します。

- ご自分だけが使用する目的の説明を追加します。
- API JSON 仕様を追加します。  
API の仕様は、[OpenAPI Specification \(OAS\)](#) が提案する仕様に従って生成します。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。

#### 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。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。

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。

The screenshot displays the Red Hat 3scale API Management developer portal. The top navigation bar includes the Red Hat 3scale logo, the text "RED HAT 3SCALE API MANAGEMENT", and the current API name "API: Echo API" with a dropdown arrow. A left-hand sidebar contains a menu with the following items: Overview, Analytics, Applications, Subscriptions, ActiveDocs (highlighted), and Integration. The main content area is titled "ActiveDocs" and shows a "Preview Service Spec (2.0)" for the "Pet Store" API. Below the title are three action buttons: "Publish" (with an eye icon), "Edit" (with a pencil icon), and "Delete" (with a trash icon). The description for the "Pet Store" API is "A sample API that uses a pet store as an example to demonstrate API specification." Under the heading "default", there is a list of API endpoints: a POST endpoint for "/user-key", a GET endpoint for "/app-id", and a GET endpoint for "/client-id". At the bottom of the list, there is a note: "[ BASE URL: , API VERSION: 1.0.0 ]".

## 第4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

- デベロッパーポータルで 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す。

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。

### 4.1. 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、OAS 3.0 が導入されましたが、ActiveDocs でのサポートは限定でした。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

- 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。

### 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、デベロッパーポータルを通じて 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」](#) を参照してください。

3scale では OAS 仕様をさまざまな方法で拡張し、デベロッパーポータルで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

## 4.2. 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。

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、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、「[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、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}"
```

## 4.3. 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

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` ファイルを例として使用し、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一覧表示します。

### 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、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 `ファイル` 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 を有効にすると、アップロードが拒否される `api-docs` ゲートウェイを通過しません。

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`

### その他のリソース

- [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](#)

## 4.4. OAS の設計および編集ツール

API を定義する 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。

## 4.5. ACTIVEDOC の API キー自動入力

API キーの自動入力は、3scale ActiveDocs の OAS に対する有用な拡張機能です。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 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

**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。

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。

パラメーターの宣言後、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)。

PARAMETER	VALUE	DESCRIPTION
word	<input type="text" value="awesome"/>	The word whose sentiment is returned
app_id	<input type="text"/>	<a href="#">Sign in</a> to you account for quick access to useful values.
app_key	<input type="text"/>	
<input type="button" value="Send Request"/>		<a href="#">HIDE RESPONSE</a>

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/ペーストせずすぐにテストすることができます。

PARAMETER	VALUE	DESCRIPTION
word	<input type="text" value="awesome"/>	The word whose sentiment is returned
app_id	<input type="text"/>	Latest 5 applications (across all accounts and services) <b>Sample App</b> cd6bac2e
app_key	<input type="text"/>	
<input type="button" value="Send Request"/>		

## 第5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ります。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 仕様の作成](#)」を参照してください。

### 5.1. 3SCALE 仕様でのクライアントクレデンシャル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、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。このクライアントクレデンシャルフローの例では、`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>

```

## 5.2. デベロッパーポータルでの ACTIVEDOCS の公開

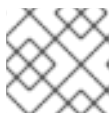
本チュートリアルでは、デベロッパーポータルで ActiveDocs を公開すると共に、API ドキュメントを自動化する方法について説明します。

### 前提条件

- デベロッパーポータルで ActiveDocs を動作させるには、REST API に対する OpenAPI Specification (OAS) 準拠の仕様が必要があります。

### 手順

- 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。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。

```

<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。
- 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。
- 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する必要があります。

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。