Red Hat Connectivity Link 1.0

API コントローラーのスタートガイド

API の作成と管理

Last Updated: 2024-12-16

Red Hat Connectivity Link 1.0 API コントローラーのスタートガイド

API の作成と管理

法律上の通知

概要

このガイドでは、Connectivity Link の使用を開始する方法を説明します。

目次

はじめに

RED HAT ドキュメントへのフィードバック

製品ドキュメントに関するご意見をお寄せください。

改善を提案するには、Jira 課題を作成し、変更案を説明してください。ドキュメントチームがご要望に迅速に対応できるよう、できるだけ詳細にご記入ください。

前提条件

- Red Hat カスタマーポータルのアカウントがある。このアカウントを使用すると、Red Hat Jira Software インスタンスにログインできます。アカウントをお持ちでない場合は、アカウントを作成するように求められます。

手順

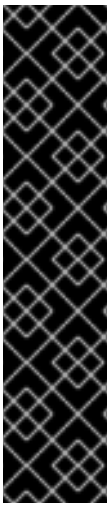
1. Create issue にアクセスします。
2. **Summary** テキストボックスに、問題の簡単な説明を入力します。
3. **Description** テキストボックスに、次の情報を入力します。
   - 問題が見つかったページの URL。
   - 問題の詳細情報。他のフィールドの情報はデフォルト値のままにすることができます。
4. **Reporter** フィールドに Jira ユーザー名を入力します。
5. **Create** をクリックして、Jira 課題をドキュメントチームに送信します。

フィードバックをご提供いただきありがとうございました。

第1章 API コントローラーとは

API コントローラーを使用して、スキーマと API 定義のライフサイクルを管理できます。Apicurio Studio を使用してアーティファクトを設計し、それらのアーティファクトを Apicurio Registry を使用して管理できます。

このリリースは、API コントローラーの開発者プレビューです。



重要

開発者プレビュー機能は、Red Hat ではいかなる形でもサポートされていません。また、機能的には完全ではなく、実稼働環境に対応していません。開発者プレビュー機能を実稼働ワークロードまたはビジネスクリティカルなワークロードには使用しないでください。開発者プレビュー機能は、Red Hat 製品に追加される可能性がある機能をいち早く提供することを目的としています。お客様はこの機能を使用してテストし、開発プロセス中にフィードバックを提供できます。

開発者プレビュー機能は、ドキュメントが提供されていない場合があり、随時変更または削除される可能性があります。また、限定的なテストしか行われていません。SLA を関連付けずに開発者プレビュー機能に関するフィードバックを送信する方法を Red Hat が提供している場合があります。詳細は、Red Hat 開発者プレビューのサポート範囲 を参照してください。

1.1. API コントローラーと CONNECTIVITY LINK の統合

API コントローラーで API またはスキーマを作成した後、そのアーティファクトを Connectivity Link で使用して以下を実行できます。

- TLSPolicy でアプリケーションを保護する
- AuthPolicy でアプリケーションを保護する
- RateLimitPolicy でアプリケーションを保護する
- アプリケーションを DNSPolicy に接続する

Connectivity Link は、Grafana、Prometheus、Alertmanager を使用して可観測性も提供します。

関連情報

- Connectivity Link のポリシー API と可観測性

1.2. APICURIO STUDIO と APICURIO REGISTRY の統合

Apicurio Studio には、API およびスキーマの設計時にドラフトアーティファクトを連携させる UI が含まれています。設計が完了した後、そのアーティファクトを Apicurio Registry で管理できます。

関連情報

- *3章Apicurio Studio の使用を開始する*
- Apicurio Registry のドキュメント

第2章 API コントローラーをインストールする

API コントローラーをインストールするには、コミュニティーの Operator を使用します。

前提条件

- OpenShift クラスターにアクセスするための **cluster-admin** 権限。

手順

1. OpenShift Container Platform Web コンソールで、**cluster-admin** 権限を使用してログインします。
2. 左側のナビゲーションメニューで、**Operators > OperatorHub** の順にクリックします。
3. **Filter by keyword** テキストボックスに **Apicurio** と入力し、**Apicurio API Controller** を見つけます。
4. Operator に関する情報を読み、**Install** をクリックして Operator サブスクリプションページを表示します。
5. 以下に示すデフォルトのサブスクリプション設定を受け入れます。
   - **Installation mode**: **All namespaces on the cluster (default)**
   - **Installed namespace**: Operator をインストールする namespace を選択します (例: **api-controller**)。まだ namespace が存在しない場合は、このフィールドをクリックし、**Create Project** を選択して namespace を作成します。
   - **Approval Strategy**: **Automatic** または **Manual** を選択します。
6. **Install** をクリックし、Operator がインストールされ、使用できる状態になるまでしばらく待ちます。
7. Operator がインストールされていることを確認します。Operator をインストールした後、**Operators > Installed Operators** をクリックし、選択した namespace (例: **api-controller**) に **Apicurio API Controller** がインストールされていることを確認します。
8. OpenShift Container Platform Web コンソールで **Developer** ビューに変更し、インストールに必要な YAML を適用します。
9. **api-controller** namespace で以下の YAML を使用して PostgreSQL データベースを作成します。

```
apiVersion: v1
kind: PersistentVolumeClaim
metadata:
  name: registry-pvc
  namespace: api-controller
spec:
  accessModes:
    - ReadWriteOnce
  resources:
    requests:
      storage: 1Gi  # Adjust the storage size as needed
---
```

```
apiVersion: apps/v1
kind: Deployment
metadata:
  namespace: "api-controller"
  labels:
    app: postgresql
  name: postgresql
spec:
  replicas: 1
  selector:
    matchLabels:
      app: postgresql
  template:
    metadata:
      labels:
        app: postgresql
    spec:
      initContainers:
        - name: init-data
          image: busybox
          command: ['sh', '-c', 'rm -rf /var/lib/postgresql/data/* && mkdir -p
/var/lib/postgresql/data/pgdata']
          volumeMounts:
            - mountPath: "/var/lib/postgresql/data"
              name: "registry-pgdata"
      containers:
        - name: postgresql
          image: quay.io/debezium/postgres:13-alpine
          ports:
            - containerPort: 5432
          env:
            - name: POSTGRES_DB
              value: registry
            - name: POSTGRES_USER
              value: apicurio
            - name: POSTGRES_PASSWORD
              value: registry
            - name: PGDATA
              value: "/var/lib/postgresql/data/pgdata"
          volumeMounts:
            - mountPath: "/var/lib/postgresql/data"
              name: "registry-pgdata"
      volumes:
        - name: registry-pgdata
          persistentVolumeClaim:
            claimName: registry-pvc
---
apiVersion: v1
kind: Service
metadata:
  namespace: "api-controller"
  labels:
    app: postgresql
  name: postgresql-service
spec:
  ports:
```

```
    - name: http
      port: 5432
      protocol: TCP
      targetPort: 5432
  selector:
    app: postgresql
  type: ClusterIP
```

10. **api-controller** namespace で次の YAML を使用して、 **apicurio** という名前の CR と必要なルートを作成します。

注記

mycluster.example.com は、クラスターのホスト名に置き換えます。

```
# Replace mycluster.example.com with your cluster hostname
# Create an API Controller custom resource
apiVersion: registry.apicur.io/v1
kind: ApicurioRegistry3
metadata:
  name: apicurio
  namespace: api-controller
spec:
  studioUi:
    enabled: true
    env:
      - name: APICURIO_REGISTRY_API_URL
        value: 'https://api-controller-app.apps.mycluster.example.com/apis/registry/v3'
      - name: APICURIO_REGISTRY_UI_URL
        value: 'https://api-controller-ui.apps.mycluster.example.com'
  ui:
    env:
      - name: REGISTRY_API_URL
        value: 'https://api-controller-app.apps.mycluster.example.com/apis/registry/v3'
  app:
    sql:
      dataSource:
        username: apicurio
        password: registry
        url: 'jdbc:postgresql://postgresql-service:5432/registry'
---
# Create a route for the Apicurio Registry API
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: api-controller-registry-api
  namespace: api-controller
spec:
  host: api-controller-app.apps.mycluster.example.com
  path: /
  to:
    kind: Service
    name: apicurio-app-service
  port:
    targetPort: http
```

```
  tls:
    termination: edge
    insecureEdgeTerminationPolicy: Redirect
  wildcardPolicy: None
---
# Create a route for the Apicurio Registry UI
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: api-controller-registry-ui
  namespace: api-controller
spec:
  host: api-controller-ui.apps.mycluster.example.com
  path: /
  to:
    kind: Service
    name: apicurio-ui-service
  port:
    targetPort: http
  tls:
    termination: edge
    insecureEdgeTerminationPolicy: Redirect
  wildcardPolicy: None
---
# Create a route for the Apicurio Studio UI
apiVersion: route.openshift.io/v1
kind: Route
metadata:
  name: api-controller-studio-ui
  namespace: api-controller
spec:
  host: api-controller-studio-ui.apps.mycluster.example.com
  path: /
  to:
    kind: Service
    name: apicurio-studio-ui-service
  port:
    targetPort: http
  tls:
    termination: edge
    insecureEdgeTerminationPolicy: Redirect
  wildcardPolicy: None
```

検証

api-controller-studio-ui ルートに移動し、**Location** URL をクリックします。Apicurio Studio コンソールが表示されるはずです。

第3章 APICURIO STUDIO の使用を開始する

3.1. APICURIO STUDIO とは

Apicurio Studio を使用して、スキーマと API 定義を設計できます。API の所有者や開発者が、イベントスキーマおよび API 定義のコンテンツを容易に管理できる Web コンソールが提供されます。

Apicurio Studio を使用して、シンプルなテンプレートからスキーマまたは API 設計を作成することも、提供される詳細なテンプレートを使用することも、既存の設計をインポートして編集することもできます。作業中の設計は、**DRAFT** として Apicurio Registry に自動的に保存されます。納得のいくコンテンツが完成したら、Apicurio Registry で設計を **ENABLED** にプロモートできます。

Apicurio Studio は、Apicurio Studio オープンソースコミュニティープロジェクトをベースとしています。

主な概念

Apicurio Studio の仕組みを理解するには、次の主要な概念を理解する必要があります。

Apicurio Studio Web コンソール

開発者が API およびスキーマ設計を作成、管理、整理する Web 環境。
Apicurio Studio Web コンソールを使用して、次のタスクを実行できます。

- Apicurio Studio に格納されているスキーマと API 設計を参照および検索する
- スキーマと API の新しい設計およびバージョンを追加する
- ファイル、URL、または Apicurio Registry インスタンスからコンテンツをインポートする
- 現在の編集セッションで加えた変更を表示する

ドラフト

API 設計またはスキーマ設計。ローカルプロジェクトにダウンロードした場合、または Apicurio Registry で使用する場合、設計は **アーティファクト** と呼ばれます。
Apicurio Studio は、次の API タイプをサポートしています。

- AsyncAPI
- OpenAPI

Apicurio Studio は、次のスキーマタイプをサポートしています。

- Apache Avro
- JSON Schema
- Google Protocol Buffers (Protobuf)

Apicurio Studio のユースケース

Apicurio Studio の主なユースケースは次のとおりです。

- **コントラクトファーストのアプリケーション開発**
  Apicurio Studio を使用して、アプリケーションコードを書く前に、アプリケーションに必要な API およびデータモデル (**コントラクト**) を視覚的に設計できます。コントラクトを定義した

後、そのコントラクトを満たすために必要なアプリケーションロジックを作成すると簡単です。Apicurio Studio で作成した設計から、Quarkus ベースのクライアントおよびサーバーアプリケーションを生成できます。

- **Apicurio Registry の設定**
  API およびスキーマの設計は、すべて Apicurio Registry に保存されます。Apicurio Registry 機能を使用できます。以下はその例です。

- 設計変更の検証ルールを作成する。

- Apicurio Registry REST API を使用して、複雑な JSON スキーマを逆参照する。

3.2. API ドラフトの作成

Apicurio Studio Web コンソールを使用して、OpenAPI または AsyncAPI 定義を作成します。

前提条件

- Apicurio Studio Web コンソールにログインしている。

手順

1. Apicurio Studio Web コンソールで、**Create draft** をクリックします。
2. ウィザードを完了し、新規ドラフトの次の詳細を指定します。
   a. **Draft Coordinates** を指定し、**Next** をクリックします。
      - **Group ID & Draft ID**: デフォルトの空の設定を使用して Draft ID を自動生成し、ドラフトを **default** Draft グループに追加します。または、オプションの Group ID または Draft ID を入力することもできます。
      - **Version number**: オプションでバージョン番号を指定します。
      - **Type**: デフォルトの **Auto-Detect** 設定を使用してドラフトタイプを自動的に検出する (空のドラフトを作成する場合は許可されません) か、リストからドラフトタイプ (**OpenAPI** など) を選択します。

   b. **Draft Content** を指定し、**Next** をクリックします。
      - **From template**: テンプレートリストから選択します。
        - **From local file**: **Browse** をクリックし、ファイルを選択するか、ファイルをドラッグアンドドロップします。たとえば、**my-openapi.json**、**my-schema.proto** などです。または、テキストボックスにファイルの内容を入力することもできます。
        - **From URL**: 有効かつアクセス可能な URL を入力し、**Fetch** をクリックします。たとえば、**https://petstore3.swagger.io/api/v3/openapi.json** です。

   c. **Draft Metadata** を指定します。
      - **Name**: オプションで、最初のアーティファクトバージョンのフレンドリーネームを入力します。
      - **Description**: オプションで、最初のアーティファクトバージョンの説明を入力します。
3. **Create** をクリックしてドラフトを作成します。ドラフトの詳細ビューが表示されます。

4. ドラフトを編集するには、**Edit draft** をクリックします。
    a. **Design** タブをクリックし、必要に応じて以下のようにドラフトを編集します。
        - バージョン番号と説明を入力します。
        - **(AsyncAPI のみ)** サービス条件を定義します。
        - 連絡先情報 (名前、メールアドレス、URL) を追加します。
        - ライセンスを選択します。
        - **(OpenAPI のみ)** タグを定義します。
        - サーバーを 1 つ以上定義します。
        - セキュリティースキームを設定します。
        - **(OpenAPI のみ)** セキュリティー要件を指定します。
        - **(OpenAPI のみ)** ベンダーエクステンションを設定します。
    b. **Source** タブをクリックして、ドラフトのライブプレビューを確認します。エディターページで値を編集すると、**Source** タブの内容が自動的に更新されます。
    c. **(オプション)** 最後に保存してから加えた変更を表示するには、**Actions > Show draft changes** をクリックします。
    d. **(オプション)** ドラフトエディターの左ペインで、以下の項目を追加できます。
        - **(OpenAPI のみ)** リソースパス、データタイプ、レスポンス
        - **(AsyncAPI のみ)** チャネル、データタイプ、メッセージ、オペレーショントレイト、メッセージトレイト
    e. **Save** をクリックします。
    f. ブレッドクラムを使用して、**Drafts** ページに戻ります。

新規ドラフトは、Group ID および Draft ID で **Drafts** ページにリスト表示されます。ドラフト詳細の表示、ドラフトコンテンツの編集、ドラフトの確定、Apicurio Registry でのドラフトの表示、ドラフトの削除を行う場合は、オプションアイコン (縦 3 点の省略記号) を使用します。

関連情報

「ドラフトの確定」 を参照

3.3. スキーマドラフトの作成

Apicurio Studio Web コンソールを使用して、Apache Avro、JSON Schema、または Google Protocol Buffers (Protobuf) のイベントスキーマを作成します。

前提条件

- Apicurio Studio Web コンソールにログインしている。

手順

1. Apicurio Studio Web コンソールで、**Create draft** をクリックします。
2. ウィザードを完了し、新規ドラフトの次の詳細を指定します。
   a. **Draft Coordinates** を指定し、**Next** をクリックします。
      - **Group ID & Draft ID**: デフォルトの空の設定を使用して Draft ID を自動生成し、ドラフトを **default** Draft グループに追加します。または、オプションの Group ID または Draft ID を入力することもできます。
      - **Version number**: オプションでバージョン番号を指定します。
      - **Type**: デフォルトの **Auto-Detect** 設定を使用してドラフトタイプを自動的に検出する (空のドラフトを作成する場合は許可されません) か、リストからドラフトタイプ (**Apache Avro** など) を選択します。
   b. **Draft Content** を指定し、**Next** をクリックします。
      - **From template**: テンプレートリストから選択します。
      - **From local file**: **Browse** をクリックし、ファイルを選択するか、ファイルをドラッグアンドドロップします。たとえば、**my-openapi.json**、**my-schema.proto** などです。または、テキストボックスにファイルの内容を入力することもできます。
      - **From URL**: 有効かつアクセス可能な URL を入力し、**Fetch** をクリックします。たとえば、**https://petstore3.swagger.io/api/v3/openapi.json** です。
   c. **Draft Metadata** を指定します。
      - **Name**: オプションで、最初のアーティファクトバージョンのフレンドリーネームを入力します。
      - **Description**: オプションで、最初のアーティファクトバージョンの説明を入力します。
3. **Create** をクリックしてドラフトを作成します。ドラフトの詳細ビューが表示されます。
4. ドラフトを編集するには、**Edit draft** をクリックします。
   a. **Design** タブをクリックし、ドラフトを編集します。
   b. **(オプション)** 最後に保存してから加えた変更を表示するには、**Actions > Show draft changes** をクリックします。
   c. **(オプション)** ドラフトエディターの左ペインで、以下の項目を追加できます。
   d. **Save** をクリックします。
   e. ブレッドクラムを使用して、**Drafts** ページに戻ります。

新規ドラフトは、Group ID および Draft ID で **Drafts** ページにリスト表示されます。ドラフト詳細の表示、ドラフトコンテンツの編集、ドラフトの確定、Apicurio Registry でのドラフトの表示、ドラフトの削除を行う場合は、オプションアイコン (縦 3 点の省略記号) を使用します。

関連情報

「ドラフトの確定」 を参照

3.4. ドラフトの確定

Apicurio Studio Web コンソールを使用して、ドラフトを確定します。このアクションにより、Apicurio Registry におけるアーティファクトのステータスが **DRAFT** から **ENABLED** に変更されます。

前提条件

- Apicurio Studio Web コンソールにログインしている。

手順

1. Apicurio Studio でドラフトに移動します。以下に例を示します。
   a. **Filter by** を **Name** に設定します。
   b. 検索フィールドにドラフトの名前を入力します。
   c. Search アイコンをクリックするか、Return を押します。
2. ドラフトは、次の 2 つの方法で確定できます。
   - 検索結果で、オプションアイコン (縦 3 点の省略記号) をクリックして **Finalize draft** を選択します。
   - 検索結果にリスト表示されているドラフトの座標をクリックしてドラフトの詳細ページを表示し、**Finalize draft** ボタンをクリックします。

   Final draft? モーダルが表示されます。
3. ドラフトを確定することを確認します。
   Dry run only オプションを使用すると、確定する前にドラフトを検証できます。

   アーティファクトの詳細が記載された確認ページが表示されます。

重要

ドラフトを確定すると、そのアーティファクトを Apicurio Studio で表示または編集できなくなります。これ以降は、Apicurio Registry を使用して編集する必要があります。

4. Apicurio Registry のアーティファクトに移動します。確認ページのオプションメニュー (縦 3 点の省略記号) で **View in Registry** を選択すると、Apicurio Registry のアーティファクトに移動できます。

第4章 APICURIO REGISTRY の使用を開始する

Apicurio Registry は、中央リポジトリーでデータスキーマ、API 設計、構造化された任意のコンテンツを管理および保存する強力なツールです。サポートされるアーティファクトタイプは、Avro、Protobuf、JSON Schema、OpenAPI 定義など多岐にわたります。

関連情報

- Apicurio Registry のドキュメント

4.1. アーティファクトとは

アーティファクトは、Apicurio Registry のビルディングブロックです。これらは、再利用および連携のためにレジストリーに格納されているデータ構造、API 設計、またはスキーマ定義を表します。アーティファクトのアップロード、更新、取得、バージョン履歴の表示、ライフサイクルの管理を実行できます。

4.2. グループの管理

グループを使用して、管理および取得しやすいようにアーティファクトを論理的に整理できます。各アーティファクトはグループに所属し、グループはドメイン、プロジェクト、またはチームを表すことができます。グループ化により、アクセスや使用量に関するポリシーを細かく制御しながらの分離が可能になります。

4.3. ルールの管理

Apicurio Registry のルールは、定義された標準に対してアーティファクトを検証して一貫性と品質を強制的に確保するために使用できます。レジストリー全体にグローバルルールを適用したり、個別のグループやアーティファクトに特定のルールを設定したりできます。

4.4. 設定

Apicurio Registry の設定オプションを使用して、特定のニーズに合わせてレジストリーを調整できます。運用パラメーターの調整、外部システムとの統合、または詳細設定を行ってパフォーマンスとセキュリティーを強化します。

4.5. API の使用

Apicurio Registry は、プログラムを使用してレジストリーと対話するための堅牢な REST API を提供します。開発者は、アーティファクト管理の自動化、レジストリー機能と CI/CD パイプラインの統合、カスタムツールのビルドを行い、スキーマガバナンスを強化できます。

付録A RED HAT サブスクリプションの使用

Red Hat Connectivity Link は、ソフトウェアサブスクリプションを通じて提供されます。サブスクリプションを管理するには、Red Hat カスタマーポータルでアカウントにアクセスします。

サブスクリプションの管理

1. access.redhat.com に移動します。
2. アカウントがない場合は作成します。
3. アカウントにログインします。
4. メニューバーで **Subscriptions** をクリックし、サブスクリプションを表示および管理します。

改訂日時: 2024-12-13