

# openSAP SAP Cloud Platform API Management Additional Download

PUBLIC





## TABLE OF CONTENTS

SYSTEM PREREQUISITES .....	3
CREATE BINDING TO PERSISTENCY SERVICE.....	4
EXTEND GENERATED CODE WITH PERSISTENCY SERVICE.....	4
TESTING YOUR API WITH PERSISTENCY SERVICE.....	9
EXTEND GENERATED CODE WITH CLOUD FOUNDRY ENVIRONMENTS.....	10
DEPLOY GENERATED TO YOUR CLOUD FOUNDRY ENVIRONMENT .....	12
FURTHER READS .....	12

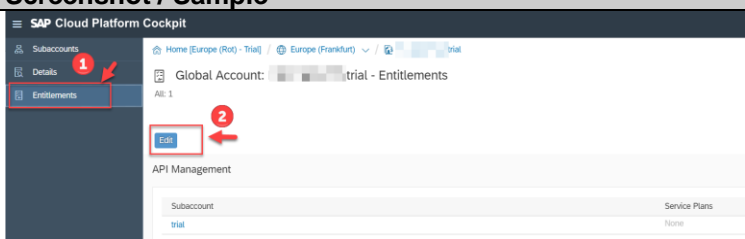
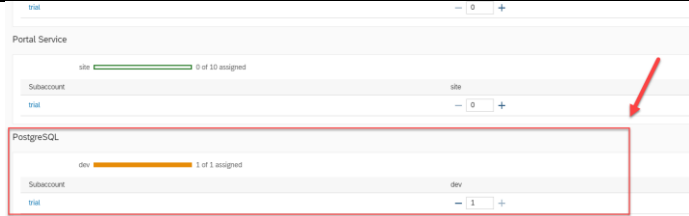
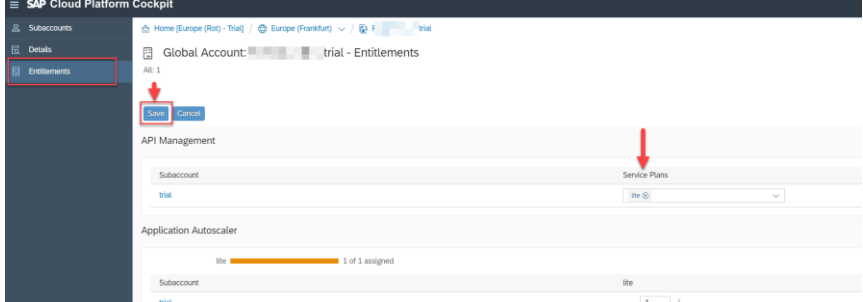
In this exercise, you'll explore the use of API first design and show how OpenAPI and the SAP API Designer can help develop your first micro service. The microservice you'll create would be for product catalog and product details like stock availability and price information.

## SYSTEM PREREQUISITES

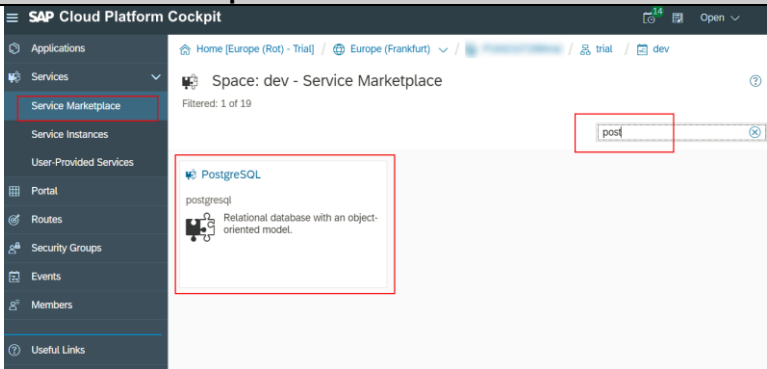
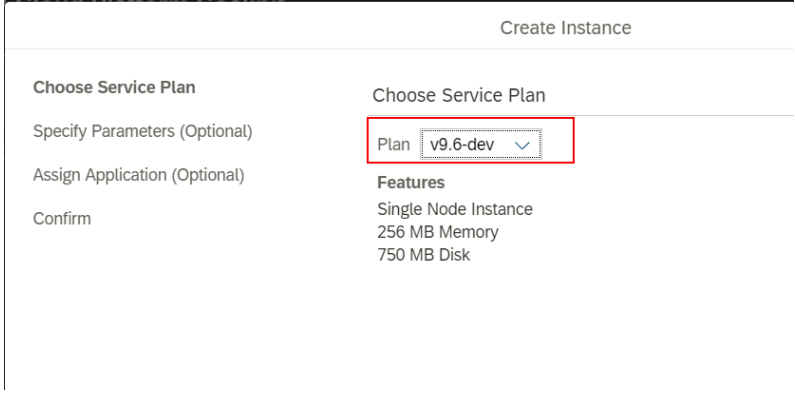
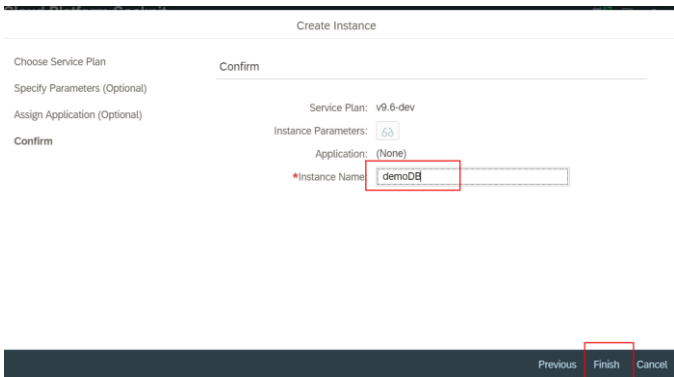
- Account on the [SAP Cloud Platform](#)
- Node JS version 6.12.3 or later installed ( <http://www.nodejs.org> )
- Node Package Manager (NPM) 3.10 or later installed (should be automatically installed when you install Node JS)
- Visual Studio Code ( <https://code.visualstudio.com/> ) or another code editor
- PostgreSQL version 10.4 or later ( <https://www.postgresql.org/download/> ) for local testing
- Cloud Foundry Command Line Interface version 6.34 or later ( <https://docs.cloudfoundry.org/cf-cli/install-go-cli.html> )

## ENABLING POSTGRES PERSISTENCE SERVICE ENTITLEMENT


If in your SAP Cloud Platform trial tenant, PostgreSQL service or SAP Cloud Platform API Management is not available in the Service Market Place, enable these services from the Entitlement tab.  
If the services are already available, then this section can be ignored.

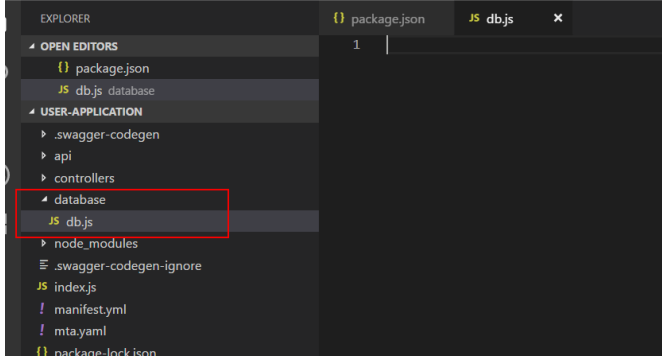
Steps	Screenshot / Sample
Logon to SAP Cloud Foundry account, from Global Account select the <b>Entitlements</b> tab and click <b>Edit</b> button	
Search for <b>PostgreSQL</b> and then add 1 for the dev plan	
Search for API Management and then select the lite plan from the drop. Click on <b>Save</b> to persist the <b>Entitlement</b> changes	

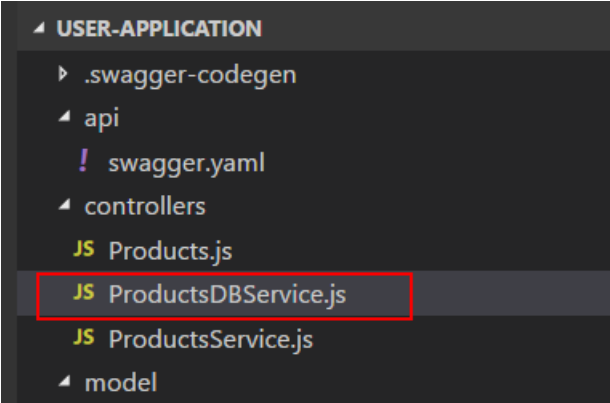

## CREATE BINDING TO PERSISTENCY SERVICE

Steps	Screenshot / Sample
Logon to SAP Cloud Foundry account, from Services -> Service Marketplace search for <b>PostgreSQL</b> persistency service	 <p>The screenshot shows the SAP Cloud Platform Cockpit interface. On the left, the 'Service Marketplace' tab is selected. The main area displays search results for 'PostgreSQL', showing a single result: 'PostgreSQL' with a description 'Relational database with an object-oriented model.' A search bar at the top right contains the text 'post'.</p>
From the Instances tab, click on New Instance and create a service instance of plan <b>v9.6-dev</b>	 <p>The screenshot shows the 'Create Instance' dialog for PostgreSQL. The 'Choose Service Plan' section is active, showing a dropdown menu with 'v9.6-dev' selected. The 'Features' section lists 'Single Node Instance', '256 MB Memory', and '750 MB Disk'.</p>
Enter the name of the MongoDB as <b>demoDB</b>	 <p>The screenshot shows the 'Create Instance' dialog for PostgreSQL, with the 'Confirm' section active. The 'Instance Name' field is highlighted with a red box and contains the text 'demoDB'. The 'Service Plan' is 'v9.6-dev' and the 'Application' is '(None)'.</p>

## EXTEND GENERATED CODE WITH PERSISTENCY SERVICE

Step	Screenshot/ Sample
Add dependency to pg library and cfenv library   package.json	<pre>{   "name": "products-catalog-api",   "version": "1.0.0",   "description": "Your first microservice using an API First approach",   "main": "index.js",</pre>

	<pre> "scripts": {   "prestart": "npm install",   "start": "node index.js" }, "keywords": [   "swagger" ], "license": "Unlicense", "private": true, "dependencies": {   "connect": "^3.2.0",   "js-yaml": "^3.3.0",   "swagger-tools": "0.10.1",   "sequelize": "~4.38.0",   "pg": "~7.4.3",   "pg-hstore": "~2.3.2",   "cfenv": "~1.1.0" } </pre>
<p>Create a folder named <b>database</b> and add a file named <b>db.js</b></p>	
<p>Open <b>db.js</b> and add in the following snippet to connect to the postgresql DB named <b>demoDB</b> and create the product schema as per the Open API specification file</p> <div data-bbox="218 1581 284 1644" data-label="Image"> </div> <p>db.js</p> <p><b>Note:-</b> For the local testing, the table named demoDB would have to be created on the local Postgres DB. In the connection string user:pass would have to be set to DB user created by you for your local DB</p>	<pre> 'use strict';  const Sequelize = require('sequelize'), appEnv = require("cfenv").getAppEnv();  let dbUrl = appEnv.getServiceURL('demoDB', {   pathname: "dbname",   auth: ["username", "password"] });  if (!dbUrl) {   dbUrl = 'postgres://user:pass@localhost:5432/demoDB'; }  const sequelize = new Sequelize(dbUrl); </pre>

	<pre> sequelize.authenticate()   .then(() =&gt; console.log('Connection has been established successfully.'))   .catch(err =&gt; console.error('Unable to connect to the database:', err));  const product = sequelize.define('product', {   id: { type: Sequelize.STRING, primaryKey: true },   name: Sequelize.STRING,   description : Sequelize.STRING,   price: Sequelize.JSON,   availability: Sequelize.JSON });  product.sync();  exports.product = product; </pre>
<p>Create a controller file(named <b>ProductsDBService.js</b>)for handling DB calls like reading of product catalogs, creating products, reading product details</p>	
<p>Add the snippet for DB handling in <b>ProductsDBService.js</b> file</p>  <p>ProductsDBService.js</p>	<pre> 'use strict'; var entityModel = require("../database/db.js").product;  function validate(res, validateFunc) {   const error = validateFunc();   if (error != null) {     setErrorResponse(res, 400, error.code, error.message);     return false;   }   return true; } </pre>

```

function validTopSkip(text, queryName) {
  let error = null;
  try {
    const value = parseInt(text);
    if (isNaN(value) || value < 0) {
      error = getError("INVALID_INPUT",
        "Incorrect format for " + queryName + " argument '" + text + "'. Provide a value positive integer value");
    }
  } catch (e) {
    error = getError("INVALID_INPUT",
      "Incorrect format for " + queryName + " argument '" + text + "'. Provide a value positive integer value");
  }
  return error
}

function getError(code, message) {
  return {
    code: code,
    message: message
  };
}

function setResponse(res, data, status) {
  res.setHeader('Content-Type',
    'application/json');
  res.statusCode = status || 200;
  res.end(JSON.stringify(data));
}

function setErrorResponse(res, status, code,
  message) {
  setResponse(res, getError(code, message),
    status);
}

exports.productsGET = (args, res, next) => {
  let skip = args.$skip;
  var query = {};

```

```

    if (skip && skip.value) {
      if (!validate(res,
        validTopSkip.bind(null, skip.value, "$skip"))) {
        return;
      }
      query.offset = parseInt(skip.value);
    }

    let top = args.$top;
    if (top && top.value) {
      if (!validate(res,
        validTopSkip.bind(null, top.value, "$top"))) {
        return;
      }
      query.limit = parseInt(top.value);
    }

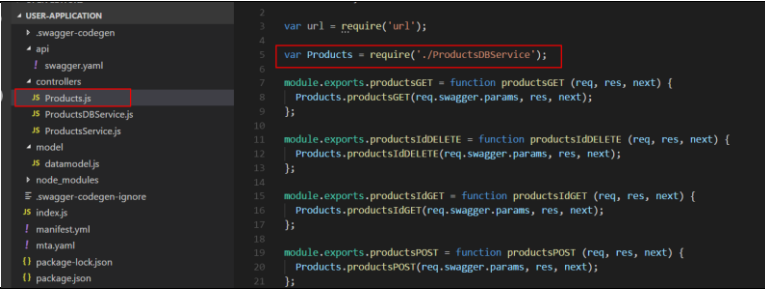
    entityModel.findAll(query)
      .then(entity => setResponse(res, entity, 200))
      .catch(error =>
        setErrorResponse(res, 500, "READ_FAILURE",
          error.message));
  }

  exports.productsPOST = (args, res, next) => {
    entityModel.create(args.payload.value)
      .then(entity => setResponse(res, entity, 201))
      .catch(error =>
        setErrorResponse(res, 500, "CREATE_FAILURE",
          error.message));
  }

  exports.productsIdGET = (args, res, next) => {
    entityModel.findById(args.id.value)
      .then(entity => {
        if(!entity){
          setErrorResponse(res, 404,
            "RESOURCE_NOT_FOUND", "Requested Product " + id +
            " not found.");
        } else {
          setResponse(res, entity, 200);
        }
      })
      .catch(error =>

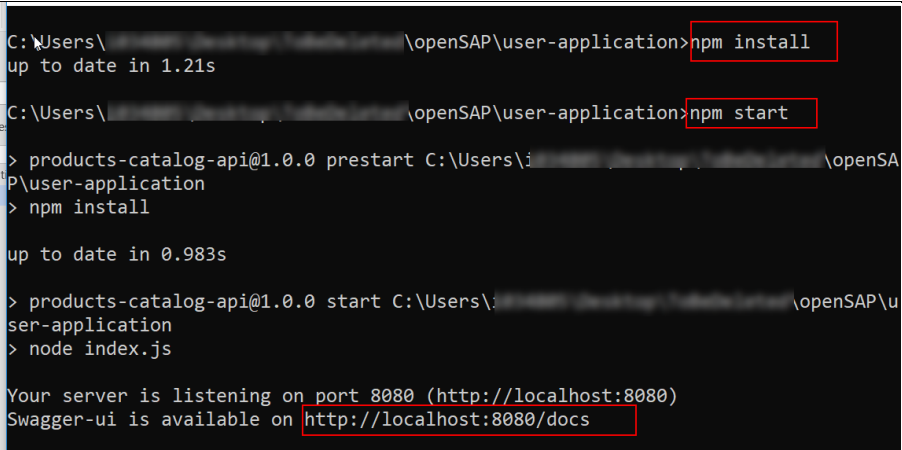
```



	<pre> setErrorResponse(res,500,"READ_FAILURE", error.message)); }  exports.productsIdDELETE = (args, res, next) =&gt; {   entityModel.findById(args.id.value)     .then(entity =&gt; {       if(!entity){         setErrorResponse(res, 404, "RESOURCE_NOT_FOUND", "Requested Product " + id + " not found.");       } else {         entity.destroy()           .then(() =&gt; {             res.statusCode = 204;             res.end();           })           .catch(error =&gt; setResponse(res,500,"DELETE_FAILURE", error.message));       }     })     .catch(error =&gt; setErrorResponse(res,500,"DELETE_FAILURE", error.message)); } </pre>
<p>Connect the generated productservice interface to the controller JS</p>	
<p>Snippet for connect Product.js to the newly created ProductsDBService.js controller</p>	<pre>var Products = require('./ProductsDBService');</pre>

## TESTING YOUR API WITH PERSISTENCY SERVICE

Step	Screenshot/ Sample
------	--------------------

<p>Open Command Line tool and then navigate to root folder of the downloaded project.</p> <p>Use command <code>npm install</code> to install all the dependent libraries</p> <p>Use command <code>npm start</code> to run the application</p>	 <pre> C:\Users\... \openSAP\user-application&gt;npm install up to date in 1.21s  C:\Users\... \openSAP\user-application&gt;npm start  &gt; products-catalog-api@1.0.0 prestart C:\Users\... \openSAP\user-application &gt; npm install up to date in 0.983s  &gt; products-catalog-api@1.0.0 start C:\Users\... \openSAP\user-application &gt; node index.js  Your server is listening on port 8080 (http://localhost:8080) Swagger-ui is available on http://localhost:8080/docs </pre>
Run the application	<code>http://localhost:8080/docs</code>

## EXTEND GENERATED CODE WITH CLOUD FOUNDRY ENVIRONMENTS

Step	Screenshot/ Sample
Open index.js from the generated code and add in the following snippet to read the server port from CF application environment	<pre> var appEnv = require("cfenv").getAppEnv(); var serverPort = appEnv.port    8080; </pre>
Add the following snippet to dynamically updated the generated swagger.yaml file with CF application host name and port	<pre> spec = spec.replace("localhost:8080", appEnv.url.split("://")[1]); spec = spec.replace("http", appEnv.url.split("://")[0]); </pre>
Updated index.js file with changes highlighted	 <pre> serverPort = appEnv.port    8080;  // The Swagger document (require it, build it programmatically, fetch it from a URL, ...) var spec = fs.readFileSync(path.join(__dirname, 'api/swagger.yaml'), 'utf8'); spec = spec.replace("localhost:8080", appEnv.url.split("://")[1]); spec = spec.replace("http", appEnv.url.split("://")[0]); var swaggerDoc = jsyaml.safeLoad(spec); </pre>
Full index.js server for is added here for reference	<pre> 'use strict';  var fs = require('fs'),     path = require('path'),     http = require('http'); </pre>



index.js

```
var app = require('connect')();
var swaggerTools = require('swagger-tools');
var jsyaml = require('js-yaml');
var appEnv = require("cfenv").getAppEnv();
var serverPort = appEnv.port || 8080;

// swaggerRouter configuration
var options = {
  swaggerUi: path.join(__dirname, '/swagger.json'),
  controllers: path.join(__dirname, './controllers'),
  useStubs: process.env.NODE_ENV === 'development' //
  Conditionally turn on stubs (mock mode)
};

// The Swagger document (require it, build it
programmatically, fetch it from a URL, ...)
var spec =
fs.readFileSync(path.join(__dirname, 'api/swagger.yaml'),
'utf8');
spec = spec.replace("localhost:8080",
appEnv.url.split("://")[1]);
spec = spec.replace("http", appEnv.url.split("://")[0]);
var swaggerDoc = jsyaml.safeLoad(spec);

// Initialize the Swagger middleware
swaggerTools.initializeMiddleware(swaggerDoc, function
(middleware) {

  // Interpret Swagger resources and attach metadata to
  request - must be first in swagger-tools middleware
  chain
  app.use(middleware.swaggerMetadata());

  // Validate Swagger requests
  app.use(middleware.swaggerValidator());

  // Route validated requests to appropriate controller
  app.use(middleware.swaggerRouter(options));

  // Serve the Swagger documents and Swagger UI
  app.use(middleware.swaggerUi());

  // Start the server
  http.createServer(app).listen(serverPort, function ()
{
```

	<pre>         console.log('Your server is listening on port %d (http://localhost:%d)', serverPort, serverPort);         console.log('Swagger-ui is available on http://localhost:%d/docs', serverPort);     });  }); </pre>
--	---

## DEPLOY GENERATED TO YOUR CLOUD FOUNDRY ENVIRONMENT

Step	Screenshot/ Sample
Open manifest.yml file from the generated code provide a unique name to your application say productcatalogs. Add in the services references to your persistency service	<pre> --- applications: - name: productcatalog   memory: 512M   buildpack: nodejs_buildpack services: - demoDB </pre>
Open Command Line tool and then navigate to root folder of the downloaded project.	
Set your Cloud Foundry Environment API endpoint	cf api https://api.cf.eu10.hana.ondemand.com
Login to your Cloud Foundry space and enter your Cloud Foundry user credentials	cf login
Deploy the application to Cloud Foundry tenant	cf push

## FURTHER READS

- [SAP Guidelines for REST API Harmonization](#)
- [OpenAPI Specification 2.0](#)
- Accessing [API Designer](#) from SAP Cloud Platform Trial Account
- Documenting APIs with [OpenAPI Specification](#) in API Designer
- [Video Tutorial](#) for Creating and Documenting APIs in API Designer
- [Blog](#) on Develop and manage API first enterprise microservices with SAP Cloud Platform and API Management



## Coding Samples

Any software coding or code lines/strings ("Code") provided in this documentation are only examples and are not intended for use in a production system environment. The Code is only intended to better explain and visualize the syntax and phrasing rules for certain SAP coding. SAP does not warrant the correctness or completeness of the Code provided herein and SAP shall not be liable for errors or damages caused by use of the Code, except where such damages were caused by SAP with intent or with gross negligence.

**[www.sap.com/contactsap](http://www.sap.com/contactsap)**

© 2018 SAP SE or an SAP affiliate company. All rights reserved.

No part of this publication may be reproduced or transmitted in any form or for any purpose without the express permission of SAP SE or an SAP affiliate company.

The information contained herein may be changed without prior notice. Some software products marketed by SAP SE and its distributors contain proprietary software components of other software vendors. National product specifications may vary.

These materials are provided by SAP SE or an SAP affiliate company for informational purposes only, without representation or warranty of any kind, and SAP or its affiliated companies shall not be liable for errors or omissions with respect to the materials. The only warranties for SAP or SAP affiliate company products and services are those that are set forth in the express warranty statements accompanying such products and services, if any. Nothing herein should be construed as constituting an additional warranty.

In particular, SAP SE or its affiliated companies have no obligation to pursue any course of business outlined in this document or any related presentation, or to develop or release any functionality mentioned therein. This document, or any related presentation, and SAP SE's or its affiliated companies' strategy and possible future developments, products, and/or platform directions and functionality are all subject to change and may be changed by SAP SE or its affiliated companies at any time for any reason without notice. The information in this document is not a commitment, promise, or legal obligation to deliver any material, code, or functionality. All forward-looking statements are subject to various risks and uncertainties that could cause actual results to differ materially from expectations. Readers are cautioned not to place undue reliance on these forward-looking statements, and they should not be relied upon in making purchasing decisions.

SAP and other SAP products and services mentioned herein as well as their respective logos are trademarks or registered trademarks of SAP SE (or an SAP affiliate company) in Germany and other countries. All other product and service names mentioned are the trademarks of their respective companies. See <http://www.sap.com/corporate-en/legal/copyright/index.epx> for additional trademark information and notices.

**THE BEST RUN**