The Compatible One Application and Platform Service[1] (COAPS) API

User Guide

Using the COAPS API (v1.5.3) to provision and manage applications on Cloud Foundry

Telecom SudParis, Computer Science Department

[1] COAPs is proposed to replace *-PaaS.

Contributors:

Mohamed Sellami, Telecom SudParis.
Sami Yangui, Telecom SudParis.
Mohamed Mohamed, Telecom SudParis.
Samir Tata, Telecom SudParis.

Contents

Introduction

The COAPS API is a generic REST API that allows a seamless interaction with different and heterogeneous PaaS. In this aim, it exposes a generic interface that can be implemented according to the different actions exposed by a PaaS (e.g. Cloud Foundry, Openshift, etc.). In this user guide, we present a COAPS API implementation to interact with a Cloud Foundry PaaS. The different steps to build the *CloudFoundry-COAPS API*, to install it and to test it are detailed below.

Building the Cloud Foundry-COAPS API

1. First, use GIT to locally clone the project's code from: git://gitorious.ow2.org/ow2-compatibleone/coaps.git. The cloned repository contains:
    a. **`coaps/spec`**: the COAPS API specifications.
    b. **`coaps/api`**: the java interfaces for the COAPS API. The current api ensures application and environment management operations.
    c. **`coaps/core`**: this folder contains two subfolders:
        i. **`CloudFoundry-api`**: the CloudFoundry-COAPS implementation.
        ii. **`OpenShift-api`**: the OpenShift-COAPS implementation.
    d. **`coaps/client`**: a generic Web-based client to test the different implementations.
    e. **`coaps/test-resources`**: the application description (i.e manifest) and a Java Web application that can be deployed on Cloud Foundry or Openshift.
2. Build the project using maven (version 3.0.4). Go to the cloned repository's root directory (i.e. coaps) and run ''`mvn install`'':

```
C:\Users\sellami\Desktop>cd coaps

C:\Users\sellami\Desktop\coaps>mvn install
```

This action will build, among others, a Web-based implementation of the CloudFoundry-COAPS API (`CF-api.war` in the `coaps/core/CloudFoundry-api/target` folder).

Deploying the Cloud Foundry-COAPS API

After the maven build, a Web application (i.e. war file) is generated in the `CloudFoundry-api` sub-folder. This application has to be deployed on a Web server and will act as a REST server which allows interacting with a CloudFoundry PaaS.

1. To deploy the application using the apache Tomcat server, the version 7 using a java 1.6 runtime is required. Deploy the Web implementation of the `CoudFoundry-api` by copying the `CF-api.war` from `GenericAPI/core/CloudFoundry-api/target` to the tomcat's `webapps` folder.

```
C:\Users\sellami\Desktop\coaps>copy core\CloudFoundry-api\target\CF-api.war "c:\
Program Files\Apache Software Foundation\Tomcat 7.0\webapps"
```

2. You can test the deployment of the application by typing http://{hostname}:{portNumber}/CF-api/ in your Web browser. You should get a message indicating that your CloudFoundry implementation is correctly running.

3. By default, the CloudFoundry-COAPS API will interact with the online VMware's Cloud Application Platform (i.e. http://api.cloudfoundry.com/) using a test account. You can sign up for a free Cloud Foundry account on this page: https://my.cloudfoundry.com/signup. The connection parameters can be updated by changing the default parameters in the credentials.properties file:
    a. To modify this file, first stop the running tomcat instance and open the CF-api.war file with your favorite compression tool.
    b. Go to `WEB-INF/classes/telecom/sudparis/eu/copaas/core/server/resources/credentials.properties`.
    c. Edit the file to specify your credentials. For example:

```
vcap.target=http://api.cloudfoundry.com
vcap.email=login
vcap.passwd=password
host=api.cloudfoundry.com
api.public.url=http://{hostname}:{portNumber}/CF-api/rest/
```

    d. Save the file and restart tomcat.
    Note: *if the old credentials persist after a change, try to delete the extracted CF-api folder and any related temporally folders.*

Interacting with the Cloud Foundry-COAPS API

To invoke the API actions, one can use any REST client (eg. CURL) or our Web-based client (generated in the `client` sub-folder).

Using CURL

The list of available operations, their type (i.e. GET, POST, etc.) and invocation path and parameters can be viewed at: http://{hostname}:{portNumber}/CF-api/rest/application.wadl. An example of the provided WADL resources and operations description is given in Figure 1.

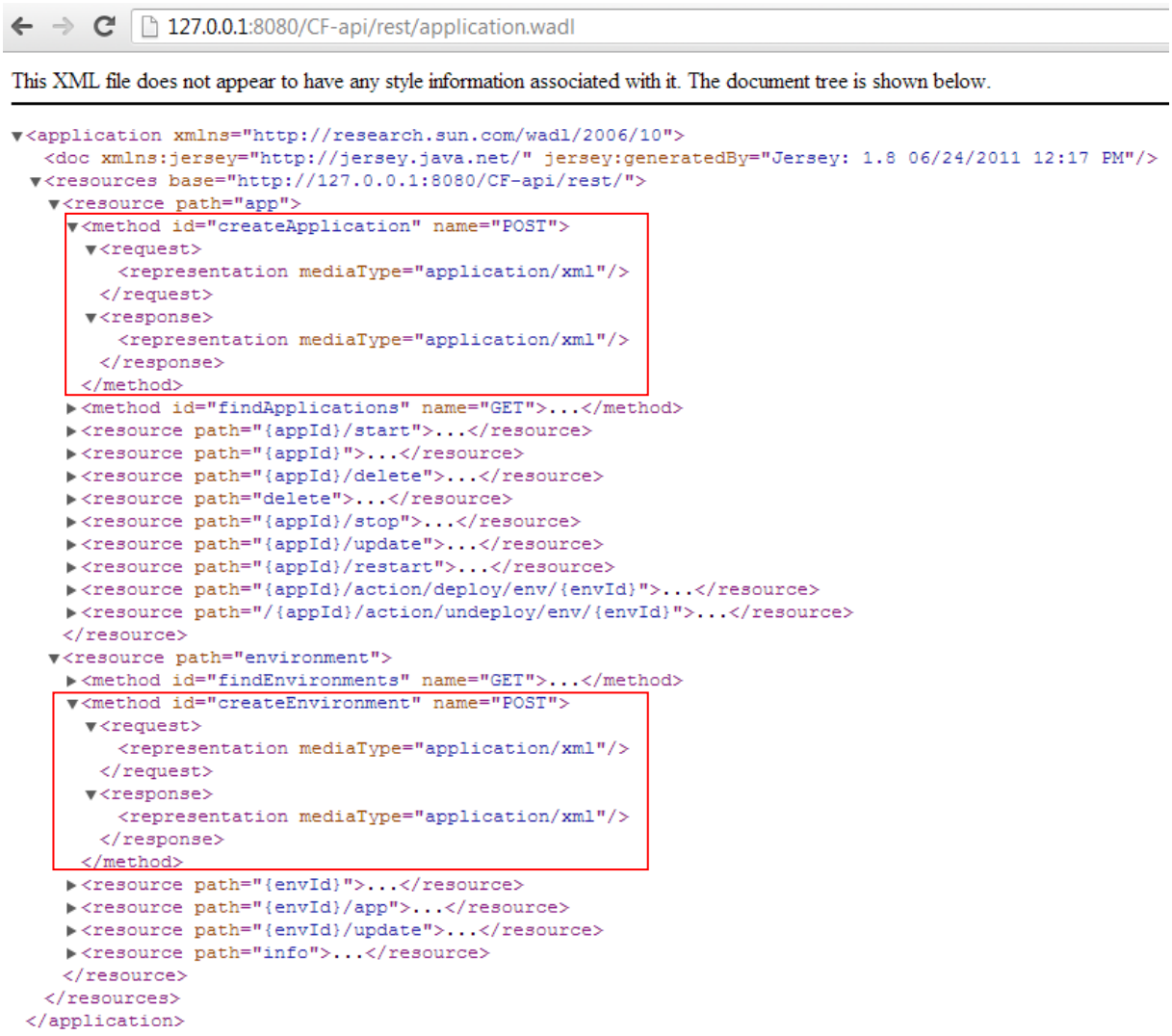


Figure 1 the WADL description of the REST resources and operations offered by the Cloud Foundry implementation

For example:

- to list all available applications on a Cloud Foundry instance we use:

```
C:\>curl -X GET http://localhost:8080/CF-api/rest/app
```

- To create an environment, we have to provide its description file in XML (i.e. its manifest) through a POST:

```
C:\>curl -X POST -d @c:\EnvironmentManifest.xml -H "Content-Type: application/xml" http://localhost:8080/CF-api/rest/environment
```

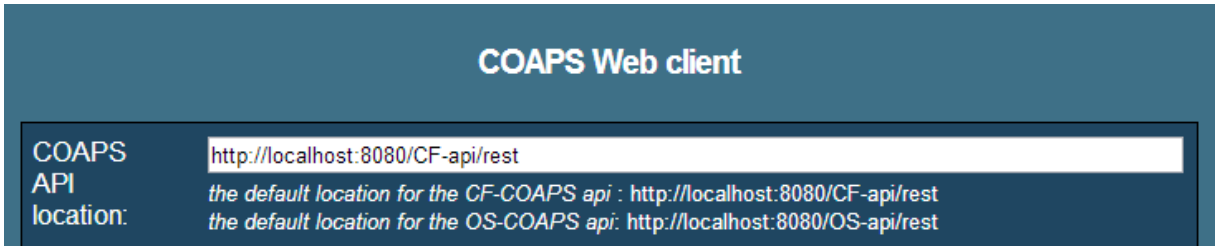
An example of a manifest is provided in `coaps/test-resources`.

Note: *If you use this manifest, do not forget to remove the license block in the beginning*.

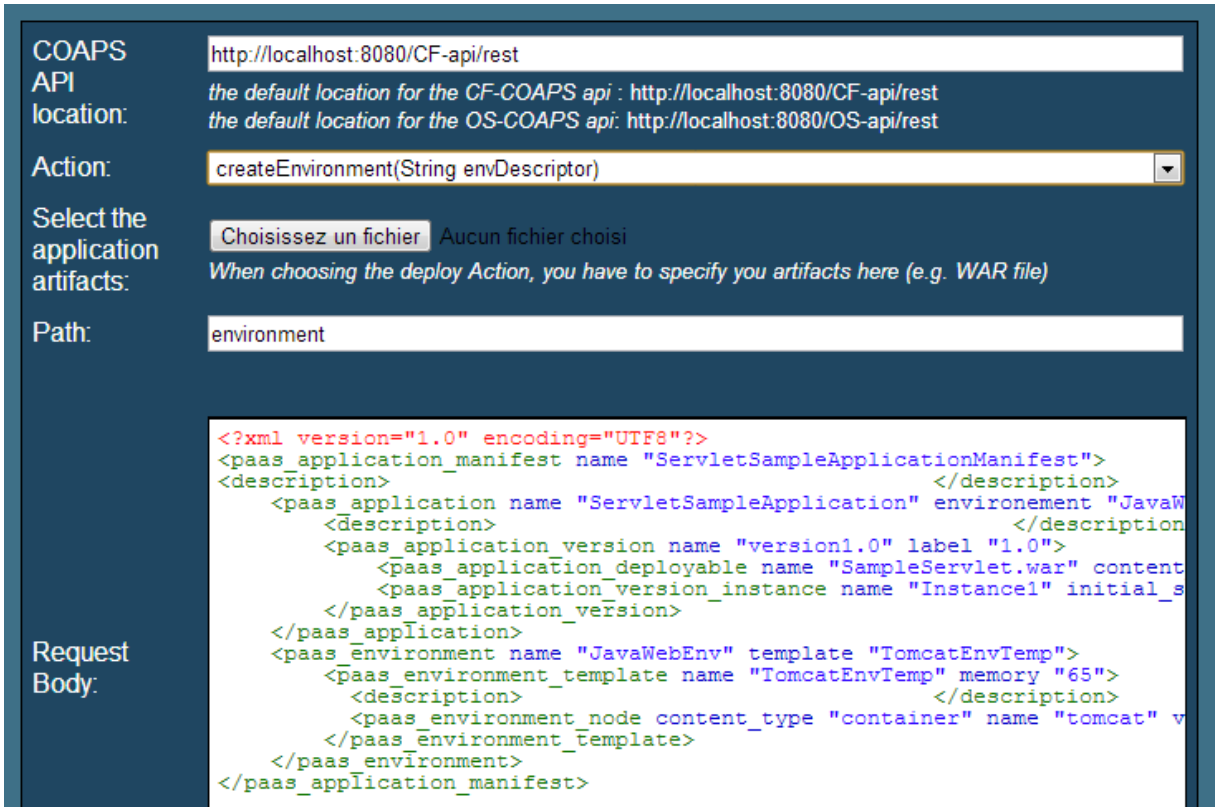
Using the Web-based client

The web archive (client.war) of the client is available at `coaps/client/target/client.war`. To deploy the client, copy the `client.war` to the tomcat's `webapps` folder. The client will be available at this address: http://{hostname}:{portNumber}/client/. In the following, we describe an application provisioning use case on Cloud Foundry:

1. Launch the client http://{hostname}:{portNumber}/client/ from a Web browser and specify the location of your CloudFoundry-COAPS API implementation[2]

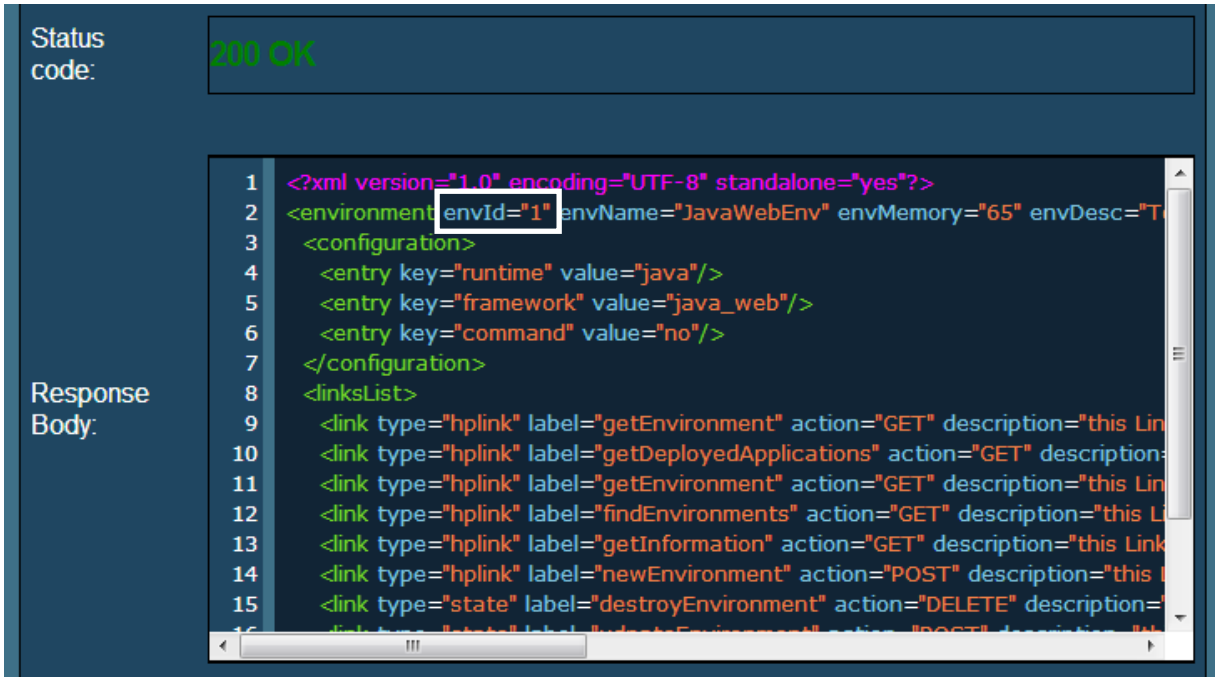


2. **Create an environment**: select the "create environment" action from the action list and provide the manifest describing, among others, the environment needed by the application to deploy (runtimes, frameworks, etc.). In the body of the request just paste the content of `coaps/test-resources/manifest.xml` (without the license comment).



After submitting this request (i.e. click on the submit button), an XML description of the created environment is returned. From this descriptor, you have to ''save'' the returned "envid" that must be provided later to link this environment to the application to deploy (deployApplication action).

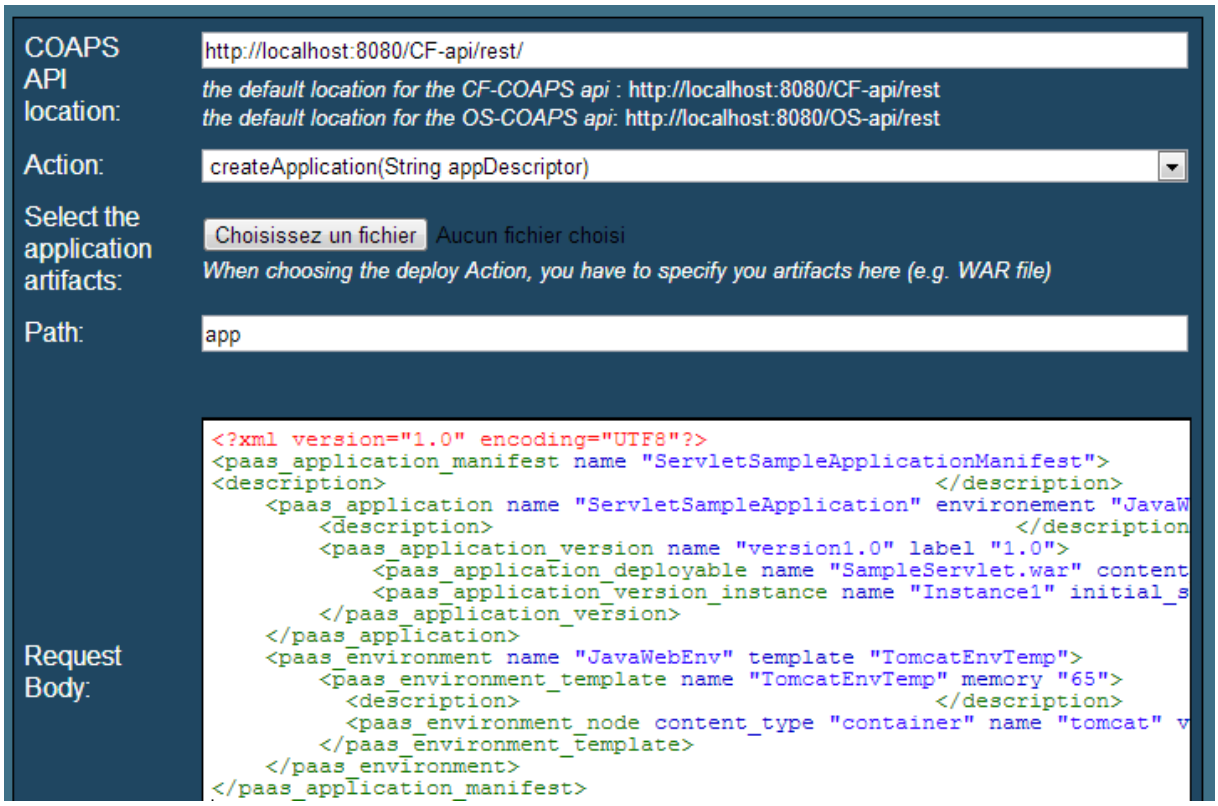
[2] Currently we provide only the Cloud Foundry and Openshift implementations. To test the Openshift-PaaS API implementation, deploy the associated war (from `coaps/core/OpenShift-api/target`) **on tomcat.**



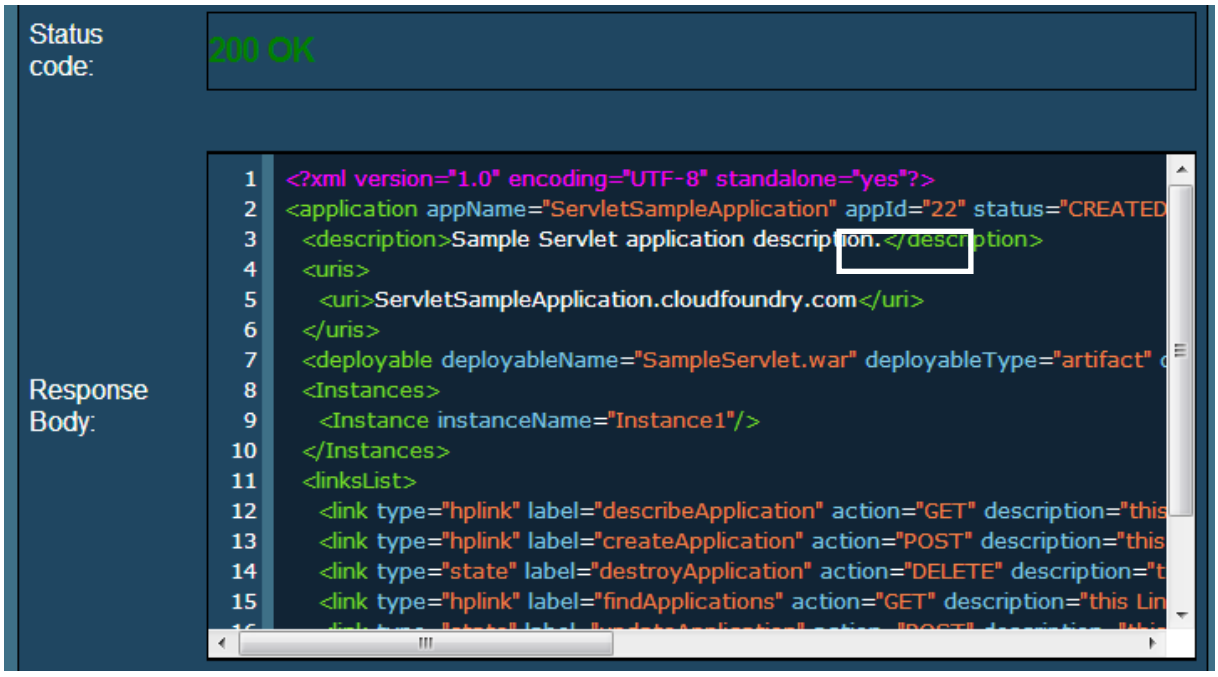
Note: If additional nodes are needed, a database for example, the Environment manifest must be updated accordingly. To add a mysql database in the environment for example, add the following element after the existing < paas_environment_node > element.

< paas_environment_node content_type="database" name="mysql" version="2.2" provider="CF"/>

3. **Create the application**: select the "create application" action from the action list and provide the manifest describing the application. In the body of the request just paste the content of `coaps/test-resources/manifest.xml` (without the license comment).



In the same way as the created environment, an "appid" will be provided by the API.

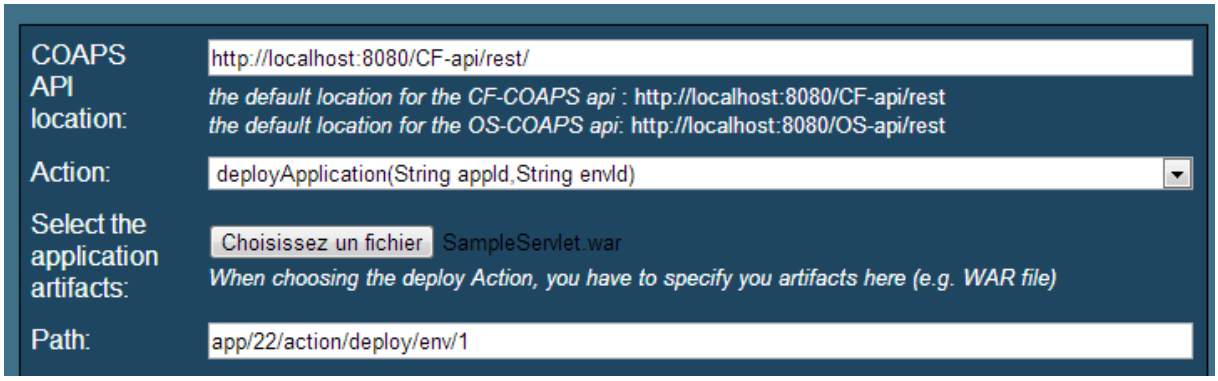


Note 1: If needed, you have to modify the manifest to update the name of the application and to specify the path of the application to deploy in the <paas_application_deployable> element.

Note 2: If additional instances are needed, the manifest must be updated accordingly. To declare an additional instance, add the following element after the existing <paas_application_version_instance> element.

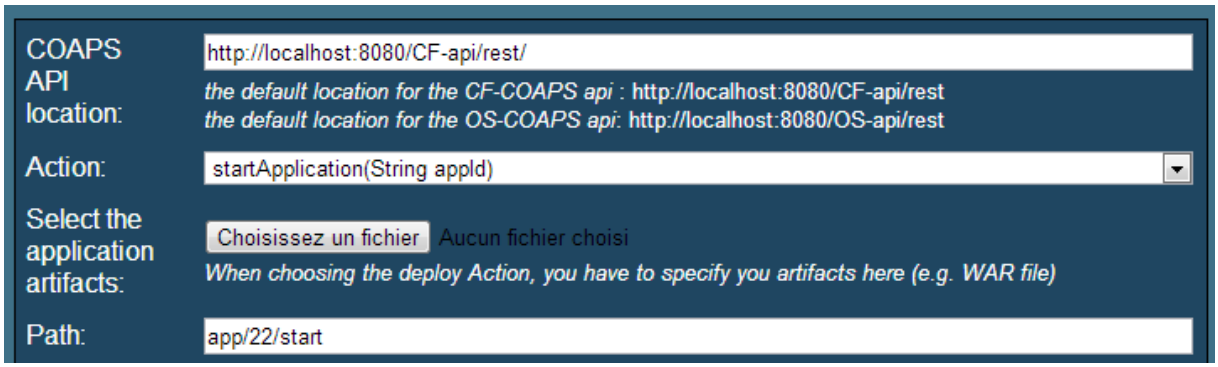
```
<paas_application_version_instance name="Instance2" initial_state="1" default_instance="false"/>
```

4. **Deploy the application** select the "deploy application" action from the action list while specifying the environment identifier "envid" and the application identifier "appid" in the path. You also have to join the application artifacts to deploy. In this example we are using the SampleServlet.war application.

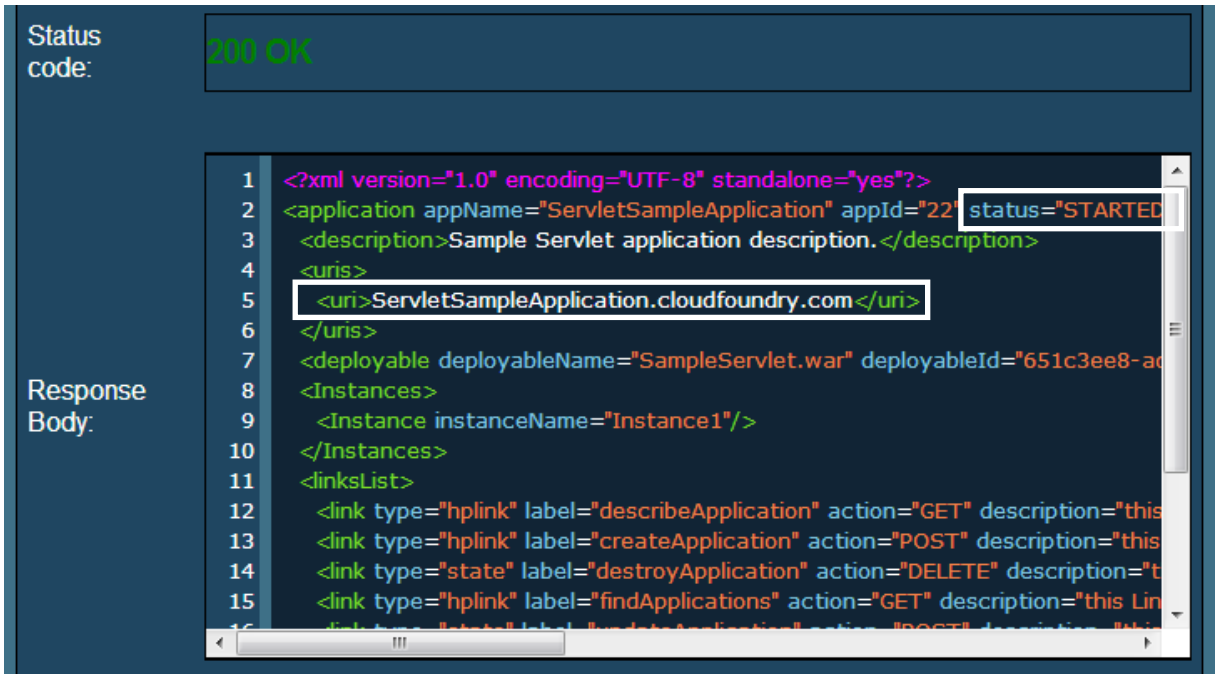


This action will upload and deploy the specified application on the Cloud Foundry PaaS. By default, the state of the application is set to “STOPPED”. As response, an XML descriptor of the deployed application is returned.

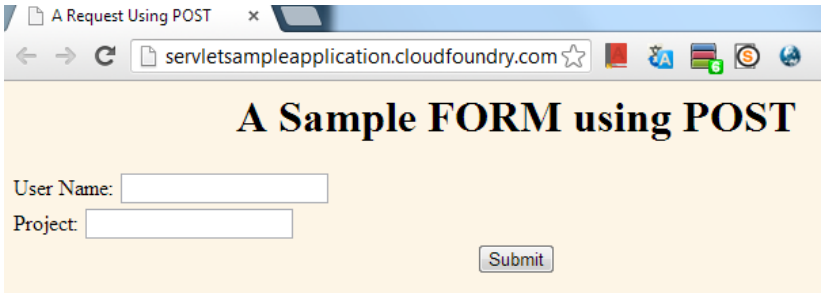
5. **Start the application**: select the "start application" action from the action list and provide the application identifier "appid" in the path.



The application is now started and its access URL http://ServletSampleApplication.cloudfoundry.com is available in the returned application description.



The application is now accessible via a browser on the returned URL.



Note: The provided scenario is a basic example that shows how to deploy a java Web application on a specific environment (java_web) and start it on one instance. However, for particular needs, more constraints can be described on the manifests (i.e. jredis environment, mysql database, more instances, etc.).

Note': Additional environment and application operations are also offered by the CloudFoundry-COAPS API implementation. For example: stop application, describe/destroy application/environment, etc.