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

Version 1.5.2

Telecom SudParis, Computer Science Department

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

Table of contents

1 Introduction ... 3

2 Overview on the COAPS API ... 3

3 The Environment Resource ... 4

3.1 Environment manifest ... 4

3.2 Environment description ... 4

3.3 Environment management methods ... 6

3.3.1 Creating Environment ... 6
3.3.2 Updating Environment ... 6
3.3.3 Destroying Environment ... 7
3.3.4 Finding Environments ... 7
3.3.5 Describing Environment ... 7
3.3.6 Getting Information ... 7
3.3.7 Getting Deployed Applications ... 8

3.4 Environment representation ... 8

3.4.1 Representation of an Environment: ... 8
3.4.2 Representation of a list of Environments: ... 9

4 The Application Manager Resource ... 9

4.1 Application manifest ... 9

4.2 Application description ... 11

4.3 Application management methods ... 12

4.3.1 Creating Application ... 12
4.3.2 Updating Application ... 13
4.3.3 Finding Applications ... 13
4.3.4 Starting Application ... 14
4.3.5 Stopping Application ... 14
4.3.6 Restarting Application ... 14
4.3.7 Describing Application ... 14
4.3.8 Destroying Application ... 15
4.3.9 Destroying Applications ... 15
4.3.10 Deploying Application ... 15
4.3.11 Undeploying Application ... 16

4.4 Application representation ... 16

4.4.1 Representation of an Application ... 16
4.4.2 Representation of a list of Application ... 17

5 Common errors ... 17

6 Link Elements ... 18

7 Annex A: Application and Environment Management Operations ... 19

1 Introduction

This document provides a description of the COAPS API based on REST/XML. This API is designed to provide an abstraction layer and a middleware for existing PaaS solutions to manage applications and environments in a generic fashion (Figure 1). To define a new connection between a novel PaaS and developer /application, one has simply to add its specific implementation.

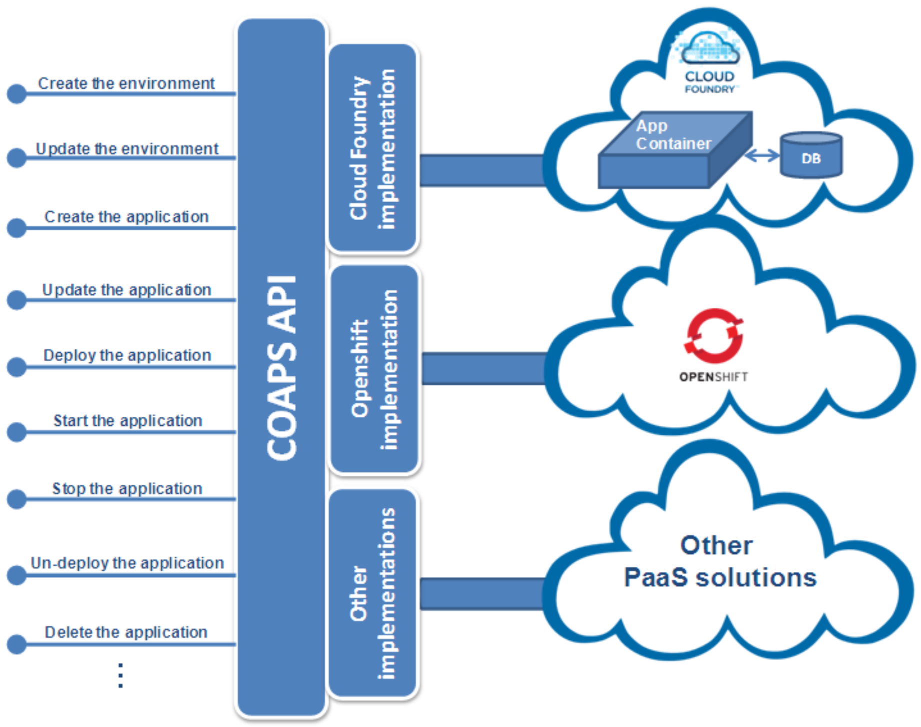

Figure 1 : Overview

Two COAPS API implementations are available (V 0.1): a Cloud Foundry implementation (CF-PaaS) and an OpenShift implementation (OS-PaaS). Both implementations are available at: http://gitorious.ow2.org/ow2-compatibleone/coaps/ and a user guide for CF-PaaS is available at: http://www-inf.it-sudparis.eu/~sellami/starPaaS/PaaSAPI-UserGuide.pdf.[2]

2 Overview on the COAPS API

There are two types of resources in COAPS API: *environment* and *application*. Environment represents a set of ''settings'' needed to host and run an application in this PaaS: *i.e.* the needed runtime (java 7, java 6, ruby, etc.), the needed frameworks/containers (spring, tomcat, ruby, etc.) and eventually needed services (databases, messaging, etc.). Application infers any computer software or program that can be deployed over a PaaS. Application source archives should be provided by the developer in a bundled format (*i.e.* war, ear, zip, etc.) or extracted format (*i.e.* a local folder with the different files and dependencies, distant URL, etc.).

To deploy an application and run it through COAPS API, one should follow the basic usage scenario illustrated in Figure 2:

[2] Both implementations and the user guide use an old version of the specification and some inconsistencies with the current specification can be found.

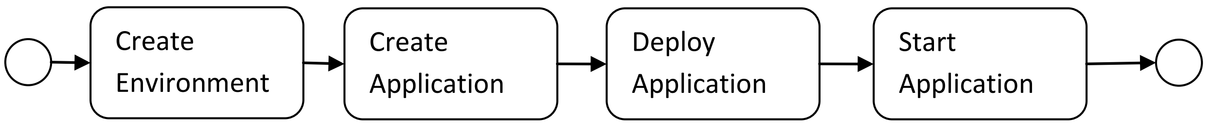

Figure 2 an application deployment scenario

3 The Environment Resource

In this section, we introduce the manifest description of environment resource, its management methods and presentations. We start this section by providing examples of an environment manifest (required as input by some of the REST methods of the environment manager resource) and of an environment description (returned as response by some of the REST methods of the environment manager resource).

3.1 Environment manifest

The *createEnvironment* and *updateEnvironment* operations require as input an environment manifest. This manifest allows developers to specify the different characteristics of the application's environment using an environment template (see Figure 3). Each environment template is composed by a set of PaaS resource nodes and PaaS relations to link these nodes. PaaS nodes can be *container* type, *database* type or *router* type while relations between them can be a binding between a container node and a database node or between two containers node through a router node for example.

> ***Note***: *the environment manifest used in this document is given as an example. While implementing our API you can specify your own manifests.*

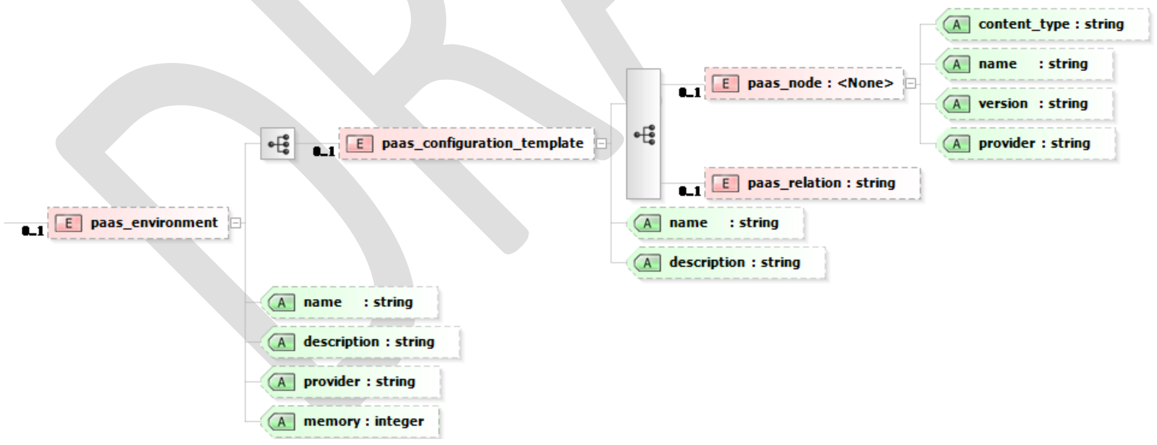

Figure 3 schema of a possible environment manifest

3.2 Environment description

Some of the environment management operations return as response an XML environment descriptor. The XML format of this descriptor is specific for the COAPS API implementation. In the following, we provide as an example the XML schema for the environment descriptor specific to a CloudFoundry implementation of the COAPS API (see Figure 4).

> ***Note***: *the environment description used in this document is given as an example. While implementing our API you can specify your proper environment description.*

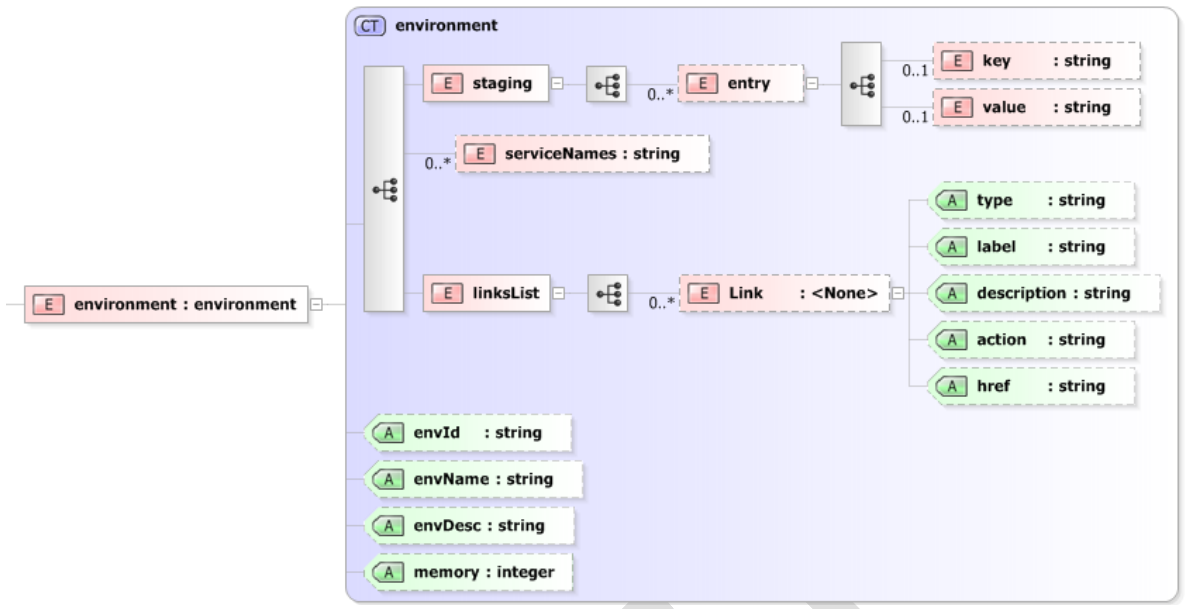

Figure 4 XML schema of a possible environment description

In Table 1, we provide the semantics of the different elements in the environment descriptor presented in Figure 4 and provide the corresponding element in the environment manifest presented in Figure 3.

Element of the environment descriptor	**Description**	**Corresponding element in the environment manifest**
E staging	This element describes the frameworks, environments and eventual commands offered by the environment.	The <paas_node> elements with *container* as content_type value.
E serviceNames	This element defines the services (database, messaging pool …) associated to the environment.	The <paas_node> elements with *database* as content_type for persistent values, *container* for hosting applications or *router* for formatted messages between Paas nodes
E linksList	The set of links associated to the environment. They are hyperlinks to either different states of the associated resources or different resources (see Section 6).	--
A envId	An automatically generated identifier for the environment.	--
A envName	The environment's name.	A name defined in <paas_environment>
A envDesc	An optional textual description of the environment.	A description defined in <paas_environment>
A memory	The physical memory that is allocated to the application expressed in Megabytes.	A memory defined in <paas_environment>

Table 1 elements and attributes of the environment description

3.3 Environment management methods

In this section, we present methods to manage an environment resource using REST architecture. They includes: creating Environment, updating Environment, destroying Environment, finding Environments, getting Environment description, getting Environment information, and getting applications deployed on an Environment.

3.3.1 Creating Environment

This method creates a new environment using an environment descriptor. An environment specifies the needed frameworks, runtimes containers and/or services required by a given application.

Resource identifier	/environment
HTTP method	POST
Input parameter	An XML environment manifest (in the body of the request)
Response	An XML environment descriptor. This descriptor contains, among other information, the created environment's ID.
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

Example of HTTP request and response:

<u>Request</u>

```
POST /CF-api/environment HTTP/1.1[3]
Host : hostname :port[4]
Accept : text/xml
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Environment manifest comes here/>
```

<u>Response</u>

```
HTTP/1.1 200 OK
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Representation of the new environment comes here/>
```

3.3.2 Updating Environment

This method updates an existing environment. The environment ID must be provided (i.e. envId) and the updates has to be specified in the input parameter (i.e. as an environment Manifest)

Resource identifier	/environment/{envId}/update
HTTP method	POST
Input parameter	An XML environment manifest (in the body of the request)
Response	The new XML environment descriptor.
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

[3] Suppose that CF-api is the application path.
[4] Hostname and port indicates the address of the server.

Example of HTTP request and response:

```
Request

POST /CF-api/environment/1/update HTTP/1.1
Host : hostname :port
Accept : text/xml
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
< Manifest of Environment 1 comes here />

Response

HTTP/1.1 200 OK
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Updated representation of Environment 1 comes here/>
```

3.3.3 Destroying Environment

This method destroys an environment given its ID.

Resource identifier	/environment/{envId}
HTTP method	DELETE
Input parameter	--
Response	The destroy discharge
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

3.3.4 Finding Environments

This method lists the available environments.

Resource identifier	/environment
HTTP method	GET
Input parameter	--
Response	A list of XML environment descriptors of the existing environments
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

3.3.5 Describing Environment

This method returns the XML environment description of an environment given its ID.

Resource identifier	/environment/{envId}
HTTP method	GET
Input parameter	--
Response	The XML environment descriptor
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

3.3.6 Getting Information

This method lists the runtimes, frameworks and services supported by the PaaS.

Resource identifier	/environment/info
HTTP method	GET
Input parameter	--
Response	The list of supported runtimes, frameworks and services
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

3.3.7 Getting Deployed Applications

This method lists the deployed application in an environment given its ID

Resource identifier	/environment/{envId}/app
HTTP method	GET
Input parameter	--
Response	A list of XML application descriptors
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

3.4 Environment representation

3.4.1 Representation of an Environment:

The representation of an Environment is presented in XML. It contains information about the Environment, actions to change its state and links to other resources, including related Environments or applications that are deployed on it.

```
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<environment envId="ID" envName="name" envDesc="description">
 <staging>
            <entry key="key" value ="value">
            …
            <entry key="key" value ="value">
  </staging>
 <serviceNames>name of the service</serviceNames>

<linksList>
            < link type="state" label="destroyEnvironment()"  action="DELETE"
                        description= " destroy Env."  href="http://hostname:port/CF-api/environment/ID"/>
            < link type="state" label="updateEnvironment()"  action="POST"
                        description= " Environment's manifest "
                        href="http://hostname:port/CF-api/environment/ID/update"/>
            …

            <link type="hplink" label="newEnvironment()"  action="POST"
                        description= "new Env."  href="http://hostname:port/CF-api/environment "/>
            <link type="hplink" label="findEnvironments()"  action="GET"
                        description= "get Envs." href="http://hostname:port/CF-api/environment "/>
            <link type="hplink" label="getInformation()"  action="GET"
                        description= " get Env. Info."  href="http://hostname:port/CF-api/environment/info "/>
            <link type="hplink" label="getDeployedApplications()"  action="GET"
                        description= "get deployed apps. "
                        href="http://hostname:port/CF-api/environment/ID/app"/>
            …
 </linksList>
</environment>
```

3.4.2 Representation of a list of Environments:

```
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<environments>
            <environment>
                        <id>envID</id>
                        <name>envName</name>
                        <description>envDesc</description>
                        <uri> http://hostname:port/CF-api/environment /envID</uri>
            </environment>
            …
            <environment>
                        …
            </environment>
</environments>
```

4 The Application Manager Resource

In this section, we introduce the application manager resource, its different child resources and their associated methods. We start this section by providing examples of an application manifest (required as input by some of the REST methods of the application manager resource) and of an application description (returned as response by some of the REST methods of the application manager resource).

4.1 Application manifest

createApplication and *updateApplication* operations requires as input an application manifest. This manifest allows developer providing information needed by the PaaS to manage its deployment and execution. It allows, among others, specifying the application name, its different versions with specific properties of each of them and a set of operational instances. It also allows specifying the type and the location of the source archives needed by the API at deployment time. An XML schema describing the various descriptive elements of the application manifest and their hierarchy is illustrated in Figure 5.

> ***Note***: *the application manifest used in this document is given as an example. While implementing our API you can specify your own manifests.*

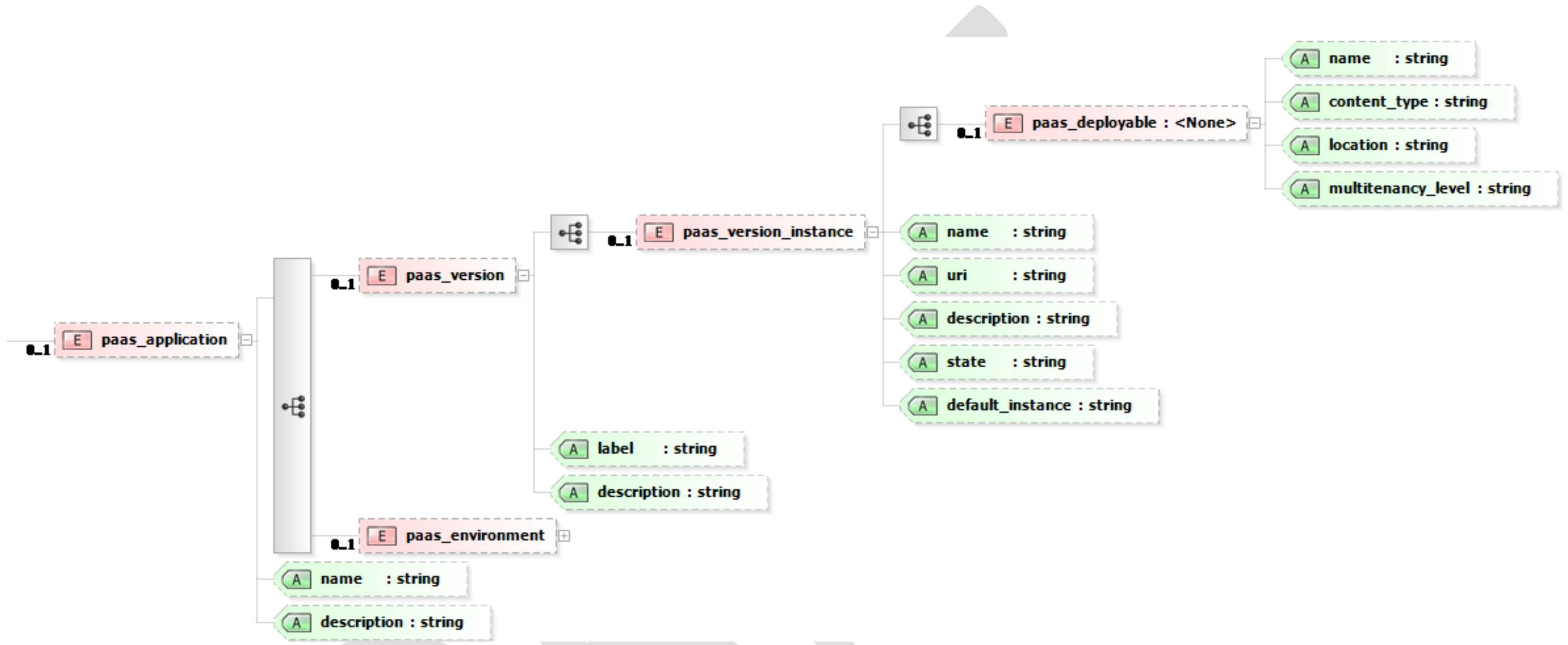

Figure 5 XML schema of an application Manifest

4.2 Application description

Some of the application management operations return as output an XML application descriptor. The XML format of this descriptor is specific and depends on the COAPS API implementation. In the following, we provide the XML schema for the application descriptor corresponding to the CloudFoundry API implementation (see Figure 6).

> ***Note**: the application description used in this document is given as an example. While implementing our API you can specify your proper application description.*

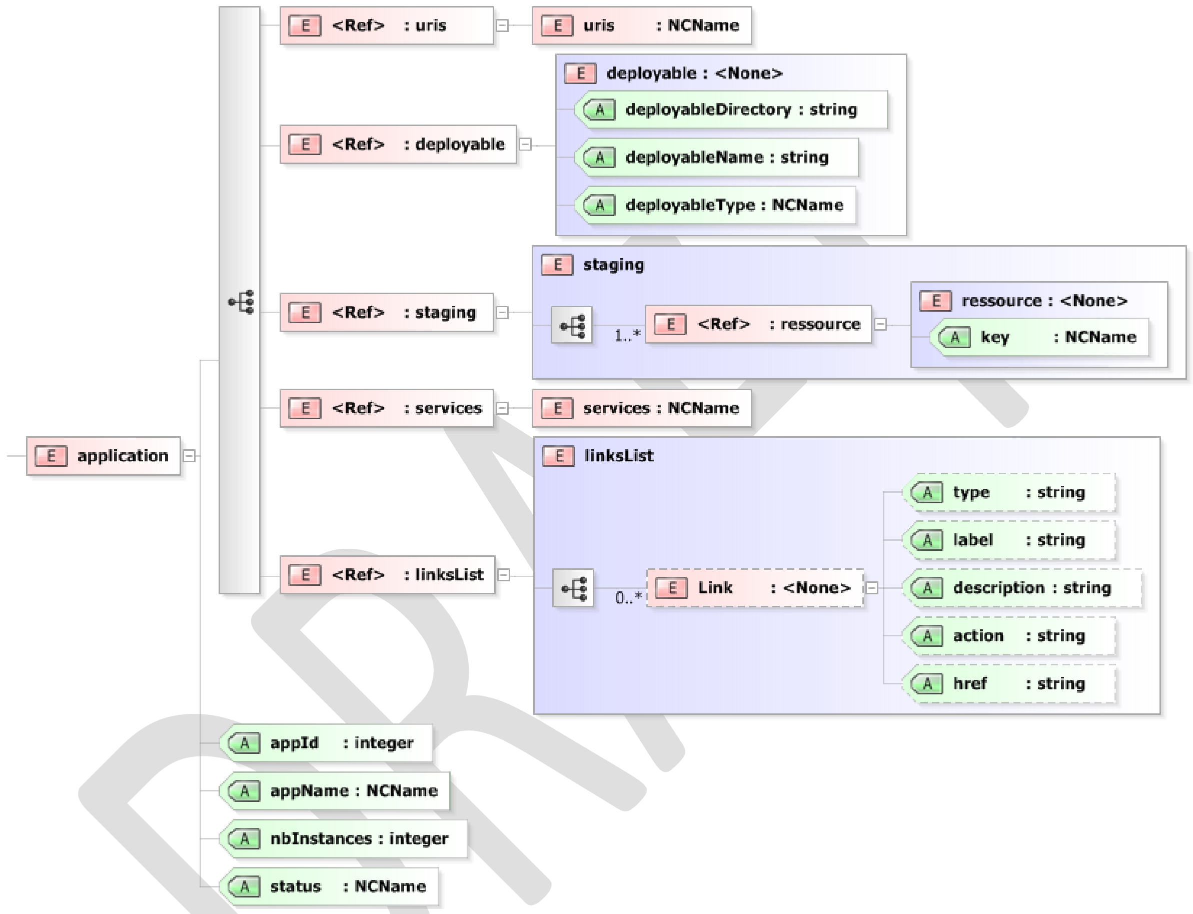

Figure 6 XML schema of a possible application description

In Table 2, we provide the semantics of the different elements (E) and attributes (A) in the application descriptor and provide the corresponding element in the application Manifest (See Figure 5 and Figure 6).

Element of the application descriptor	Description	Corresponding element in the application manifest
E uris	The URI of the deployed application. This URI is automatically generated using the provided application name and the PaaS URI.	--
E deployable	This element describes the application deployable (e.g. artifact, source files …).	E <paas_deployable>
E Staging	This element describes the runtime, framework and commands required by the	--

	application. This information is retrieved from the <paas_node> element in the environment manifest.	
E services	This element describes the services (e.g. messaging, databases …) used by the application. This information is retrieved from the <paas_node> element in the environment manifest.	--
E linksList	The set of links associated to the environment. They are hyperlinks to either different states of the associated resources or different resources (see Section 6).	--
A appId	An automatically generated identifier for the application.	--
A appName	The application's name.	A name defined in <paas_application>
A nbInstances	The number of the application instances.	The number of <paas_version_instance> elements.
A status	This attribute indicates the status (STARTED/STOPPED) of the application. When an application is created, the default value is STOPPED.	--

Table 2 elements and attributes of the application description

4.3 Application management methods

4.3.1 Creating Application

This method creates a new application using an application descriptor.

Resource identifier	/app
HTTP method	POST
Input parameter	An XML application manifest (in the body of the request)
Response	An XML application descriptor. This descriptor contains, among other information, the created application's ID.
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

Example of HTTP request and response:

Request

```
POST /CF-api/app HTTP/1.1
Host : hostname :port
Accept : text/xml
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Application manifest comes here/>
```

Response

```
HTTP/1.1 200 OK
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Representation of the new application comes here/>
```

4.3.2 Updating Application

This method updates an existing application. The application ID must be provided (*i.e.* appId) and the updates has to be specified in the input parameter (*i.e*. as an application Manifest).

Resource identifier	/app/{appId}/update
HTTP method	POST
Input parameter	An XML application manifest (in the body of the request)
Response	The new XML application descriptor
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

Example of HTTP request and response:

```
Request

POST /CF-api/app/2/update HTTP/1.1
Host : hostname :port
Accept : text/xml
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Updated manifest of Application 2 comes here/>

Response

HTTP/1.1 200 OK
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Updated representation of Application 2 comes here/>
```

4.3.3 Finding Applications

This method lists the available applications.

Resource identifier	/app
HTTP method	GET
Input parameter	--
Response	A list of XML application descriptors of the existing applications
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

Example of HTTP request and response:

```
Request

GET /CF-api/app HTTP/1.1
Host : hostname :port

Response

HTTP/1.1 200 OK
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Representation of a list of applications comes here/>
```

4.3.4 Starting Application

This method starts a deployed application.

Resource identifier	/app/{appId}/start
HTTP method	POST
Input parameter	--
Response	The XML application descriptor with the value of the status attribute set to **STARTED**
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

4.3.5 Stopping Application

This method stops a started application.

Resource identifier	/app/{appId}/stop
HTTP method	POST
Input parameter	--
Response	The XML application descriptor with the value of the status attribute set to **STOPPED**
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

4.3.6 Restarting Application

This method restarts a deployed application.

Resource identifier	/app/{appId}/restart
HTTP method	POST
Input parameter	--
Response	The XML application descriptor with the value of the status attribute set to **STARTED**
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

4.3.7 Describing Application

This method returns the XML application description for an application given its ID.

Resource identifier	/app/{appId}
HTTP method	GET

Input parameter	--
Response	The XML application descriptor
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

4.3.8 Destroying Application

This method deletes an application given its ID.

Resource identifier	/app/{appId}
HTTP method	DELETE
Input parameter	--
Response	The destroy discharge
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

Example of HTTP request and response:

Request

```
DELETE /CF-api/app/3 HTTP/1.1
Host : hostname :port
```

Response

```
HTTP/1.1 200 OK
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<message>The application with ID 3 was successfully destroyed</message>
```

4.3.9 Destroying Applications

This method deletes all existing applications.

Resource identifier	/app/delete
HTTP method	DELETE
Input parameter	--
Response	The destroy discharge
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

4.3.10 Deploying Application

This method deploys an application identified by its ID (i.e. appId) on an existing environment also identified by its ID (i.e. envId). The application artifact to deploy must also be included.

Resource identifier	/app/{appId}/action/deploy/env/{envId}
HTTP method	POST
Input parameter	The application artifacts (as a file)
Response	An XML application descriptor
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

Example of HTTP request and response:

```
Request

POST /CF-api/app/1/action/deploy/env/2 HTTP/1.1
Host : hostname :port
Accept : text/xml
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<Application manifest with a path to the deployed application comes here/>

Response

HTTP/1.1 200 OK
Content-Type : text/xml

<?xml version="1.0" encoding="UTF-8"?>
<message>The application 2 was successfully deployed on the environment 1<message/>
<Representation of the application 2 comes here/>
```

4.3.11 Undeploying Application

This method un-deploys an application identified by its ID (i.e. appId) already deployed on an existing environment also identified by its ID (i.e. envId).

Resource identifier	/app/{appId}/action/undeploy/env/{envId}
HTTP method	POST
Input parameter	--
Response	The un-deployment discharge
Status code	200 if OK the error code otherwise (see Section 4.4 for possible error codes)

4.4 Application representation

4.4.1 Representation of an Application

```
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<application xmlns:ns2="resource namespace" appName="name" appId="ID" status="STOP/START"
memory="512" checkExists="true" nbInstances="1">
        <uris>SampleApplication.cloudfoundry.com</uris>
        <deployable deployableName="SampleServlet.war" deployableType="artifact"
                deployableDirectory="APPLICATION_PATH"/>
        <versionInstances instanceName="Instance1"/>

<linksList>
        < link type= “state” label="startApplication()"  action="POST"
                description= “start App ”   href="http://hostname:port/CF-api/app/ID/start"/>
        < link type= “state” label="stopApplication()" action="POST"
                description= “ stop App”   href="http://hostname:port/CF-api/app/ID/stop"/>
        < link type= “state” label="restartApplication()" action="POST"
                description= “ restart App”  href="http://hostname:port/CF-api/app/ID/restart"/>
```

```
            < type= “state” label="updateApplication()"   action="POST"
                        description= “ application’s manifest” href="http://hostname:port/CF-api/app/ID/update"/>
            < type= “state” label="destroyApplication()"    action="DELETE"
                        description= “ destroy App” href="http://hostname:port/CF-api/app/ID"/>
            …

            <link type=“hplink” label="createApplication()"  action="POST"
                        description = “application’s manifest” href="http://hostname:port/CF-api/app "/>
            <link type=“hplink” label="findApplications()"  action="GET"
                        description = “ ” href="http://hostname:port/CF-api/app "/>
            <link type=“hplink” label="getEnvironmentsOfAnApp()"  action="GET"
                        Description = “ ”  href="http://hostname:port/CF-api/app/ID/environment"/>
            …
  </linksList>
 </application>
```

4.4.2 Representation of a list of Application

```
Content-Type: application/xml

<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<applications>
            <application>
                        <id>appID</id>
                        <name>appName</name>
                        <description>appDesc</description>
                        <uri> http://hostname:port/CF-api/app/appID</uri>
            </ application>
            …
            < application>
                        …
            </ application>
</ applications>
```

5 Common errors

This section lists common errors, based on the HTTP status codes[5], which can be returned by the management operations.

	Error	Description	HTTP Status Code
Client errors	Bad Request	The request has a syntax error (invalid action, missing parameter …).	400
	Resource Not Found	The requested resource (environment or application) was not found.	404
	Method Not Allowed	The used REST action (i.e. GET, POST …) is not allowed on that resource.	405
Server errors	Internal Failure	The internal processing has failed due to some unexpected errors.	500
	Service Unavailable	The request has failed due to a temporary failure on	503

[5] Fielding, et al. HTTP/1.1, Internet RFC 2616, available at: http://www.ietf.org/rfc/rfc2616.txt.

		the server	
	Timeout exception	The server took long time to respond.	504

6 Link Elements

The COAPS API uses link elements to connect: (1) application objects to environment objects and (2) different management methods to application and environment objects. The aim of links is to ease the retrieval, by a human or software agent, of the information associated to an environment or an application object.

The structure of the link element is described in Figure 7.

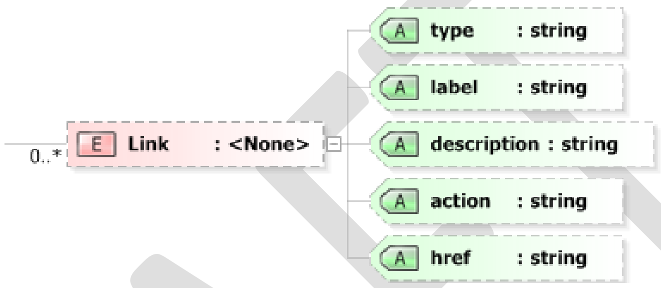

Figure 7 XML schema of the link element

In Table 3, we provide the semantics of the different attributes (A) of the link element.

Attribute	Description
A type	Type of the link: *state* or *hplink*. *state* is a link to other states in the lifecycle of the associate resource. *hplink* is a link to another resource.
A label	The name of the management method (e.g. describeApplication(), destroyEnvironment(), etc.)
A description	Description of the link (text, metadata, manifest data, etc.).
A action	The associated HTTP action (e.g. GET, POST, etc.)
A href	The URI, including the resource identifier, of the management method

Table 3 attributes of the link element

7 Annex A: Application and Environment Management Operations

The following Table provides a summary of the different Application and environment management operations and their associated REST method and resource identifiers.

Application management operations	
Operation	**Method**
Create Application	POST /app
Update Application	POST /app/{appId}/update
Find Applications	GET /app
Start Application	POST /app/{appId}/start
Stop Application	POST /app/{appId}/stop
Restart Application	POST /app/{appId}/restart
Describe Application	GET /app/{appId}
Destroy Application	DELETE /app/{appId}
Destroy Applications	DELETE /app/delete
Deploy Application	POST /app/{appId}/action/deploy/env/{envId}
Undeploy Application	POST /app/{appId}/action/undeploy/env/{envId}
Environment management operations	
Operation	**Method**
Create Environment	POST /environment
Update Environment	POST /environment/{envId}/update
Destroy Environment	DELETE /environment/{envId}
Find Environments	GET /environment
Describe Environment	GET /environment/{envId}
Get Deployed Applications	GET /environment/{envId}/app
Get information	GET /environment/info