Package Development in APS for PA

Lab Guide

1.0.1

ISBN: N/A
Parallels
500 SW 39th Street
Suite 200
Renton, WA 98057
USA
Tel: +1 (703) 815 5670
Fax: +1 (703) 815 5675

Contents

Preface

General Conventions

Be aware of the following conventions used in this book.

- The content of this guide is divided into modules, which, in turn, are subdivided into sub modules.
- When following steps or using examples, be sure to type double-quotes ("), left single-quotes (`), and right single-quotes (') exactly as shown.
- Commands in directories included in a PATH variable are used without absolute path names.

```
# $PRODUCT_ROOT_D/bin/<utility name> [parameters] [options]
```

- Steps that use commands in other, less common, directories show the absolute paths in the examples.

```
# /usr/local/pem/bin/<utility name> [parameters] [options]
```

Typographical Conventions

The following formatting conventions in the text identify special information.

Formatting convention	Type of Information	Example
Special Bold	Items you must select, such as menu options, command buttons, or items in a list.	Navigate to the QoS tab.
	Titles of modules, sections, and subsections.	Read the Basic Administration module.
Italics	Used to emphasize the importance of a point, to introduce a term or to designate a command-line placeholder, which is to be replaced with a real name or value.	These are the so-called *shared containers*. To stop the service, type `/etc/init.d/`*<service script>*
`Monospace`	The names of commands, files, and directories.	Use `less /var/log/messages` to investigate startup issues.
`Preformatted`	On-screen computer output in your command-line sessions; source code in XML, C++, or other programming languages.	`Saved parameters for Container 101`
`Preformatted Bold`	What you type, contrasted with on-screen computer output.	**`# rpm -qa | grep -i pba`**
CAPITALS	Names of keyboard keys.	SHIFT, CTRL, ALT
KEY+KEY	Key combinations - user must press and hold down one key and then press another.	CTRL+P, ALT+F4

Shell Prompts and Command Examples

Command-line examples throughout this guide presume that you are using the Bourne-again shell (bash). Whenever a command can be run as a regular user, we will display it with a dollar sign prompt. When a command is meant to be run as root, we will display it with a hash mark prompt:

Bourne-again shell prompt	`$`
Bourne-again shell root prompt	`#`

Lab Directions

To perform all the exercises, you will be working in the isolated demo-system.

Connecting to Sandbox

To connect to your Sandbox, you need to install and use a VNC client (e.g., TightVNC) on your local workstation. To access the Sandbox, use the directions below:

1 Run the VNC-client software on your local workstation

2 Use the following connection parameters:

a Jumper IP-address: provided by Instructor

NOTE: If you do not know the jumper's IP-address, ask the Trainer to provide.

b VNC-server port: 5901

Ask the instructor to provide you a password.

An example connection string would look like this one: `10.111.122.121:5901`

NOTE: You may use your browser with the java plug-in to connect to the VNC-server.
NOTE: To install the TightVNC client visit `http://www.tightvnc.com/download.php`.

Working with Sandbox

During this hands-on practice you will be working with the isolated environment (Sandbox) that includes the following virtual environments:

Hostname	Operating System	FrontNet	BackNet	Description
pba.edu.trn	CentOS 6	10.111.11.11	10.111.22.11	PBA Application Server
pbadb.edu.trn	CentOS 6		10.111.22.12	PBA Database Server
store.edu.trn	CentOS 6	10.111.11.13	10.111.22.13	PBA Online Store
poamn.edu.trn	CentOS 6		10.111.22.14	POA Management Node
poauibr.edu.trn	CloudLinux 6	10.111.11.15	10.111.22.15	POA UI server + Branding
poadns.edu.trn	CentOS 6	10.111.11.16	10.111.22.16	POA DNS Server
web01.edu.trn	CloudLinux 6	10.111.11.17	10.111.22.17	POA Linux Shared Hosting NG Server
jumper.edu.trn, ns1.edu.trn	CentOS 6	10.111.11.1	10.111.22.1	Entry point to above nodes

The *jumper.edu.trn* VE also has a special interface providing access to the Internet for all other VEs within a Sandbox. You will be using this VE to:

- Access other VEs through *ssh* connection using command line interface (CLI).
- Access the PA control panel through the browser (e.g., Chrome).

Accessing CP

To access the POA control panel through the browser:

1 On the *jumper.edu.trn* VE, open the browser.

2 To open the POA control panel through the brand access point, use the following URL:
`https://cp.edu.trn`

3 To log into the control panel, use the following credentials:
   - Login: *admin*
   - Password: *setup*

Accessing VE

To access a particular server through `ssh`:

1 Open the command line environment (CLE) and type the following command:
`ssh root@<Server_IP_Address>`, e.g., to connect to the POA management node, type:

```
$ ssh root@10.111.22.14
```

2 If prompted, accept the key.

3 Use the following credentials to get authorized:

- Login: *root*
- Password: *sw1q2w3e*

NOTE: Alternatively, you can use aliases: `poamn`, `poadns`, `web01`, etc.

Lab Prerequisites

The host, on which the APS packages will be build (the *jumper* host) requires the following RPM packages to be installed:

- `aps-php-runtime`
- `apstools`
- `php`
- `php-xml`

Performing APS development and provisioning requires the *APS PHP Runtime* and *APS Tools* packages. Download and install the required packages if tasked by Instructor.

1 On the jumper VE, download the latest versions (to date) from the following link: `http://doc.apsstandard.org/tools/`
   - `apstools-x.x-xxx.noarch.rpm`
   - `aps-php-runtime-x.x-xxx.noarch.rpm`

2 Install them:

```
$ sudo rpm -Uvh package_name.rpm
```

3 By using yum, install the `php` and `php-xml` packages:

```
$ sudo yum install php
$ sudo yum install php-xml
```

Ensure the packages are installed, for example, for the `aps-php-runtime` package:

```
$ yum list | grep aps-php-runtime
aps-php-runtime.noarch      2.0-247     installed
```

The following set of lab exercises applies to POA 5.5 deployed in the training environment.

The system infrastructure should include the following servers:

- POA management node with the name *poamn.edu.trm* (BackNet 10.111.22.14)
- POA UI/Branding NG node with the name *poauibr.edu.trn* (BackNet 10.111.22.15)
- DNS server with the name *poadns.edu.trn* (BackNet 10.111.22.16)
- Apache Webserver NG with the name *web01.edu.trn* (BackNet 10.111.22.17)

The *poadns.edu.trn* server (BackNet 10.111.22.16) will serve as the end-point host for APS provisioning, for what it requires the following RPM packages to be pre-installed:

- `aps-php-runtime`
- `php`
- `php-xml`

Be sure to enable the following options in the POA System Properties available in the provider control panel at System > Settings > System Properties:

- APS development mode
- Customers management from POA UI
- Resellers management from POA UI

Also be sure to set the Password Quality Level for Child Accounts to *None* in the System > Settings > Security > Setup screen.

In POA, make sure that the following account entities are pre-created:

- For the Provider account, create the staff member under the name *Training Admin* with the password '*password*'
- Create the customer account under the name *Training Customer* with the staff member of the same name with the password '*password*'

For the *Training Admin* be sure to do the following:

- Assign the *Account Administrator* role

Before proceeding to perform exercise, copy the content from the `Desktop/APS.Development` directory (`Lab-1`, `Lab-2`, ... directories) to the `Desktop/APS2` directory on the jumper VE.

NOTE: Make sure that the write permission is added to the files.

LAB 1

VPS Cloud (Starter) Project

In this set of exercises, you will learn and explore:

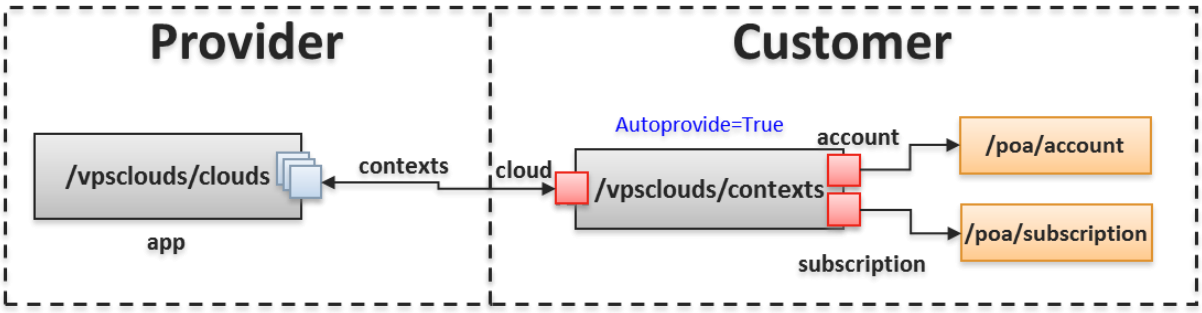
- The general structure of the APS application project
- The approach to the business logic implementation
- How to build and deploy the package
- How to provision the APS application through the POA control panel.

For the demo purpose, you will assemble, deploy, and provision the *VPS Cloud* application.

In This Lab

Application Structure

This exercise uses the following resource model:



The application end-point has the `/vpsclouds/` alias as the document root. Below it, you will provide access to the services that allows creating resources.

- The `APP-META.xml` file declares two services - *clouds* and *contexts*.
- For each service declared in the metadata file, a provisioning script must implement the provisioning logic of the service.
  - On the provider side, the `clouds.php` script implements the `/vpsclouds/clouds` service.
  - On the customer side, the `contexts.php` script implements the `/vpsclouds/contexts` service.
- The *clouds* type must be provisioned along with the reference to the *contexts* (`vpsclouds/contexts`) type.
- The *context* type must be provisioned along with the required references:
  - Reference to the *clouds* type (`vpsclouds/clouds`) is required.
  - Reference to the built-in POA account (`/poa/account`) defines the POA owner of the context.
  - Reference to the built-in POA subscription (`/poa/subscription`) defines the subscription from which the context is created.

Exercise Directions

By completing this set of exercises, you will gain practical skills in assembling of a simple APS application. You will experience the following development stages:

- Setting an application structure.
- Building a package.
- Setting an application end-point.
- Configuring and provisioning an application.

The files, which are required for these exercises are located in the `~/Desktop/APS2/Lab-1` directory:

File	Description
`APP-META.xml`	The metadata file
`clouds.php`	The script implements the *clouds* service
`contexts.php`	The script implements the *contexts* service

Below is an overview of actions you are going to perform to assemble the VPS Cloud "starter" package and provision the APS application:

1 Configure the package structure and create a package:
   - In the lab directory, create the application structure directory and add files to it.
   - Build the application package.

2 Set and configure the application end-point.
   - Create the end-point document root and copy the scripts of the project to this directory.
   - In the end-point document root directory, create the required file and specify the rewrite rules for the application services in it.

3 Deploy and provision the application in POA.
   - Import the application package and install the application instance.
   - Create and configure the application resources and the service template.
   - Create an application subscription and check it.

Setting Package Structure

In this exercise, you will learn and explore:

- The general structure of the APS project.

To manually create the APS project structure:

1 On the jumper VE, navigate to the ~/`Desktop/APS2/Lab-1` directory.

```
$ cd ~/Desktop/APS2/Lab-1
```

2 In the `Lab-1` directory, create the `scripts` directory that will keep the scripts implementing the business logic.

```
$ mkdir scripts
```

3 Into the `scripts` directory, put the following PHP scripts implementing the application business logic:

- `clouds.php`
- `contexts.php`

```
$ mv ~/Desktop/APS2/Lab-1/*.php ~/Desktop/APS2/Lab-1/scripts
```

NOTE: You can make use of the *Midnight Commander* tool for file management.

As a result, you will get the basic structure of the APS project:

```
$ ll -aR
.:
total 20
drwxrwxr-x. 3 palab palab 4096 Oct 24 09:49 .
drwxrwxr-x. 3 palab palab 4096 Oct 24 09:43 ..
-r-xr-xr-x. 1 palab palab 1667 Oct 24 09:49 APP-META.xml
drwxrwxr-x. 2 palab palab 4096 Oct 24 09:49 scripts

./scripts:
total 24
drwxrwxr-x. 2 palab palab 4096 Oct 24 09:49 .
drwxrwxr-x. 3 palab palab 4096 Oct 24 09:49 ..
-r-xr-xr-x. 1 palab palab 1666 Oct 24 09:49 clouds.php
-r-xr-xr-x. 1 palab palab 3443 Oct 24 09:49 contexts.php
```

You will use it further to build the package.

Understanding Provisioning Logic

In this exercise, you will learn and explore:

- The general approach to defining the APS application logic.
- The content and structure of the PHP file implementing this logic.
- The cloud application class definition and functions.
- The context class definitions and functions.

To verify and examine the application and context business logic:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-1/scripts` project directory.
   - Open the `clouds.php` and `context.php` files:

2 Ensure that both files contain the reference to PHP runtime.

```
require "aps/2/runtime.php"
```

3 Ensure that `clouds.php` contains the following main declarations and definitions:
   - The declaration of the *clouds* application type.

   What is the inherited *aps* type? _____
   - The definition of the *clouds* application class.

   What is the base class of the *clouds* application type? _____

4 Ensure the *clouds* class contains the following definitions:
   - The link to the *contexts* type.
   - The global connection settings.

   NOTE: The functions processing links and the CRUD operations should not be declared if they do not contain custom logic.

5 Ensure `contexts.php` contains the following main declarations and definitions:
   - The declaration of the *contexts* type.

   NOTE: The context type version must be the same as one of the cloud application.

   What is the inherited *aps* type? _____
   - The definition of the *contexts* class.

   What is the base class of the *contexts* type? _____

6 Ensure the *contexts* class contains the following definitions:
   - The link to the *clouds* type.
   - The link to the subscription type.

   What is the inherited aps type? _____
   - The link to the account type.

   What is the inherited aps type? _____

   - What are the attributes for these declarations? _____
   - The functions processing these links.

7 For the *contexts* class, note the additional application-specific definitions:
   - The context identifier

- The context password

Further on, you will trace what for and how these definitions are used.

Understanding Metadata

In this exercise, you will learn and explore:

- The general approach to defining the APS application metadata
- The content and structure of the `APP-META.xml` file containing the metadata.

To examine the metadata:

1 On the jumper VE, navigate to the project directory: `~/Desktop/APS2/Lab-1`

2 Open the `APP-META.xml`.

3 Ensure that `APP-META.xml` contains the general sections defining:

- The application general parameters

> Note: Make sure that the app version coincides with the version of the types.

- The application vendor-specific information
- The application packager-specific information
- The application presentation settings
- The application license agreement information

4 Ensure that `APP-META.xml` contains the sections defining the application services:

- The *clouds* and *contexts* in this case.

Building Package

In this exercise, you will learn and explore:

- How to build the APS package from the project.

To build the package and examine the updated project structure:

1 Build the package using the `apsbuild` utility.

a On the jumper VE, navigate to the following directory: `~/Desktop/APS2`

```
$ cd ~/Desktop/APS2
```

b Run the following command:

```
$ apsbuild Lab-1
Building package.......
Generating APP-LIST.xml.......
Validating package.......
Package format version - 2.0
Validation complete: 0 Error(s), 0 Warning(s)
APS package has been built successfully.
Package full path is: /home/palab/Desktop/APS2/VPS_Cloud-1.1-0.app.zip
```

As the result of the successful package building, you will see the output as above.

2 Note the newly created `schemas` folder and the schema files inside the package.

The package is now ready to be installed in POA.

Setting Application End-Point

In this exercise, you will learn and explore how to set the end-point for the demo project on a service host. This includes the package installation and the web server configuration.

NOTE: We will use the *poadns.edu.trn* with the BackNet IP address *10.111.22.16* as the end-point host.

Follow the directions below:

1 Log in to the end-point host via *ssh*:

```
$ ssh root@10.111.22.16
```

2 Create the end-point document root.

```
# cd /var/www/html
# mkdir vpsclouds
```

3 Copy the PHP scripts of the project to the document root, e.g., using the *scp* utility.

a ON THE JUMPER VE, open CLI and run the following command:

```
$ scp ~/Desktop/APS2/Lab-1/scripts/*.php root@10.111.22.16:/var/www/html/vpsclouds
```

4 RETURN TO THE END-POINT HOST and check the Apache configuration.

a Open the Apache configuration file for editing, e.g.:

```
# vim /etc/httpd/conf/httpd.conf
```

b Ensure the parent directory section is configured as follows:

```
<Directory "/var/www/html">
.................
AllowOverride All
.................
</Directory>
```

c Save changes if needed.

NOTE: Ask the Instructor to help if you are not familiar with the *vim* editor.

5 For each application service, set the rewrite rule.

a In the document root, create the `.htaccess` file, e.g.:

```
# vim /var/www/html/vpsclouds/.htaccess
```

b In this file, set the rewrite rules.

```
Options +FollowSymLinks +ExecCGI
<IfModule mod_rewrite.c>
   RewriteEngine On
   RewriteBase /vpsclouds
   RewriteRule ^clouds(|/.*)$ clouds.php?q=$1 [L,QSA]
   RewriteRule ^contexts(|/.*)$ contexts.php?q=$1 [L,QSA]
</IfModule>
```

c Save changes.

6 For all site files, assign *owner:group* as *apache:apache*:

```
# chown -R apache:apache /var/www/html/vpsclouds
```

7 Ensure the file structure look as follows:

```
# ls -laR /var/www/html
/var/www/html:
total 12
drwxr-xr-x 3 root   root   4096 Oct 12 06:13 .
drwxr-xr-x 6 root   root   4096 Oct 12 05:49 ..
drwxr-xr-x 2 apache apache 4096 Oct 12 06:15 vpscloud

/var/www/html/vpscloud:
total 20
drwxr-xr-x 2 apache apache 4096 Oct 12 06:15 .
drwxr-xr-x 3 root   root   4096 Oct 12 06:13 ..
-rw-r--r-- 1 apache apache 1610 Oct 12 06:13 clouds.php
-rw-r--r-- 1 apache apache 3349 Oct 12 06:13 contexts.php
-rw-r--r-- 1 apache apache  317 Oct 12 06:15 .htaccess
```

8 Restart the *httpd* service on the end-point host:

```
# service httpd restart
```

9 Ensure that the end-point correctly responds to HTTP requests.

a Ensure the output of the following commands:

```
# curl http://poadns.edu.trn/vpsclouds/contexts/
# curl http://poadns.edu.trn/vpsclouds/clouds/
```

looks like follows:

```
{"code": 404, "type": "RuntimeException", "message": "Not Found: No appropriate
method found for url vpsclouds\/clouds"}
```

This concludes setting the endpoint for this project.

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Importing Application Package

In this exercise, you will learn and explore how to import the APS package through the POA control panel. To import the package:

1 Login to the POA provider control panel.

2 Navigate to Services > Applications.

3 Click Import Package.

4 Select local file and seek for the package on your jumper VE, e.g.:
`~/Desktop/APS2/VPS_Cloud-1.1-0.app.zip`

5 Press the Submit button.

6 In the Applications screen, click Refresh.

As a result, the package version (e.g., 1.1-0) appears in the Latest Version column.

NOTE: If it has not appeared, you should proceed with the troubleshooting starting from the POA task manager.

Setting Application Instance

In this exercise, you will learn and explore how to create an APS application instance.

To create an application instance:

1 In the POA provider control panel, navigate to Services > Applications.
2 Open a profile of the VPS Cloud package.
3 On the Instances tab, click Install.
4 Enter the following end-point URI:
`http://10.111.22.16/vpsclouds`
and click Next.
5 Enter the global settings if required, e.g.:
a In the apphost field, type the IP-address of the end-point host: 10.111.22.16
b Provide the application administrator's credentials:
Cloud Admin: *cloudadmin*
Cloud Password: *password*
c Click Next.
6 Click Finish to confirm the settings and complete the installation.

As a result, you will notice the following message: *Installation of application VPS Cloud successfully scheduled*. Refresh the screen and the status is changed to provisioning or ready.

NOTE: Check if there is a failed task. If so, proceed to the troubleshooting section.

Setting Resource Types

In this exercise, you will learn and explore:

- The general approach of creating resource types for APS application services through the control panel.
- How to create the resource types for the VPS Cloud application services through the control panel.

To create the resource types for the application services.

1 In the POA provider control panel, navigate to Services > Applications.

2 Open a profile of the APS package and switch to the Resource Types tab.

3 Create the application resource on the basis of the *Application Service Reference* class.

a Click Create and select *Application Service Reference*.

b In the Name field, enter *VPS Clouds - Application* and click Next.

c Select *VPS cloud globals* as the APS type.

NOTE: In `APS-META.xml`, you can find the `VPS cloud globals` name in the `presentation` sub-section of the section of the `"clouds"` service.

d In the Resource column, click on the instance ID.

e Click Finish.

4 Create the resource type that will be used for the provisioning of the management contexts for customers.

a Click Create and select *Application Service*.

b In the Name field, enter *VPS Clouds - VPS Management* and click Next.

c Select *VPS Management* as the application service.

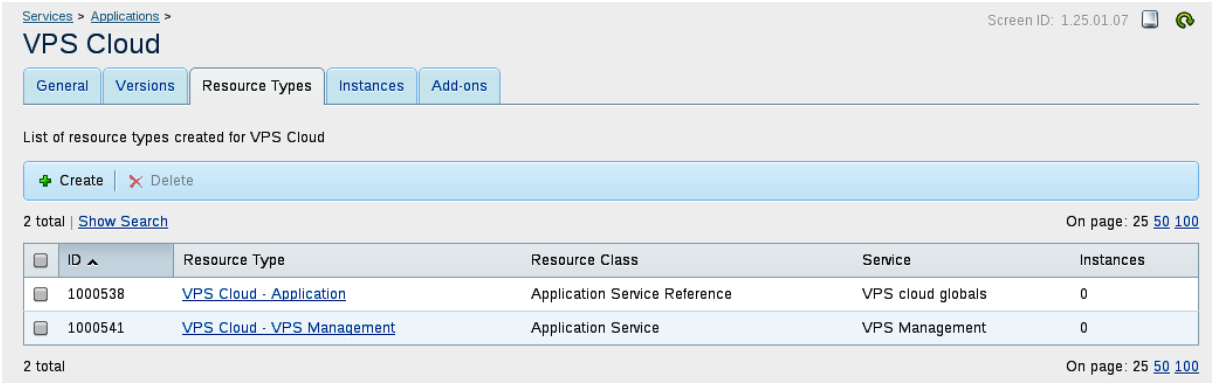
d Leave the Priority field as is.

e Select the Automatically provision service check-box and click Next.

NOTE: You will not be able to manage an application without a context.

f Click Finish.

As a result, we have two resources ready to be provisioned.



1 Make sure that the *clouds* resource is already available on the service bus:

a Log in to the POA management node via *ssh*:

```
$ ssh root@10.111.22.14
```

b Run the command:

```
# curl -E /usr/local/pem/APS/certificates/poa.pem -k
https://localhost:6308/aps/2/resources?aps.type=http://edu.trn/vpsclouds/clouds/1.1
```

c Find the response:

```
[
        {
                "aps":
                {
                        "type": "http://edu.trn/vpsclouds/clouds/1.1",
                        "id": "12fb8b58-305e-41f8-a680-c951048dd6b0",
                        "status": "aps:ready",
                        "revision": 4,
                        "modified": "2013-12-26T09:30:13Z"
                },
                "apphost": "10.111.22.16",
                "cloudadmin": "cloudadmin",
                "cloudpass": "password"
        }
]
```

2 Using the same approach, check if the *contexts* resource is available on the service bus.

- Which response did you get? _____
- Why? _____

Setting Service Template

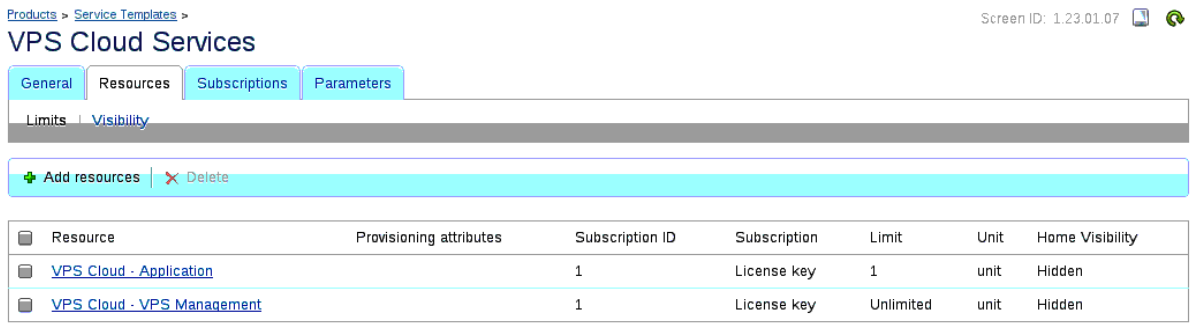
In this exercise, you will learn and explore:

- The general approach to creating a service template for an APS application.
- How to create the service template for the VPS Cloud application.

To create a POA service template for the application:

1 In the POA provider control panel, navigate to Products > Service Templates.
2 Start creating the new service template
   a Click Add New Service Template.
3 Configure the general parameters.
   a In the Name field, enter *VPS Clouds Services*
   b Select the Autoprovisioning check-box.
   c Set the Type option as Custom.
   d Click Next.
4 In the list of resource types, select both resource types created earlier and click Next.
5 For each resource, uncheck the Unlimited checkbox and set the Limit as 1.
6 Click Next, then Finish.
7 Activate the service template.
   a Open the newly created service template.
   b In the General section, click Activate.

As the result, you can now use the service template for resource provisioning.



Provisioning Application

In this exercise, you will subscribe to the service on behalf of a customer to verify the successful provisioning. To create a subscription and verify service provisioning:

1 Switch to the Subscriptions tab in the *VPS Cloud Services* service template.
2 Click Create New Subscription.
3 Select the *Training Customer*.
4 Do not change the limits and click Next.
5 Click Finish and wait until the screen refreshes.
6 Verify the successful subscription provisioning:
   a Navigate to Operations > Subscriptions.
   b Locate the *VPS Cloud Services* subscription in the list.

As a result, POA creates the new subscription and provisions the service.

1 Make sure that the *contexts* resource has also became available on the service bus:
   a Log in to the POA management node via *ssh*:

```
$ ssh root@10.111.22.14
```

   b Run the command:

```
# curl -E /usr/local/pem/APS/certificates/poa.pem -k
https://localhost:6308/aps/2/resources?aps.type=http://edu.trn/vpsclouds/contexts/1
.1
```

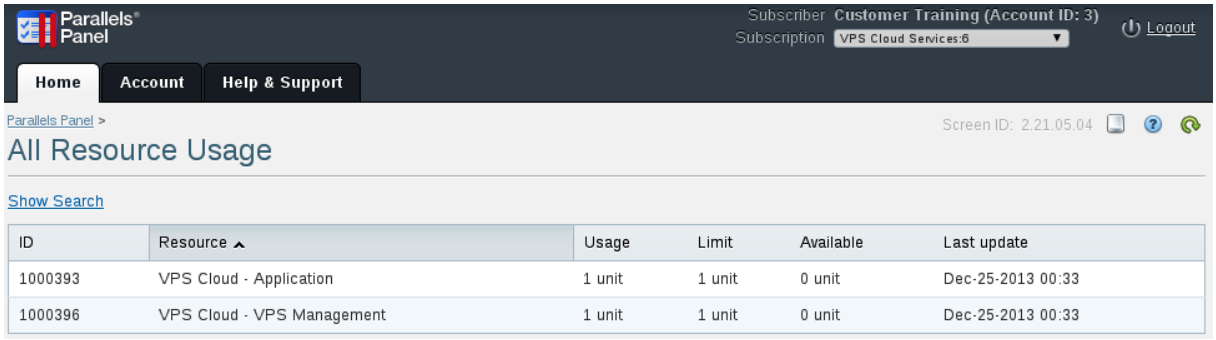
   c Find the response, e.g.:

```
[
        {
                "aps":
                {
                        "type": "http://edu.trn/vpsclouds/contexts/1.1",
                        "id": "001ee76c-3931-4a4c-9187-b25dcc974f43",
                        "status": "aps:ready",
                        "revision": 4,
                        "modified": "2013-12-26T09:35:38Z"
                },
                "contextid": "11",
                "contextpass": "kpcOA6%ISRN2"
        }
]
```

Exploring Application Features

In this exercise, you will explore the application features. Follow the directions below.

1 Login to the customer control panel.
   a In the POA provider control panel, navigate to Operations > Customers.
   b Open the profile of the *Training Customer* customer.
   c On the General tab, click Staff Members.
   d Click the Login as Customer link.
2 Verify what is available for the *Training Customer*.
   a In the top-right subscription selector, select the *VPS Cloud Services* subscription.
   b On the Home tab, click Resource Usage.
   c Verify that POA provisioned both resources.
   d Examine the Usage, Limit, and Available columns.



This concludes the development, deployment, and provisioning of the "starter" demo APS application.

Answer Keys

Understanding Provisioning Logic

What is the inherited *aps* type? - *application*

What is the base class of the *clouds* application type? - *ResourceBase*

What is the inherited *aps* type? - *resource*

What is the base class of the *contexts* type? - *ResourceBase*

What is the inherited aps type? - *subscription*

What is the inherited aps type? - *account*

What are the attributes for these declarations? - *required*

LAB 2

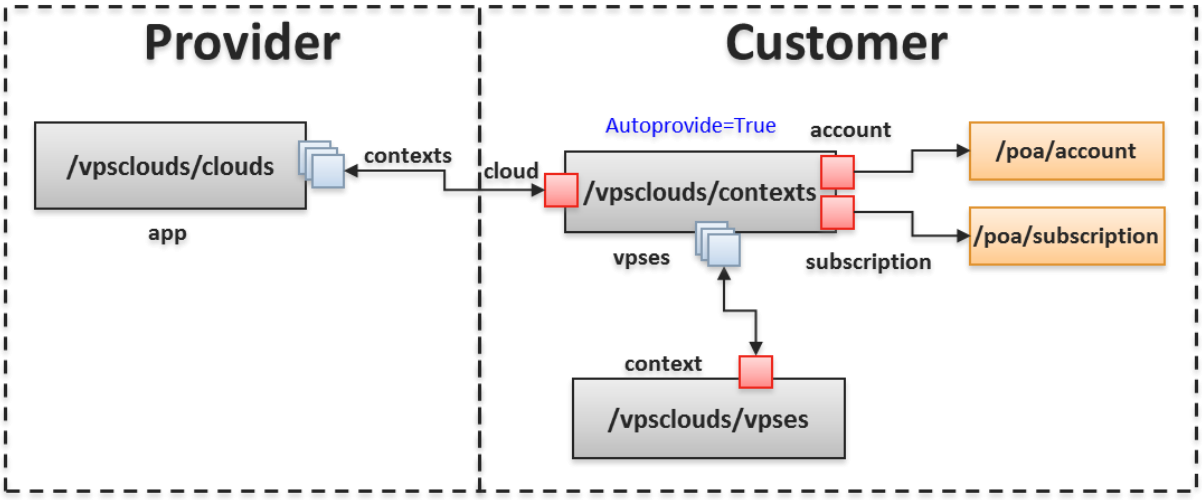
VPS Cloud (Basic) Project

In the following set of exercises, you will create the new extended mockup project on the basis of the existing one. You will learn and explore how to modify the copy of the existing APS project adding to it the new functionality that allows selling VPSs through the control panel.

In This Lab

Application Structure

In this exercise, you will build a new application with the following resource model:



This package contains three APS types. In addition to the types in the *VPS Cloud (Starter)* project, you will add the resource type presenting VPS.

- The new resource type *vpses* must be provisioned along with the reference to the *contexts* (`vpsclouds/contexts`) type.

Exercise Directions

In this part of the lab you will extend the “starter” project adding new feature: VPS management. Here you gain new skills in embedding UI into the customer control panel. You will experience the following development stages:

- Setting an application structure.
- Configuring application scripts.
- Configuring the application metadata.
- Building a package.
- Setting an application end-point.
- Configuring and provisioning an application.

You need to use the following data to perform the set of exercises:

- Previously created project files containing in the `~/Desktop/APS2/Lab-1` directory.
- Files located in the `~/Desktop/APS2/Lab-2` directory:

File	Description
`vpses.php`	The script implements the *vpses* service
`servers.html`	The file implements a VPS management screen
`server.edit.html`	The file implements a VPS editing screen
`server.new-1.html`	The file implements a first step of VPS creation screen
`server.new-last.html`	The file implements a last step of VPS creation screen
`displayError.js`	The file containing the *displayError* function
`server-wizard.js`	The file containing description of the VPS creation steps

You will update the structure of the VPS Cloud “starter” project to extend both presentation and business logic by performing the following actions:

1 Configure the package structure and build the package:
   - Update the project structure by adding new/modifying existing project files.
   - Modify the metadata file for the new project.
   - Build the application package.
2 Update and configure the application end-point.
3 Deploy and provision the application in POA.
   - Import the application package and update the application instance.
   - Create and configure the new application resources and update the service template.
   - Update the subscription and check the new functionality.

Setting Package Structure

In this exercise, you will update a project by extending the structure of the existing *VPS Cloud (Starter)* project. Follow the directions below.

1 Copy the content of the previously created project to the `basic` directory:

```
$ cp -r ~/Desktop/APS2/Lab-1/scripts ~/Desktop/APS2/Lab-2
$ cp ~/Desktop/APS2/Lab-1/APP-META.xml ~/Desktop/APS2/Lab-2
```

2 Update the structure and contents of the new project:

a In the `Lab-2` directory, create the `ui` directory.

```
$ mkdir ~/Desktop/APS2/Lab-2/ui
```

b Move the following UI files into the `ui` directory:

```
server.edit.html
server.new-1.html
server.new-last.html
servers.html
displayError.js
server-wizard.js
```

```
$ mv ~/Desktop/APS2/Lab-2/*.html ~/Desktop/APS2/Lab-2/ui
$ mv ~/Desktop/APS2/Lab-2/*.js ~/Desktop/APS2/Lab-2/ui
```

c From the same source, move the `vpses.php` script to the `scripts` directory.

```
$ mv ~/Desktop/APS2/Lab-2/vpses.php ~/Desktop/APS2/Lab-2/scripts
```

Although you have got now the new project structure, the new project is not yet ready for building a new package. In the following exercises, you will update the scripts and the `APP-META.xml` file to reflect the changes, i.e., VPS management logic and UI elements.

Understanding Provisioning Logic

In this exercise, you will learn and explore the content and structure of the additional PHP file containing the VPS management logic.

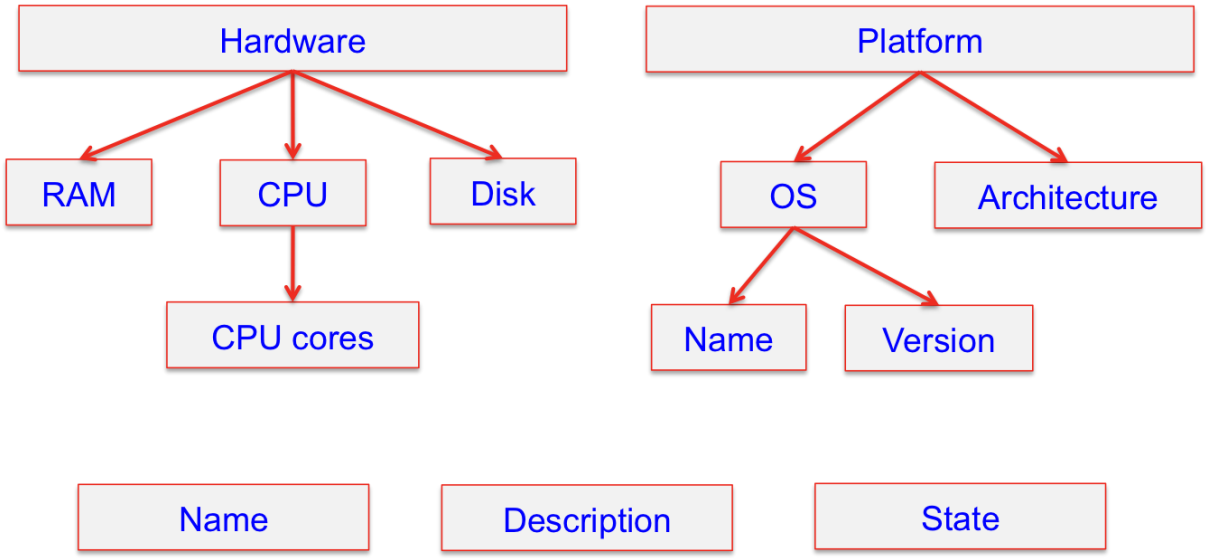
To verify and examine the VPS management business logic:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-2/scripts` project directory.

2 Open the `vpses.php` files.

3 Ensure that this file contains the reference to the PHP runtime.

4 Ensure that this file contains the main declarations and definitions reflecting the accepted for VPS hierarchical structure of properties.



5 Ensure the *vpses* class contains the following declarations and definitions:

- The link to the *contexts* type.
- The declarations of the all VPS attributes.

Modifying Scripts

In this exercise you will perform the following operations with the scripts:

- Update resource version in all scripts.
- In the `contexts.php` script add the relation with *vpses*.

Follow the directions below:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-2/scripts/` project directory.

2 Open the `clouds.php` file for editing.

a Update the type version as follows.

```
 * @type("http://edu.trn/vpsclouds/clouds/1.2")

 * @link("http://edu.trn/vpsclouds/contexts/1.2[]")
```

3 Open the `contexts.php` file for editing.

a Update the type version as follows.

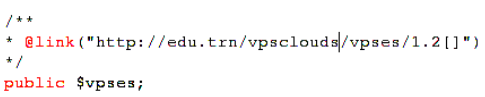
```
 * @type("http://edu.trn/vpsclouds/contexts/1.2")

 * @link("http://edu.trn/vpsclouds/clouds/1.2")
```

NOTE: Missing an update of the type version could result in an issue afterwards.

b Add the relation to the new *vpses* type.

```
/**
* @link("http://edu.trn/vpsclouds/vpses/1.2[]")
*/
public $vpses;
```



4 Save changes and close the files.

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Modifying Metadata/Building Package

In this exercise, you will learn and explore how to modify the meta data to:

- Make the package upgradable.
- Declare the navigation tree plugged into the customer control panel.
- Declare the new *vpses* service and specify by the `<code>` element that the JSON schema will be declared in the `vpses.php` script.

Follow the directions below:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-2` project directory.

2 Open the `APP-META.xml` file for editing.

a Ensure the application version is updated.
`<version>1.2</version>`

b Under the *Infrastructure Management* category, add the new section that provides the VPS management logic.

```
        <navigation id="ccp" label="VPS Management">
                <var name="context" type-
id="http://edu.trn/vpsclouds/contexts/1.2"/>
                <plugs-to id="http://www.aps-standard.org/ui/service"/>
                <item id="servers" label="Servers">
                        <view id="servers" label="Servers">
                                <view id="server.new-1" label="Edit (step 1)">
                                        <controls>
                                                <cancel/>
                                                <next/>
                                        </controls>
                                </view>
                                <view id="server.new-last" label="Review">
                                        <controls>
                                                <prev/>
                                                <finish/>
                                        </controls>
                                </view>
                                <view id="server.edit" label="Edit">
                                        <var name="vps" type-
id="http://edu.trn/vpsclouds/vpses/1.2"/>
                                        <controls>
                                                <cancel/>
                                                <finish/>
                                        </controls>
                                </view>
                        </view>
                </item>
        </navigation>
```

```
        <category>Infrastructure/Management</category>
    </categories>
    <navigation id="ccp" label="VPS Management">
        <var name="context" type-id="http://edu.trn/vpsclouds/contexts/1.2"/>
        <plugs-to id="http://www.aps-standard.org/ui/service"/>

        <item id="servers" label="Servers">
            <view id="servers" label="Servers">
                <view id="server.new-1" label="Edit (step 1)">
                    <controls>
                        <cancel/>
                        <next/>
                    </controls>
                </view>
                <view id="server.new-last" label="Review">
                    <controls>
                        <prev/>
                        <finish/>
                    </controls>
                </view>
                <view id="server.edit" label="Edit">
                    <var name="vps" type-id="http://edu.trn/vpsclouds/vpses/1.2"/>
                    <controls>
                        <cancel/>
                        <finish/>
                    </controls>
                </view>
            </view>
        </item>
    </navigation>
```

Note that each view element is implemented by the corresponding HTML file, for example:

- The *server.new-1* view will be implemented by the `server.new-1.html` file and used to display the first step in creating a new VPS.

  Which buttons it declares? _____

a Make the package upgradable by adding the string:

```
    <upgrade match="version=ge=1.0"/>
```

```
        <name>End-User License Agreement</name>
        <file>http://opensource.org/licenses/bsd-license</file>
    </text>
    </license-agreement>
<upgrade match="version=ge=1.0"/>
        <service id="clouds">
```

b To the list of services, add the new service.

```
<service id="vpses">
        <code engine="php" path="scripts/vpses.php"/>
    <presentation>
        <name>Virtual Private Server</name>
        <summary>Cloud virtual private server</summary>
    </presentation>
</service>
```

```
    <service id="contexts">
        <code engine="php" path="scripts/contexts.php"/>
        <presentation>
            <name>VPS Management</name>
            <summary>VPS management environment</summary>
        </presentation>
    </service>
    <service id="vpses">
        <code engine="php" path="scripts/vpses.php"/>
        <presentation>
            <name>Virtual Private Server</name>
            <summary>Cloud virtual private server</summary>
        </presentation>
    </service>
</application>
```

3 Save the file and build the package:

a On the jumper VE, navigate to the following directory: `~/Desktop/APS2`.

```
$ cd ~/Desktop/APS2
```

b Run the following command:

```
$ apsbuild Lab-2
```

As a result, you can find the new package at `~/Desktop/APS2/VPS_Cloud-1.2-0.app.zip`

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Setting Application End-Point

In this exercise, you will learn and explore how to update the end-point for the extended project on a service host. To install the new application instance:

1 Copy the updated and new PHP scripts to the `/var/www/html/vpsclouds/` directory, e.g., using the *scp* utility.

a On the jumper VE, run the following command:

```
$ scp ~/Desktop/APS2/Lab-2/scripts/*.php root@10.111.22.16:/var/www/html/vpsclouds
```

2 Update the `.htaccess` file to make HTTP requests for the *vpses* service redirected to the `vpses.php` script:

a Connect to the service host through *ssh* as you did earlier:

```
$ ssh root@10.111.22.16
```

b In the document root, open the `.htaccess` file for editing, e.g.:

```
# vim /var/www/html/vpsclouds/.htaccess
```

c In this file, add the following rewrite rule.

```
RewriteRule ^vpses(|/.*)$ vpses.php?q=$1 [L,QSA]
```

d Save changes.

3 For all site files, assign *owner:group* as *apache:apache*:

```
# chown -R apache:apache /var/www/html/vpsclouds
```

4 Restart the *httpd* service on the end-point host:

```
# service httpd restart
```

5 Ensure that the end-point correctly responds to HTTP requests.

a Ensure the output of the following command:

```
# curl http://poadns.edu.trn/vpsclouds/vpses/
```

looks like follows:

```
{"code": 404, "type": "RuntimeException", "message": "Not Found: No appropriate method found for url vpsclouds\/vpses"}
```

6 In the end-point directory, clean a type cache in order to force the PHP runtime use the new objects instead of the cached ones:

```
$ rm -rf /var/www/html/vpsclouds/typeCache/
```

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Deployment and Provisioning Application

In this exercise, you will learn and explore how to further deploy and then provision the application, including:

- Import the updated APS package to POA.
- Upgrade application instance.
- Add resources.
- Upgrade the service template.
- Upgrade the subscription.

To update and make the application ready for sale:

1 Import the new application package as you did earlier, and then refresh the screen.

2 Upgrade the application instance:
   a Open the updated package and click the Upgrade Instances button.
   b Select Upgrade all instances if requested and click Next.
   c Click Finish.

3 For provisioning VPS, create the additional resource type.
   a In the POA provider control panel, navigate Services > Applications.
   b Open a profile of the APS package and switch to the Resource Types tab.
   c Click Create and select *Application Service*.
   d In the Name field, enter *VPS Cloud - Virtual Server* and click Next.
   e Select *Virtual Private Server* as the application service.
   f Leave the Priority field as is.
   g Leave the Automatically provision service check-box unmarked and click Next.
   h Click Finish.

4 Update a service template for the VPS Cloud application.
   a Navigate to Products > Service Templates.
   b Open the profile of the *VPS Cloud Services* service template.
   c In the service template profile, open the Resources tab.
   d Click Add Resources to add the new one.
   e Select the *VPS Clouds - Virtual Server* and click Submit. Do not limit this resource.
   f Click Submit.

5 For *Training Customer*, upgrade the subscription:
   a Navigate to Operations > Subscriptions.
   b Open the profile of the *VPS Cloud Services* subscription.
   c Click the Upgrade button.

As a result, the new application features become available in the customer CP.

Exploring Application Features

In this exercise, you will explore the new application features. Follow the directions below.

1 Login to the customer control panel.

a IN the subscription profile, open the Managed By tab and click *Login as Customer*.

2 Verify the services available for the *Training Customer*.

a In the subscription selector, select the upgraded subscription.

3 Create a new VPS.

a On the VPS Management tab, click New and follow the wizard.

b Set the VPS's hardware parameters you want and click Next.

c Click Finish.

4 Make sure that the *vpses* resource is also became available on the service bus:

a Log in to the POA management node via *ssh*:

```
$ ssh root@10.111.22.14
```

b Run the command:

```
$ curl -E /usr/local/pem/APS/certificates/poa.pem -k
https://localhost:6308/aps/2/resources?aps.type=http://edu.trn/vpsclouds/vpses/1.2
```

c Find the response of the following kind:


```
[
	{
		"aps":
		{
			"type": "http://edu.trn/vpsclouds/vpses/1.2",
			"id": "95885df9-6f0a-4e81-bb70-3af7a3332588",
			"status": "aps:ready",
			"revision": 3,
			"modified": "2013-12-26T12:16:15Z"
		},
		"description": null,
		"hardware":
		{
			"CPU":
			{
				"number": 4
			},
			"diskspace": 32,
			"memory": 512
		},
		"name": "VPS",
		"platform":
		{
			"OS":
			{
				"name": "centos6",
				"version": null
			},
			"arch": null
		},
		"state": "stopped"
	}
]
```

5 Test the management facilities and check the updated status of the *contexts* resource:

a Set the check-box near the new VPS and click Start.

b Log in to the POA management node via *ssh*:

```
$ ssh root@10.111.22.14
```

c Run the following command one more time:

```
$ curl -E /usr/local/pem/APS/certificates/poa.pem -k
https://localhost:6308/aps/2/resources?aps.type=http://edu.trn/vpsclouds/vpses/1.2
```

d Find the response of the following kind:


```
[
	{
		"aps":
		{
			"type": "http://edu.trn/vpsclouds/vpses/1.2",
			"id": "95885df9-6f0a-4e81-bb70-3af7a3332588",
			"status": "aps:ready",
			"revision": 3,
			"modified": "2013-12-26T12:16:15Z"
		},
		"description": null,
		"hardware":
		{
			"CPU":
			{
				"number": 4
			},
			"diskspace": 32,
			"memory": 512
		},
		"name": "VPS",
		"platform":
		{
			"OS":
			{
				"name": "centos6",
				"version": null
			},
			"arch": null
		},
		"state": "Running"
	}
]
```


e Pay attention to the "state" field. In which status the resource is now? _____

f Set the check-box near the new VPS and click Stop.

g Set the check-box near the new VPS and click Delete.

This concludes the stage of the demo APS application development, deployment, and provisioning.

Answer Keys

Modifying Metadata/Building Package

Which buttons it declares? - *cancel* and *next*

Exploring Application Features

In which status the resource is now? - Running

LAB 3

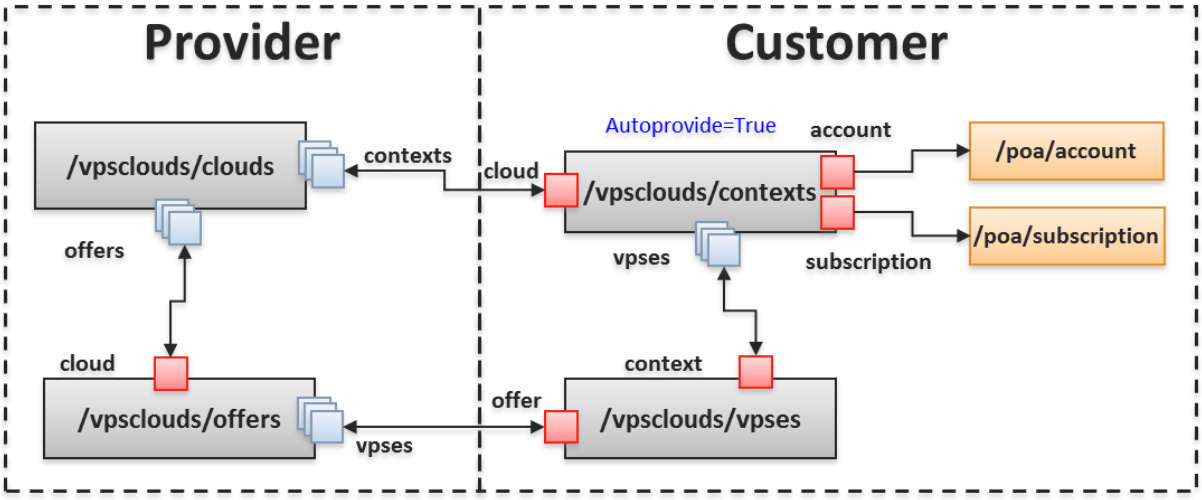
VPS Cloud (Offers) Project

In the following exercises, you will experience APS application development, deployment and provisioning stages to gradually enrich the project with the new feature that enable the provider to use custom UI in the provider control panel to create different configurations for VPSes that will be offered for customers. Customers will be able to choose desired object properties from predefined set of VPSes.

In This Lab

Application Structure

In this exercise, you will build a new application with the following resource model:



This package will contain four APS types. In addition to the types in the *VPS Cloud Basic* project, you will add the resource type presenting VPS offers.

The new resources must be connected with the application from the one side and with customer VPSes from the other side, i.e.:

- The new *offers* type must be provisioned along with the two new pairs of relations:
  - Reference to the *clouds* type (`vpsclouds/clouds`) is required.
  - Reference to the *vpses* type (`vpsclouds/vpses`) is required.
- The resource type *vpses* must be provisioned with the reference to the *offers* (`vpsclouds/offers`) type.
- The resource type *clouds* must be provisioned with the reference to the *offers* (`vpsclouds/offers`) type.

Exercise Directions

In this part of the lab, you will extend the “basic” project adding new feature: offers management - customers will be able to choose offers when purchasing and creating VPSes. You will experience the following development stages:

- Setting an application structure.
- Configuring application scripts.
- Configuring the application metadata.
- Building a package.
- Setting an application end-point.
- Configuring and provisioning an application.

You need to use the following data to perform the set of exercises:

- Previously created project files containing in the `~/Desktop/APS2/Lab-2` directory.
- Files located in the `~/Desktop/APS2/Lab-3` directory:

File	Description
`offers.php`	The script implements the *offers* service
`offers.html`	The file implements an offer management screen on the provider side
`offer.edit.html`	The file implements an offer editing screen on the provider side
`offer.new.html`	The file implements an offer configuration screen on the provider side
`server.new-1.html`	The file implements a first step of VPS creation screen
`server.edit.html`	The file implements a VPS editing screen
`server.new-last.html`	The file implements a last step of VPS creation screen
`newvps.json`	The file provides initial values for a VPS
`newoffer.json`	The file provides initial values for an offer

Below is the overview of actions, you will need to perform to update the “basic” project up to its new version - “offers”:

1 Configure the package structure and build the package:
   - Update the project structure by adding new/modifying existing project files.
   - Modify other scripts for the new project.
   - Modify metadata to embed new service and extend navigation section for new project.
   - Build the application package.
2 Update and configure the application end-point.
3 Deploy and provision the application in POA.
   - Import the application package and update the application instance.
   - Create and configure the new application resources and update the service template.
   - Update the subscription and check the new functionality.

Setting Package Structure

In this exercise, you will create a new project by copying and extending the structure of the existing VPS Cloud project. Follow the directions below.

1 Copy the content of the previously created project to the `Lab-3` directory:

```
$ cp -r ~/Desktop/APS2/Lab-2/* ~/Desktop/APS2/Lab-3
```

2 Update the structure and contents of the new project:

a Move the following HTML pages and JSON files into the `ui` directory:

```
newoffer.json
newvps.json
offers.html
offer.new.html
offer.edit.html
server.new-1.html
server.edit.html
```

```
$ mv ~/Desktop/APS2/Lab-3/*.json ~/Desktop/APS2/Lab-3/ui
$ mv ~/Desktop/APS2/Lab-3/*.html ~/Desktop/APS2/Lab-3/ui
```

NOTE: These files will override those moved from Lab-2.

b From the same source, move the new `offers.php` script to the `scripts` directory.

```
$ mv ~/Desktop/APS2/Lab-3/offers.php ~/Desktop/APS2/Lab-3/scripts
```

Understanding Provisioning Logic

In this exercise, you will learn and explore the content and structure of the additional PHP file containing the management logic of the VPS configuration offers.

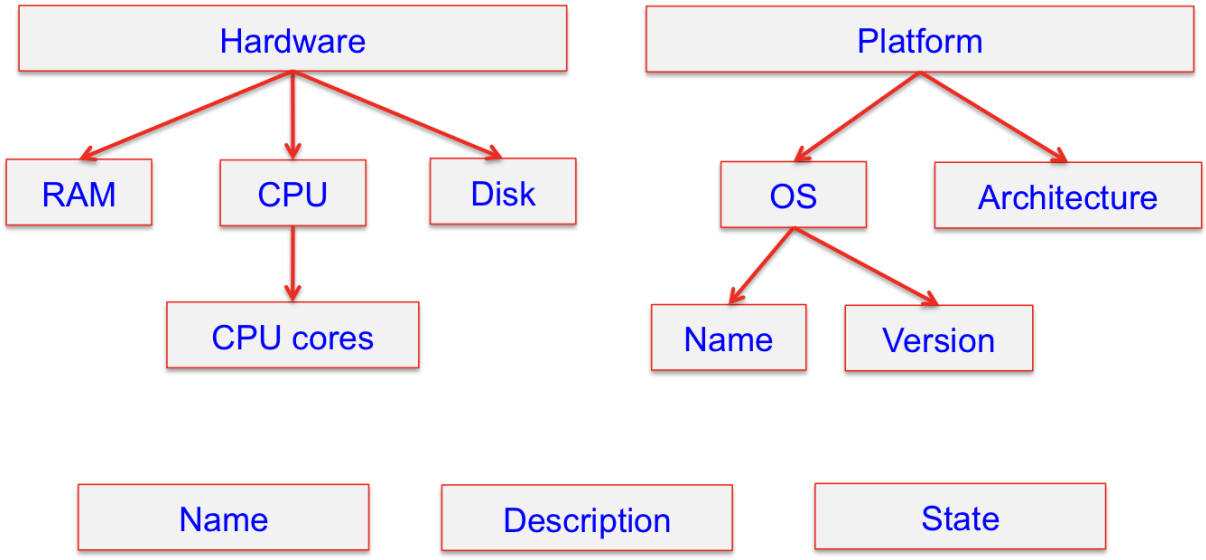
To verify and examine the VPS management business logic:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-3/scripts` project directory.

2 Open the `offers.php` files.

3 Ensure that this file contains the reference to the PHP runtime.

4 Ensure that this file contains the main declarations and definitions reflecting the accepted for this project hierarchical structure.



5 Ensure the `offers` class contains the following declarations and definitions:

- The declarations of the links to the application and vpses.
- What is the type of link to the application? _____
- What is the type of link to vpses? _____
- The functions used to process this link.
- The declarations of the all offer's attributes, e.g. name and description.

Modifying Scripts

In this exercise, you will perform the following operations with the scripts:

- Update resource version in all scripts.
- Modify the `vpses.php` script to add a relation with the new *offers* type.
- Modify the `clouds.php` script to add a relation with the new *offers* type.

Follow the directions below:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-3/scripts` project directory.

2 Open the `contexts.php` file for editing.

a Update the type version as follows.

```
* @type("http://edu.trn/vpsclouds/contexts/1.3")

* @link("http://edu.trn/vpsclouds/clouds/1.3")

* @link("http://edu.trn/vpscloud/vpses/1.3[]")
```

3 Open the `vpses.php` file for editing.

a Update the type version as follows.

```
* @type("http://edu.trn/vpsclouds/vpses/1.3")

* @link("http://edu.trn/vpsclouds/contexts/1.3")
```

b In the *vpses* class definition, add the link to the *offers* type.

```
/**
* @link("http://edu.trn/vpsclouds/offers/1.3")
* @required
*/
public $offer;
```

```
class vpses extends APS\ResourceBase {
    /**
    * @link("http://edu.trn/vpsclouds/offers/1.3")
    * @required
    */
    public $offer;

    /**
     * @link("http://edu.trn/vpsclouds/contexts/1.3")
     * @required
    */
    public $context;
```

4 Open the `clouds.php` file for editing.

a Update the type version as follows.

```
 * @type("http://edu.trn/vpsclouds/clouds/1.3"

 * @link("http://edu.trn/vpsclouds/contexts/1.3[]")
```

b In the *clouds* class definition, add the link to the *offers* type.

```
/**
 * @link("http://edu.trn/vpsclouds/offers/1.3[]")
 */
public $offers;
```

```
class clouds extends APS\ResourceBase {
# Link to collection of offers. Pay attention to [] brackets at the end of the @link line.
        /**
         * @link("http://edu.trn/vpsclouds/offers/1.3[]")
         */
        public $offers;

# Link to collection of contexts. Pay attention to [] brackets at the end of the @link line.
        /**
         * @link("http://edu.trn/vpsclouds/contexts/1.3[]")
         */
        public $contexts;
```

5 Save changes and close the files.

6 Navigate to the `~/Desktop/APS2/Lab-3/ui/` project directory and ensure that the type version in all the files is updated up to 1.3. You can modify the type version in each file automatically using the command:

```
$ find -type f -name \* -exec sed -i -r 's/1\.2/1\.3/g' {} \;
```

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Modifying Metadata/Building Application

In this exercise, you will learn and explore:

- How to update resource version.
- How to reflect the new *offers* service in the application metadata.
- How to define the UI navigation structure in the application metadata.

To perform this:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-3` project directory.

2 Open the `APP-META.xml` file for editing.

a Ensure the application version is updated.

```
<version>1.3</version>
```

b Make sure that the type version is updated to 1.3 throughout the file.

NOTE: Missing an update of the type version could result in an issue afterwards.

c Under t*he Infrastructure Managemen*t category, add the new section that provides the offers management logic.

```
        <navigation id="pcp" label="Offers Management">
                <var name="cloud" type-id="http://edu.trn/vpsclouds/clouds/1.3"/>
                <plugs-to id="http://www.aps-standard.org/ui/application"/>
                <item id="offer" label="Offers">
                        <view id="offers" label="Offers List">
                                <view id="offer.edit" label="Offer
{offer.offername}">
                                        <var name="offer" type-
id="http://edu.trn/vpsclouds/offers/1.3"/>
                                        <controls>
                                                <cancel label="Back to Offers
list"/>
                                                <submit label="Save Changes"/>
                                        </controls>
                                </view>
                                <view id="offer.new" label="New Offer">
                                        <controls>
                                                <cancel label="Back to Offers
list"/>
                                                <submit label="Save Offer"/>
                                        </controls>
                                </view>
                        </view>
                </item>
        </navigation>
```

```
        <category>Infrastructure/Management</category>
    </categories>
    <navigation id="pcp" label="Offers Management">
        <var name="cloud" type-id="http://edu.trn/vpsclouds/clouds/1.3"/>
        <plugs-to id="http://www.aps-standard.org/ui/application"/>
        <item id="offers" label="Offers">
            <view id="offers" label="Offers List">
                <view id="offer.edit" label="Offer {offer.offername}">
                    <var name="offer" type-id="http://edu.trn/vpsclouds/offers/1.3"/>
                    <controls>
                        <cancel label="Back to Offers list"/>
                        <submit label="Save Changes"/>
                    </controls>
                </view>
                <view id="offer.new" label="New Offer">
                    <controls>
                        <cancel label="Back to Offers list"/>
                        <submit label="Save Offer"/>
                    </controls>
                </view>
            </view>
        </item>
    </navigation>
    <navigation id="ccp" label="VPS Management">
```

d Add the new service to the list of services.

```
    <service id="offers">
        <code engine="php" path="scripts/offers.php"/>
        <presentation>
            <name>VPS Parameters</name>
            <summary>Set of VPS parameters</summary>
        </presentation>
    </service>
```

3 Save the file and build the package:

a On the jumper VE, navigate to the directory: `~/Desktop/APS2`.

```
$ cd ~/Desktop/APS2
```

b Run the following command:

```
$ apsbuild Lab-3
```

As a result, you can find the new package at `~/Desktop/APS2/VPS_Cloud-1.3-0.app.zip`

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Setting Application End-Point

In this exercise, you will update the end-point for the extended project on a service host as you did earlier:

1 Copy the updated and new PHP scripts to the `/var/www/html/vpsclouds/` directory, e.g., using the *scp* utility.

a On the jumper VE, run the following command:

```
$ scp ~/Desktop/APS2/Lab-3/scripts/*.php root@10.111.22.16:/var/www/html/vpsclouds
```

2 Update the `.htaccess` file to make HTTP requests for the *offers* service redirected to the `offers.php` script:

a Connect to the service host through *ssh* as you did earlier:

```
$ ssh root@10.111.22.16
```

b In the document root, open the `.htaccess` file for editing, e.g.:

```
# vim /var/www/html/vpsclouds/.htaccess
```

c In this file, add the following rewrite rule.

```
RewriteRule ^offers(|/.*)$ offers.php?q=$1 [L,QSA]
```

d Save changes.

3 For all site files, assign *owner:group* as *apache:apache*:

```
# chown -R apache:apache /var/www/html/vpsclouds
```

4 Restart the *httpd* service on the end-point host:

```
# service httpd restart
```

5 Ensure that the end-point correctly responds to HTTP requests.

a Ensure the output of the following command:

```
# curl http://poadns.edu.trn/vpsclouds/offers/
```

looks like follows:

```
{"code": 404, "type": "RuntimeException", "message": "Not Found: No appropriate
method found for url vpsclouds\/offers"}
```

6 In the end-point directory, clean a type cache in order to force the PHP runtime use the new objects instead of the cached ones:

```
$ rm -rf /var/www/html/vpsclouds/typeCache/
```

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Deploying and Provisioning Application

In this exercise, you will learn and explore how to further deploy and then provision the application:

- Import the updated APS package to POA.
- Upgrade application instance.
- Add resources.
- Upgrade the service template.
- Upgrade the subscription.

To update and make the application ready for sale:

1 Import the new application package as you did earlier, and then refresh the screen.

2 Upgrade the application instance:

   a Open the updated package and click the Upgrade Instances button.

   b Select Upgrade all instances if requested and click Next.

   c Click Finish.

3 Start creating a new offer.

   a On the Instances tab, open the application instance (click the VPS Cloud link).

   b On the Offers tab, click New to create new offers.

4 Fill out the offer's parameters.

   a In the Offer name field, type *VPS Offer Basic*

   b Leave the OS parameter unchanged.

   c For the rest parameters set the values as below:

   CPU Number: 4

   Disk Space: 32

   Ram: 512

   d Click Save Offer.

5 Make sure that the *offers* resource is already available on the service bus:

   a Log in to the POA management node via *ssh*:

```
$ ssh root@10.111.22.14
```

   b Run the command:

```
# curl -E /usr/local/pem/APS/certificates/poa.pem -k
https://localhost:6308/aps/2/resources?aps.type=http://edu.trn/vpsclouds/offers/1.3
```

c Find the response of the following kind:


```
[
	{
		"aps":
		{
			"type": "http://edu.trn/vpsclouds/offers/1.3",
			"id": "f0e2976f-49c6-452a-816f-47ced8eb5884",
			"status": "aps:ready",
			"revision": 3,
			"modified": "2013-12-27T09:00:26Z"
		},
		"hardware":
		{
			"CPU":
			{
				"number": 4
			},
			"diskspace": 32,
			"memory": 2048
		},
		"offerdscr": null,
		"offername": "VPS Offer Basic",
		"platform":
		{
			"OS":
			{
				"name": "centos6",
				"version": null
			},
			"arch": null
		}
	}
]
```


6 Create two additional offers. Use the following values for their parameters:

- *VPS Offer Silver* (CPU cores: 8; Diskspace: 64; RAM:4096)
- *VPS Offer Gold* (CPU cores: 16; Diskspace: 100; RAM:8192)

7 For VPS offers, create the additional resource types.

a In the POA provider control panel, navigate Services > Applications.

b Open a profile of the APS package and switch to the Resource Types tab.

c Click Create and select *Application Service Reference*.

d In the Name field, enter *VPS Cloud - Offer Basic*

e Select the *VPS Parameters* as the APS type.

f Click on the instance ID for *VPS Offer Basic*.

g Click Finish.

h Create two more resource types for the rest VPS offers:
VPS Cloud - Offer Silver
VPS Cloud - Offer Gold

8 Update a service template for the VPS Cloud application.

a Navigate to Products > Service Templates.

b Open the profile of the *VPS Cloud Services* service template.

c In the service template profile, open the Resources tab.

d Click Add Resources to add the new resources. Do not limit the *offer* resources.

e Click Submit.

9 For *Training Customer*, upgrade the subscription:
   a Navigate to Operations > Subscriptions.
   b Open the profile of the *VPS Cloud Services* subscription.
   c Click the Upgrade button.

As a result, new application feature becomes available in the customer CP.

Exploring Application Features

In this exercise, you will explore the new application features. Follow the directions below.

1 Login to the customer control panel.
   a On the Managed By tab, click *Login as Customer*.
2 Verify the services available for the *Training Customer*.
   a In the subscription selector, select the upgraded subscription.
3 Create a new VPS.
   a On the VPS Management tab, click New and follow the wizard.
   b In the Name field, type *VPS Basic*.
   c In the Offers section, select the VPS Offer Basic offer.
   d Try to change the offer's hardware parameters within their limits.

   NOTE: Here, you will find the limits set during offers' parameters configuration.

   e Set the VPS's hardware parameters you want and click Next.
   f Click Finish.
4 Create another VPS using the other offer type.
5 Test the new management facilities.
   a Open a profile of the VPS.
   b Change the parameters and save changes.

This concludes the stage of the extended demo APS application development, deployment, and provisioning.

Answer Keys

Understanding Provisioning Logic

What is the type of link to the application? - *singular-required*

What is the type of link to vpses? - *collection-optional*

LAB 4

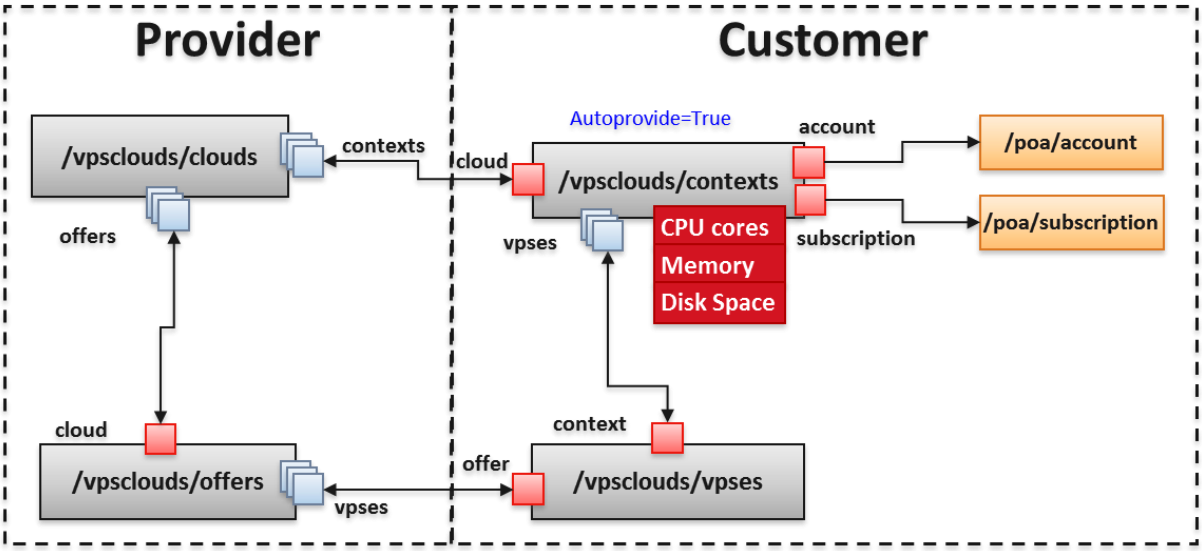
VPS Cloud (Counters) Project

In this set of exercises, you will create the new project on the basis of the previous one and enrich it with the new feature that will add the ability to monitor resource consumption.

In This Lab

Application Structure

In this exercise, you will build a new application with the following resource model:



This package will contain four APS types, i.e., we do not need to add any new types here. We are going to count CPU, RAM, and disk usage. Therefore, the respective properties and functions must be added to the VPSes provisioning logic.

The *VPS* type must be provisioned along with the following resource counter properties:

- CPU
- RAM
- Diskspace

Exercise Directions

You will gain new practical skills in adding counters into your project that is needed to set and control resource limits and collect resource usage. You will experience the following development stages:

- Setting an application structure.
- Configuring application scripts.
- Configuring the application metadata.
- Building a package.
- Setting an application end-point.
- Configuring and provisioning an application.

You need to use the following data to perform the set of exercises:

- Previously created project files containing in the `~/Desktop/APS2/Lab-3` directory.
- Files located in the `~/Desktop/APS2/Lab-4` directory:

File	Description
`contexts.php`	The script implements the *contexts* service
`vpses.php`	The script implements the *vpses* service
`counters.html`	The file implements a resource usage statistics screen

Below is an overview of actions you are going to perform to assemble the VPS Cloud "counters" package and provision the APS application:

1 Configure the package structure and build the package:
   - Update the project structure by adding new/modifying existing project files.
   - Modify the metadata file and scripts for the new project.
   - Build the application package.
2 Update and configure the application end-point.
3 Deploy and provision the application in POA.
   - Import the application package and update the application instance.
   - Create and configure the new application resources and update the service template.
   - Update the subscription and check the new functionality.

Setting Package Structure

In this exercise, you will create a new project by copying and extending the structure of the existing project. Follow the directions below.

1 On the jumper VE, copy the previously created project to the `Lab-4` directory:

```
$ cp -r ~/Desktop/APS2/Lab-3/* ~/Desktop/APS2/Lab-4
```

2 Update the structure and contents of the new project:

a Move the `counters.html` file into the `ui` directory:

```
$ mv ~/Desktop/APS2/Lab-4/*.html ~/Desktop/APS2/Lab-4/ui
```

b Move the `vpses.php` page into the `scripts` directory:

```
$ mv ~/Desktop/APS2/Lab-4/*.php ~/Desktop/APS2/Lab-4/scripts
```

Understanding Provisioning Logic

In this exercise, you will learn and explore:

- The additional content of the `contexts.php` file defining the counters.

Follow the directions below:

1 On the jumper VE, navigate to the project directory:

```
$ cd ~/Desktop/APS2/Lab-4/scripts
```

2 Open the new version of the `contexts.php` script.

3 Ensure the script contains the following declarations and definitions:

- The declarations of counters for each hardware component.
- The definition and implementation of the function that returns the resource usage (the *retrieve()* function).

Modifying Metadata and Scripts/Building Application

In this exercise, you will learn and explore:

- How to reflect the new counters UI feature in the application metadata.
- How to update resource version in all scripts.

Follow the directions below:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-4` project directory.

2 Open `APP-META.xml` for editing.

a Ensure the application version is updated.
`<version>1.4</version>`

b Make sure that the type version is updated to 1.4.

c Under the *VPS Management* navigation section, add a new item as follows.

```
<item id="counters" label="Resource Usage">
      <view id="counters" label="Resource Usage"/>
</item>
```

```
                </view>
            </item>
            <item id="counters" label="Resource Usage">
                <view id="counters" label="Resource Usage"/>
            </item>
        </navigation>
    </presentation>
    <license-agreement must-accept="true">
```

3 Save changes.

4 Navigate to the `~/Desktop/APS2/Lab-4` project directory and ensure that the type version in all files is updated up to 1.4.

```
$ cd ~Desktop/APS2/Lab-4
$ find -type f -name \* -exec sed -i -r 's/1\.3/1\.4/g' {} \;
```

5 Build the package:

a On the jumper VE, navigate to the following directory: `~/Desktop/APS2`.

```
$ cd ~/Desktop/APS2/
```

b Run the following command:

```
$ apsbuild Lab-4
```

As a result, you can find the new package at `~/Desktop/APS2/VPS_Cloud-1.4-0.app.zip`

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Setting Application End-Point

In this exercise, you will set the end-point for the extended project on a service host as you did earlier:

1 Copy the updated and new PHP scripts to the `/var/www/html/vpsclouds/` directory, e.g., using the *scp* utility.

   a On the jumper VE, run the following command:

```
$ scp ~/Desktop/APS2/Lab-4/scripts/*.php root@10.111.22.16:/var/www/html/vpsclouds
```

2 For all site files, ensure the *owner:group* is set as *apache:apache*.

3 Restart the *httpd* service on the end-point host if needed

4 In the end-point directory, clean a type cache in order to force the PHP runtime use the new objects instead of the cached ones:

```
$ rm -rf /var/www/html/vpsclouds/typeCache/
```

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Deploying and Provisioning Application

In this exercise, you will learn and explore how to further deploy and then provision the application:

- Import the updated APS package to POA.
- Upgrade application instance.
- Add resources.
- Upgrade the service template.
- Upgrade the subscription.

To update and make the application ready for sale:

1 Import the new application package as you did earlier, and then refresh the screen.

2 Upgrade the application instance:
   a Open the updated package and click the Upgrade Instances button.
   b Select Upgrade all instances if requested and click Next.
   c Click Finish.

3 For the CPU resource counter, create the additional resource type.
   a In the POA provider control panel, navigate Services > Applications.
   b Open a profile of the APS package and switch to the Resource Types tab.
   c Click Create and select *Application Counter (unit)*.
   d In the Name field, enter *VPS Cloud - CPU Usage*
   e Select the *cpuusage* resource as the APS type.
   f Click Finish.

4 For the memory resource counter, create the additional resource type.
   a Click Create and select *Application Counter (KB)*.
   b In the Name field, enter *VPS Cloud - Memory Usage*
   c Select the *memoryusage* resource as the APS type.
   d Click Finish.

5 For the disk space resource counter, create the additional resource type.
   a Click Create and select *Application Counter (KB)*.
   b For the resource type, use the *VPS Cloud - Disk Usage* name.
   c Select the *diskusage* resource as the APS type.
   d Click Finish.

6 Update a service template for the application.
   a Navigate to Products > Service Templates.
   b Open the profile of the *VPS Cloud Services* service template.
   c In the service template profile, open the Resources tab.
   d Click Add Resources to add the new resources. For the counter resources, set the unlimited value.
   e Click Submit.

7 For *Training Customer*, upgrade the subscription:
   a Navigate to Operations > Subscriptions.
   b Open the profile of the *VPS Cloud Services* subscription.
   c Click the Upgrade button.

As a result, new application feature becomes availbale in the customer CP

Exploring Application Features

To ensure the provisioning of the APS application and test its new functionality:

1 Login to the customer control panel on behalf of *Training Customer*.
2 Ensure the *VPS Cloud Services* subscription is selected.
3 Create a new VPS on the basis of any of the offers.
4 Force the resource usage collection in POA.
   a Open the provider POA control panel.
   b Navigate to Operations > Tasks.
   c On the Periodic tab, select the check box for the task named *Synchronize resource usage for instances of APS applications.*
   d On the upper pane, click Run Tasks and then click OK in the pop-up window to confirm the operation.
5 Verify the VPS resource usage
   a Open the customer control panel.
   b Open the VPS Management tab and click the Resource Usage tab.

   NOTE: It is the current usage of the resources for the VPS.

6 Verify the resource usage total for the subscription.
   a On the Home tab, click the Resource Usage link.
   b Note the total usage for each counter resource.

This concludes the stage of the demo APS application development, deployment, and provisioning.

LAB 5

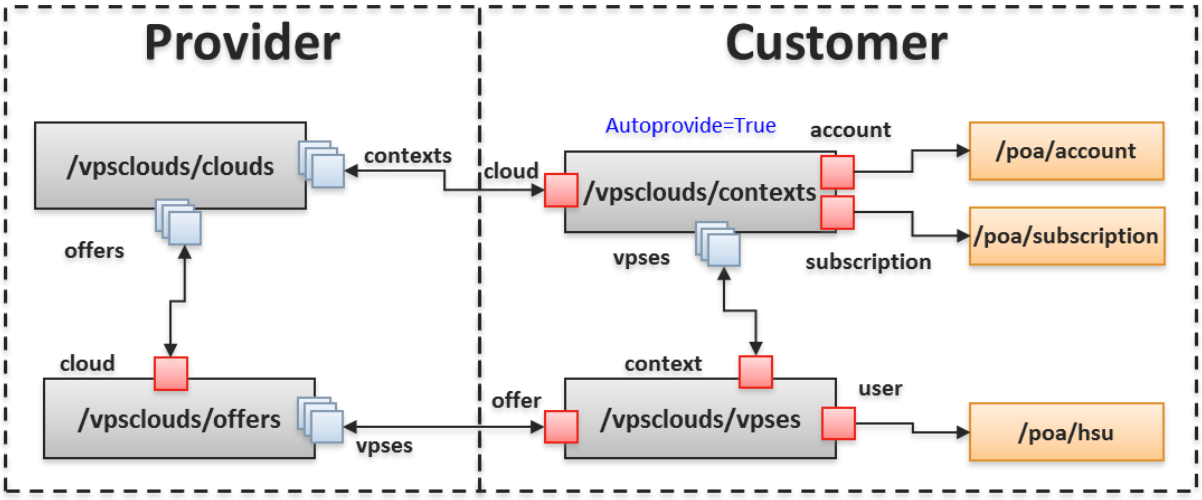
VPS Cloud (Users) Project

In this set of exercises, you will create the new project on the basis of the previous and enrich it with the new feature that will allow a customer to create users and assign them as VPS owners.

In This Lab

Application Structure

In this exercise, you will build a new application with the following resource model:



This package will contain four APS types, i.e., we do not need to add any new types here. In this resource model, the *vpses* service will require a relation with a POA user, i.e.:

The *vpses* type must be provisioned along with the new relation:

- Reference to the *user* type (`poa/user`) is required.

Exercise Directions

By completing this set of exercises, you will gain practical skills in embedding application UI into MyCP:

- Setting an application structure.
- Configuring application scripts.
- Configuring the application metadata.
- Building a package.
- Setting an application end-point.
- Configuring and provisioning an application.

You need to use the following data to perform the set of exercises:

- Previously created project files containing in the `~/Desktop/APS2/Lab-4` directory.
- Files located in the `~/Desktop/APS2/Lab-5` directory:

File	Description
`myservers.html`	The file implements a MyCP screen
`servers.html`	The file implements a VPS management screen
`server.edit.html`	The file implements a VPS editing screen
`server.new-1.html`	The file implements a first step of VPS creation screen
`server.new-last.html`	The file implements a last step of VPS creation screen
`newvps.json`	The file provides initial values for a VPS
`getUserList.js`	The file retrieve a list of users

Below is an overview of actions you are going to perform to assemble the VPS Cloud “users” package and provision the application:

1 Configure the package structure and build the package:
   - Copy the previously created project files to the lab directory.
   - Add the above mentioned files to the `ui` directory.
   - Modify the metadata file and scripts for the new project.
   - Build the application package.
2 Set and configure the application end-point.
3 Update and configure the application end-point.
4 Deploy and provision the application in POA.
   - Import the application package and update the application instance.
   - Create and configure the new application resources and update the service template.
   - Update the subscription and check the new functionality.

Setting Package Structure

In this exercise, you will create a new project by copying and extending the structure and contents of the existing project. Follow the directions below.

1 On the jumper VE, copy the previously created project to the `Lab-5` directory:

```
$ cp -r ~/Desktop/APS2/Lab-4/* ~/Desktop/APS2/Lab-5
```

2 Update the structure and contents of the new project:

a Move `*.html`, `*.js`, and `*.json` files into the `ui` directory:

```
$ mv ~/Desktop/APS2/Lab-5/*.json ~/Desktop/APS2/Lab-5/ui
$ mv ~/Desktop/APS2/Lab-5/*.js ~/Desktop/APS2/Lab-5/ui
$ mv ~/Desktop/APS2/Lab-5/*.html ~/Desktop/APS2/Lab-5/ui
```

NOTE: These files will override those moved from Lab-4.

Modifying Metadata and Scripts/Building Application

In this exercise, you will learn and explore:

- How to define the UI navigation structure in the application metadata.
- How to modify the `vpses.php` script in order to add a relation with *users*.

Also, you will update resource version in all scripts.

Follow the directions:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-5` project directory.

2 Open `APP-META.xml` for editing.

a Ensure the application version is updated.

```
<version>1.5</version>
```

b Make sure that the version of all types changed from 1.4 to 1.5.

c Under the *VPS Management* navigation section, add a new item as follows.

```
        <navigation id="mycp" label="Virtual Servers">
                <var name="user" type-id="http://aps-standard.org/types/core/service-
user/1.0"/>
                <plugs-to id="http://www.aps-standard.org/ui/user"/>
                <item id="myservers" label="Virtual Servers">
                    <view id="myservers" label="Own VPSes"/>
                </item>
        </navigation>
```

```
                </item>
                <item id="counters" label="Resource Usage">
                    <view id="counters" label="Resource Usage"/>
                </item>
            </navigation>
            <navigation id="mycp" label="Virtual Servers">
                <var name="user" type-id="http://aps-standard.org/types/core/service-user/1.0"/>
                <plugs-to id="http://www.aps-standard.org/ui/user"/>
                <item id="myservers" label="Virtual Servers">
                    <view id="myservers" label="Own VPSes"/>
                </item>
            </navigation>
        </presentation>
```

3 Save changes. The new navigation item in MyCP will reflect the above changes.

4 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-5/scripts` project directory.

5 Open the `vpses.php` file for editing.

a Update the type version as follows.

```
     * @type("http://edu.trn/vpsclouds/vpses/1.5")

     * @link("http://edu.trn/vpsclouds/contexts/1.5")

     * @link("http://edu.trn/vpsclouds/offers/1.5")
```

b In the *vpses* class, add the link to the *users* type.

```
    /**
     * @link("http://aps-standard.org/types/core/service-user/1.0")
     * @required
    */
    public $user;
```

```
/**
 * @link("http://edu.trn/vpsclouds/offers/1.5")
 * @required
 */
public $offer;

/**
 * @link("http://aps-standard.org/types/core/service-user/1.0")
 * @required
*/
public $user;
```

c Add the VPS owner name as the VPS class property.

```
/**
* @type("string")
* @title("User Name")
* @description("VPS Owner Name")
*/
public $userName;
```

d Implement the custom start() function and allow users to access it.

```
/**
* @verb(GET)
* @path("/start")
* @return(string,text/json)
* @access(referrer, true)
*/
public function start() {
        $this->state = 'Running';
        $apsc = \APS\Request::getController();
        $apsc->updateResource($this);
  }
```

e Same way, implement the custom `stop()` function and allow users to access it.

```
/**
* @verb(GET)
* @path("/stop")
* @return(string,text/json)
* @access(referrer, true)
*/
public function stop() {
        $this->state = 'Stopped';
        $apsc = \APS\Request::getController();
        $apsc->updateResource($this);
}
```

6 Save changes.

7 Make sure that the type version is updated to 1.5 in all files contained in the `/scripts` and `/ui` directories.

8 Build the package:

a On the jumper VE, navigate to the following directory: `~/Desktop/APS2`.

```
$ cd ~/Desktop/APS2
```

b Run the following command:

```
$ apsbuild Lab-5
```

As a result, you can find the new package at `~/Desktop/APS2/VPS_Cloud-1.5-0.app.zip`

NOTE: Mistakes and typos occurred while performing this exercise can result in any issues. See the *Troubleshooting* section at the end of this guide in order to locate how to fix them.

Deploying and Provisioning Application

Before you continue on to the last set of exercises, perform the following actions on your own:

1 Refer to the previous section to update the end-point for the new project on the service host.

2 Refer to the previous section to update the instance.

NOTE: No additional resource types are to be added.

Exploring Application Features

In these exercises, you will learn and explore the new features of the of the *VPS Cloud Services* subscription.

Preliminary Actions

Prior to testing the new functionality of the application, you will need to create a domain subscription, which is required for creating users in POA for the *Training Customer* account. Follow the directions below:

1 In the POA provider control panel, create a service template for a domain hosting.
   a Navigate to Products > Service Templates and click Add New Service Template.
   b In the Name field, type DNS Hosting.
   c Set the Autoprovisioning check-box and click Next.
   d In the resource field, use the "`DNS*`" as a search pattern to locate the DNS Hosting resource.
   e For this resource, set the check-box and click Next.
   f Leave all the limits as is, click Next and then click Finish.

2 Create a domain subscription for *Training Customer*.
   a Open a profile of the newly created template.
   b On the General tab, click Activate.
   c On the Subscriptions tab, click Create New Subscription.
   d Select *Training Customer*.
   e Leave the resource values as is and click Next.
   f Click Finish.

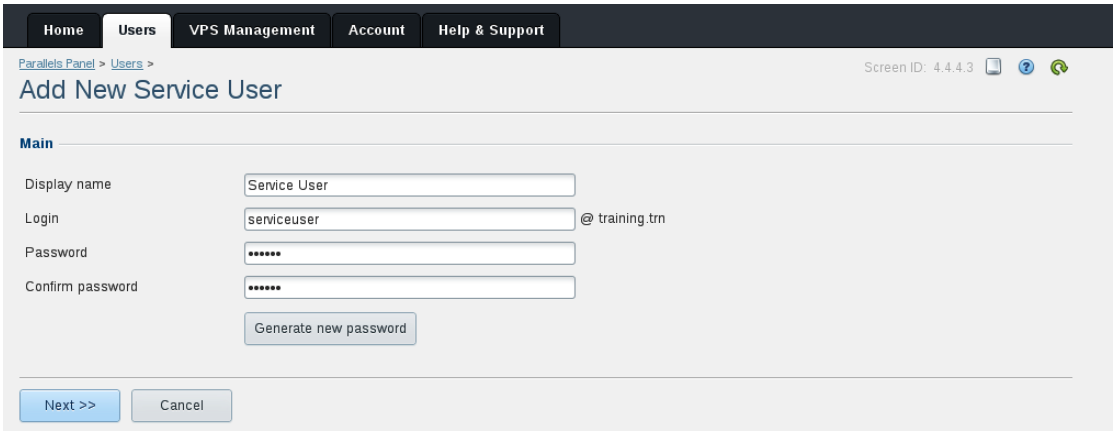
3 Open the customer control panel.
   a Navigate to Operations > Customers.
   b Open a profile of *Training Customer*.
   c On the General tab, click the Staff Members sub-tab.
   d Click the Login as Customer link.

4 Create a user for *Training Customer*.

a On the Users tab, click Add New Service User.

b Fill out the appeared form and click Next.

c Click Finish.



d Repeat the actions to create a second user.

New service users become available in the customer control panel.

Managing VPS

To provision the application and test its new features:

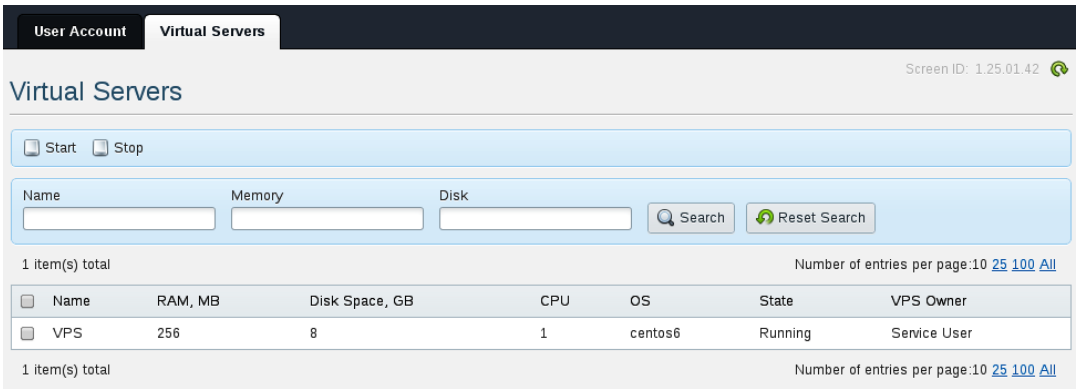
1 In customer control panel, create a new VPS on the basis of an offer. (Select a user for the VPS being created.)

2 Explore the management facilities in MyCP.

a On the Users tab, click the Login link for the user you have chosen on the previous step.

b In MyCP, open the Virtual Servers tab and select the VPS.

c Click Start to change the VPS's status.



This concludes the stage of the demo APS application development, deployment, and provisioning.

Troubleshooting

The following troubleshooting scenarios cover the common cases, which can arise during the lab flow.

Case 1

In this scenario, we will consider the case when the end-point configuration check fails.

- Symptom
  - As a result of the command:

```
# curl http://poadns.edu.trn/vpsclouds/contexts/
```

  - you see the message like this:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>500 Internal Server Error</title>
</head><body>
<h1>Internal Server Error</h1>
<p>The server encountered an internal error or misconfiguration and was
unable to complete your request.</p>
<p>Please contact the server administrator, root@localhost and inform them of
the time the error occurred,
and anything you might have done that may have caused the error.</p>
<p>More information about this error may be available in the server error
log.</p>
<hr>
<address>Apache/2.2.15 (CentOS) Server at poadns.edu.trn Port 80</address>
</body></html>
```

  - or:

```
<!DOCTYPE HTML PUBLIC "-//IETF//DTD HTML 2.0//EN">
<html><head>
<title>404 Internal Server Error</title>
</head><body>
<h1>Not Found</h1>
<>The requested URL /vpsclouds/contexts/ was not found on this server.</p>
<hr>
<address>Apache/2.2.15 (CentOS) Server at poadns.edu.trn Port 80<address>
</body><html>
```

- Cause
  - This mistake can arise due to improper end-point configuration. For example, because of any mistake in the `.htaccess` file.
- Resolution
  - Follow the directions below:

1 Check all the configuration steps one more time.

Especially, pay attention on the `.htaccess` file, i.e, check mistakes in rewrite rules or any typos.

Case 2

In this scenario, we will consider the case when the package creation fails.

- Symptom
  - As a result of the application instance installation, you see the message like this:

```
$ apsbuild Lab-x
Building package.......
Generating APP-LIST.xml.......
Validating package.......
Package format version - 2.0
schemas/contexts.schema.gen:1: Error: No content to map due to end-of-input
Validation complete: 1 Error(s), 0 Warning(s)
Error: Failed to build package. Package is not valid.
```

- Cause
  - This mistake can arise due to any typo or incorrect definitions on annotation blocks so the APS Runtime cannot parse them, in one of a script during their modification.
- Resolution
  - Follow the directions below:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-x/scripts/` project directory.

2 Check all the `*.php` files trying to find any mistakes such as the following:

```
* @type("http://edu.trn/vpsclouds/contexts/1.x
```

instead of

```
* @type("http://edu.trn/vpsclouds/contexts/1.x")
```

After the issue will be found, recreate the package.

Note: You can also find the similar case at `http://kb.parallels.com/en/117500`.

Case 3

In this scenario, we will consider the case when the application instance installation fails.

- Symptom
  - As a result of the application instance installation, you see the message like this:

```
      [APSC] REST Application returns error code=500 type=Exception:
file_put_contents(/var/www/html/scripts/certificate.pem): failed to open stream:
Permission denied at      /usr/share/aps/php/aps/2/types.php:215.
```

- Cause
  - POA sends a certificate to endpoint but it cannot be stored because webserver does not have write permissions on the folder where endpoint scripts are stored. This mistake could be done during end-point configuration.
- Resolution
  - Follow the directions below:

1 Log in to the end-point host via *ssh*:

```
$ ssh root@10.111.22.16
```

2 For end-point scripts, assign *owner:group* as *apache:apache*:

```
# chown -R apache:apache /var/www/html/vpsclouds
```

After the issue will be found, reinstall the instance.

Note: You can also find this case at `http://kb.parallels.com/en/117497`, and similar case at `http://kb.parallels.com/en/117495`.

Case 4

In this scenario, we will consider the case when the application instance configuration fails.

- Symptom
  - When you try to configure offers in the application instance in PCP you do not see the Offer tab.
- Cause
  - Mistake can be caused by not updated type versions during scripts modification.
- Resolution
  - Follow the directions below:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-x/scripts` project directory.
   - Check all the type versions in all the scripts.

2 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-x/ui` project directory.
   - Check all the type versions in all the scripts.

After mistakes were found and corrected successfully, you need to:

1 Update the release version in the `APP-META.xml` file to the next value.

```
    <release>x</release>
```

2 Rebuild the package.

3 Import the updated package.

4 Upgrade the instance.

Case 5

In this scenario, we will consider the case when the offers configuration in application instance fails.

- Symptom
  - When you try to configure offers in the application instance in PCP you see an error message that looks like:

```
      RequestError: Unable to load /aps/2/resources/be2a6cfb-a9e8-4258-9901-
663333bae7a3/offers/ status: 500
      Appropriate HTTP method POST is not found in the resource type declaration.
```

- Cause
  - Mistake can arise because the link to the *offers* type was not added to the `clouds.php` file during scripts modification.
- Resolution
  - Follow the directions below:

1 On the jumper VE, navigate to the `~/Desktop/APS2/Lab-x/scripts` project directory.
   - Open the `clouds.php` file for editing.
   - Check if the link to the *offers* type was added.

After the link was added successfully, you need to:

1 Update the release version in the `APP-META.xml` file to the next value.

```
      <release>x</release>
```

2 Rebuild the package.

3 Import the updated package.

Upgrade the instance.

Case 6

In this scenario, we will consider the case when the VPS creation for a user fails.

- Symptom
  - When you try to create a VPS for a user it fails for any reason.
- Cause
  - Mistake can arise because a type cache was not cleaned in the end-point directory.
- Resolution
  - Follow the directions below:

1 Connect to the service host through *ssh*:

```
$ ssh root@10.111.22.16
```

2 Clean a type cache:

```
$ rm -rf /var/www/html/vpsclouds/typeCache/
```

Note: You can also find the similar case at `http://kb.parallels.com/en/117616`.