The Aculab Cloud Guide to IVR

How to create an IVR application in 7 simple steps

For this code walkthrough have used the PHP wrapper for Aculab Cloud's REST API.

Code examples for this IVR Demo application using other languages can be found on the Aculab Cloud website:

https://cloud.aculab.com/documents/IVRApplication.html

Writing an IVR system in Aculab Cloud

Aculab Cloud provides a simple, easy to learn API for rapid development of telephony applications.

Using the features provided by Aculab Cloud you can rapidly develop an Interactive Voice Response (IVR) system which could be used to automatically provide services or direct callers to the correct department to meet their needs. Such systems can increase both staff productivity and customer satisfaction.

For this example we are going to build an IVR demo application for a fictitious delivery company, 'Aculab Cloud Logistics'. We'll be enabling the IVR to perform a series of typical tasks:

- Find out company opening times
- Locate your package
- Speak to an agent if you have a more complex enquiry

This walkthrough will provide you with all the information you need to successfully create your first IVR system on Aculab Cloud. Let's get started!

Before we start

The structure of our IVR system and example code

Where to find code examples

For this code walkthrough we are using the PHP wrapper for Aculab Cloud's REST API. Code examples for this IVR demo application using other languages can be found on the Aculab Cloud website:

https://www.aculab.com/cloud/demos-all/ivr

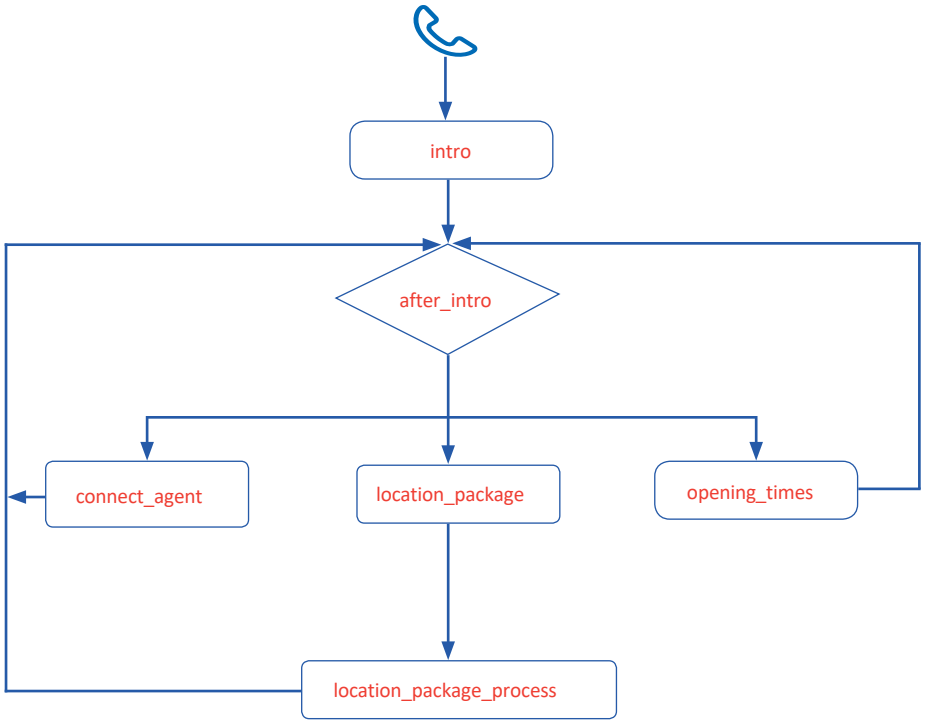

Figure 1: IVR web page flow

The structure of our IVR system

The IVR system consists of a number of web pages.

Each web page is made up of a list of actions which we are asking Aculab Cloud to perform. An action could be something such as playing a message, or requesting user input.

Once Aculab Cloud has completed the list of actions provided, it requests the next page from your webserver. The web page Aculab Cloud requests may differ depending on the result of an action. For example, pressing different keys during a RunMenu action will result in a different page being requested.

Figure 1 shows the web pages which make up our IVR, and the order in which they will be requested by Aculab Cloud.

In the subsequent sections of this document we shall walkthrough the construction of each web page in turn.

Page 1: intro

Creating an IVR system in Aculab Cloud

Page structure

The structure of each web page looks similar to the code below, which is for the first page in our IVR application, intro.php.

```
<?php
declare(encoding='UTF-8');
require_once __DIR__.'/vendor/autoload.php';

use \Aculab\TelephonyRestAPI\Response;
use \Aculab\TelephonyRestAPI\Play;
use \Aculab\TelephonyRestAPI\Redirect;

$response = new Response();
$response->addAction(Play::sayText('Hello and welcome to the Aculab Cloud Logistics IVR system '));
$response->addAction(new Redirect('after_intro.php'));

print $response;
?>
```

Figure 2: intro.php

Page components

Looking at the code, the first thing to do on each page is set the character encoding to Unicode, which allows for international characters in any text. For this example, the required php libraries have been installed using Composer (https://getcomposer.org/). Therefore, we also auto-load any classes installed via composer.

```
declare(encoding='UTF-8');
require_once __DIR__.'/vendor/autoload.php';
```

Next, import the Aculab classes to be used within the web page:

```
use \aculab\telephonyrestapi\response;
use \aculab\telephonyrestapi\play;
use \aculab\telephonyrestapi\redirect;
```

Create a new response object, to which will be added the actions we want Aculab Cloud to perform:

```
$response = new Response();
```

Page 1: intro

Creating an IVR system in Aculab Cloud

Next, create each action we want Aculab Cloud to perform, and add it to the response. In this case, Aculab Cloud will be required to speak some text to our caller:

```
$response->addAction(Play::sayText('Hello and welcome to the Aculab Cloud
Logistics IVR system '));
```

Then, tell Aculab Cloud the next web page it should request after completing the actions initially requested:

```
$response->addAction(new Redirect('after_intro.php'));
```

Finally, send the list of actions to Aculab Cloud by outputting them to standard output:

```
print $response;
```

A similar structure is seen throughout the web pages within the IVR application.

Page 2: after_intro

Creating an IVR system in Aculab Cloud

Page structure

The page requested once the actions in the intro page have been executed is the after_intro.php page.

This page provides a number of menu options to the caller, and requires an input via a telephone keypad. Depending on the user input, the appropriate page is requested by Aculab Cloud.

```
<?php
declare(encoding='UTF-8');
require_once __DIR__.'/vendor/autoload.php';

use \aculab\telephonyrestapi\response;
use \aculab\telephonyrestapi\play;
use \aculab\telephonyrestapi\runmenu;

$response = new response();
$response->addAction(Play::sayText('Main menu. '));
$menu_text = Play::sayText("Press 1 to hear our store location and opening times, press 2
        to find the location of your package, or press 3 to talk to one of our highly
        experienced and friendly agents. At any time, just hit star to hear the options
        again.");
$main_menu = new RunMenu ($menu_text);
$main_menu->setPrompt($menu_text);
$main_menu->addMenuOption ('1', 'opening_times.php');
$main_menu->addMenuOption ('2', 'location_package.php');
$main_menu->addMenuOption ('3', 'connect_agent.php');
$response->addAction($main_menu);

print $response;
?>
```

Figure 3: after_intro.php

Page components

This page contains a number of actions. First, some text is played to the caller, announcing the main menu:

```
$response->addAction(Play::sayText('Main menu. '));
```

Page 2: after_intro

Creating an IVR system in Aculab Cloud

Secondly, a menu which consists of the text containing the menu options is played to the caller:

```
$response->addAction(Play::sayText('Main menu. '));
$menu_text = Play::sayText("Press 1 to hear our store location and opening times, press 2
        to find the location of your package, or press 3 to talk to one of our highly
        experienced and friendly agents. At any time, just hit star to hear the options
        again.");
$main_menu = new RunMenu ($menu_text);
$main_menu->setPrompt($menu_text);
```

Code is required to handle the selected option and present the appropriate page depending on the input provided by the caller:

```
$main_menu->addMenuOption ('1', 'opening_times.php');
$main_menu->addMenuOption ('2', 'location_package.php');
$main_menu->addMenuOption ('3', 'connect_agent.php');
```

By default, if a caller hits the '*' key on their phone, or enters an invalid menu option, the menu options will be replayed to the caller.

Menu Option Pages 1: opening_times

Creating an IVR system in Aculab Cloud

Page structure

The first menu option is to tell the caller the company opening times. This is easily achieved by saying some text to the caller. In a real world IVR application, the opening times may be stored in a database, and your application could decide what times to say to the caller, based on a look up in that database.

```
<?php
declare (encoding='UTF-8');
require_once __DIR__.'/vendor/autoload.php';

use \aculab\telephonyrestapi\response;
use \aculab\telephonyrestapi\play;
use \aculab\telephonyrestapi\redirect;

$response = new response();
$response->addAction(Play::sayText('Aculab Cloud Logistics are located at 2 Bramley '.
        ' Road, Milton Keynes, MK1 1PT, United Kingdom. Our working hours are Monday '
to Friday, 9 a.m. to 5 p.m. '));
$response->addAction(new Redirect('after_intro.php'));

print $response;
?>
```

Figure 4: opening_times.php

After playing the opening times, Aculab Cloud is requested to direct the caller back to the main menu provided in after_info.php.

```
$response->addAction(new Redirect('after_intro.php'));
```

Menu Option Pages 2: location_package

Creating an IVR system in Aculab Cloud

Page structure

The next menu option is for finding the location of a package. The user is asked to enter a 4-digit package number, and then Aculab Cloud informs them of the location of their package.

This process is **split into two web pages -** location_package.php and location_package_process.php

```
<?php
declare (encoding='UTF-8');
require_once __DIR__.'/vendor/autoload.php';


use \aculab\telephonyrestapi\response;
use \aculab\telephonyrestapi\play;
use \aculab\telephonyrestapi\getnumber;


$response = new Response();
$get_number = new GetNumber('location_package_process.php');
$get_number->setPrompt(Play::sayText('Please enter your four digit package '.'number to
locate. For demonstration purposes, you can enter any number you'.'like. '));
$get_number->setDigitCount (4);
$response->addAction ($get_number);


print $response;
?>
```

Figure 5: location_package.php

Page components

In location_package.php an action is sent to Aculab Cloud, asking the caller to enter a 4-digit number. This is done using the GetNumber action:

```
$get_number = new GetNumber('location_package_process.php');
$get_number->setPrompt(Play::sayText('Please enter your four digit package '.
'number to locate. For demonstration purposes, you can enter any number you '.
'like. '));
$get_number->setDigitCount (4);
$response->addAction ($get_number);
```

Menu Option Pages 2: location_package_process

Creating an IVR system in Aculab Cloud

```
<?php
declare (encoding='UTF-8');
require_once __DIR__.'/vendor/autoload.php';

use \aculab\telephonyrestapi\InstanceInfo;
use \aculab\telephonyrestapi\response;
use \aculab\telephonyrestapi\play;
use \aculab\telephonyrestapi\getnumberresult;
use \aculab\telephonyrestapi\redirect;

$info = InstanceInfo::getInstanceInfo();
$number_result = $info->getActionResult();
$entered_number = $number_result->getEnteredNumber();

if ($entered_number)
{
                $msg = 'Package '.implode(' ', str_split ($entered_number)).' was signed for
                '. 'by Allan yesterday at 3:30 pm. ';

                $response = new Response();
                $response->addAction(Play::sayText ($msg));
                $response->addAction(new Redirect('after_intro.php'));

}               print $response;
?>
```

Figure 6: location_package_process.php

Aculab Cloud sends the caller's digit input to location_package_process.php where the action result is retrieved, which in this case is the digits the caller has pressed:

```
$info = InstanceInfo::getInstanceInfo();
$number_result = $info->getActionResult();
$entered_number = $number_result->getEnteredNumber();
```

In this IVR example the caller is told that the package has been signed for:

```
                $msg = 'Package '.implode(' ', str_split ($entered_number)).' was signed for
                '. 'by Allan yesterday at 3:30 pm. ';
                $response = new response();
                $response->addAction(Play::sayText ($msg));
```

In a real world IVR application, a lookup would be performed via a database, to find the current location of the package.

Menu Option Pages 3: connect_agent

Creating an IVR system in Aculab Cloud

Page structure

The final menu option enables a caller to be connected to an agent. This is so that a caller can speak to someone when needing to resolve a more complicated query.

```
<?php
declare (encoding='UTF-8');
require_once __DIR__.'/vendor/autoload.php';

use \aculab\telephonyrestapi\response;
use \aculab\telephonyrestapi\play;
use \aculab\telephonyrestapi\connect;
use \aculab\telephonyrestapi\redirect;

$response = new response();
$connect = new Connect();
$connect->addDestination ('agent_ivr@sip-0-2-0.aculab.com');
$connect->setHoldMedia (Play::sayText ('All of our operators are currently busy. '.
                'Your call is important to us. Please hold the line, and an agent will answer '.
                'shortly. '));

$response->addAction ($connect);
$response->addAction(new Redirect('after_intro.php'));
print $response;
?>
```

Figure 7: connect_agent.php

Page components

When it is necessary to connect the initial caller to a 3rd party (Agent), the connection is performed using the connect action.

The phone number of the agent is passed to the connect action. This can be either a SIP address, or a telephone number. In this example, we pass a SIP address:

```
$connect->addDestination ('agent_ivr@sip-0-0-0.aculab.com');
```

While the call to the agent is being established, a hold message is played to the caller. This could be text, or an audio file:

```
$connect->setHoldMedia (Play::sayText ('All of our operators are currently busy. '.
                'Your call is important to us. Please hold the line, and an agent will answer '.
                'shortly. '));
```

Menu Option Pages 3: connect_agent

Creating an IVR system in Aculab Cloud

Once the connect action has been completed a request is made to be directed to the main menu.:

```
$response->addAction(new Redirect('after_intro.php'));
```

That completes the walkthrough of this IVR application.

What to do next?

As mentioned earlier, in this application note we have focused on PHP. However, Aculab provides wrappers around its REST API in a number of languages.

Code samples for the other wrappers can be found on the Aculab Cloud website. https://www.aculab.com/cloud/demos-all/ivr

If you have any questions, you can email Aculab's support team via support@aculab.com

Aculab Cloud

Three decades of innovation - in the cloud

About Aculab

Aculab provides deployment proven telephony products to the global communications market

Whether you need telephony resources on a board, on a host server processor or from a cloud-based platform, Aculab ensures that you have the choice. We are an innovative, market leading company that places product quality and support right at the top of our agenda. With over 35 years of experience in helping to drive our customers' success, our technology is used to deliver multimodal voice, data and fax solutions for use within IP, PSTN and mobile networks – with performance levels that are second to none.

For more information

To learn more about Aculab Cloud and Aculab's extensive telephony solutions visit:

www.aculab.com

aculab.com

AMERICAS
+1 (781) 352 3550 | sales@aculab.com

EMEAA
+44 (0) 1908 273802 | sales@aculab.com

Leverage the heritage of Aculab when you move to the cloud

Moving your application development environment to a cloud infrastructure is a big step. Despite the clear benefits of cloud migration, it's natural for developers of hardware-based solutions to be concerned about the risks of moving their technology IP – and the years of investment and knowledge that has gone into creating it – to a new cloud development platform. Most of the big names in cloud communications are relatively new entrants to the communications market; some are working with open source technologies and, as the market consolidates, it is likely that many will not be in business in just a few years' time. So how do you know that a cloud platform can deliver the same level of reliability and performance that you've come to expect from a hardware deployment, and that it will be around for decades?

Three decades of innovation — the next chapter

Aculab Cloud deploys Aculab's industry benchmark technology and has been built organically out of more than 35 years' worth of experience in the communications enablement market. Put simply, it's the result of more than three decades of experience and innovation.

Aculab Cloud developers can be assured that the technology that powers Aculab Cloud has been used to enable tens of thousands of mission-critical applications across the world. Aculab Cloud features robust, field-proven protocols that have been developed and honed in conjunction with thousands of developers and deployed across hundreds of networks.

It's the only cloud communications platform that delivers the expertise, experience and reliability that you get from working with a proven communications enabler.

Leverage our heritage when you move to the cloud.