

# Implementing a RESTful Service Objects Web Service into a PHP Project

## Introduction

This guide provides a step by step on how to implement a Service Objects DOTS web service into a simple web form using Java. This project will be using the REST implementation of BIN Validation but will be similar for any REST implementation of a Service Objects web service. This guide will be very basic and should help serve as a foundation to integrating one of Service Object’s web services into your PHP application.

## What to Expect:

We will demonstrate the implementation in two parts. The first part will demonstrate the process of calling and displaying the results from a Service Objects web service and the second part will include a more applicable example of how to integrate some useful business logic around the validated data from the Service Objects web service.

## Audience

Any developer (beginner to expert), that is interested in integrating a DOTS web service into PHP web application.

## Requirements

- PHP IDE – *NetBeans was used in this example, but the steps should be relatively similar for another PHP IDE*
- Basic Working Knowledge of PHP and server side scripting.
- A working License Key(trial or production) from Service Objects. [Click here](#) to obtain a free trial key for the service you are most interested in.

## Contents

Setting Up Your Project.....	2
Designing Your Web Form and Adding the Code.....	6
Failover Configuration.....	7
Processing the Results From A Service Objects Web Service .....	8
Implementing Business Logic into Your Web Application. ....	9
Conclusion.....	11

# Implementing a RESTful Service Objects Web Service into a PHP Project

## Setting Up Your Project

Launch NetBeans and select **File**, and then **New Project** to create the web application. In the New Project window select **PHP Application** and then click next as seen in the figure below.

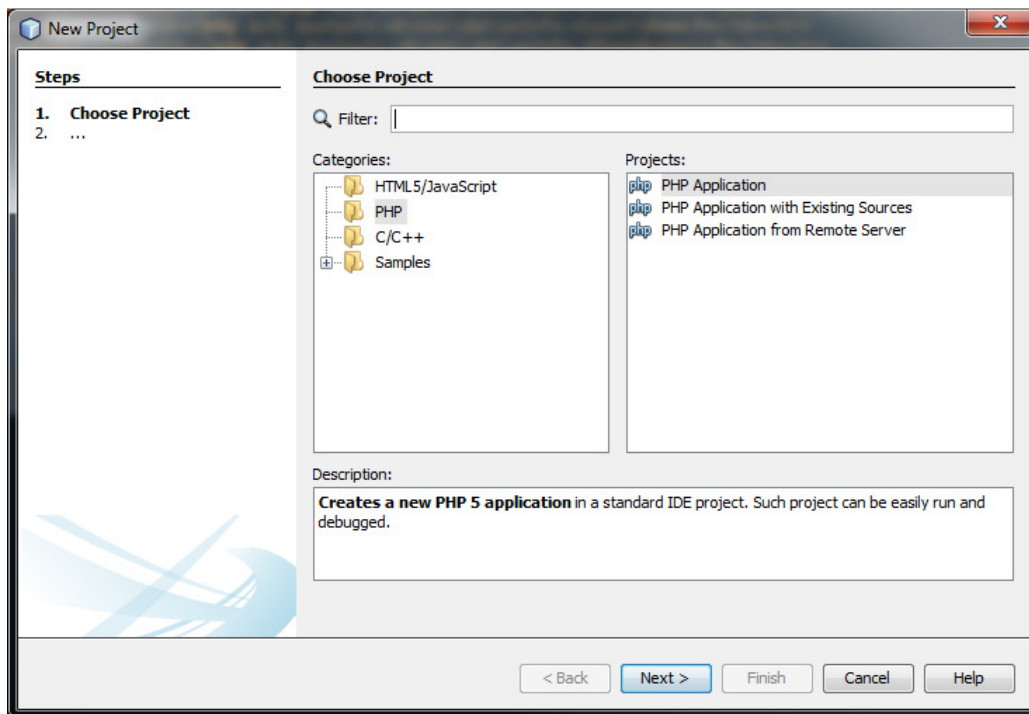


Figure 1

On the next screen, give the project an appropriate name and location of the project.

## Implementing a RESTful Service Objects Web Service into a PHP Project

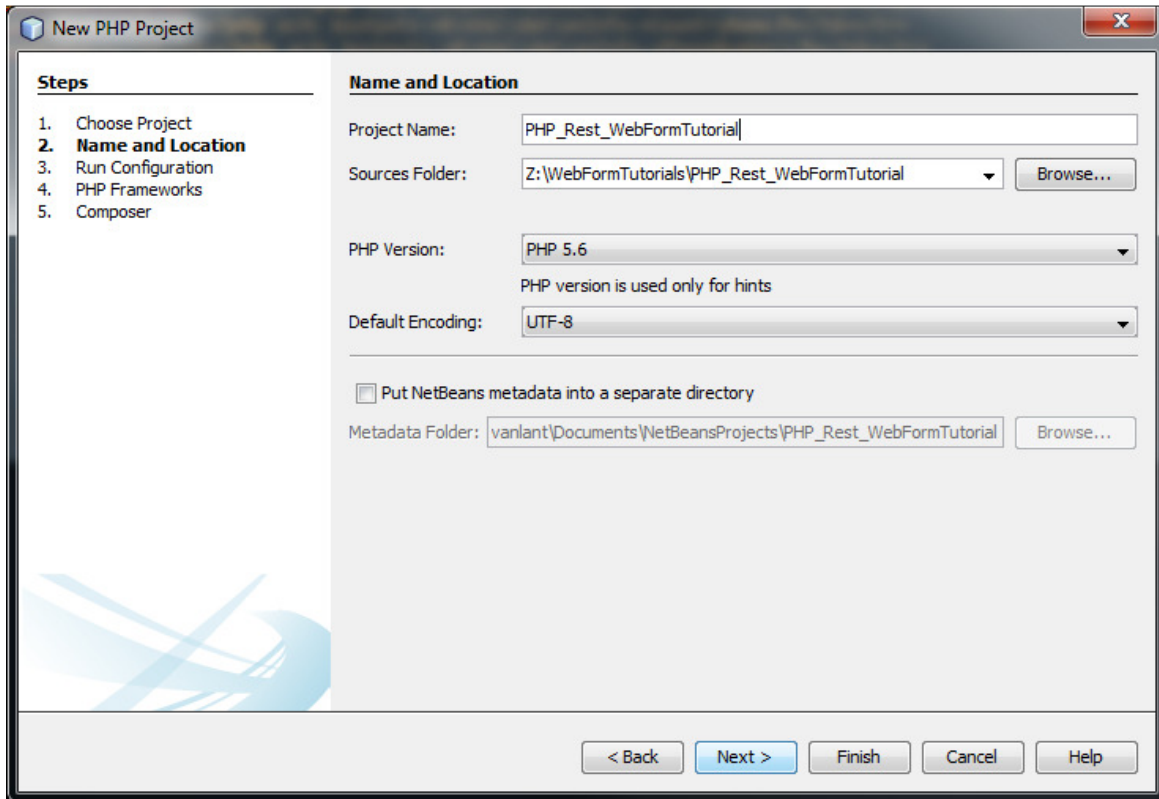


Figure 2

Now that the project has been created, we will need to add a php file to function as our input form and to handle the call to the Service Objects web service. To do this, right click the project folder, select **New** and then select **PHP File** and you will be prompted to name the file to be added. For this example, we will name the PHP file called **restCall.php** which will serve as our input form, make the call to the web service and then process the results. This can be seen in figure 3.

## Implementing a RESTful Service Objects Web Service into a PHP Project

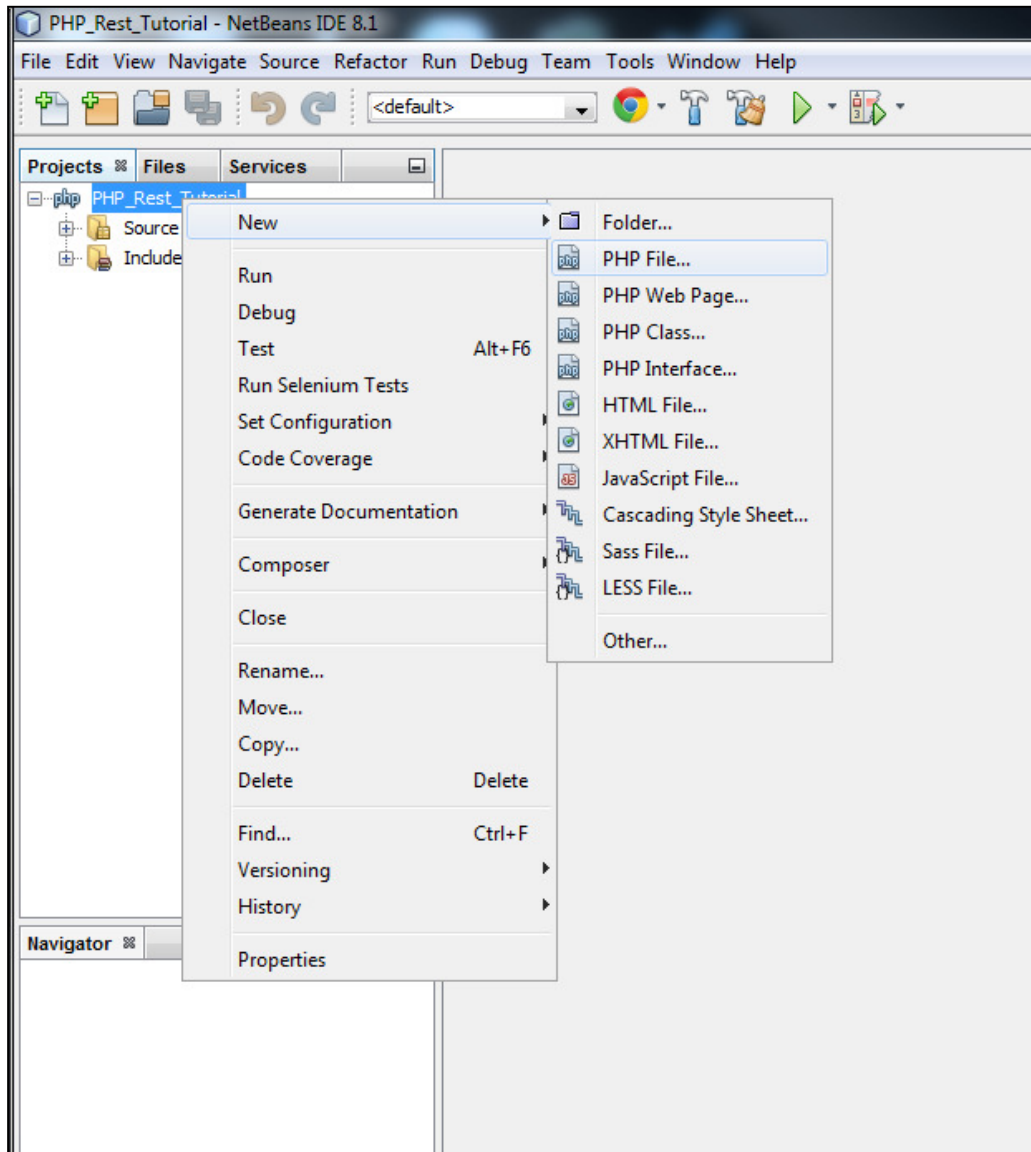


Figure 3

Now that our PHP file has been properly added, we'll need to configure it so that it will run properly. To do this select **Run** at the top of the NetBeans program and then select **Set Project Configuration** and then **Customize...** In the **Run As** selection, highlight **PHP Built-in Web Server(running on built-in web server)**. Ensure that the name of the php file that was just added to the project is placed in the **Router Script** section. This can be seen in the screen shot below.

## Implementing a RESTful Service Objects Web Service into a PHP Project

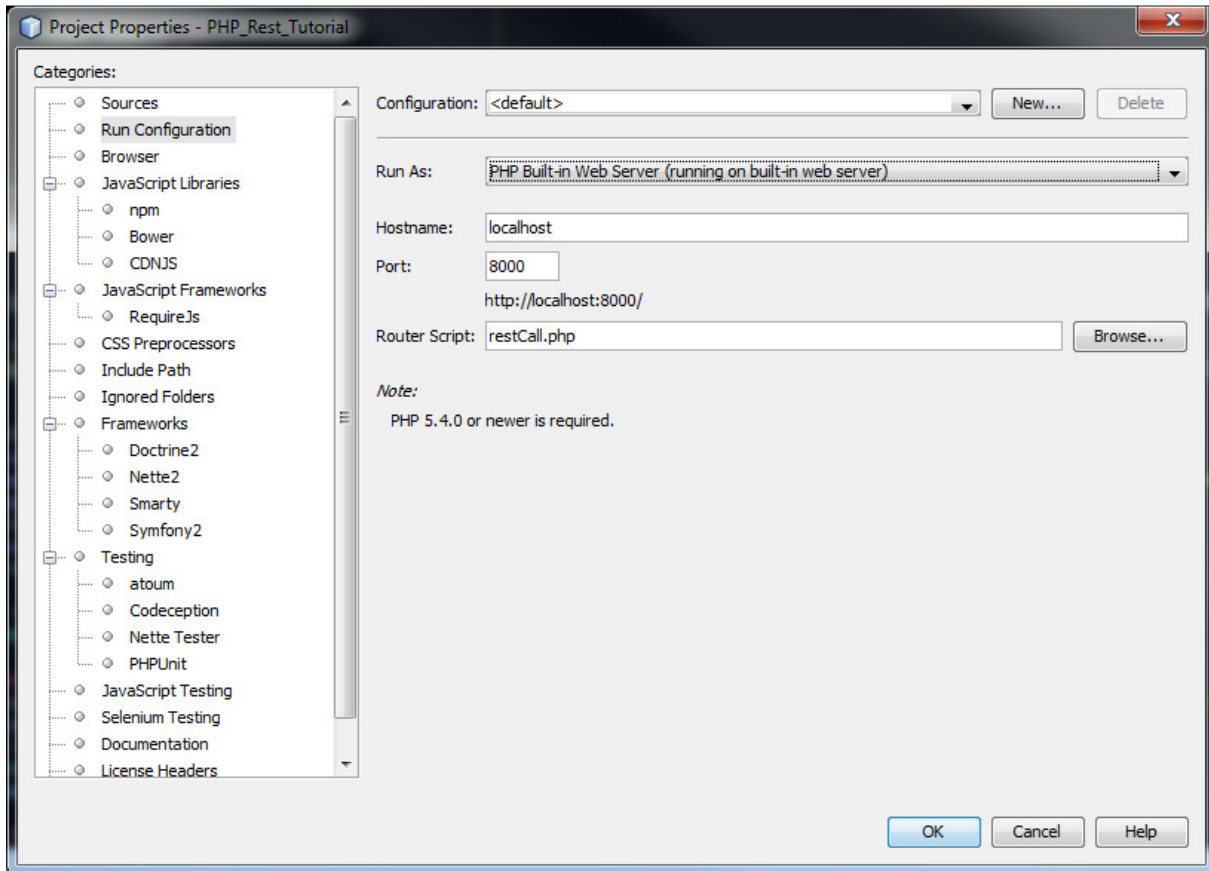


Figure 4

# Implementing a RESTful Service Objects Web Service into a PHP Project

## Designing Your Web Form and Adding the Code

Now that we have successfully created the web application and added the necessary php page, we need to fill in script with the necessary code that will receive the inputs, make the call to the Service Objects web service and process the results for the user. This code be found in the **restCall\_Contents.txt** file that is included with this tutorial.

The first thing the code does is instantiate the variables that input form will be passing to the rest call. It also sets the input-form parameters to our PHP variables.

```
// variable cleanup before generating url
$BIN = trim($BIN);
$LicenseKey = trim($LicenseKey);

$url = "http://trial.serviceobjects.com/bv/BinValidation.asmx/ValidateBIN?BinNumber=".urlencode($BIN)."&LicenseKey=".urlencode($LicenseKey);
//use backup url once given purchased license key
$backupURL = "http://trial.serviceobjects.com/bv/BinValidation.asmx/ValidateBIN?BinNumber=".urlencode($BIN)."&LicenseKey=".urlencode($LicenseKey);

try {
    // Get cURL resource
    $curl = curl_init();
    curl_setopt_array($curl, array(CURLOPT_RETURNTRANSFER => 1, CURLOPT_URL => $url, CURLOPT_USERAGENT => 'Service Objects BIN Validation'));
    curl_setopt($curl, CURLOPT_TIMEOUT, 50); //timeout in seconds
    // Send the request & save response to $resp
    $resp = curl_exec($curl);
    $outputs = new SimpleXMLElement($resp);
}
```

Figure 5: Basic CURL request

The code will also display a simple input form with a table that has inputs for BIN, and LicenseKey. There is also submit button that will send the user entered information to the script behind the form and make the actual rest call to the web service.

After the user has entered a valid BIN number and LicenseKey into the input form, the script will then trim any whitespace on the input variables and then pass then create the backup and primary by encoding the inputs for the curl request. The code will then initialize the curl request and pass in the **primaryURL** to the curl request. After the curl request is completed, the code will instantiate a SimpleXMLElement to process the outputs.

# Implementing a RESTful Service Objects Web Service into a PHP Project

## Failover Configuration

Within the code that makes the call to the web service, we've included failover configuration. In the event that you make your own REST call, we highly recommend implementing a failover call into your code. This is done by checking if the response from the webservice is null or if the error object has a **TypeCode** of 3; which would indicate a fatal error. In the event that either of these instances occurs, the code will call the backup service. We strongly recommend implementing this fail over configuration into your project. With failover enabled you can ensure that your application will function as normal in the event that the production endpoint is unavailable.

```
if($outputs == null || ($outputs->Error != null && $outputs->Error->TypeCode == "3"))
{
    echo "IN back up block";
    curl_setopt_array($curl, array(CURLOPT_RETURNTRANSFER => 1, CURLOPT_URL => $backupURL, CURLOPT_USERAGENT => 'Service Objects BIN Validation'));
    curl_setopt($curl, CURLOPT_TIMEOUT, 50); //timeout in seconds
    // Send the request & save response to $resp
    $resp = curl_exec($curl);
    $outputs = new SimpleXMLElement($resp);

    if($resp == false)
    {
        echo "<b> Both rest calls failed </b>";
        curl_close($curl);
        return;
    }
}
```

Figure 6: Failover Configuration

As noted in the screen shot above, if the code detects a null response or **TypeCode** of 3 it will call the backupURL and attempt to get a valid response. After this, the code checks if there was an error response returned; if so, then it will proceed as normal and display the response to the user. If no error response is detected then the code will display a message saying that both rest calls have failed.

*In this code both the primary and backup URLs are pointed to the same server, but once a production key is purchased there should be 2 service references added to the project: one for the primary Service Objects endpoint and one for the backup:*

<http://ws.serviceobjects.com/bv/BinValidation.asmx>

<http://wsbackup.serviceobjects.com/bv/BinValidation.asmx>



# Implementing a RESTful Service Objects Web Service into a PHP Project

## Processing the Results From A Service Objects Web Service

After our curl request has completed the code will then move onto displaying the results from the webservice to the user. If an error is returned from the web service, then code will display the error that was returned from the service.

Web Service Results	
Outputs	Values
Type	User Input
TypeCode	2
Desc	Improper BIN format. Please input a valid BIN.
DescCode	2

Figure 7

If the user input is valid, the project will output all the results from the service. Below is a screen shot illustrating a valid response from the BIN Validation web service.

Web Service Results	
Outputs	Values
BIN	
BankName	KEYPOINT CREDIT UNION
PaymentMethod	DEBIT
CardType	PLATINUM
CountryAbbreviation	US
CountryName	UNITED STATES
PhoneNumbers	

Figure 8

If some other type of error happened in this process it will be caught in the **catch** statement and the resulting error will be displayed to the user.

This code can be used for testing purposes to experiment with the different types of results that service will output depending on what the user enters.



# Implementing a RESTful Service Objects Web Service into a PHP Project

## Implementing Business Logic into Your Web Application.

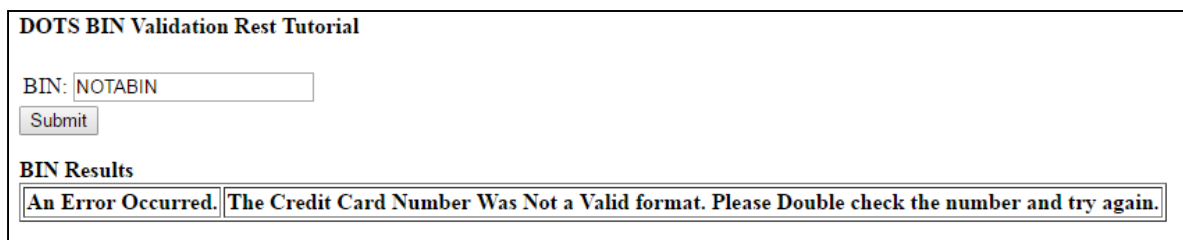
Simply displaying the results to the end user isn't a necessarily a good use of the Service Objects web service, so in this section we will go over an example of what it may look like to integrate some business logic around the results that web service returns. For this example we'll assume that a user is completing an order by inputting a credit card number. The application will take the user entered credit card number and run the first 6 digits through the BIN Validation web service and determine whether or not a given BIN number is valid. Along with this logic, we'll integrate some logic that will display error responses based on potential error messages from the service.

To implement this functionality, we'll need to update the code in our **restCall.php** page. The new code contents for this page can be found in the **Updated\_restCall\_Contents.txt** file that is included with this tutorial.

The project will have similar structure as it takes the input, makes the call to the web service and then displays the results. A few changes have been made to this page to accommodate our new functionality. The first is that we have removed the **LicenseKey** input from the web form. This will be hard coded into our application. The next major changes that we have implemented are in how the application will process an error. We have added several if statements to check for different errors that the service will return and then return various bits of information based on particular error found.

The first error that the application checks for is if an error with a **TypeCode** of 1 is returned. If error appears, then this indicates that something went wrong with the **LicenseKey** for the service. This would indicate a number of things about the **LicenseKey** (i.e. not enough transactions on the key, the key being in valid, etc). In the event that this error appears the code will display an error message the user. It may also be helpful to notify the developer or tech support automatically if this error is reached.

If the code finds an error with a TypeCode and DescCode of 2, this indicates that the BIN was not formatted correctly. This can occur if letters or special characters are put into BIN field. In this event, the code will display a message to the user. Below is a screenshot of that message.



The screenshot shows a web form titled "DOTS BIN Validation Rest Tutorial". It contains a text input field labeled "BIN:" with the value "NOTABIN". Below the input field is a "Submit" button. Underneath the button, there is a section titled "BIN Results" which contains a message box with the text: "An Error Occurred. The Credit Card Number Was Not a Valid format. Please Double check the number and try again."

Figure 9: TypeCode and DescCode of 2

The next check the code will make will be for a fatal error with is a **TypeCode** of 3. Since we have already implemented failover configuration into our project this code should only be reached in the unlikely event that the primary and secondary calls to the service have failed. In the event this happens, this error should be logged and then sent to Service Objects for review.

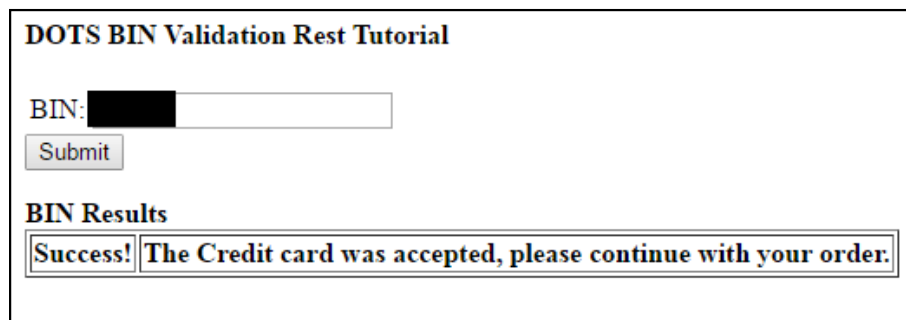
## Implementing a RESTful Service Objects Web Service into a PHP Project

The next set of error messages will display if the Error response from the web service had a **TypeCode** of 4. This code indicates that BIN was properly formatted but there was not enough information available to call it valid. If a **DescCode** of 1 is returned in this instance, then the service indicates that the BIN number was not a valid number. If a **DescCode** of 2 is returned along with a **TypeCode** of 4, then this indicates that there may be conflicting information available about the BIN number. For example one source of data may say the specific card has been issued by one bank and another source may say the card was issued by a different bank.

A final else statement has been implemented in this block in the event that some other error occurs that was not caught in the previous set of code. A full list of error codes that the BIN Validation service can return can be found on the Developer's Guide for the BIN Validation Service. Below is a link to that guide.

<https://docs.serviceobjects.com/display/devguide/DOTS++BIN++Validation#DOTSBINValidation-Errors>

If no error was found, then the code will display a message indicating the credit card number was accepted. If no error is found in a production environment, then this can be used to proceed with a purchase that the user is completing. Below is a screen shot of the message indicating a valid response.



**DOTS BIN Validation Rest Tutorial**

BIN:

**BIN Results**

**Success!** The Credit card was accepted, please continue with your order.

Figure 10

# Implementing a RESTful Service Objects Web Service into a PHP Project

## Conclusion

That concludes our tutorial on how to create a web form that uses a RESTful call with PHP. If you have any questions please email [support@serviceobjects.com](mailto:support@serviceobjects.com) and we would gladly address any issues you may have.