# PHP SOAP Server mit WSDL und API Schlüssel

In diesem Tutorial wird die Einrichtung eine kennwortgeschützten SOAP Servers mit WSDL Unterstützung erklärt.

Damit wir den SOAP Server erweitern können, erstellen wir erstmal eine Kindklasse vom SOAP Server. Dieser bringen wir nichts weiteres bei als die WSDL Funktion.
An der WSDL Funktion scheitern leider die meisten Implementierungen. Sie gibt die Spezifikationen vor, die es ermöglichen einen SOAP Server auch mit anderen Clients zu nutzen - wie z.B. Java.

== SOAP Server ==

### Quellcode

```
/**
* Extends SoapServer with wsdl
*/
class APIServer extends SoapServer {
/**
* use reflection api to print supported methods
*/
public function wsdl() {
$url = 'http://'.$_SERVER['HTTP_HOST'].$_SERVER['PHP_SELF'];
$classNam = 'APIService';
$messageMethods = '';
$portTypeOperations = '';
$bindingOperations = '';
$class = new ReflectionClass($className);
$methods = $class->getMethods();
foreach($methods as $methodKey => $methodValue) {
if(!$methodValue->isPublic()) continue;
$messageMethodParts = '';
$params = $methodValue->getParameters();
foreach($params as $paramKey => $paramValue) {
$messageMethodParts .= ' <part name="'.$paramValue->name.'" type="xsd:anyType"/>';
}
$messageMethods .= ' <message name="'.$methodValue->name.'Request">
'.$messageMethodParts.'
</message>
<message name="'.$methodValue->name.'Response">
<part name="Result" type="xsd:anyType"/>
</message>';
$portTypeOperations .= ' <operation name="'.$methodValue->name.'">
<input message="tns:'.$methodValue->name.'Request"/>
<output message="tns:'.$methodValue->name.'Response"/>
</operation>';
$bindingOperations .= ' <operation name="'.$methodValue->name.'">
<soap:operation soapAction="urn:xmethods-delayed-quotes#'.$methodValue->name.'"/>
<input>
<soap:body use="encoded" namespace="urn:xmethods-delayed-quotes"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</input>
<output>
<soap:body use="encoded" namespace="urn:xmethods-delayed-quotes"
encodingStyle="http://schemas.xmlsoap.org/soap/encoding/"/>
</output>
</operation>';
}
$s = '<?xml version ='.'"'.'1.0'.'"'.' encoding='.'"'.'UTF-8'.'"'.' ?>
```

```
<definitions name="'.$className.'"
targetNamespace="'.$url.'"
xmlns:tns="'.$url.'"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns="http://schemas.xmlsoap.org/wsdl/">
'.$messageMethods.'
<portType name="'.$className.'PortType">
'.$portTypeOperations.'
</portType>
<binding name="'.$className.'Binding" type="tns:'.$className.'PortType">
<soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
'.$bindingOperations.'
</binding>
<service name="'.$className.'Service">
<port name="'.$className.'Port" binding="tns:'.$className.'Binding">
<soap:address location="'.$url.'"/>
</port>
</service>
</definitions>';
header("Content-Type: text/xml");
header(sprintf("Content-Length: %d", strlen($s)));
echo $s;
}
}
```

Alles anzeigen

== SOAP Authentifizierung ==
Wir werden den eigentlich SOAP Service kapseln. Jeder Aufruf muss einen SOAP Header zur Authentifizierung enthalten. Im Beispiel muss ein SOAP Header mit dem Namen "apikey" angegeben werden. Der einzig mögliche Wert im Beispiel ist "hello".
Diese Stelle kann natürlich durch ein komplexeren Authentifizierungsverfahren ersetzt werden. So könnt ihr auch einige Funktionen gar nicht erlauben.

## Quellcode

```
<?php
/**
* API Service base system with authentification
*/
class APIServiceAuth {
private $auth = false;
/**
*
*/
public function __construct() {
$this->apiservice = new APIService();
}
/*
* Authentication function
* This is called by the Soap header of the same name. The function name is a Ws-security standard auth tag
* and corresponds to the header tag.
*/
public function APIToken($apikey) {
```

```
22. $this->apikey = is_object($apikey) ? $apikey->apikey : $apikey;
23. if($this->apikey == 'hello') {
25. $this->auth = true;
26. } else {
27. $this->auth = false;
28. }
29. }
30. /**
32. * wrapper for real api service - checks authentication first
33. *
34. * @param function string
35. * @param arguments mixed[]
36. */
37. public function __call($function, $arguments) {
39. // authenticated
40. if(!$this->auth) {
41. $this->mismatchlog($function, $arguments);
42. throw new SoapFault("Server", "Authentication error.");
43. }
45. // function exists
46. if (method_exists($this->apiservice, $function)) {
47. return call_user_func_array(array($this->apiservice, $function), $arguments);
48. }
49. else {
50. throw new SoapFault("Server", "Call to undefined Method.");
51. }
52. }
53. }
```

Alles anzeigen

== SOAP Service ==
Der SOAP Service bietet die Implementierung aller Funktionen an, die ihr für öffentliche Aufrufe erlauben wollt.

## Quellcode

```
1. <?php
3. /**
4. * API Service
5. */
6. class APIService {
8. /**
9. * gets foo
10. */
11. public function getFoo($param1, $param2) {
12. return 'hello '.$param1.' / '.$param2;
13. }
15. /**
16. * sets foo
17. */
18. public function setFoo($param1, $param2) {
20. }
21. }
```

Alles anzeigen

== http server ==

Damit das ganze aufrufbar wird, muss nun alles zusammengefügt werden:

## Quellcode

```
<?php
require_once('APIServer.php');
require_once('APIService.php');
require_once('APIServiceAuth.php');
$server = new APIServer(null, array(
'uri' => 'demo.easy-coding.de')
);
// output wsdl specs
if(isset($_GET['wsdl'])) {
$server->wsdl();
}
// real handler
else {
$server->setClass('APIServiceAuth');
$server->handle();
}
```

Alles anzeigen

== http client ==
Der Client ist für unser Beispiel wie folgt zu konfigurieren:

## Quellcode

```
<?php
/*
* Login credentials to be supplied as a SOAP header
* Class used to ensure Soap Head tags in correct format.
*/
class SoapHeaderAPIToken {
public $apikey;
/**
*
* @param apikey
*/
public function __construct($apikey) {
$this->apikey = $apikey;
}
}
$uri = 'demo.easy-coding.de';
$wsdl = 'http://demo.easy-coding.de/php/soap-server-wsdl-auth/server.php?wsdl';
$location = 'http://demo.easy-coding.de/php/soap-server-wsdl-auth/server.php';
// Set the login headers
$wsu = 'http://schemas.xmlsoap.org/ws/2002/07/utility';
$usernameToken = new SoapHeaderAPIToken('hello');
$soapHeaders[] = new SoapHeader($wsu, 'APIToken', $usernameToken);
$client = new SoapClient($wsdl, array(
"location" => $location,
"uri" => $uri
));
$client->__setSoapHeaders($soapHeaders);
// access the client
echo $client->getFoo('a', 'b');
```

Alles anzeigen

== Demo + Download ==
Die Demo und alle Downloads findet ihr unter
demo.easy-coding.de/php/soap-server-wsdl-auth/client.php und demo.easy-coding.de/php/soap-server-wsdl-auth/download.zip