SOAP

Introduction

The SOAP extension can be used to write SOAP Servers and Clients. It supports subsets of » SOAP 1.1, » SOAP 1.2 and » WSDL 1.1 specifications.

Installing/Configuring

Requirements

This extension makes use of the » GNOME xml library. Download and install this library. You will need at least libxml-2.5.4.

Installation

This extension is only available if PHP was configured with *--enable-soap*.

Runtime Configuration

The behaviour of these functions is affected by settings in *php.ini*.

SOAP Configuration Options

Name	Default	Changeable	Changelog
soap.wsdl_cache_enabled	"1"	PHP_INI_ALL	Available since PHP 5.0.0.
soap.wsdl_cache_dir	"/tmp"	PHP_INI_ALL	Available since PHP 5.0.0.
soap.wsdl_cache_ttl	"86400"	PHP_INI_ALL	Available since PHP 5.0.0.
soap.wsdl_cache_limit	"5"	PHP_INI_ALL	Available since PHP 5.1.5.
soap.wsdl_cache	"1"	PHP_INI_ALL	Available since PHP 5.1.5.

For further details and definitions of the PHP_INI_* constants, see the php.ini directives.

Here's a short explanation of the configuration directives.

`soap.wsdl_cache_enabled` boolean
: Enables or disables the WSDL caching feature.

`soap.wsdl_cache_dir` string
: Sets the directory name where the SOAP extension will put cache files.

soap.wsdl_cache_ttl integer
: Sets the number of seconds (time to live) that cached files will be used instead of the originals.

soap.wsdl_cache_limit integer
: Maximum number of in-memory cached WSDL files. Adding further files into a full memory cache will delete the oldest files from it.

soap.wsdl_cache integer
: If *soap.wsdl_cache_enabled* is on, this setting determines the type of caching. It can be any of: **WSDL_CACHE_NONE** (*0*), **WSDL_CACHE_DISK** (*1*), **WSDL_CACHE_MEMORY** (*2*) or **WSDL_CACHE_BOTH** (*3*). This can also be set via the *options* array in the SoapClient or SoapServer constructor.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

SOAP_1_1 (integer)

SOAP_1_2 (integer)

SOAP_PERSISTENCE_SESSION (integer)

SOAP_PERSISTENCE_REQUEST (integer)

SOAP_FUNCTIONS_ALL (integer)

SOAP_ENCODED (integer)

SOAP_LITERAL (integer)

SOAP_RPC (integer)

SOAP_DOCUMENT (integer)

SOAP_ACTOR_NEXT (integer)

SOAP_ACTOR_NONE (integer)

SOAP_ACTOR_UNLIMATERECEIVER (integer)

SOAP_COMPRESSION_ACCEPT (integer)

SOAP_COMPRESSION_GZIP (integer)

SOAP_COMPRESSION_DEFLATE (integer)

SOAP_WAIT_ONE_WAY_CALLS (integer)
Added in PHP 5.1.0.

UNKNOWN_TYPE (integer)

XSD_STRING (integer)

XSD_BOOLEAN (integer)

XSD_DECIMAL (integer)

XSD_FLOAT (integer)

XSD_DOUBLE (integer)

XSD_DURATION (integer)

XSD_DATETIME (integer)

XSD_TIME (integer)

XSD_DATE (integer)

XSD_GYEARMONTH (integer)

XSD_GYEAR (integer)

XSD_GMONTHDAY (integer)

XSD_GDAY (integer)

XSD_GMONTH (integer)

XSD_HEXBINARY (integer)

XSD_BASE64BINARY (integer)

XSD_ANYURI (integer)

XSD_ANYXML (integer)
Added in PHP 5.1.0.

XSD_QNAME (integer)

XSD_NOTATION (integer)

XSD_NORMALIZEDSTRING (integer)

XSD_TOKEN (integer)

XSD_LANGUAGE (integer)

XSD_NMTOKEN (integer)

XSD_NAME (integer)

XSD_NCNAME (integer)

XSD_ID (integer)

XSD_IDREF (integer)

XSD_IDREFS (integer)

XSD_ENTITY (integer)

XSD_ENTITIES (integer)

XSD_INTEGER (integer)

XSD_NONPOSITIVEINTEGER (integer)

XSD_NEGATIVEINTEGER (integer)

XSD_LONG (integer)

XSD_INT (integer)

XSD_SHORT (integer)

XSD_BYTE (integer)

XSD_NONNEGATIVEINTEGER (integer)

XSD_UNSIGNEDLONG (integer)

XSD_UNSIGNEDINT (integer)

XSD_UNSIGNEDSHORT (integer)

XSD_UNSIGNEDBYTE (integer)

XSD_POSITIVEINTEGER (integer)

XSD_NMTOKENS (integer)

XSD_ANYTYPE (integer)

SOAP_ENC_OBJECT (integer)

SOAP_ENC_ARRAY (integer)

XSD_1999_TIMEINSTANT (integer)

XSD_NAMESPACE (string)

XSD_1999_NAMESPACE (string)

WSDL_CACHE_NONE (integer)

Switches off WSDL caching even if soap.wsdl_cache_enabled is on. Available since PHP 5.1.5.

WSDL_CACHE_DISK (integer)
Available since PHP 5.1.5.

WSDL_CACHE_MEMORY (integer)
Caches WSDL data in process memory. Available since PHP 5.1.5.

WSDL_CACHE_BOTH (integer)
Available since PHP 5.1.5.

SOAP Functions

Predefined Classes

SoapClient

Constructor

- SoapClient->__construct() - constructs a new SoapClient object

Methods

- SoapClient->__call() - Calls a SOAP function (deprecated)
- SoapClient->__doRequest() - Performs a SOAP request
- SoapClient->__getFunctions() - Returns list of SOAP functions
- SoapClient->__getLastRequest() - Returns last SOAP request
- SoapClient->__getLastRequestHeaders() - Returns last SOAP request headers
- SoapClient->__getLastResponse() - Returns last SOAP response
- SoapClient->__getLastResponseHeaders() - Returns last SOAP response headers
- SoapClient->__getTypes() - Returns list of SOAP types
- SoapClient->__setCookie() - Sets the cookie that will be sent with the SOAP request
- SoapClient->__soapCall() - Calls a SOAP function

SoapFault

Constructor

- SoapFault->__construct() - construct a new SoapFault object

SoapHeader

SoapHeader is a special low-level class for passing or returning SOAP headers. It's just a data holder and it does not have any special methods except its constructor. It can be used in the SoapClient->__soapCall() method to pass a SOAP header or in a SOAP header handler to return the header in a SOAP response.

Constructor

- SoapHeader->__construct() - construct a new SoapHeader object

SoapParam

SoapParam is a special low-level class for naming parameters and returning values in *non-WSDL* mode. It's just a data holder and it does not have any special methods except its constructor.

Constructor

- SoapParam->__construct() - construct a new SoapParam object

SoapServer

Constructor

- SoapServer->__construct() - construct a new SoapServer object

Methods

- SoapServer->addFunction() - Adds one or several functions those will handle SOAP requests
- SoapServer->fault() -
- SoapServer->getFunctions() - Returns list of defined functions
- SoapServer->handle() - Handles a SOAP request
- SoapServer->setClass() - Sets class which will handle SOAP requests

- SoapServer->setPersistence() - Sets persistence mode of SoapServer

SoapVar

SoapVar is a special low-level class for encoding parameters and returning values in *non-WSDL* mode. It's just a data holder and does not have any special methods except the constructor. It's useful when you want to set the type property in SOAP request or response.

Constructor

- SoapVar->__construct() - construct a new SoapVar object

is_soap_fault

is_soap_fault -- Checks if SOAP call was failed

Description

bool **is_soap_fault** (mixed `$obj`)

This function is useful when you like to check if the SOAP call failed, but don't like to use exceptions. To use it you must create a SoapClient object with the *exceptions* option set to zero or **FALSE**. In this case, the SOAP method will return a special SoapFault object which encapsulates the fault details (faultcode, faultstring, faultactor and faultdetails).

If *exceptions* is not set then SOAP call will throw an exception on error. is_soap_fault() checks if the given parameter is a SoapFault object.

Parameters

obj
The tested object.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #1 - is_soap_fault() example

```
<?php
$client = new SoapClient("some.wsdl", array('exceptions' => 0));
$result = $client->SomeFunction();
if (is_soap_fault($result)) {
   trigger_error("SOAP Fault: (faultcode: {$result->faultcode}, faultstring:
{$result->faultstring})", E_USER_ERROR);
}
?>
```

Example #2 - SOAP's standard method for error reporting is exceptions

```
<?php
try {
   $client = new SoapClient("some.wsdl");
   $result = $client->SomeFunction(/* ... */);
```

```
} catch (SoapFault $fault) {
   trigger_error("SOAP Fault: (faultcode: {$fault->faultcode}, faultstring:
{$fault->faultstring})", E_USER_ERROR);
}
?>
```

See Also

- SoapClient->__construct()
- SoapFault->__construct()

SoapClient->__call()

SoapClient->__call() -- Calls a SOAP function (deprecated)

Description

SoapClient

mixed **__call** (string `$function_name`, array `$arguments` [, array `$options` [, array `$input_headers` [, array `$output_headers`]]])

This method is deprecated. Use SoapClient->__soapCall() instead of it.

SoapClient->__construct()

SoapClient->__construct() -- SoapClient constructor

Description

SoapClient

__construct (mixed `$wsdl` [, array `$options`])

This constructor creates SoapClient objects in *WSDL* or *non-WSDL* mode.

Parameters

`wsdl`
URI of the *WSDL* file or **NULL** if working in *non-WSDL* mode.

Note

During development stage, you may want to disable WSDL caching by the mean of the *soap.wsdl_cache_ttl php.ini* setting, otherwise changes made to the WSDL file will have no effect until *soap.wsdl_cache_ttl* is expired.

`options`
An array of options. If working in WSDL mode, this parameter is optional. If working in non-WSDL mode, you must set the *location* and *uri* options, where *location* is the URL to request and *uri* is the target namespace of the SOAP service. The *style* and *use* options only work in non-WSDL mode. In WSDL mode, they come from the WSDL file. The *soap_version* option specifies whether to use SOAP 1.1, or SOAP 1.2 client. For HTTP authentication, you may use the *login* and *password* options. For making an HTTP connection through a proxy server, use the options *proxy_host*, *proxy_port*, *proxy_login* and *proxy_password*. For HTTPS client certificate authentication use *local_cert* and *passphrase* options. The *compression* option allows to use compression of HTTP SOAP requests and responses. The *encoding* option defines internal character encoding. This option does not change the encoding of SOAP requests (it is always utf-8), but converts strings into it. The *classmap* option can be used to map some WSDL types to PHP classes. This option must be an array with WSDL types as keys and names of PHP classes as values. Setting the boolean *trace* option enables use of the methods SoapClient->__getLastRequest, SoapClient->__getLastRequestHeaders, SoapClient->__getLastResponse and SoapClient->__getLastResponseHeaders. The *exceptions* option is a boolean value defining whether soap errors throw exceptions of type SoapFault. The

connection_timeout option defines a timeout in seconds for the connection to the SOAP service. This option does not define a timeout for services with slow responses. To limit the time to wait for calls to finish the default_socket_timeout setting is available. The *typemap* option is an array of type mappings. Type mapping is an array with keys *type_name*, *type_ns* (namespace URI), *from_xml* (callback accepting one string parameter) and *to_xml* (callback accepting one object parameter). Other options are *stream_context*, *features*, *cache_wsdl* and *user_agent*.

Examples

Example #3 - SoapClient examples

```
<?php

$client = new SoapClient("some.wsdl");

$client = new SoapClient("some.wsdl", array('soap_version'   => SOAP_1_2));

$client = new SoapClient("some.wsdl", array('login'          => "some_name",
                                            'password'       =>
"some_password"));

$client = new SoapClient("some.wsdl", array('proxy_host'     => "localhost",
                                            'proxy_port'     => 8080));

$client = new SoapClient("some.wsdl", array('proxy_host'     => "localhost",
                                            'proxy_port'     => 8080,
                                            'proxy_login'    => "some_name",
                                            'proxy_password' =>
"some_password"));

$client = new SoapClient("some.wsdl", array('local_cert'     =>
"cert_key.pem"));

$client = new SoapClient(null, array('location' =>
"http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));

$client = new SoapClient(null, array('location' =>
"http://localhost/soap.php",
                                     'uri'      => "http://test-uri/",
                                     'style'    => SOAP_DOCUMENT,
                                     'use'      => SOAP_LITERAL));

$client = new SoapClient("some.wsdl",
 array('compression' => SOAP_COMPRESSION_ACCEPT | SOAP_COMPRESSION_GZIP));

$server = new SoapClient("some.wsdl", array('encoding'=>'ISO-8859-1'));

class MyBook {
   public $title;
   public $author;
}

$server = new SoapClient("books.wsdl", array('classmap' => array('book' =>
```

```
"MyBook")));

?>
```

SoapClient->__doRequest()

SoapClient->__doRequest() -- Performs a SOAP request

Description

SoapClient

string **__doRequest** (string `$request`, string `$location`, string `$action`, int `$version` [, int `$one_way`])

Performs SOAP request over HTTP.

This method can be overridden in subclasses to implement different transport layers, perform additional XML processing or other purpose.

Parameters

`request`
The XML SOAP request.

`location`
The URL to request.

`action`
The SOAP action.

`version`
The SOAP version.

`one_way`

Return Values

The XML SOAP response.

ChangeLog

Version	Description

5.1.3	The *one_way* parameter was added.

Examples

Example #4 - Some examples

```
<?php
function Add($x,$y) {
 return $x+$y;
}

class LocalSoapClient extends SoapClient {

 function __construct($wsdl, $options) {
   parent::__construct($wsdl, $options);
   $this->server = new SoapServer($wsdl, $options);
   $this->server->addFunction('Add');
 }

 function __doRequest($request, $location, $action, $version) {
   ob_start();
   $this->server->handle($request);
   $response = ob_get_contents();
   ob_end_clean();
   return $response;
 }

}

$x = new LocalSoapClient(NULL,array('location'=>'test://',
                                    'uri'=>'http://testuri.org'));
var_dump($x->Add(3,4));
?>
```

SoapClient->__getFunctions()

SoapClient->__getFunctions() -- Returns list of SOAP functions

Description

SoapClient

array __**getFunctions** (void)

Returns the list of SOAP functions.

Note

This function works only in WSDL mode.

Return Values

The list of SOAP functions.

Examples

Example #5 - SoapClient->__getFunctions() example

```
<?php
$client = new SoapClient('some.wsdl');
var_dump($client->__getFunctions());
?>
```

See Also

- SoapClient->__construct()

SoapClient->__getLastRequest()

SoapClient->__getLastRequest() -- Returns last SOAP request

Description

SoapClient

string **__getLastRequest** (void)

Note

This method works only if the SoapClient object was created with the *trace* option.

Return Values

The last SOAP request.

Examples

Example #6 - SoapClient->__getLastRequest() example

```
<?php
$client = SoapClient("some.wsdl", array('trace' => 1));
$result = $client->SomeFunction();
echo "REQUEST:\n" . $client->__getLastRequest() . "\n";
?>
```

See Also

- SoapClient->__construct()
- SoapClient->__getLastRequestHeaders()
- SoapClient->__getLastResponse()
- SoapClient->__getLastResponseHeaders()

SoapClient->__getLastRequestHeaders()

SoapClient->__getLastRequestHeaders() -- Returns last SOAP request headers

Description

SoapClient

string **__getLastRequestHeaders** (void)

Note
This method works only if the SoapClient object was created with the *trace* option.

Return Values

The last SOAP request headers.

See Also

- SoapClient->__construct()
- SoapClient->__getLastRequest()
- SoapClient->__getLastResponse()
- SoapClient->__getLastResponseHeaders()

SoapClient->__getLastResponse()

SoapClient->__getLastResponse() -- Returns last SOAP response.

Description

SoapClient

string **__getLastResponse** (void)

Note

This method works only if the SoapClient object was created with the *trace* option.

Return Values

The last SOAP response.

Examples

Example #7 - SoapClient->__getLastResponse() example

```
<?php
$client = SoapClient("some.wsdl", array('trace' => 1));
$result = $client->SomeFunction();
echo "RESPONSE:\n" . $client->__getLastResponse() . "\n";
?>
```

See Also

- SoapClient->__construct()
- SoapClient->__getLastResponseHeaders()
- SoapClient->__getLastRequest()
- SoapClient->__getLastRequestHeaders()

SoapClient->__getLastResponseHeaders()

SoapClient->__getLastResponseHeaders() -- Returns last SOAP response headers.

Description

SoapClient

string **__getLastResponseHeaders** (void)

Note
This method works only if the SoapClient object was created with the *trace* option.

Return Values

The last SOAP response headers.

See Also

- SoapClient->__construct()
- SoapClient->__getLastResponse()
- SoapClient->__getLastRequest()
- SoapClient->__getLastRequestHeaders()

SoapClient->__getTypes()

SoapClient->__getTypes() -- Returns list of SOAP types

Description

SoapClient

array __**getTypes** (void)

This function works only in WSDL mode.

Return Values

The list of SOAP types.

Examples

Example #8 - SoapClient->__getTypes() example

```
<?php
$client = new SoapClient("some.wsdl");
var_dump($client->__getTypes());
?>
```

See Also

- SoapClient->__construct()

SoapClient->__setCookie()

SoapClient->__setCookie() -- Sets the cookie that will be sent with the SOAP request

Description

SoapClient

void **__setCookie** (string `$name` [, string `$value`])

Defines a cookie to be sent along with the SOAP requests.

Note

Calling this method will affect all following calls to SoapClient methods.

Parameters

`name`
The name of the cookie.

`value`
The value of the cookie. If not specified, the cookie will be deleted.

Return Values

No value is returned.

SoapClient->__soapCall()

SoapClient->__soapCall() -- Calls a SOAP function

Description

SoapClient

mixed **__soapCall** (string `$function_name`, array `$arguments` [, array `$options` [, mixed `$input_headers` [, array `&$output_headers`]]])

This is a low level API function that is used to make a SOAP call. Usually, in WSDL mode, you can simply call SOAP functions as SoapClient methods. This method useful in non-WSDL mode when *soapaction* is unknown, *uri* differs from the default or when sending and/or receiving SOAP Headers.

On error, a call to a SOAP function can cause PHP to throw exceptions or return a SoapFault object if exceptions are disabled. To check if the function call failed to catch the SoapFault exceptions, check the result with is_soap_fault().

Return Values

SOAP functions may return one, or multiple values. If only one value is returned by the SOAP function, the return value of *__soapCall* will be a simple value (e.g. an integer, a string, etc). If multiple values are returned, *__soapCall* will return an associative array of named output parameters.

Examples

Example #9 - SoapClient->__soapCall() Examples

```
<?php

$client = new SoapClient("some.wsdl");
$client->SomeFunction($a, $b, $c);

$client->__soapCall("SomeFunction", array($a, $b, $c));
$client->__soapCall("SomeFunction", array($a, $b, $c), NULL,
                    new SoapHeader(), $output_headers);


$client = new SoapClient(null, array('location' =>
"http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));
$client->SomeFunction($a, $b, $c);
```

```
$client->__soapCall("SomeFunction", array($a, $b, $c));
$client->__soapCall("SomeFunction", array($a, $b, $c),
                    array('soapaction' => 'some_action',
                          'uri'        => 'some_uri'));
?>
```

See Also

- SoapClient->__construct()
- SoapParam->__construct()
- SoapVar->__construct()
- SoapHeader->__construct()
- SoapFault->__construct()
- is_soap_fault()

SoapFault->__construct()

SoapFault->__construct() -- SoapFault constructor

Description

SoapFault

__construct (string `$faultcode`, string `$faultstring` [, string `$faultactor` [, mixed `$detail` [, string `$faultname` [, SoapHeader `$headerfault`]]]])

This class is useful when you would like to send SOAP fault responses from the PHP handler. *`faultcode`*, *`faultstring`*, *`faultactor`* and *`details`* are standard elements of SOAP Fault;

Parameters

`faultcode`
The error code of the SoapFault.

`faultstring`
The error message of the SoapFault.

`faultactor`
A string identifying the actor that caused the error.

`detail`

`faultname`
Can be used to select the proper fault encoding from WSDL.

`headerfault`
Can be used during SOAP header handling to report an error in the response header.

Examples

Example #10 - Some examples

```
<?php
function test($x)
```

```
{
   return new SoapFault("Server", "Some error message");
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

It is possible to use PHP exception mechanism to throw SOAP Fault.

Example #11 - Some examples

```
<?php
function test($x)
{
   throw new SoapFault("Server", "Some error message");
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

See Also

- SoapClient->__construct()
- SoapClient->__soapCall()
- SoapVar->__construct()
- SoapParam->__construct()
- SoapFault->__construct()
- is_soap_fault()

SoapHeader->__construct()

SoapHeader->__construct() -- SoapHeader constructor

Description

SoapHeader

__construct (string `$namespace`, string `$name` [, mixed `$data` [, bool `$mustUnderstand` [, mixed `$actor`]]])

Constructs a new SoapHeader object.

Parameters

`namespace`
The namespace of the SOAP header element.

`name`
The name of the SOAP header element.

`data`
A SOAP header's content. It can be a PHP value or a SoapVar object.

`mustUnderstand`
Value of the *mustUnderstand* attribute of the SOAP header element.

`actor`
Value of the *actor* attribute of the SOAP header element.

Examples

Example #12 - Some examples

```
<?php
$client = new SoapClient(null, array('location' =>
"http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));
$client->__soapCall("echoVoid", null, null,
               new SoapHeader('http://soapinterop.org/echoheader/',
                              'echoMeStringRequest',
                              'hello world'));
```

```
?>
```

See Also

- SoapClient->__soapCall()
- SoapVar->__construct()
- SoapParam->__construct()

SoapParam->__construct()

SoapParam->__construct() -- SoapParam constructor

Description

SoapParam

__construct (mixed `$data`, string `$name`)

Constructs a new SoapParam object.

Parameters

`data`
: The data to pass or return. You can pass this parameter directly as PHP value, but in this case it will be named as *paramN* and the SOAP Service may not understand it.

`name`
: The parameter name.

Examples

Example #13 - Some examples

```
<?php
$client = new SoapClient(null,array('location' =>
"http://localhost/soap.php",
                                    'uri'      => "http://test-uri/"));
$client->SomeFunction(new SoapParam($a, "a"),
                      new SoapParam($b, "b"),
                      new SoapParam($c, "c"));
?>
```

See Also

- SoapClient->__soapCall()
- SoapVar->__construct()

SoapServer->addFunction()

SoapServer->addFunction() -- Adds one or several functions those will handle SOAP requests

Description

SoapServer

void **addFunction** (mixed `$functions`)

Exports one or more functions for remote clients.

Parameters

`functions`
To export one function, pass the function name into this parameter as a string. To export several functions, pass an array of function names. To export all the functions, pass a special constant **SOAP_FUNCTIONS_ALL**.

Note

`functions` must receive all input arguments in the same order as defined in the WSDL file (They should not receive any output parameters as arguments) and return one or more values. To return several values they must return an array with named output parameters.

Return Values

No value is returned.

Examples

Example #14 - Some examples

```
<?php

function echoString($inputString)
{
```

```
   return $inputString;
}

$server->addFunction("echoString");

function echoTwoStrings($inputString1, $inputString2)
{
   return array("outputString1" => $inputString1,
                "outputString2" => $inputString2);
}
$server->addFunction(array("echoString", "echoTwoStrings"));

$server->addFunction(SOAP_FUNCTIONS_ALL);

?>
```

See Also

- SoapServer->__construct()
- SoapServer->setClass()

SoapServer->__construct()

SoapServer->__construct() -- SoapServer constructor

Description

SoapServer

__construct (mixed `$wsdl` [, array `$options`])

This constructor allows the creation of SoapServer objects in WSDL or non-WSDL mode.

Parameters

`wsdl`
: If you want the WSDL mode, you must set this to the URI of a WSDL file. In the other case, you must set this to **NULL** and set the *uri* option.

`options`
: Allow setting a default SOAP version (*soap_version*), internal character encoding (*encoding*), and actor URI (*actor*). The *classmap* option can be used to map some WSDL types to PHP classes. This option must be an array with WSDL types as keys and names of PHP classes as values. The *typemap* option is an array of type mappings. Type mapping is an array with keys *type_name*, *type_ns* (namespace URI), *from_xml* (callback accepting one string parameter) and *to_xml* (callback accepting one object parameter). Other options are *features* and *cache_wsdl*.

Examples

Example #15 - Some examples

```
<?php

$server = new SoapServer("some.wsdl");

$server = new SoapServer("some.wsdl", array('soap_version' => SOAP_1_2));

$server = new SoapServer("some.wsdl", array('actor' =>
"http://example.org/ts-tests/C"));

$server = new SoapServer("some.wsdl", array('encoding'=>'ISO-8859-1'));
```

```
$server = new SoapServer(null, array('uri' => "http://test-uri/"));

class MyBook {
   public $title;
   public $author;
}

$server = new SoapServer("books.wsdl", array('classmap' => array('book' =>
"MyBook")));

?>
```

SoapServer->fault()

SoapServer->fault() -- Issue SoapServer fault indicating an error

Description

SoapServer

void **fault** (string `$code`, string `$string` [, string `$actor` [, mixed `$details` [, string `$name`]]])

Warning

This function is currently not documented; only its argument list is available.

See Also

- SoapFault->__construct()

SoapServer->getFunctions()

SoapServer->getFunctions() -- Returns list of defined functions

Description

SoapServer

array **getFunctions** (void)

This method returns the list of all functions added by SoapServer->addFunction() or SoapServer->setClass().

Return Values

The list of all functions.

Examples

Example #16 - Some examples

```
<?php
$server = new SoapServer(NULL, array("uri" => "http://test-uri"));
$server->addFunction(SOAP_FUNCTIONS_ALL);
if ($_SERVER["REQUEST_METHOD"] == "POST") {
 $server->handle();
} else {
 echo "This SOAP server can handle following functions: ";
 $functions = $server->getFunctions();
 foreach($functions as $func) {
   echo $func . "\n";
 }
}
?>
```

See Also

- SoapServer->__construct()
- SoapServer->addFunction()
- SoapServer->setClass()

SoapServer->handle()

SoapServer->handle() -- Handles a SOAP request

Description

SoapServer

void **handle** ([string `$soap_request`])

Processes a SOAP request, calls necessary functions, and sends a response back.

Parameters

`soap_request`
The SOAP request. If this argument is omitted, the request is supposed to be in the *$HTTP_RAW_POST_DATA* PHP variable.

Return Values

No value is returned.

Examples

Example #17 - Some examples

```
<?php
function test($x)
{
   return $x;
}

$server = new SoapServer(null, array('uri' => "http://test-uri/"));
$server->addFunction("test");
$server->handle();
?>
```

See Also

- SoapServer->__construct()

SoapServer->setClass()

SoapServer->setClass() -- Sets class which will handle SOAP requests

Description

SoapServer

void **setClass** (string `$class_name` [, mixed `$args` [, mixed `$...`]])

Exports all methods from specified class.

The object can be made persistent across request for a given PHP session with the SoapServer->setPersistence() method.

Parameters

`class_name`
The name of the exported class.

`args`
These optional parameters will be passed to the default class constructor during object creation.

Return Values

No value is returned.

Examples

Example #18 - Some examples

```
<?php

class foo {
   function foo()
   {
   }
}
$server->setClass("foo");

class bar {
```

```
    function bar($x, $y)
    {
    }
}
$server->setClass("bar", $arg1, $arg2);

?>
```

See Also

- SoapServer->__construct()
- SoapServer->addFunction()
- SoapServer->setPersistence()

SoapServer->setPersistence()

SoapServer->setPersistence() -- Sets persistence mode of SoapServer

Description

SoapServer

void **setPersistence** (int `$mode`)

This function allows saving data between requests in a PHP session. It works only with a server that exports functions from a class with SoapServer->setClass().

Parameters

`mode`
One of the *SOAP_PERSISTENCE_XXX* constants.

Return Values

No value is returned.

Examples

Example #19 - Some examples

```
<?php

$server->setPersistence(SOAP_PERSISTENCE_SESSION);

$server->setPersistence(SOAP_PERSISTENCE_REQUEST);

?>
```

Note

The persistence **SOAP_PERSISTENCE_SESSION** makes persistent only object of given class, but not the class static data. You may use $this->bar instead of self::$bar.

See Also

- SoapServer->__construct()
- SoapServer->setClass()

SoapVar->__construct()

SoapVar->__construct() -- SoapVar constructor

Description

SoapVar

__construct (mixed `$data`, int `$encoding` [, string `$type_name` [, string `$type_namespace` [, string `$node_name` [, string `$node_namespace`]]]])

Constructs a new SoapVar object.

Parameters

`data`
The data to pass or return.

`encoding`
The encoding ID, one of the *XSD_...* constants.

`type_name`
The type name.

`type_namespace`
The type namespace.

`node_name`
The XML node name.

`node_namespace`
The XML node namespace.

Examples

Example #20 - Some examples

```
<?php
class SOAPStruct {
   function SOAPStruct($s, $i, $f)
   {
```

```
        $this->varString = $s;
        $this->varInt = $i;
        $this->varFloat = $f;
    }
}
$client = new SoapClient(null, array('location' =>
"http://localhost/soap.php",
                                     'uri'      => "http://test-uri/"));
$struct = new SOAPStruct('arg', 34, 325.325);
$soapstruct = new SoapVar($struct, SOAP_ENC_OBJECT, "SOAPStruct",
"http://soapinterop.org/xsd");
$client->echoStruct(new SoapParam($soapstruct, "inputStruct"));
?>
```

See Also

- SoapClient->__soapCall()
- SoapParam->__construct()

use_soap_error_handler

use_soap_error_handler -- Set whether to use the SOAP error handler and return the former value

Description

bool **use_soap_error_handler** ([bool `$handler`])

Warning
This function is currently not documented; only its argument list is available.