* [Welcome to CodeIgniter](http://docs.google.com/general/welcome.html)
* [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Downloading CodeIgniter](http://docs.google.com/installation/downloads.html)
  + [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Upgrading From a Previous Version](http://docs.google.com/installation/upgrading.html)
  + [Troubleshooting](http://docs.google.com/installation/troubleshooting.html)
* [CodeIgniter Overview](http://docs.google.com/overview/index.html)
  + [Getting Started](http://docs.google.com/overview/getting_started.html)
  + [CodeIgniter at a Glance](http://docs.google.com/overview/at_a_glance.html)
  + [Supported Features](http://docs.google.com/overview/features.html)
  + [Application Flow Chart](http://docs.google.com/overview/appflow.html)
  + [Model-View-Controller](http://docs.google.com/overview/mvc.html)
  + [Architectural Goals](http://docs.google.com/overview/goals.html)
* [Tutorial](http://docs.google.com/tutorial/index.html)
  + [Static pages](http://docs.google.com/tutorial/static_pages.html)
  + [News section](http://docs.google.com/tutorial/news_section.html)
  + [Create news items](http://docs.google.com/tutorial/create_news_items.html)
  + [Conclusion](http://docs.google.com/tutorial/conclusion.html)
* [Contributing to CodeIgniter](http://docs.google.com/contributing/index.html)
  + [Writing CodeIgniter Documentation](http://docs.google.com/documentation/index.html)
  + [Developer’s Certificate of Origin 1.1](http://docs.google.com/DCO.html)
* [General Topics](http://docs.google.com/general/index.html)
  + [CodeIgniter URLs](http://docs.google.com/general/urls.html)
  + [Controllers](http://docs.google.com/general/controllers.html)
  + [Reserved Names](http://docs.google.com/general/reserved_names.html)
  + [Views](http://docs.google.com/general/views.html)
  + [Models](http://docs.google.com/general/models.html)
  + [Helpers](http://docs.google.com/general/helpers.html)
  + [Using CodeIgniter Libraries](http://docs.google.com/general/libraries.html)
  + [Creating Libraries](http://docs.google.com/general/creating_libraries.html)
  + [Using CodeIgniter Drivers](http://docs.google.com/general/drivers.html)
  + [Creating Drivers](http://docs.google.com/general/creating_drivers.html)
  + [Creating Core System Classes](http://docs.google.com/general/core_classes.html)
  + [Creating Ancillary Classes](http://docs.google.com/general/ancillary_classes.html)
  + [Hooks - Extending the Framework Core](http://docs.google.com/general/hooks.html)
  + [Auto-loading Resources](http://docs.google.com/general/autoloader.html)
  + [Common Functions](http://docs.google.com/general/common_functions.html)
  + [Compatibility Functions](http://docs.google.com/general/compatibility_functions.html)
  + [URI Routing](http://docs.google.com/general/routing.html)
  + [Error Handling](http://docs.google.com/general/errors.html)
  + [Caching](http://docs.google.com/general/caching.html)
  + [Profiling Your Application](http://docs.google.com/general/profiling.html)
  + [Running via the CLI](http://docs.google.com/general/cli.html)
  + [Managing your Applications](http://docs.google.com/general/managing_apps.html)
  + [Handling Multiple Environments](http://docs.google.com/general/environments.html)
  + [Alternate PHP Syntax for View Files](http://docs.google.com/general/alternative_php.html)
  + [Security](http://docs.google.com/general/security.html)
  + [PHP Style Guide](http://docs.google.com/general/styleguide.html)
* [Libraries](http://docs.google.com/index.html)
  + [Benchmarking Class](http://docs.google.com/benchmark.html)
  + [Caching Driver](http://docs.google.com/caching.html)
  + [Calendaring Class](http://docs.google.com/calendar.html)
  + [Shopping Cart Class](http://docs.google.com/cart.html)
  + [Config Class](http://docs.google.com/config.html)
  + [Email Class](http://docs.google.com/email.html)
  + [Encrypt Class](http://docs.google.com/encrypt.html)
  + [Encryption Library](http://docs.google.com/encryption.html)
  + [File Uploading Class](http://docs.google.com/file_uploading.html)
  + [Form Validation](http://docs.google.com/form_validation.html)
  + [FTP Class](http://docs.google.com/ftp.html)
  + [Image Manipulation Class](http://docs.google.com/image_lib.html)
  + [Input Class](http://docs.google.com/input.html)
  + [Javascript Class](http://docs.google.com/javascript.html)
  + [Language Class](http://docs.google.com/language.html)
  + [Loader Class](http://docs.google.com/loader.html)
  + [Migrations Class](http://docs.google.com/migration.html)
  + [Output Class](http://docs.google.com/output.html)
  + [Pagination Class](http://docs.google.com/pagination.html)
  + [Template Parser Class](http://docs.google.com/parser.html)
  + [Security Class](http://docs.google.com/security.html)
  + [Session Library](http://docs.google.com/sessions.html)
  + [HTML Table Class](http://docs.google.com/table.html)
  + [Trackback Class](http://docs.google.com/trackback.html)
  + [Typography Class](http://docs.google.com/typography.html)
  + [Unit Testing Class](http://docs.google.com/unit_testing.html)
  + [URI Class](http://docs.google.com/uri.html)
  + [User Agent Class](http://docs.google.com/user_agent.html)
  + XML-RPC and XML-RPC Server Classes
  + [Zip Encoding Class](http://docs.google.com/zip.html)
* [Database Reference](http://docs.google.com/database/index.html)
  + [Quick Start: Usage Examples](http://docs.google.com/database/examples.html)
  + [Database Configuration](http://docs.google.com/database/configuration.html)
  + [Connecting to a Database](http://docs.google.com/database/connecting.html)
  + [Running Queries](http://docs.google.com/database/queries.html)
  + [Generating Query Results](http://docs.google.com/database/results.html)
  + [Query Helper Functions](http://docs.google.com/database/helpers.html)
  + [Query Builder Class](http://docs.google.com/database/query_builder.html)
  + [Transactions](http://docs.google.com/database/transactions.html)
  + [Getting MetaData](http://docs.google.com/database/metadata.html)
  + [Custom Function Calls](http://docs.google.com/database/call_function.html)
  + [Query Caching](http://docs.google.com/database/caching.html)
  + [Database Manipulation with Database Forge](http://docs.google.com/database/forge.html)
  + [Database Utilities Class](http://docs.google.com/database/utilities.html)
  + [Database Driver Reference](http://docs.google.com/database/db_driver_reference.html)
* [Helpers](http://docs.google.com/helpers/index.html)
  + [Array Helper](http://docs.google.com/helpers/array_helper.html)
  + [CAPTCHA Helper](http://docs.google.com/helpers/captcha_helper.html)
  + [Cookie Helper](http://docs.google.com/helpers/cookie_helper.html)
  + [Date Helper](http://docs.google.com/helpers/date_helper.html)
  + [Directory Helper](http://docs.google.com/helpers/directory_helper.html)
  + [Download Helper](http://docs.google.com/helpers/download_helper.html)
  + [Email Helper](http://docs.google.com/helpers/email_helper.html)
  + [File Helper](http://docs.google.com/helpers/file_helper.html)
  + [Form Helper](http://docs.google.com/helpers/form_helper.html)
  + [HTML Helper](http://docs.google.com/helpers/html_helper.html)
  + [Inflector Helper](http://docs.google.com/helpers/inflector_helper.html)
  + [Language Helper](http://docs.google.com/helpers/language_helper.html)
  + [Number Helper](http://docs.google.com/helpers/number_helper.html)
  + [Path Helper](http://docs.google.com/helpers/path_helper.html)
  + [Security Helper](http://docs.google.com/helpers/security_helper.html)
  + [Smiley Helper](http://docs.google.com/helpers/smiley_helper.html)
  + [String Helper](http://docs.google.com/helpers/string_helper.html)
  + [Text Helper](http://docs.google.com/helpers/text_helper.html)
  + [Typography Helper](http://docs.google.com/helpers/typography_helper.html)
  + [URL Helper](http://docs.google.com/helpers/url_helper.html)
  + [XML Helper](http://docs.google.com/helpers/xml_helper.html)

Toggle Table of Contents

[CodeIgniter](http://docs.google.com/index.html)

* [Welcome to CodeIgniter](http://docs.google.com/general/welcome.html)
* [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Downloading CodeIgniter](http://docs.google.com/installation/downloads.html)
  + [Installation Instructions](http://docs.google.com/installation/index.html)
  + [Upgrading From a Previous Version](http://docs.google.com/installation/upgrading.html)
  + [Troubleshooting](http://docs.google.com/installation/troubleshooting.html)
* [CodeIgniter Overview](http://docs.google.com/overview/index.html)
  + [Getting Started](http://docs.google.com/overview/getting_started.html)
  + [CodeIgniter at a Glance](http://docs.google.com/overview/at_a_glance.html)
  + [Supported Features](http://docs.google.com/overview/features.html)
  + [Application Flow Chart](http://docs.google.com/overview/appflow.html)
  + [Model-View-Controller](http://docs.google.com/overview/mvc.html)
  + [Architectural Goals](http://docs.google.com/overview/goals.html)
* [Tutorial](http://docs.google.com/tutorial/index.html)
  + [Static pages](http://docs.google.com/tutorial/static_pages.html)
  + [News section](http://docs.google.com/tutorial/news_section.html)
  + [Create news items](http://docs.google.com/tutorial/create_news_items.html)
  + [Conclusion](http://docs.google.com/tutorial/conclusion.html)
* [Contributing to CodeIgniter](http://docs.google.com/contributing/index.html)
  + [Writing CodeIgniter Documentation](http://docs.google.com/documentation/index.html)
  + [Developer’s Certificate of Origin 1.1](http://docs.google.com/DCO.html)
* [General Topics](http://docs.google.com/general/index.html)
  + [CodeIgniter URLs](http://docs.google.com/general/urls.html)
  + [Controllers](http://docs.google.com/general/controllers.html)
  + [Reserved Names](http://docs.google.com/general/reserved_names.html)
  + [Views](http://docs.google.com/general/views.html)
  + [Models](http://docs.google.com/general/models.html)
  + [Helpers](http://docs.google.com/general/helpers.html)
  + [Using CodeIgniter Libraries](http://docs.google.com/general/libraries.html)
  + [Creating Libraries](http://docs.google.com/general/creating_libraries.html)
  + [Using CodeIgniter Drivers](http://docs.google.com/general/drivers.html)
  + [Creating Drivers](http://docs.google.com/general/creating_drivers.html)
  + [Creating Core System Classes](http://docs.google.com/general/core_classes.html)
  + [Creating Ancillary Classes](http://docs.google.com/general/ancillary_classes.html)
  + [Hooks - Extending the Framework Core](http://docs.google.com/general/hooks.html)
  + [Auto-loading Resources](http://docs.google.com/general/autoloader.html)
  + [Common Functions](http://docs.google.com/general/common_functions.html)
  + [Compatibility Functions](http://docs.google.com/general/compatibility_functions.html)
  + [URI Routing](http://docs.google.com/general/routing.html)
  + [Error Handling](http://docs.google.com/general/errors.html)
  + [Caching](http://docs.google.com/general/caching.html)
  + [Profiling Your Application](http://docs.google.com/general/profiling.html)
  + [Running via the CLI](http://docs.google.com/general/cli.html)
  + [Managing your Applications](http://docs.google.com/general/managing_apps.html)
  + [Handling Multiple Environments](http://docs.google.com/general/environments.html)
  + [Alternate PHP Syntax for View Files](http://docs.google.com/general/alternative_php.html)
  + [Security](http://docs.google.com/general/security.html)
  + [PHP Style Guide](http://docs.google.com/general/styleguide.html)
* [Libraries](http://docs.google.com/index.html)
  + [Benchmarking Class](http://docs.google.com/benchmark.html)
  + [Caching Driver](http://docs.google.com/caching.html)
  + [Calendaring Class](http://docs.google.com/calendar.html)
  + [Shopping Cart Class](http://docs.google.com/cart.html)
  + [Config Class](http://docs.google.com/config.html)
  + [Email Class](http://docs.google.com/email.html)
  + [Encrypt Class](http://docs.google.com/encrypt.html)
  + [Encryption Library](http://docs.google.com/encryption.html)
  + [File Uploading Class](http://docs.google.com/file_uploading.html)
  + [Form Validation](http://docs.google.com/form_validation.html)
  + [FTP Class](http://docs.google.com/ftp.html)
  + [Image Manipulation Class](http://docs.google.com/image_lib.html)
  + [Input Class](http://docs.google.com/input.html)
  + [Javascript Class](http://docs.google.com/javascript.html)
  + [Language Class](http://docs.google.com/language.html)
  + [Loader Class](http://docs.google.com/loader.html)
  + [Migrations Class](http://docs.google.com/migration.html)
  + [Output Class](http://docs.google.com/output.html)
  + [Pagination Class](http://docs.google.com/pagination.html)
  + [Template Parser Class](http://docs.google.com/parser.html)
  + [Security Class](http://docs.google.com/security.html)
  + [Session Library](http://docs.google.com/sessions.html)
  + [HTML Table Class](http://docs.google.com/table.html)
  + [Trackback Class](http://docs.google.com/trackback.html)
  + [Typography Class](http://docs.google.com/typography.html)
  + [Unit Testing Class](http://docs.google.com/unit_testing.html)
  + [URI Class](http://docs.google.com/uri.html)
  + [User Agent Class](http://docs.google.com/user_agent.html)
  + XML-RPC and XML-RPC Server Classes
  + [Zip Encoding Class](http://docs.google.com/zip.html)
* [Database Reference](http://docs.google.com/database/index.html)
  + [Quick Start: Usage Examples](http://docs.google.com/database/examples.html)
  + [Database Configuration](http://docs.google.com/database/configuration.html)
  + [Connecting to a Database](http://docs.google.com/database/connecting.html)
  + [Running Queries](http://docs.google.com/database/queries.html)
  + [Generating Query Results](http://docs.google.com/database/results.html)
  + [Query Helper Functions](http://docs.google.com/database/helpers.html)
  + [Query Builder Class](http://docs.google.com/database/query_builder.html)
  + [Transactions](http://docs.google.com/database/transactions.html)
  + [Getting MetaData](http://docs.google.com/database/metadata.html)
  + [Custom Function Calls](http://docs.google.com/database/call_function.html)
  + [Query Caching](http://docs.google.com/database/caching.html)
  + [Database Manipulation with Database Forge](http://docs.google.com/database/forge.html)
  + [Database Utilities Class](http://docs.google.com/database/utilities.html)
  + [Database Driver Reference](http://docs.google.com/database/db_driver_reference.html)
* [Helpers](http://docs.google.com/helpers/index.html)
  + [Array Helper](http://docs.google.com/helpers/array_helper.html)
  + [CAPTCHA Helper](http://docs.google.com/helpers/captcha_helper.html)
  + [Cookie Helper](http://docs.google.com/helpers/cookie_helper.html)
  + [Date Helper](http://docs.google.com/helpers/date_helper.html)
  + [Directory Helper](http://docs.google.com/helpers/directory_helper.html)
  + [Download Helper](http://docs.google.com/helpers/download_helper.html)
  + [Email Helper](http://docs.google.com/helpers/email_helper.html)
  + [File Helper](http://docs.google.com/helpers/file_helper.html)
  + [Form Helper](http://docs.google.com/helpers/form_helper.html)
  + [HTML Helper](http://docs.google.com/helpers/html_helper.html)
  + [Inflector Helper](http://docs.google.com/helpers/inflector_helper.html)
  + [Language Helper](http://docs.google.com/helpers/language_helper.html)
  + [Number Helper](http://docs.google.com/helpers/number_helper.html)
  + [Path Helper](http://docs.google.com/helpers/path_helper.html)
  + [Security Helper](http://docs.google.com/helpers/security_helper.html)
  + [Smiley Helper](http://docs.google.com/helpers/smiley_helper.html)
  + [String Helper](http://docs.google.com/helpers/string_helper.html)
  + [Text Helper](http://docs.google.com/helpers/text_helper.html)
  + [Typography Helper](http://docs.google.com/helpers/typography_helper.html)
  + [URL Helper](http://docs.google.com/helpers/url_helper.html)
  + [XML Helper](http://docs.google.com/helpers/xml_helper.html)

[CodeIgniter](http://docs.google.com/index.html)

* [Docs](http://docs.google.com/index.html) »
* [Libraries](http://docs.google.com/index.html) »
* XML-RPC and XML-RPC Server Classes

classic layout

XML-RPC and XML-RPC Server Classes[**¶**](#gjdgxs)

CodeIgniter’s XML-RPC classes permit you to send requests to another server, or set up your own XML-RPC server to receive requests.

* [What is XML-RPC?](#30j0zll)
* [Using the XML-RPC Class](#1fob9te)
  + [Initializing the Class](#3znysh7)
  + [Sending XML-RPC Requests](#2et92p0)
    - [Explanation](#tyjcwt)
  + [Anatomy of a Request](#3dy6vkm)
  + [Creating an XML-RPC Server](#1t3h5sf)
  + [Processing Server Requests](#4d34og8)
    - [Notes:](#2s8eyo1)
  + [Formatting a Response](#17dp8vu)
  + [Sending an Error Response](#3rdcrjn)
  + [Creating Your Own Client and Server](#26in1rg)
    - [The Client](#lnxbz9)
    - [The Server](#35nkun2)
    - [Try it!](#1ksv4uv)
  + [Using Associative Arrays In a Request Parameter](#44sinio)
  + [Data Types](#2jxsxqh)
* [Class Reference](#z337ya)

[**What is XML-RPC?**](#3j2qqm3)[**¶**](#30j0zll)

Quite simply it is a way for two computers to communicate over the internet using XML. One computer, which we will call the client, sends an XML-RPC **request** to another computer, which we will call the server. Once the server receives and processes the request it will send back a **response** to the client.

For example, using the MetaWeblog API, an XML-RPC Client (usually a desktop publishing tool) will send a request to an XML-RPC Server running on your site. This request might be a new weblog entry being sent for publication, or it could be a request for an existing entry for editing. When the XML-RPC Server receives this request it will examine it to determine which class/method should be called to process the request. Once processed, the server will then send back a response message.

For detailed specifications, you can visit the [XML-RPC](http://www.xmlrpc.com/) site.

[**Using the XML-RPC Class**](#1y810tw)[**¶**](#1fob9te)

[**Initializing the Class**](#4i7ojhp)[**¶**](#3znysh7)

Like most other classes in CodeIgniter, the XML-RPC and XML-RPCS classes are initialized in your controller using the $this->load->library function:

To load the XML-RPC class you will use:

$this->load->library('xmlrpc');

Once loaded, the xml-rpc library object will be available using: $this->xmlrpc

To load the XML-RPC Server class you will use:

$this->load->library('xmlrpc');  
$this->load->library('xmlrpcs');

Once loaded, the xml-rpcs library object will be available using: $this->xmlrpcs

Note

When using the XML-RPC Server class you must load BOTH the XML-RPC class and the XML-RPC Server class.

[**Sending XML-RPC Requests**](#2xcytpi)[**¶**](#2et92p0)

To send a request to an XML-RPC server you must specify the following information:

* The URL of the server
* The method on the server you wish to call
* The *request* data (explained below).

Here is a basic example that sends a simple Weblogs.com ping to the [Ping-o-Matic](http://pingomatic.com/)

$this->load->library('xmlrpc');  
  
$this->xmlrpc->server('http://rpc.pingomatic.com/', 80);  
$this->xmlrpc->method('weblogUpdates.ping');  
  
$request = array('My Photoblog', 'http://www.my-site.com/photoblog/');  
$this->xmlrpc->request($request);  
  
if ( ! $this->xmlrpc->send\_request())  
{  
 echo $this->xmlrpc->display\_error();  
}

[**Explanation**](#1ci93xb)[**¶**](#tyjcwt)

The above code initializes the XML-RPC class, sets the server URL and method to be called (weblogUpdates.ping). The request (in this case, the title and URL of your site) is placed into an array for transportation, and compiled using the request() function. Lastly, the full request is sent. If the send\_request() method returns false we will display the error message sent back from the XML-RPC Server.

[**Anatomy of a Request**](#3whwml4)[**¶**](#3dy6vkm)

An XML-RPC request is simply the data you are sending to the XML-RPC server. Each piece of data in a request is referred to as a request parameter. The above example has two parameters: The URL and title of your site. When the XML-RPC server receives your request, it will look for parameters it requires.

Request parameters must be placed into an array for transportation, and each parameter can be one of seven data types (strings, numbers, dates, etc.). If your parameters are something other than strings you will have to include the data type in the request array.

Here is an example of a simple array with three parameters:

$request = array('John', 'Doe', 'www.some-site.com');  
$this->xmlrpc->request($request);

If you use data types other than strings, or if you have several different data types, you will place each parameter into its own array, with the data type in the second position:

$request = array(  
 array('John', 'string'),  
 array('Doe', 'string'),  
 array(FALSE, 'boolean'),  
 array(12345, 'int')  
);  
$this->xmlrpc->request($request);

The [Data Types](#2bn6wsx) section below has a full list of data types.

[**Creating an XML-RPC Server**](#qsh70q)[**¶**](#1t3h5sf)

An XML-RPC Server acts as a traffic cop of sorts, waiting for incoming requests and redirecting them to the appropriate functions for processing.

To create your own XML-RPC server involves initializing the XML-RPC Server class in your controller where you expect the incoming request to appear, then setting up an array with mapping instructions so that incoming requests can be sent to the appropriate class and method for processing.

Here is an example to illustrate:

$this->load->library('xmlrpc');  
$this->load->library('xmlrpcs');  
  
$config['functions']['new\_post'] = array('function' => 'My\_blog.new\_entry');  
$config['functions']['update\_post'] = array('function' => 'My\_blog.update\_entry');  
$config['object'] = $this;  
  
$this->xmlrpcs->initialize($config);  
$this->xmlrpcs->serve();

The above example contains an array specifying two method requests that the Server allows. The allowed methods are on the left side of the array. When either of those are received, they will be mapped to the class and method on the right.

The ‘object’ key is a special key that you pass an instantiated class object with, which is necessary when the method you are mapping to is not part of the CodeIgniter super object.

In other words, if an XML-RPC Client sends a request for the new\_post method, your server will load the My\_blog class and call the new\_entry function. If the request is for the update\_post method, your server will load the My\_blog class and call the update\_entry() method.

The function names in the above example are arbitrary. You’ll decide what they should be called on your server, or if you are using standardized APIs, like the Blogger or MetaWeblog API, you’ll use their function names.

There are two additional configuration keys you may make use of when initializing the server class: debug can be set to TRUE in order to enable debugging, and xss\_clean may be set to FALSE to prevent sending data through the Security library’s xss\_clean() method.

[**Processing Server Requests**](#3as4poj)[**¶**](#4d34og8)

When the XML-RPC Server receives a request and loads the class/method for processing, it will pass an object to that method containing the data sent by the client.

Using the above example, if the new\_post method is requested, the server will expect a class to exist with this prototype:

class My\_blog extends CI\_Controller {  
  
 public function new\_post($request)  
 {  
  
 }  
}

The $request variable is an object compiled by the Server, which contains the data sent by the XML-RPC Client. Using this object you will have access to the *request parameters* enabling you to process the request. When you are done you will send a Response back to the Client.

Below is a real-world example, using the Blogger API. One of the methods in the Blogger API is getUserInfo(). Using this method, an XML-RPC Client can send the Server a username and password, in return the Server sends back information about that particular user (nickname, user ID, email address, etc.). Here is how the processing function might look:

class My\_blog extends CI\_Controller {  
  
 public function getUserInfo($request)  
 {  
 $username = 'smitty';  
 $password = 'secretsmittypass';  
  
 $this->load->library('xmlrpc');  
  
 $parameters = $request->output\_parameters();  
  
 if ($parameters[1] != $username && $parameters[2] != $password)  
 {  
 return $this->xmlrpc->send\_error\_message('100', 'Invalid Access');  
 }  
  
 $response = array(  
 array(  
 'nickname' => array('Smitty', 'string'),  
 'userid' => array('99', 'string'),  
 'url' => array('http://yoursite.com', 'string'),  
 'email' => array('jsmith@yoursite.com', 'string'),  
 'lastname' => array('Smith', 'string'),  
 'firstname' => array('John', 'string')  
 ),  
 'struct'  
 );  
  
 return $this->xmlrpc->send\_response($response);  
 }  
}

[**Notes:**](#1pxezwc)[**¶**](#2s8eyo1)

The output\_parameters() method retrieves an indexed array corresponding to the request parameters sent by the client. In the above example, the output parameters will be the username and password.

If the username and password sent by the client were not valid, and error message is returned using send\_error\_message().

If the operation was successful, the client will be sent back a response array containing the user’s info.

[**Formatting a Response**](#49x2ik5)[**¶**](#17dp8vu)

Similar to *Requests*, *Responses* must be formatted as an array. However, unlike requests, a response is an array **that contains a single item**. This item can be an array with several additional arrays, but there can be only one primary array index. In other words, the basic prototype is this:

$response = array('Response data', 'array');

Responses, however, usually contain multiple pieces of information. In order to accomplish this we must put the response into its own array so that the primary array continues to contain a single piece of data. Here’s an example showing how this might be accomplished:

$response = array(  
 array(  
 'first\_name' => array('John', 'string'),  
 'last\_name' => array('Doe', 'string'),  
 'member\_id' => array(123435, 'int'),  
 'todo\_list' => array(array('clean house', 'call mom', 'water plants'), 'array'),  
 ),  
 'struct'  
);

Notice that the above array is formatted as a struct. This is the most common data type for responses.

As with Requests, a response can be one of the seven data types listed in the [Data Types](#2bn6wsx) section.

[**Sending an Error Response**](#2p2csry)[**¶**](#3rdcrjn)

If you need to send the client an error response you will use the following:

return $this->xmlrpc->send\_error\_message('123', 'Requested data not available');

The first parameter is the error number while the second parameter is the error message.

[**Creating Your Own Client and Server**](#147n2zr)[**¶**](#26in1rg)

To help you understand everything we’ve covered thus far, let’s create a couple controllers that act as XML-RPC Client and Server. You’ll use the Client to send a request to the Server and receive a response.

[**The Client**](#3o7alnk)[**¶**](#lnxbz9)

Using a text editor, create a controller called Xmlrpc\_client.php. In it, place this code and save it to your application/controllers/ folder:

<?php  
  
class Xmlrpc\_client extends CI\_Controller {  
  
 public function index()  
 {  
 $this->load->helper('url');  
 $server\_url = site\_url('xmlrpc\_server');  
  
 $this->load->library('xmlrpc');  
  
 $this->xmlrpc->server($server\_url, 80);  
 $this->xmlrpc->method('Greetings');  
  
 $request = array('How is it going?');  
 $this->xmlrpc->request($request);  
  
 if ( ! $this->xmlrpc->send\_request())  
 {  
 echo $this->xmlrpc->display\_error();  
 }  
 else  
 {  
 echo '<pre>';  
 print\_r($this->xmlrpc->display\_response());  
 echo '</pre>';  
 }  
 }  
}  
?>

Note

In the above code we are using a “url helper”. You can find more information in the [*Helpers Functions*](http://docs.google.com/general/helpers.html) page.

[**The Server**](#23ckvvd)[**¶**](#35nkun2)

Using a text editor, create a controller called Xmlrpc\_server.php. In it, place this code and save it to your application/controllers/ folder:

<?php  
  
class Xmlrpc\_server extends CI\_Controller {  
  
 public function index()  
 {  
 $this->load->library('xmlrpc');  
 $this->load->library('xmlrpcs');  
  
 $config['functions']['Greetings'] = array('function' => 'Xmlrpc\_server.process');  
  
 $this->xmlrpcs->initialize($config);  
 $this->xmlrpcs->serve();  
 }  
  
  
 public function process($request)  
 {  
 $parameters = $request->output\_parameters();  
  
 $response = array(  
 array(  
 'you\_said' => $parameters[0],  
 'i\_respond' => 'Not bad at all.'  
 ),  
 'struct'  
 );  
  
 return $this->xmlrpc->send\_response($response);  
 }  
}

[**Try it!**](#ihv636)[**¶**](#1ksv4uv)

Now visit the your site using a URL similar to this:

example.com/index.php/xmlrpc\_client/

You should now see the message you sent to the server, and its response back to you.

The client you created sends a message (“How’s is going?”) to the server, along with a request for the “Greetings” method. The Server receives the request and maps it to the process() method, where a response is sent back.

[**Using Associative Arrays In a Request Parameter**](#32hioqz)[**¶**](#44sinio)

If you wish to use an associative array in your method parameters you will need to use a struct datatype:

$request = array(  
 array(  
 // Param 0  
 array('name' => 'John'),  
 'struct'  
 ),  
 array(  
 // Param 1  
 array(  
 'size' => 'large',  
 'shape'=>'round'  
 ),  
 'struct'  
 )  
);  
  
$this->xmlrpc->request($request);

You can retrieve the associative array when processing the request in the Server.

$parameters = $request->output\_parameters();  
$name = $parameters[0]['name'];  
$size = $parameters[1]['size'];  
$shape = $parameters[1]['shape'];

[**Data Types**](#1hmsyys)[**¶**](#2jxsxqh)

According to the [XML-RPC spec](http://www.xmlrpc.com/spec) there are seven types of values that you can send via XML-RPC:

* *int* or *i4*
* *boolean*
* *string*
* *double*
* *dateTime.iso8601*
* *base64*
* *struct* (contains array of values)
* *array* (contains array of values)

[**Class Reference**](#41mghml)[**¶**](#z337ya)

*class* CI\_Xmlrpc[¶](#2grqrue) initialize([*$config = array()*])[¶](#vx1227)

| Parameters: | * **$config** (*array*) – Configuration data |
| --- | --- |
| Return type: | void |

Initializes the XML-RPC library. Accepts an associative array containing your settings.

server(*$url*[, *$port = 80*[, *$proxy = FALSE*[, *$proxy\_port = 8080*]]])[¶](#3fwokq0)

| Parameters: | * **$url** (*string*) – XML-RPC server URL * **$port** (*int*) – Server port * **$proxy** (*string*) – Optional proxy * **$proxy\_port** (*int*) – Proxy listening port |
| --- | --- |
| Return type: | void |

Sets the URL and port number of the server to which a request is to be sent:

$this->xmlrpc->server('http://www.sometimes.com/pings.php', 80);

Basic HTTP authentication is also supported, simply add it to the server URL:

$this->xmlrpc->server('http://user:pass@localhost/', 80);

timeout(*$seconds = 5*)[¶](#1v1yuxt)

| Parameters: | * **$seconds** (*int*) – Timeout in seconds |
| --- | --- |
| Return type: | void |

Set a time out period (in seconds) after which the request will be canceled:

$this->xmlrpc->timeout(6);

This timeout period will be used both for an initial connection to the remote server, as well as for getting a response from it. Make sure you set the timeout before calling send\_request().

method(*$function*)[¶](#4f1mdlm)

| Parameters: | * **$function** (*string*) – Method name |
| --- | --- |
| Return type: | void |

Sets the method that will be requested from the XML-RPC server:

$this->xmlrpc->method('method');

Where method is the name of the method.

request(*$incoming*)[¶](#2u6wntf)

| Parameters: | * **$incoming** (*array*) – Request data |
| --- | --- |
| Return type: | void |

Takes an array of data and builds request to be sent to XML-RPC server:

$request = array(array('My Photoblog', 'string'), 'http://www.yoursite.com/photoblog/');  
$this->xmlrpc->request($request);

send\_request()[¶](#19c6y18)

| Returns: | TRUE on success, FALSE on failure |
| --- | --- |
| Return type: | bool |

The request sending method. Returns boolean TRUE or FALSE based on success for failure, enabling it to be used conditionally.

display\_error()[¶](#3tbugp1)

| Returns: | Error message string |
| --- | --- |
| Return type: | string |

Returns an error message as a string if your request failed for some reason.

echo $this->xmlrpc->display\_error();

display\_response()[¶](#28h4qwu)

| Returns: | Response |
| --- | --- |
| Return type: | mixed |

Returns the response from the remote server once request is received. The response will typically be an associative array.

$this->xmlrpc->display\_response();

send\_error\_message(*$number*, *$message*)[¶](#nmf14n)

| Parameters: | * **$number** (*int*) – Error number * **$message** (*string*) – Error message |
| --- | --- |
| Returns: | XML\_RPC\_Response instance |
| Return type: | XML\_RPC\_Response |

This method lets you send an error message from your server to the client. First parameter is the error number while the second parameter is the error message.

return $this->xmlrpc->send\_error\_message(123, 'Requested data not available');

[Next](http://docs.google.com/zip.html)   [Previous](http://docs.google.com/user_agent.html)

© Copyright 2014 - 2016, British Columbia Institute of Technology. Last updated on Oct 28, 2016.

Built with [Sphinx](http://sphinx-doc.org/) using a [theme](https://github.com/snide/sphinx_rtd_theme) provided by [Read the Docs](https://readthedocs.org).