Recite CMS Web Services PHP Client Guide

Recite CMS Web Services Client

Recite CMS Web Services PHP Client Guide

Table of Contents

1. Getting Started 1
    Adding the Bundled Library to Your Application 1
        Using the Bundled Library 1
        Handling Exceptions 3
        Using the Bundled Test Suite 4
        API Documentation 4
2. Class Mappings 5
    WSDL Data Type Mappings 5
3. Sample Web Service Usage 7

List of Examples

1.1. Setting Web Service URL and Credentials 1
1.2. Calling the `getReciteVersion()` Web Service Method 2
1.3. Retrieving a List of Available Web Services 2
1.4. Using the `Factory` method 2
1.5. Handling Exceptions 3
1.6. Handling Exceptions 4
1.7. Sample Apache Configuration for Web Services Client 4
2.1. Using Mapped Classes 5
3.1. Sample Web Service Usage 7

Chapter 1. Getting Started

The simplest way to consume Recite web services is to use the bundled PHP 5 library. If you are not using PHP or don't want to use the bundled library most of this section won't apply to you. However, you can access the main web service WSDL descriptor file to discover which services and methods are available at the following URL (using our example web site URL of `http://xyz.example.com`):

- `http://xyz.example.com/__/webservices?wsdl`

Adding the Bundled Library to Your Application

To use the bundled library, copy the `./lib/Recite` directory into your application include directory. For example, if your application's include directory is `/var/www/wsdemo/include`, then the bundled library would be found in `/var/www/wsdemo/include/Recite`.

In this example, the PHP include path would be `/var/www/wsdemo/include`.

> **Tip**
>
> The Apache directive to achieve this is `php_value include_path /var/www/wsdemo/include`.

Using the Bundled Library

To make use of the bundled library, you must first include the `Recite_Webservices` class (located in `./Recite/Webservices.php`). This is a static class you use to set your username, password and web site URL, as well as to build a `SoapClient` object to access the required service.

Following is a script that shows how this achieved, using the example URL and credentials mentioned earlier in this guide.

Example 1.1. Setting Web Service URL and Credentials

```
<?php
    require_once('Recite/Webservices.php');

    Recite_Webservices::SetCredentials('wsuser', 'wspass');
    Recite_Webservices::SetUrl('http://xyz.example.com');
?>
```

To retrieve a configured instance of `SoapClient` with which to access web services, call the `Factory()` method on the `Recite_Webservices` class. This method takes a single argument, which is the name of the service to access. If no service is specified, the default web service is used. This service can be used to tell you which other services are available.

> **Caution**
>
> You must set the credentials and URL prior to calling the `Recite_Webservices::Factory()` method. The `Recite_Webservices_Exception` will be thrown when you call `Factory()` otherwise.

For example, to access the default web service and then call the `getReciteVersion()` method on it, you would use the following code:

Example 1.2. Calling the `getReciteVersion()` Web Service Method

```
<?php
    require_once('Recite/Webservices.php');

    Recite_Webservices::SetCredentials('wsuser', 'wspass');
    Recite_Webservices::SetUrl('http://xyz.example.com');

    $service = Recite_Webservices::Factory();

    echo $service->getReciteVersion();
?>
```

You can also see the other available services by calling `getServices()` on the default service (index). This will return an `array` with the names of the other services.

Example 1.3. Retrieving a List of Available Web Services

```
<?php
    require_once('Recite/Webservices.php');

    Recite_Webservices::SetCredentials('wsuser', 'wspass');
    Recite_Webservices::SetUrl('http://xyz.example.com');

    $service = Recite_Webservices::Factory();

    var_dump($service->getServices());
?>
```

This would result in the following output:

```
array
    0 => string 'pages' (length=5)
    1 => string 'users' (length=5)
    2 => string 'ecommerce' (length=9)
    3 => string 'assets' (length=6)
    4 => string 'search' (length=6)
    5 => string 'forms' (length=5)
    6 => string 'calendar' (length=8)
    7 => string 'listings' (length=8)
    8 => string 'templates' (length=9)
    9 => string 'index' (length=5)
    10 => string 'auth' (length=4)
```

You can use any of these values to access the respective web service. For instance, to use the Page Manager web service, you would pass pages to the `Factory()` method.

Example 1.4. Using the `Factory` method

```
<?php
    // ...

    $service = Recite_Webservices::Factory('pages');
?>
```

> **Note**
>
> The index web service is the default web service. Meaning if you don't pass any argument to `Factory()` it is the same as passing *`index`*.

To determine which methods and data types are available for each web service, you can either browse the WSDL file directly, or you can call the factory method for the required service then call `$service->__getFunctions()` or `$service->__getTypes()`. Alternatively, you can use the bundled test suite to browse and try the available methods on each service (see later in this chapter).

> **Note**
>
> These functions are documented at http://php.net/manual/en/function.soap-soapclient-getfunctions.php and http://php.net/manual/en/function.soap-soapclient-gettypes.php respectively.

Handling Exceptions

When using both the `Recite_Webservices` class and the `SoapClient` class (for calling web service functions), exceptions are thrown if any operations fail.

If an error occurs using `Recite_Webservices` (or one of the bundled type classes), the `Recite_Webservices_Exception` is thrown.

If an error occurs in calling a SOAP method, the built-in `SoapFault` exception type is thrown.

For example, an error setting the client URL:

Example 1.5. Handling Exceptions

```
<?php
    // ...

    try {
        Recite_Webservices::SetUrl('12345');
    }
    catch (Exception $ex) {
        die('Error setting the client URL: ' . $ex->getMessage());
    }

    // ...
?>
```

An example of an error occurring using the web service (in this example a non-existent method is called):

Example 1.6. Handling Exceptions

```
<?php
    // ...

    try {
        $service->madeUpMethod();
    }
    catch (Exception $ex) {
        die('Error in SOAP request: ' . $ex->getMessage());
    }

    // ...
?>
```

Using the Bundled Test Suite

It is highly recommended that if you're going to use the bundled PHP library to access Recite web services that you first install the test suite to help become familiar with the services available. There are many sample scripts available that show you how to build requests and handle returned data nicely.

Let's assume you install the test suite in `/var/www/wstest`. You would need to set up `/var/www/wstest/htdocs` as the web root, and the include path as both `/var/www/wstest/include` and `/var/www/wstest/lib`.

> **Note**
>
> The include path is split up into two directories so you can easily copy the `./lib/Recite` directory into your application.

A sample Apache configuration `<VirtualHost>` is as follows:

Example 1.7. Sample Apache Configuration for Web Services Client

```
<VirtualHost *:80>
    ServerName wstest.example.com
    DocumentRoot /var/www/wstest/htdocs

    php_value include_path /var/www/wstest/include:/var/www/wstest/lib
</VirtualHost>
```

> **Note**
>
> You'll need to update the server name and paths to suit your own environment.

Once you've set up this host and restarted your web server, you can view the test suite by visiting `http://wstest.example.com` in your browser (of course substituting in your own host name).

API Documentation

The test suite comes bundled with full API documentation for the provided data types. These data types map directly back to the types returned by the Recite web services. The API guide also contains extensive description of the data available. Assuming you've set up the test suite as described in the previous section, you can visit `http://wstest.example.com/api` to view the API.

Chapter 2. Class Mappings

In addition to primitive data types (such as integers or strings) Recite web services uses a number of custom data types. Each custom type that is used is specified in the WSDL of the web service that uses it. In order to make use of these custom types we must map the type to a local PHP class.

For example, if you call the `getPage()` method of the pages web service, according to the WSDL the type returned is `Module_Pages_Webservies_Page`. In order to easily make use of the data included in this data type, we create a local PHP to map this class.

Fortunately, the bundled class does this for us automatically. In this particular example, we have a PHP class called `Recite_Pages_Page` (located in `./Recite/Pages/Page.php`). When we use the `Recite_Webservices::factory('pages')` call to use the pages web service it will automatically create the mapping between the PHP class and the custom type in the WSDL.

Class mapping does not only correspond to data returned from web services; many of the methods in Recite web services accept complex types as arguments to the function call. For example, the pages method `getPage()` accepts an instance of `Module_Pages_Webservices_Options_GetPages`. The bundled client automatically maps this to the local class `Recite_Pages_Options_GetPages`.

Here's an example of using the mapped classes:

Example 2.1. Using Mapped Classes

```
<?php
    require_once('Recite/Webservices.php');

    // ... set the end point and authentication

    // create the SOAP client
    $client = Recite_Webservices::Factory('pages');

    // build the options for the getPages() call
    $options = new Recite_Pages_Options_GetPages();

    // retrieve the pages, an array of Recite_Pages_Page
    $pages = $client->getPages($options);
?>
```

You don't even need to reference the type name as it appears in the WSDL file. You will however need to know which class to use based on the mappings.

WSDL Data Type Mappings

WSDL Data Type	Mapped PHP Class Name
Module_Assets_Webservices_Options_GetFiles	Recite_Assets_Options_GetFiles
Module_Assets_Webservices_File	Recite_Assets_File
Module_Assets_Webservices_GetFilesResult	Recite_Assets_GetFilesResult
Module_Webservices_Auth_AuthenticateRequest	Recite_Auth_AuthenticateRequest
Module_Webservices_Auth_AuthenticateResponse	Recite_Auth_AuthenticateResponse
Module_Calendar_Webservices_Event_List	Recite_Calendar_Event_List
Module_Calendar_Webservices_Event_Type	Recite_Calendar_Event_Type
Module_Calendar_Webservices_Options_GetCalendars	Recite_Calendar_Options_GetCalendars
Module_Calendar_Webservices_Options_GetEvents	Recite_Calendar_Options_GetEvents
Module_Calendar_Webservices_Calendar	Recite_Calendar_Calendar
Module_Calendar_Webservices_Category	Recite_Calendar_Category
Module_Calendar_Webservices_DateRange	Recite_Calendar_DateRange

WSDL Data Type	Mapped PHP Class Name
Module_Calendar_Webservices_Event	Recite_Calendar_Event
Module_Ecommerce_Webservices_Options_GetCategories	Recite_Ecommerce_Options_GetCategories
Module_Ecommerce_Webservices_Options_GetProducts	Recite_Ecommerce_Options_GetProducts
Module_Ecommerce_Webservices_Product_Type	Recite_Ecommerce_Product_Type
Module_Ecommerce_Webservices_Category	Recite_Ecommerce_Category
Module_Ecommerce_Webservices_GetProductsResult	Recite_Ecommerce_GetProductsResult
Module_Ecommerce_Webservices_Product	Recite_Ecommerce_Product
Module_Forms_Webservices_Element_Container	Recite_Forms_Element_Container
Module_Forms_Webservices_FieldList	Recite_Forms_FieldList
Module_Forms_Webservices_FieldList_Field	Recite_Forms_FieldList_Field
Module_Forms_Webservices_Processor_Options	Recite_Forms_Processor_Options
Module_Forms_Webservices_Element	Recite_Forms_Element
Module_Forms_Webservices_Error	Recite_Forms_Error
Module_Forms_Webservices_Form	Recite_Forms_Form
Module_Forms_Webservices_Response	Recite_Forms_Response
Module_Listings_Webservices_Listing_Type	Recite_Listings_Listing_Type
Module_Listings_Webservices_Options_GetCategories	Recite_Listings_Options_GetCategories
Module_Listings_Webservices_Options_GetListings	Recite_Listings_Options_GetListings
Module_Listings_Webservices_Category	Recite_Listings_Category
Module_Listings_Webservices_GetListingsResult	Recite_Listings_GetListingsResult
Module_Listings_Webservices_Listing	Recite_Listings_Listing
Module_Pages_Webservices_Options_GetPages	Recite_Pages_Options_GetPages
Module_Pages_Webservices_Page	Recite_Pages_Page
Module_Search_Webservices_Options_Search	Recite_Search_Options_Search
Module_Search_Webservices_SearchResult	Recite_Search_SearchResult
Module_Search_Webservices_SearchResult_Item	Recite_Search_SearchResult_Item
Module_Search_Webservices_SearchIndex	Recite_Search_SearchIndex
Module_Templates_Webservices_Options_GetTemplates	Recite_Templates_Options_GetTemplates
Module_Templates_Webservices_Template	Recite_Templates_Template
Module_Users_Webservices_Options_GetUsers	Recite_Users_Options_GetUsers
Module_Users_Webservices_User_Directory	Recite_Users_User_Directory
Module_Users_Webservices_GetUsersResult	Recite_Users_GetUsersResult
Module_Users_Webservices_User	Recite_Users_User
Module_Webservices_Types_ContentBlock	Recite_Webservices_Types_ContentBlock
Module_Webservices_Types_Pager	Recite_Webservices_Types_Pager

Chapter 3. Sample Web Service Usage

This section contains a sample usage of web services to show you how most web service calls work. In this example we're going to retrieve a list of files using the assets web service.

The algorithm for achieving this is as follows:

1. Build the options used for retrieving the list of files. We use the `Recite_Assets_Options_GetFiles` class for this.
2. Create the `SoapClient` object used for accessing the assets web service. We use the `Recite_Webservices::Factory()` method for this.
3. Call the `getFiles()` method on the SOAP client.
4. Use returned files as required. The returned files are contained in an instance of `Recite_Assets_GetFilesResult`.

Example 3.1. Sample Web Service Usage

```
<?php
    require_once('Recite/Webservices.php');

    // set credentials
    Recite_Webservices::SetCredentials('wsuser', 'wspass');
    Recite_Webservices::SetUrl('http://xyz.example.com');

    // build the SoapClient object for assets
    $client = Recite_Webservices::Factory('assets');

    $options = new Recite_Assets_Options_GetFiles();

    // return all files
    $options->includeSubFolders = true;

    // return paged results with a limit of 10 per page
    $options->paged = true;
    $options->pageNumber = 1;
    $options->limit = 10;

    // perform the web service method
    $result = $client->getFiles($options);

    // use the returned data
    foreach ($result->files as $file) {
        // output file details
    }

    // you can also output paging data from $result->pager
?>
```