Recite CMS Web Services Development Guide

Recite CMS 1.7.4

Recite CMS Web Services Development Guide

Table of Contents

Chapter 1. Getting Started

This document is a guide to using web services with Recite CMS. The fundamental idea behind using web services is that you create, update and manage your content via the Recite Control Panel (just as you would when not using web services), then access that data using web services.

The mechanism Recite uses for its web services is SOAP. Your language of choice must have a SOAP implementation in order to be able to consume these web services.

There is a complete sample library and test suite provided for use with PHP 5.2+. This library relies on the `SoapClient` class bundled with PHP. The PHP client library has separate documentation associated with it on how to interact with the web services described in this document.

Nearly all Recite web services are read-only, meaning that you can only consume data and not create or update data. The only exception to this is in form handling, in that any forms created in the Recite Form Manager can be used in Web Services.

This guide assumes Recite has already been installed and a client has been set up in the Recite Administration Site. For the purposes of this guide we'll assume the client is called XYZ Company and the URL being managed by Recite is `http://xyz.example.com`.

Chapter 2. Setting up Recite for Web Services

Before you can begin consuming the web services provided by Recite you must set up your Recite installation accordingly.

The key steps involved in this include:

- Installing the Web Services module and associated web service drivers for each of Recite's other modules (where available)
- Creating a user account with which to access web services. All web services are password protected and require a user with sufficient permissions to access them.
- Configuring an assets mirror where files uploaded to Recite will be stored.

Installing Web Services

The web services module should be installed from the Recite library into the `./lib/modules/webservices` directory.

Additionally, the `./lib/modules/webservices` must also be installed from the Recite library. This directory contains a number of different drivers, each of which is responsible for communicating with another Recite module (such as the file manager).

Only the drivers that relate to modules you have installed should be included. For example, if your Recite installation doesn't include the Listings module, then the listings web services driver should not be included.

Once the Web Services module has been installed it must be added to the client in the Recite Administration site.

Creating a Web Services User Account

In order to access web services you need a Recite Control Panel user with the appropriate permissions. This is achieved by going to the Users tab in Recite. Once the User Manager is loaded, click the Control Panel Users tab in the left frame. You can then either create a new user role, or modify an existing one.

To create a new role, click User Roles in the left frame. Next, enter the name of the new rule (such as `Web Services`) in the form in the right-hand frame. This role doesn't need to inherit from any other role. You can then click Create Role.

After the role has been created, the list of roles will refresh showing the newly created role. Click on the new role to display a list of available permissions. From the modules drop-down, select Web Services then click Assign Module. Any users you now create using the `Web Services` user role will have access to the web services.

Now we will create a new user. Click on the Control Panel Users link in the left frame. This will then show a list of users that can access the control panel. Click the Create User tab in the right frame to create a new user. Firstly, select the newly created role from the User Role drop-down. Next, enter a username and password. For the sake of this guide I will assume you have entered `wsuser` for the username and `wspass` for the password. You can skip the other fields if you like. Click Save User to create the user account.

At this point you have created a user account with adequate access to use the web services. Later in this guide I will show you where you use the username and password.

Configuring an Assets Mirror

Since the point of web services is to recreate a site using the remote data, you need some way to access data for uploaded files (it would be extremely inefficient and difficult to send file data over web services).

Therefore, you need to use the Recite asset mirroring functionality in order to send assets to the location of the web services client.

When an asset is created, modified or deleted, each of the asset mirrors are notified accordingly. Currently there is only one mirror type, which is the "File System" mirror.

Tip

Since the web services client will typically not be on the same server as your Recite installation, the File System mirror driver on its own may not be sufficient. If you want to mirror files using SSH/SFTP or FTP then Fuse (or similar) should be used. More details can be found at `http://ubuntu.wordpress.com/2005/10/28/how-to-mount-a-remote-ssh-filesystem-using-sshfs/`. Alternatively, if the server resides on the same network you may be able to mount a network drive in a similar fashion.

To create an asset mirror, go the Admin tab in the Recite Control Panel. In the left frame under the Configuration heading, select File Mirrors.

Note

If you don't see this heading check your permissions for the File Manager module.

Next, click the Create Mirror tab. You will then prompted to choose a mirror type. From here, select File System and click Create Mirror. You will then be required to enter a title by which to identify this mirror, and also the local file system path where the assets will be mirrored.

When you've entered these options, click Save File Mirror. If the path is valid (and is writable), the new asset mirror will be saved. After the mirror is saved, click the Synchronise With Mirror button to send existing files to the selected path.

Now whenever a new file is uploaded, it will be sent to the asset mirror. If a file is deleted from the File Manager it will also be deleted from the mirror.

Setting up the Client Site

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`

There are several different web services available (as you can determine by calling the `getServices` method on the above web service). You can access these using the following URL:

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

For example, if you want to use the pages web service, the end point would be:

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

For full details on using the bundled PHP 5 library refer to the document "Recite Web Services PHP Client Guide".

Chapter 3. Web Service: index

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

This is the default web service used to get information about the Recite installation and about other available web services.

getReciteVersion

- Returns: `string`

This method returns the version of Recite that you're connected to with Web Services.

getReciteTitle

- Returns: `string`

This method returns a formatted description of the Recite version you're connected to.

getSystemSummary

- Returns: `string`

This method returns a description of the system software Recite is running on.

getClientName

- Returns: `string`

This method returns the internal name of the client as created in Recite.

getClientId

- Returns: `int`

This method returns the internal client ID number of the Recite client.

getServices

- Returns: `Array of string`

This method returns a list of the available web services that can be connected to using Recite Web Services.

Chapter 4. Web Service: assets

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

This web service is used to retrieve data about files uploaded to the file manager.

getFiles

- Argument 1: `Module_Assets_Webservices_Options_GetFiles`

  The list of options by which to retrieve files

- Returns: `Module_Assets_Webservices_GetFilesResult`

This method returns a list of pages based on the options specified in the first argument.

getFile

- Argument 1: `int`

  The internal Recite file ID of the file to load

- Returns: `Module_Assets_Webservices_File`

This method returns a single file, based on its ID.

getFileByPath

- Argument 1: `string`

  The path of the file to return

- Returns: `Module_Assets_Webservices_File`

This method returns a single file, based on its path within the Recite file manager.

Chapter 5. Web Service: calendar

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

This web service is used to retrieve calendar events.

getCategories

- Argument 1: `Module_Calendar_Webservices_Options_GetCategories`

  The options to determine which categories to return

- Returns: `Array of Module_Calendar_Webservices_Category`

This method returns a list of event categories based on the specified options.

getCalendars

- Argument 1: `Module_Calendar_Webservices_Options_GetCalendars`

  The options to determine which calendars to return

- Returns: `Array of Module_Calendar_Webservices_Calendar`

This method returns a list of calendars that exist in Recite.

getEvents

- Argument 1: `Module_Calendar_Webservices_Options_GetEvents`

  The options to determine which events to return

- Returns: `Module_Calendar_Webservices_Event_List`

This method returns a list of events based on provided options.

getCalendar

- Argument 1: `int`

  The ID of the calendar to return

- Returns: `Module_Calendar_Webservices_Calendar`

This method returns details about a single calendar.

getEvent

- Argument 1: `int`

  The ID of the event to return

- Returns: `Module_Calendar_Webservices_Event`

This method returns details about a single event.

getEventTypes

- Returns: `Array of Module_Calendar_Webservices_Event_Type`

This method returns all available event types.

Chapter 6. Web Service: ecommerce

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

This web service is used to interact with the ecommerce module. This currently supports only reading product information.

getCategories

- Argument 1: `Module_Ecommerce_Webservices_Options_GetCategories`

  The options to determine which categories to return

- Returns: `Array of Module_Ecommerce_Webservices_Category`

This method returns a list of product categories based on the specified options.

getProducts

- Argument 1: `Module_Ecommerce_Webservices_Options_GetProducts`

  The options to determine which products to return

- Returns: `Module_Ecommerce_Webservices_GetProductsResult`

This method returns a list of products based on the specified options.

getProduct

- Argument 1: `int`

  The ID of the product to return

- Returns: `Module_Ecommerce_Webservices_Product`

This method returns a single product based on the ID supplied.

getProductTypes

- Returns: `Array of Module_Ecommerce_Webservices_Product_Type`

This method returns a list of product types that are available.

Chapter 7. Web Service: forms

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

This web service is used to submit form data to Recite. For example, if you create an email form in Recite you can submit data to this form using this web service.

getForms

- Returns: `Array of Module_Forms_Webservices_Form`

This method returns all forms that have been created in Recite.

processForm

- Argument 1: `Module_Forms_Webservices_Processor_Options`

  The options to use for processing the form, such as submitted form values.

- Returns: `Module_Forms_Webservices_Response`

This method will process a form based on the supplied values. If any errors occur processing the form (such as invalid values), the errors will be supplied in the response data.

Chapter 8. Web Service: listings

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

This web service is used to retrieve data from the lisings module. This includes retrieving listings, listing types and categories.

getCategories

- Argument 1: `Module_Listings_Webservices_Options_GetCategories`

  The options to determine which categories to return

- Returns: `Array of Module_Listings_Webservices_Category`

This method returns a list of categories based on the specified options.

getListings

- Argument 1: `Module_Listings_Webservices_Options_GetListings`

  The options to determine which listings to return

- Returns: `Module_Listings_Webservices_GetListingsResult`

This method returns listings based on the specified options.

getListing

- Argument 1: `int`

  The ID of the listing to return

- Returns: `Module_Listings_Webservices_Listing`

This method returns a single listing based on the supplied listing ID.

getListingTypes

- Returns: `Array of Module_Listings_Webservices_Listing_Type`

This method returns all available listing types.

Chapter 9. Web Service: pages

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

This web service is used to retrieve pages created in the Page Manager.

getPages

- Argument 1: `Module_Pages_Webservices_Options_GetPages`

  The options to determine which pages to return

- Returns: `Array of Module_Pages_Webservices_Page`

This method returns a list of pages based on the specified options.

getPage

- Argument 1: `int`

  The ID of the page to return

- Returns: `Module_Pages_Webservices_Page`

This method returns a single page from the Recite page manager based on its ID.

getPageByUrl

- Argument 1: `string`

  The URL of the page to return

- Returns: `Module_Pages_Webservices_Page`

This method returns a single page from the Recite page based on its URL.

Chapter 10. Web Service: search

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

This web service is used to perform searches on web site content. Search indexes are created and managed in Recite. Using this web service you can query any or all of these services.

getSearchIndexes

- Returns: `Array of Module_Search_Webservices_SearchIndex`

This method returns a list of search indexes that can be searched.

search

- Argument 1: `Module_Search_Webservices_Options_Search`

  The options used to perform the search, such as which indexes to search and the search query.

- Returns: `Module_Search_Webservices_SearchResult`

This method performs a search and returns a list of results.

Chapter 11. Web Service: templates

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

This web service is used to retrieve templates from the template manager. The templates in Recite are validated as Smarty templates when created but theoretically you could store templates in any required format.

getTemplate

- Argument 1: `int`

  The ID of the template to return

- Returns: `Module_Templates_Webservices_Template`

This method returns a single template based on its ID.

getTemplateByPath

- Argument 1: `string`

  The path of the template to return

- Returns: `Module_Templates_Webservices_Template`

This method returns a template based on its path.

getTemplates

- Argument 1: `Module_Templates_Webservices_Options_GetTemplates`

  The options to determine which templates to return

- Returns: `Array of Module_Templates_Webservices_Template`

This method returns a list of templates based on the specified options.

Chapter 12. Web Service: users

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

This web service is used to retrieve user data with Recite. Users can be created with the forms web service, or they can be created in Recite.

getUsers

- Argument 1: `Module_Users_Webservices_Options_GetUsers`
- Returns: `Module_Users_Webservices_GetUsersResult`

This method returns a list of users based on the specified options.

getUser

- Argument 1: `int`

  The ID of the user to return

- Returns: `Module_Users_Webservices_User`

This method retrieves a user based on its ID.

getDirectories

- Returns: `Array of Module_Users_Webservices_User_Directory`

This method returns a list of user directories created in the User Manager.

Chapter 13. Web Service: auth

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

This web service is used to authenticate users on the client site. For users to authenticate, they must be part of a user directory created in the control panel for the given client.

Additionally, user functions such as logging in, updating details or resetting password must all have a corresponding form created the control panel.

Note

For a user to log in, there must be a log in form present in the control panel, however you cannot use the forms web service to log them in. You must use the auth web service as outlined in this chapter to authenticate users.

When a user successfully logs in, an authentication token is returned which must be remembered by the client site for future requests. The token is included in subsequent requests using a cookie called `authToken`.

Note

Currently the only web service that requires presence of this authentication cookie is the update details form.

The bundled PHP library has built-in support for authentication and sending the authentication cookie in web service requests.

Important

If you're using the bundled PHP client you must enable sessions prior to using the bundled authentication manager. You can do so using `session_start()`. If you try to use authentication without starting sessions an exception will be thrown.

Tokens automatically expire if they are unused for more than an hour. Every time a token is passed using the `authToken` cookie, or as an argument to the `isValidToken()` or `getIdentity()` methods the time stamp of the token is updated (thereby preventing expiry).

authenticate

- Argument 1: `Module_Webservices_Auth_AuthenticateRequest`

  The authentication data including username and password

- Returns: `Module_Webservices_Auth_AuthenticateResponse`

This method is used to authenticate a user. For this method to work you must have at least one log in form created in the control panel. The form defines which user directory the authenticating user must belong to. If you have multiple log in forms you can specify which one to use with the `formId` property of the request.

If authentication was successful, the return data will contain the authentication token and identity data of the authenticated user which the web service client site must then store and handle as required.

If authentication was not successful then the return data will contain one or more error messages.

isValidToken

- Argument 1: `string`

  The token to check

- Returns: `boolean`

  True if the token is valid, false if not

This method checks whether or not a token is valid. For the token to validate, it must exist in the list of authenticate users and must also not be expired.

getIdentity

- Argument 1: `string`

  The token to retrieve the identity for

- Returns: `Module_Users_Webservices_User`

This method returns the identity of the user that corresponds to the given token (assuming the token is invalid). If you are storing the identity of the user locally (such as in session, as the bundled PHP library does), you should update your local storage when you receive the data from this method.

Tip

If you update a user's details with the user details form, you should call this method afterwards so your locally stored data accurately reflects the data stored in Recite.

---

destroyToken

- Argument 1: `string`

  The token to destroy

This method destroys a token, rendering it no longer usable. This is the effectively the same as logging out a user.