



Atrixware Weblearning 9.6

API Documentation

Contents

Atrixware Weblearning 9.6 API Overview	4
Course Administrator Functions	6
get_course_admin_count.....	6
get_course_admin_list.....	6
get_course_admin_record	6
Course Functions.....	7
get_course_count	7
get_course_list.....	7
get_course_record.....	7
remove_course	7
Module Functions	8
get_module_count	8
get_module_list.....	8
get_module_list_from_course.....	8
get_module_record.....	8
Slide Functions	9
get_slide_count.....	9
get_slide_list	9
get_slide_record	9
User Functions	10
add_user.....	10
add_user_into_course	10
add_user_into_system.....	11
add_user_into_usergroup	11
get_user_count	12
get_user_list.....	12
get_user_list_from_course.....	12
get_user_list_from_usergroup	12
get_user_record.....	12
remove_user.....	13
remove_user_from_course	13
remove_user_from_system	13
remove_user_from_usergroup.....	13
update_user	14
Usergroup Functions.....	15
add_usergroup.....	15
get_user_list_from_usergroup	15
get_usergroup_count.....	15
get_usergroup_list	15
remove_usergroup.....	16
Reporting Functions	17
get_course_scores.....	17
get_item_activity	17
get_quiz_scores.....	17
get_user_activity.....	17
get_user_course_results.....	18

get_user_quiz_results.....	18
get_user_scores	18
Other Functions	19
get_api_version.....	19
PHP Sample Code.....	20
Example 1.....	21
Example 2.....	21

Atrixware Weblearning 9.6 API Overview

This document covers the API functionality for the **Atrixware Weblearning 9.6** System (*Professional and Enterprise Systems Only*).

The Weblearning API is located here: <http://your.server.name/lms/api.php>

The minimum parameters (via POST or GET) are as follows:

- action = **apicall** (*literal*)
- apicall = *{name of api function}*
- account = *{course admin account id}*
- pass = *{course admin account password}*

The optional parameters (which differ for each api function) are as follows:

- p1 = parameter 1 value (optional)
- p2 = parameter 2 value (optional)
- p3 = parameter 3 value (optional)
- p4 = parameter 4 value (optional)
- p5 = parameter 5 value (optional)

Simple Example (querystring):

<http://your.server.name/lms/api.php?action=apicall&apicall=showemail&account=samplouser&pass=12345>

In the above example, it is calling the api on the **your.server.name** server, the action is **apicall**, the apicall is the function **showemail** (*not an actual api function, just for demonstration*), the course admin account id is **samplouser**, and the course admin password is **12345**.

Some functions will require **p1**, p2, p3, p4 and/or **p5** parameters to have values (*each API function description will indicate what is required, if anything*).

Each API call will return an XML result inside a top node named **<api_result>**, and often (*if not otherwise specified*) in a node named **<result>**, resembling this:

```
<api_result>
  <result>The Result</result>
</api_result>
```

“The Result” can be the data you asked for, or simply a Boolean **true** or **false** result (*to tell you if the function accomplished the task*).

For example, a function like **get_course_count** will return a number (*indicating number of courses*), but a function like **add_user_to_system** will return **true** or **false** (*indicating whether user was added or not*).

For some functions, the XML returned also includes what we refer to as a *simpleXML* text section. This is a section that is XML-like, but not actually valid XML. It's easy to parse through though – have a look at the PHP sample code section for a wrapper function you can use to read that data (along with a few examples).

The API Functions are separated in this documentation based on what part of the Weblearning system they access:

- Course Admin Functions
- Course Functions
- Module Functions
- Slide Functions
- User Functions
- Usergroup Functions
- Reporting Functions
- Other Functions

Note on the Weblearning 9.5 API functions

If you have used the Weblearning 9.5 API, you will notice that everything from that API no longer exists in the 9.6 API. Good news – all 9.5 functions still exist in the 9.6 API, but none of them are documented (as to steer users towards the newer API functions).

Course Administrator Functions

get_course_admin_count

Returns count – total number of course admin accounts on the system.

get_course_admin_list

Returns list of course admins on the system – each item is the account name.

get_course_admin_record

Parameters Needed:

- p1 - account name

Returns **entire account record** as it exists in the database (*as multiple XML nodes*).

Course Functions

get_course_count

Parameters Needed:

- p1 - account name

Returns total number of courses in specified account.

get_course_list

Parameters Needed:

- p1 - account name

Returns list of courses (as course names) in specified account.

get_course_record

Parameters Needed:

- p1 - account name
- p2 - course name

Returns XML dataset with complete course information.

remove_course

Parameters Needed:

- p1 - account name
- p2 - course name

This will remove *course* named **p2** from *course admin* named **p1**.

The return value will be **true** if course was removed, and also if course never existed in the first place. If it returns **false**, then the course was unsuccessfully removed.

Module Functions

get_module_count

Parameters Needed:

- p1 - account name

Returns total number of modules in specified account.

get_module_list

Parameters Needed:

- p1 - account name

Returns list of modules, each XML node result contains the location number and title of the module inside of *simpleXML* text data.

get_module_list_from_course

Parameters Needed:

- p1 - account name
- p2 - course name

Returns list of modules inside of specified course, each result contains the location number and title of the module.

get_module_record

Parameters Needed:

- p1 - account name
- p2 - module location

Returns a complete module record inside an XML dataset, which contains 2 main nodes (module_config, and module_content), each which contains *simpleXML* text data.

Slide Functions

get_slide_count

Parameters Needed:

- p1 - account name

Returns count – total number of slides in the specified account.

get_slide_list

Parameters Needed:

- p1 - account name

Returns list of slides (*each as the slide id inside of an XML result node*) in the specified account.

get_slide_record

Parameters Needed:

- p1 - account name
- p2 - slide id

Returns **entire slide record** as it exists in the database (*as multiple XML nodes*).

User Functions

add_user

Parameters Needed:

- p1 - account name
- p2 - username
- p3 - password
- p4 - name/value pairs (*optional*)

This function is an alias for the **add_user_into_system** function (*see section on that function for details*).

add_user_into_course

Parameters Needed:

- p1 - account name
- p2 - username
- p3 - course name
- p4 - start date (*optional*)
- p5 - expire date (*optional*)

This function enrolls **an existing user** into a *course* (*the user must exist in the system first before calling this function*).

You can optionally send a start and expiration date (p4 and p5 parameters) – they must be in the format of *YYYYMMDD* to be valid.

If user was successfully enrolled, this function will return **true**, otherwise it will return **false**.

add_user_into_system

Parameters Needed:

- p1 - account name
- p2 - username
- p3 - password
- p4 - name/value pairs (*optional*)

This function **adds a new user** into the system. The account name (*course admin*), desired username and desired password are all required.

The name/value pair parameter is optional, and can be a series of name+value pairs to set specific values for data for the user. Each name/value pair looks like this:

name=value

.. and is separated by double-pipe symbols like this:

name1=value1||name2=value2

Each 'name' is a data row in the user database, and each 'value' is the value you want to assign to that row. *Listed below are the valid names you can use:*

first_name, last_name, email, middle_init, date_start, date_expire, company, address, city, state, zip, phone, custom1, custom2, custom3, custom4, custom5, custom6, custom7, custom8, custom9, custom10, custom11, custom12, custom13, custom14, custom15, notes

As an example, if you wanted to set the users first name to John and last name to Smith, your query strings p4 value would look like this:

&p4=first_name=John||last_name=Smith

Returns **true** if user was added, and **false** if not.

add_user_into_usergroup

Parameters Needed:

- p1 - account name
- p2 - username
- p3 - group name

This function adds an **existing user** into an **existing usergroup** (*both the user and the usergroup must exist in the system first before calling this function*).

Returns **true** if user was added to the usergroup, and **false** if not.

get_user_count

Parameters Needed:

- p1 - account name

Returns number of users in specified account.

get_user_list

Parameters Needed:

- p1 - account name

Returns list of users (as usernames) in specified account.

get_user_list_from_course

Parameters Needed:

- p1 - account name
- p2 - course name

Returns list of users (as usernames) in specified course.

get_user_list_from_usergroup

Parameters Needed:

- p1 - account name
- p2 - usergroup name

Returns list of users (as usernames) in specified usergroup.

get_user_record

Parameters Needed:

- p1 – account name
- p2 – username

Returns a complete XML record of all data fields from a user record in the database. If the record is not found, the result returned will **false**.

remove_user

Parameters Needed:

- p1 - account name
- p2 - username

This function is an alias for the **remove_user_from_system** function (see section on that function for details).

remove_user_from_course

Parameters Needed:

- p1 - account name
- p2 - username
- p3 - course name

This function removes a user (*username*) from a course (*course name*). Returns **true** if user has been removed (*or was never enrolled in the course in the first place*), or **false** if the user was not removed.

remove_user_from_system

Parameters Needed:

- p1 - account name
- p2 - username

This function removes a user (*username*) from the system. Returns **true** if user has been removed (*or was never in the system in the first place*), or **false** if the user was not removed.

remove_user_from_usergroup

Parameters Needed:

- p1 - account name
- p2 - username
- p3 - usergroup name

This function removes a user (*username*) from a usergroup (*usergroup name*). Returns **true** if user has been removed (*or was never enrolled in the usergroup in the first place*), or **false** if the user was not removed.

update_user

Parameters Needed:

- p1 - account name
- p2 - username
- p3 – name/value pairs

This function updates an existing user with the name/value pairs specified. If the user does not yet exist in the system, this function will return **false**.

The name/value pairs are the same as are used in the **add_user_into_system** function (but you can also use **password** as a valid field name to update here), so you can refer to the section on that function for the list of valid names you can use as data rows.

Usergroup Functions

add_usergroup

Parameters Needed:

- p1 - account name
- p2 - usergroup name

Adds a usergroup to the specified account. Returns **true** if usergroup was added (*or already existed*), and **false** otherwise.

get_user_list_from_usergroup

Parameters Needed:

- p1 - account name
- p2 - usergroup name

Returns list of users (as usernames) in specified usergroup.

get_usergroup_count

Parameters Needed:

- p1 - account name

Returns count – total number of usergroups in the specified account.

get_usergroup_list

Parameters Needed:

- p1 - account name

Returns list of usergroups (*each as the usergroup name inside of an XML result node*) in the specified account.

remove_usergroup

Parameters Needed:

- p1 - account name
- p2 - usergroup name

Removes a usergroup from the specified account (*this will not remove students in the group – it just removes the usergroup itself*). Returns **true** if usergroup was removed (or *did not exist in the first place*), and **false** otherwise.

Reporting Functions

get_course_scores

Parameters Needed:

- p1 - account name
- p2 - course name

Returns scores for specified course as an XML dataset, each node containing a *simpleXML* text section with the quiz title, student username, course name, score, passing score, participation time, and date

get_item_activity

Parameters Needed:

- p1 - account name
- p2 - item name

Returns activity for specified item (which can be a course title, quiz title, or presentation title) as an XML dataset, each node containing a *simpleXML* text section with the item title, student, description of activity, time/date, and ip address.

get_quiz_scores

Parameters Needed:

- p1 - account name
- p2 - quiz title

Returns scores for specified course as an XML dataset, each node containing a *simpleXML* text section with the quiz title, student username, course name, score, passing score, participation time, and date

get_user_activity

Parameters Needed:

- p1 - account name
- p2 - username

Returns activity for specified user as an XML dataset, each node containing a *simpleXML* text section with the item (course, quiz or presentation), student, description of activity, time/date, and ip address.

get_user_course_results

Parameters Needed:

- p1 - account name
- p2 - username
- p3 - course name

Returns XML dataset with total_quiz_count, passed_quiz_count, failed_quiz_count, total_minutes, and average.

get_user_quiz_results

Parameters Needed:

- p1 - account name
- p2 - username
- p3 - quiz name
- p4 - course name

Returns XML dataset, each result contains a simpleXML text with the quiz id, title, student_id (username), course_name, module_id, score_auto (main score), score_final (score after manual overrides), passing_score, time_to_complete, item_data (internal info on choices picked), certificate id, and date.

get_user_scores

Parameters Needed:

- p1 - account name
- p2 - username

Returns scores for specified user as an XML dataset, each node containing a *simpleXML* text section with the quiz title, student username, course name, score, passing score, participation time, and date

Other Functions

get_api_version

Returns value (inside <result> node) representing API Version. You can use this to make sure the functions you call are available (*for example, some API versions may have additional API functions not available in earlier versions*).

PHP Sample Code

If you are using PHP 5+ on your server to access the API, the following may help.

Tip: You can download these files (and more) from this link:

<http://files.atrrixware.com/files/weblearning/version9/resources/api96kit.zip>

1. Create a folder on your web server to place the PHP files
2. Create a file named **api_wrapper.php** (*inside folder*)
3. Create a file named **api_example.php** (*inside folder*)

Inside the **api_wrapper.php** file, add this code:

```
<?php
/*
 * Weblearning API Wrapper Created on 2/16/2010 by Atrixware / AWD
 *
 */
// setup global variables - expects $account, $pass,
// and $server to already be defined
$account      = urlencode($account);
$pass         = urlencode($pass);
$server_api   =
"{$server}lms/api.php?account={$account}&pass={$pass}&action=apicall&apicall=";

// functions ----
function get_api_value () {
    // usage examples:
    // $x = get_api_value( "function" );
    // $x = get_api_value( "function", "parameter1" ); --up to 5 parameters
    //
    global $server_api;
    $in_function_and_parameters = urlencode(func_get_arg(0));
    if ( func_num_args() >= 2 ) { $in_function_and_parameters .= "&p1=" .
urlencode( func_get_arg(1) ); }
    if ( func_num_args() >= 3 ) { $in_function_and_parameters .= "&p2=" .
urlencode( func_get_arg(2) ); }
    if ( func_num_args() >= 4 ) { $in_function_and_parameters .= "&p3=" .
urlencode( func_get_arg(3) ); }
    if ( func_num_args() >= 5 ) { $in_function_and_parameters .= "&p4=" .
urlencode( func_get_arg(4) ); }
    if ( func_num_args() >= 6 ) { $in_function_and_parameters .= "&p5=" .
urlencode( func_get_arg(5) ); }
    //
    return file_get_contents( "{$server_api}{$in_function_and_parameters}" );
}

function get_tag_data( $in_tag_name, $in_xml_string ) {
    // usage example: $x = get_tag_data( "color", $the_xml_string );
    if( strlen( strstr($in_xml_string, "<{$in_tag_name}>") ) > 0 ) {
        $start = strpos( $in_xml_string, "<{$in_tag_name}>" );
        $start = $start + strlen( "<{$in_tag_name}>" );
        $end = ( strpos( $in_xml_string, "</{$in_tag_name}>" ) );
        $num = ( $end - $start );
        $valore = substr( $in_xml_string, $start, $num );
        return $valore;
    } else {
        return "";
    }
}

?>
```

Inside the `api_example.php` file, enter this code:

```
<?php
// setup weblearning account info
$account      = "your_account_name";
$pass         = "your_password";
$server       = "http://your.server.url/"; // include trailing slash

// include the api wrapper module
require_once "api_wrapper.php";

// START CODE HERE--

?>
```

Make sure you enter in your **account name**, **password**, and **server URL**, and then save both files.

Now try these examples to see how it works (enter the example code right beneath the `// START CODE HERE--` section of the `api_example.php` file:

Example 1

Item(s) in yellow need to be entered by you:

```
// get xml return value
$s      = get_api_value( "get_user_list", "account_name" );
$xml    = simplexml_load_string( $s );

// loop through each xml node
foreach( $xml->children() as $child ) {
    // echo each item and its contents -----
    echo "<b>" . $child->getName() . " : </b>{$child}<br />";
}
}
```

Example 2

Item(s) in yellow need to be entered by you:

```
// get xml return value
$s      = get_api_value( "get_quiz_scores", "account_name", "quiz_name" );
$xml    = simplexml_load_string( $s );

// loop through each xml node
foreach( $xml->children() as $child ) {

    // echo login and password for each item -----
    // by getting data from simple-xml structure
    $username    = get_tag_data( "student", $child );
    $score       = get_tag_data( "score", $child );
    echo "<b>STUDENT:</b> { $username} <b>SCORE:</b> { $score}<br />";
}
}
```