

# Atrixware Weblearning 9.5

## **API Documentation**

<b>Weblearning 9.5 API.....</b>	<b>3</b>
<b>Course Administrator Functions .....</b>	<b>5</b>
get_course_admin_record .....	5
get_passing_score.....	5
get_email .....	5
get_expiration_date .....	5
get_limited_user_password .....	5
<b>Course Functions.....</b>	<b>6</b>
course_exists.....	6
delete_course .....	6
get_course_description.....	6
get_course_id .....	6
get_course_names .....	7
<b>Module/Quiz Functions.....</b>	<b>8</b>
get_module_name .....	8
get_module_names .....	8
get_modules_in_course.....	8
get_modules_not_in_course.....	8
<b>Questions Functions .....</b>	<b>9</b>
get_question_text.....	9
<b>Student/User Functions.....</b>	<b>10</b>
add_student_to_all_courses.....	10
copy_students_from_course.....	10
get_students_list.....	10
get_student_record.....	11
is_student_expired.....	11
is_student_in_course .....	11
remove_student_from_course .....	11
<b>Score/Reporting Functions .....</b>	<b>12</b>
get_activity_for_student.....	12
get_activity_for_item.....	12
get_scores_for_course .....	12
get_scores_for_quiz.....	12
get_scores_for_student .....	13
<b>Other/Misc. Functions .....</b>	<b>14</b>
get_api_version.....	14
<b>PHP Sample Code.....</b>	<b>15</b>
Example 1.....	16
Example 2.....	16

## Weblearning 9.5 API

This document covers the API functionality for the **Atrixware Weblearning 9.5** System (*Professional and Enterprise Licenses 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)
- ne1 = parameter 1 override flag (optional)
- ne2 = parameter 2 override flag (optional)
- ne3 = parameter 3 override flag (optional)
- ne4 = parameter 4 override flag (optional)
- ne5 = parameter 5 override flag (optional)

**Simple Example (querystring):**

<http://your.server.name/lms/api.php?action=apicall&apicall=showemail&account=sampleuser&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**, the course admin account id is **sampleuser**, and the course admin password is **12345**.

Some functions will require **p1** -> **p5** parameters to have values, and/or **ne1** -> **ne5** to be set (*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>xxx</result>
</api_result>
```

**xxx** can be the result, or simply **SUCCESS: 0 results returned** to indicate that the call was successful (*but returned 0 results, either because no results were found, or, the function called does not return any result*). For example, you may call a function that returns a list of students (*for example*) that is a successful call, but returns 0 results.

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

- Course Administrator Functions
- Course Functions
- Module/Quiz Functions
- Question Functions
- Student/User Functions
- Score/Reporting Functions
- Other/Misc Functions

Note that not every function is yet available via the API. With each new release, we add more and more API functionality.

As it stands now, the majority of the API is for **getting** data from the system (as opposed to **setting** data). However, we have added quite a few **set** based functions in the student functions for this release, since this is where most set based functions are used.

We have also added reporting API functions as well in this release.

## Course Administrator Functions

### **get\_course\_admin\_record**

Returns **entire account record** as it exists in the database (as multiple XML nodes), which includes all information from the other course administrator functions here, and other data not available from individual API functions.

### **get\_passing\_score**

Returns account default passing score (inside <result> node).

### **get\_email**

Returns account email address (inside <result> node).

### **get\_expiration\_date**

Returns account expiration date (inside <result> node).

### **get\_limited\_user\_password**

Returns account limited user (*aka – reports*) password (inside <result> node).

## Course Functions

### course\_exists

**Parameters Needed:**

- p1 - account name
- p2 - course name

Returns true or false (inside <result> node).

### delete\_course

**Parameters Needed:**

- p1 - account name
- p2 - course name
- p3 - un-enroll students (**yes** or **no**)

This will remove *course* named **p2** from *course admin* named **p1**. If **p3** is set to **yes**, then students enrolled in the course will un-enrolled as well (*a student can be associated with a non-existent course, and will be, if p3 is set to no*).

The return value will always be **SUCCESS: 0 results returned**. So, in order to determine if the course was deleted, you should call the **course\_exists** API function before and/or after this API function.

### get\_course\_description

**Parameters Needed:**

- p1 – course name

Returns a simple-xml document (inside <result> node) containing all layout and content data for a course.

### get\_course\_id

**Parameters Needed:**

- p1 – course name

Returns course database ID (inside <result> node).

## **get\_course\_names**

Returns course listing (inside <result> node) like this.

```
CourseName1 || CourseName2 || CourseName3 ||
```

## Module/Quiz Functions

### get\_module\_name

**Parameters Needed:**

- p1 – quiz location (1 to 100)

Returns quiz/module title (inside <result> node).

### get\_module\_names

Returns string containing all quizzes & modules (inside <result> node) like this:

```
Module1name || module2name || module3name ||
```

### get\_modules\_in\_course

**Parameters Needed:**

- p1 – course name

Returns a result (inside <result> node) like this:

```
[xxx] COURSE_NAME[-----][xxx] COURSE_NAME[-----]
```

xxx represents the location of the module (001 – 100).

### get\_modules\_not\_in\_course

**Parameters Needed:**

- p1 – course name

Returns a result (inside <result> node) like this:

```
[xxx] COURSE_NAME[-----][xxx] COURSE_NAME[-----]
```

xxx represents the location of the module (001 – 100).

## Questions Functions

### `get_question_text`

**Parameters Needed:**

- p1 – question id

Returns html code of question text (inside <result> node).

## Student/User Functions

**Note:** There is a separate and comprehensive student enrollment API ~ information is available here:

<http://www.atrixware.com/support/hs/index.php?pg=forums.posts&id=13077&pc=1>

### add\_student\_to\_all\_courses

**Parameters Needed:**

- p1 – student login id
- p2 – student password
- p3 – student email

This function enrolls a new (or existing) student of login id **p1** and password of **p2** and email of **p3** into all courses.

If you are using an existing student, then the **p1** must exactly match the login id already in the system, and **p2** must exactly match the password already in the system, or the function will not effect the student record. Also, the email address will not be changed if you enter a different one than already exists for that student.

Returned value is always **<result>SUCCESS: 0 results returned</result>** so use the **is\_student\_in\_course** API function to determine the success of this call.

### copy\_students\_from\_course

**Parameters Needed:**

- p1 – source course name
- p2 – destination course name

This enrolls all students from course p1 into course p2. The return value is an HTML string, with the result text inside of a **<strong>** tag.

### get\_students\_list

**Parameters Needed:**

- p1 – account name
- (optional) p2 = course name (ALL COURSES if not set)

Returns XML document containing multiple result nodes each containing a simple-xml string containing all known information about the student (representing each and every data field available on the student entry form).

## get\_student\_record

### Parameters Needed:

- p1 – account name
- p2 – student login id

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

## is\_student\_expired

### Parameters Needed:

- p1 – student login id
- p2 – account name

This returns one of the following:

**<result>SUCCESS: 0 results returned</result>** (means no expire date set)  
**<result>>true</result>** (means is expired)  
**<result>>false</result>** (means is not expired)

## is\_student\_in\_course

### Parameters Needed:

- p1 – student login id
- p2 – course name

Returns **<result>1</result>** if student **p1** is in course **p2**, or **<result>0</result>** otherwise.

## remove\_student\_from\_course

### Parameters Needed:

- p1 – course name
- p2 – student login id

This removes student id of p2 from course named p1. The returned value is either an HTML string with the text result message inside of **<strong>** tags, or if the student did not exist in the course, the result returned will be **<result>SUCCESS: 0 results returned</result>**.

## Score/Reporting Functions

### get\_activity\_for\_student

**Parameters Needed:**

- p1 – student login id

Returns one or more **<result>** nodes, inside each one, a *simple-xml* string containing student, item (*quiz, presentation or course name*), description of activity, time/date, and ip address.

### get\_activity\_for\_item

**Parameters Needed:**

- p1 – item (quiz, presentation, or course) name

Returns one or more **<result>** nodes, inside each one, a *simple-xml* string containing student, item (quiz, presentation or course name), description of activity, time/date, and ip address.

### get\_scores\_for\_course

**Parameters Needed:**

- p1 – course name

Returns one or more **<result>** nodes, inside each one, a *simple-xml* string containing quiz, student, course, score, passing score, date, and participation time.

### get\_scores\_for\_quiz

**Parameters Needed:**

- p1 –quiz title

Returns one or more **<result>** nodes, inside each one, a *simple-xml* string containing quiz, student, course, score, passing score, date, and participation time.

## get\_scores\_for\_student

### Parameters Needed:

- p1 – student login id

Returns one or more **<result>** nodes, inside each one, a *simple-xml* string containing quiz, student, course, score, passing score, date, and participation time.

## Other/Misc. 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 or better) on your server to access the API, the following may help.

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 4/24/2009 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_students_list", "your_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_students_list", "your_account_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
    $login_id      = get_tag_data( "login", $child );
    $password      = get_tag_data( "password", $child );
    echo "<b>LOGIN:</b> {$login_id} <b>PASSWORD:</b> {$password}<br />";
}
}
```