Atrixware Weblearning LMS

API 9.64 Documentation

Contents

Weblearning API Overview

What this Document Covers
This document covers the API functionality for the **Atrixware Weblearning 9.6 LMS**. More specifically, it covers the **9.64** version of the API (*older API versions do exist, and are still available in the Weblearning LMS 9.6x builds, the 9.64 API represents the most up-to-date version of the API at the time of this writing*).

What does API Stand For?
API is an acronym for **A**pplication **P**rogramming **I**nterface, and it is the most widely used way to enable different systems communicate with each other.

When would I use the API?
Some typical uses for the Weblearning API would be integrating your **shopping cart** software, **CRM**, or **HR** web application with Weblearning and letting those systems either enroll users, or get user data FROM the Weblearning LMS. Other frequent uses are **creating external web applications** for user enrollments, message monitoring, and custom reporting that work with the Atrixware Weblearning LMS.

Do Older API Versions Work?
The 9.50, 9.51 and 9.60 versions of the API are still available. We do not offer documentation on those versions any longer, but some older external applications and plug-in apps still use those API versions, so we have left them in for backwards-compatibility reasons.

If you have old code that uses any of these older versions of the API, there is no urgent need to rewrite your code to use the new API. However, moving forward, you should strongly consider using the new 9.64 API functions since they are more secure, faster, more optimized, and work with the various new features included in the Weblearning 9.64 LMS.

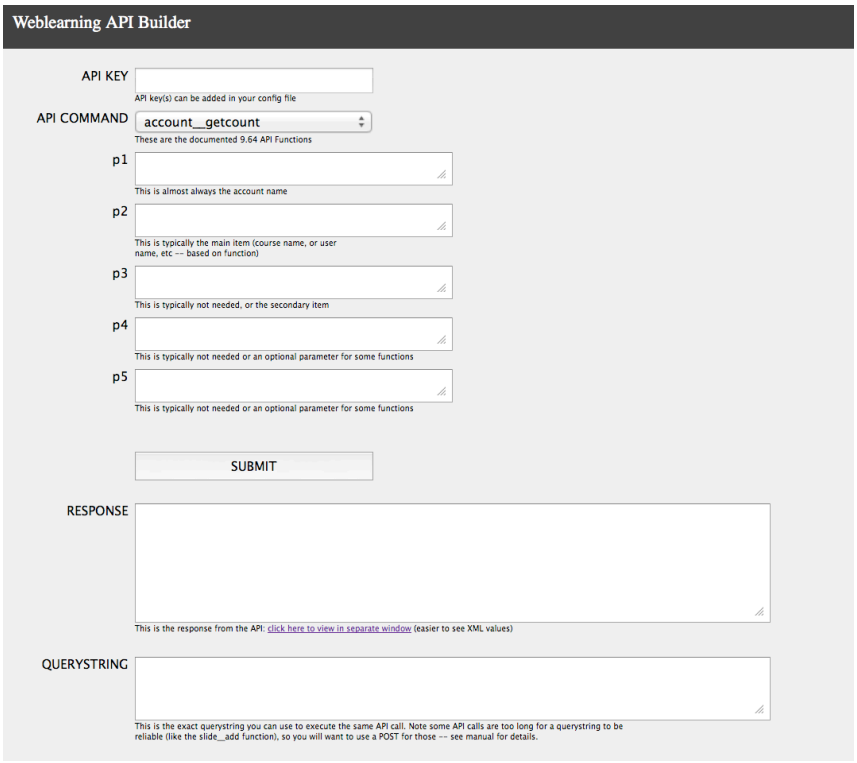
Getting Started

Atrixware has created some tools to make it easier for you to learn about and use the API.

Weblearning API Builder App
Your system comes equipped with an API Builder app. It is located at
http : // your.weblearingsystem.com / lms / apibuilder.php

Simply open a web browser and navigate to the address above (*replace the your.weblearningsystem.com piece with your own system address*).

You will see a form similar to the following:



This form is designed to enable you to quickly test out API functions. It lets you enter an API key (*you set up API keys in the global system config file -- discussed later*), an API command, and the parameters. When you submit, it echo's the response, as well as the querystring you can use (*or paste in to your code*) to execute that exact API call.

The API Code Kit
The API Code Kit is a PHP 5 based library that makes it easy to use the Weblearning API in your PHP code. You can download the files here:
http://files.atrixware.com/files/weblearning/version9/resources/apicodekit964.zip

The code kit consists of 2 library files you will use in your projects (**api_wrapper.php** and **api_template.php**) and a few example files**.**

To set it up, follow these steps:

1. Create a folder on your web server to put the PHP files.
2. Copy the **api_wrapper.php** and **api_template.php** files over into the folder.
3. Enter the proper values for **$api_key** and **$api_server** inside the **api_template.php** file (open the file using a text editor).
4. Use the **api_template.php** file as a starting point for any PHP script that needs to use the Weblearning API in your project.

To try out the example1 file:

1. Copy the file named **api_example1.php** and **api_wrapper.php** into a folder on your web server.
2. Open **api_example1.php** in a text editor, and enter the proper values for **$api_key** and **$api_server.**
3. On line #12, change **account_name** to that of your course admin account name.
4. Save.
5. Open your web browser and navigate to the file.

To try out the example2 file:

1. Copy the file named **api_example2.php** and **api_wrapper.php** into a folder on your web server.
2. Open **api_example2.php** in a text editor, and enter the proper values for **$api_key** and **$api_server.**
3. On line #12, change **account_name** to that of your course admin account name.
4. Also on line #12, change **course_name** to that of a course name with some activity.
5. Save.
6. Open your web browser and navigate to the file.

Understanding Return Values
When you run an API call, the data returned is always in XML, which looks something like this:

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

.. or it could look like this (*multiple results*) ...

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

.. or like this (*specific tags determined by exact API function*)...

```
<api_result>
       <some_tag>The Result</some_tag>
       <another_tag>The Result</another_tag>
       <third_tag>The Result</third_tag>
</api_result>
```

As you can see, there can be more than one <result> returned (*typically for the getList functions*), or named tags (*usually for getRecord functions*). The data returned inside each <result> or named tag can be a boolean value (*true/false*), an integer value, a string, or a tagged (*xml-like*) string. Each command explained later reveals what kind of data it will return.

A special note on the tagged (*xml-like*) string that is returned by some functions. The structure is very similar to XML, but it is not necessarily valid XML. It's easy enough to parse through though - take a look at the **api_wrapper.php** file for a function named **get_tag_data()** to see how it works (*if you are using PHP, you won't have to worry about writing the function, but you you are using another language, you can use that as the basis for writing your own function*).

Setting Up an API Key
Inside your config file, there is a section to set up one or more API keys. In general, it involved populating an array with one or more keys you want to allow as access keys to the API.

Information on this (*and the entire config file*) is available in the **Weblearning Server Admin Guide** - available for download from the Weblearning website at http://weblearning.atrixware.com/.

Account Functions

These functions enable you to work with Course Administrator Account data.

account__getCount

Parameters:

None

Return Type:

Integer

Description:

Returns the number of accounts are in the system.

Example (*using Code Kit*):

```
$data  = APICommand( "account__getCount" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

account__getList

Parameters:

None

Return Type:

String

Description:

Returns each account username as an XML <result> node

Example (*using Code Kit*):

```
$data  = APICommand( "account__getList" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

account__getRecord

Parameters:
p1: account name

Return Type:
XML Dataset

Description:
Returns an XML dataset detailing information on specified account

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "account__getRecord", "anthony" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

User Functions

These functions enable you to work with User accounts.

user__getCount

Parameters:

p1: account name

Return Type:

Integer

Description:

Returns the number of users that exist inside specified account.

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "user__getCount", "anthony" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

user__getList

Parameters:

p1: account name

Return Type:

String

Description:

Returns each user (as a username) from specified account as an XML <result> node

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "user__getList", "anthony" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

user__getRecord

Parameters:
p1: account name
p2: user's username

Return Type:
XML Dataset

Description:
Returns an XML dataset detailing information on user from specified account

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// assumes user username "chris" exists
$data  = APICommand( "user__getRecord", "anthony", "chris" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

user__add

Parameters:
p1: account name
p2: desired user's username
p3: desired user's password
p4: OPTIONAL name=value pairs

Return Type:
Boolean (*true or false*)

Description:
Adds a user into specified account. The **p2** (*username*) and **p3** (*password*) parameters are required parameters, but the 4th parameter (**p4**) of *name=value* pairs is optional, and worth discussing a bit further.

Each *name=value* should be separated by a two pipe characters (**||**), and each pair looks like this: **name=value** where name is the row name (*like address, or first_name for example*), and value is the value for that row.

For example, to set a user's first name and last name, **p4** will look like this:

```
first_name=Bob||last_name=Jones
```

Here are the possible *name* values you can use for the *name=value* pairs:

```
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
```

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
// should return true if username Bob does not exist, or false if it does
// try to add user Bob, with password of bob1, name of Robert Smith
// and Start Date of May 1, 2012
$data  = APICommand( "user__add", "anthony",
                     "Bob", "bob1",
                     "first_name=Robert||last_name=Smith||date_start=20120501"
                    );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

user__update

Parameters:
p1: account name
p2: desired user's username
p3: name=value pairs

Return Type:
Boolean (*true or false*)

Description:
Updates an existing user from specified account. Returns true if user was updated, or false if it failed (*or if user does not exist*).

The **p3** parameter uses the sane *name=value* pair format as described in the user__add function. They should be separated by a two pipe characters (**||**), and each pair looks like this: **name=value** where name is the row name (*like address, or first_name for example*), and value is the value for that row.

For example, to update a user's first name and last name, **p3** will look like this:

```
first_name=Bob||last_name=Jones
```

Here are the possible *name* values you can use for the *name=value* pairs:

```
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
```

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
// try to update user Bob, change name to Roberto Gonzales
// and Start Date of May 1, 2012
$data  = APICommand( "user__update", "anthony", "Bob",
                     "first_name=Roberto||last_name=Gonzales"
                   );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

user__enroll

Parameters:
p1: account name
p2: username
p3: course name to enroll user into
p4: OPTIONAL start date YYYYMMDD
p5: OPTIONAL end date YYYYMMDD

Return Type:
Boolean (*true or false*)

Description:
Enrolls an existing user from specified account into specified course. Returns true if user was enrolled, or false if it failed (*or if user does not exist*).

The **p4** and **p5** parameters are optional. They are used for setting an enrollment window of dates -- the date format should be YYYYMMDD, so for example a date of May 1, 2012 would be represented like this: 20120501.

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// try to enroll username Bob into course named Sample Course
$data  = APICommand( "user__enroll", "anthony", "Bob", "Sample Course" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

user__disenroll

Parameters:
p1: account name
p2: username
p3: course name to enroll user into

Return Type:
Boolean (*true or false*)

Description:
Dis-enrolls an existing user from specified account out of the specified course. Returns true if user was dis-enrolled (*or was not enrolled in the first place*).

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// try to disenroll username Bob out of course named Sample Course
$data  = APICommand( "user__disenroll", "anthony", "Bob", "Sample Course" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

Usergroup Functions

These functions enable you to work with Usergroups.

usergroup__getCount

Parameters:

p1: account name

Return Type:

Integer

Description:

Returns the number of usergroups that exist inside specified account.

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "usergroup__getCount", "anthony" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

usergroup__getList

Parameters:

p1: account name

Return Type:

String

Description:

Returns each usergroup name from specified account as an XML <result> node

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "usergroup__getList", "anthony" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

usergroup__add

Parameters:
p1: account name
p2: usergroup name to add

Return Type:
Boolean (*true or false*)

Description:
Adds usergroup into specified account.

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
$data  = APICommand( "usergroup__add", "anthony", "Building 101" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

usergroup__remove

Parameters:
p1: account name
p2: usergroup name to remove

Return Type:
Boolean (*true or false*)

Description:
Removes usergroup from specified account.

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
$data  = APICommand( "usergroup__remove", "anthony", "Building 101" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

usergroup__getUserList

Parameters:
p1: account name
p2: usergroup name

Return Type:
String

Description:
Returns each user (as a username) from usergroup and from specified account as an XML <result> node

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
$data  = APICommand( "usergroup__getUserList", "anthony", "Building 101" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

usergroup__addUser

Parameters:
p1: account name
p2: usergroup name
p3: username to add to group

Return Type:
Boolean (*true or false*)

Description:
Adds user into usergroup in specified account.

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// adding username Bob to Building 101 usergroup (assumes both exist)
$data  = APICommand( "usergroup__addUser", "anthony", "Building 101", "Bob" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

usergroup__removeUser

Parameters:
p1: account name
p2: usergroup name
p3: username to remove from group

Return Type:
Boolean (*true or false*)

Description:
Removes user from usergroup in specified account.

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// removing username Bob from Building 101 usergroup (assumes both exist)
$data  = APICommand( "usergroup__removeUser", "anthony", "Building 101", "Bob" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

Course Functions

These functions enable you to work with course data.

course__getCount

Parameters:

p1: account name

Return Type:

Integer

Description:

Returns the number of courses that exist inside specified account.

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "course__getCount", "anthony" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

course__getList

Parameters:

p1: account name

Return Type:

String

Description:

Returns each course name from specified account as an XML <result> node

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "course__getList", "anthony" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

course__getRecord

Parameters:
p1: account name
p2: course name

Return Type:
XML Dataset

Description:
Returns an XML dataset detailing information on course from specified account

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// assumes course names "Sample Course" exists
$data  = APICommand( "course__getRecord", "anthony", "Sample Course" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

course__getUserList

Parameters:
p1: account name
p2: course name

Return Type:
String

Description:
Returns each user (as a username) from course and from specified account as an XML <result> node

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
$data  = APICommand( "course__getUserList", "anthony", "Sample Course" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

course__enrollUser

Parameters:
p1: account name
p2: course name
p3: username to enroll
p4: OPTIONAL start date YYYYMMDD
p5: OPTIONAL end date YYYYMMDD

Return Type:
Boolean (*true or false*)

Description:
See the **user__enroll** API command.

course__removeUser

Parameters:
p1: account name
p2: course name
p3: username to enroll

Return Type:
Boolean (*true or false*)

Description:
See the **user__disenroll** API command.

Slide Functions

These functions enable you to work with slides and questions.

slide__getCount

Parameters:

p1: account name

Return Type:

Integer

Description:

Returns the number of slides that exist inside specified account.

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "slide__getCount", "anthony" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

slide__getList

Parameters:

p1: account name

Return Type:

String

Description:

Returns each slide id from specified account as an XML <result> node

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
$data  = APICommand( "slide__getList", "anthony" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

slide__getRecord

Parameters:
p1: account name
p2: slide id

Return Type:
XML Dataset

Description:
Returns an XML dataset detailing information on slide from specified account

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// assumes slide id 1 exists
$data  = APICommand( "slide__getRecord", "anthony", "1" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

slide__add

Parameters:
p1: account name
p2: slide data AS string (tagged)

Return Type:
Integer

Description:
Inserts a new slide into the slide bank of specified account. Quite often, the data that gets submitted using this API command can grow much larger than a querystring-based send can handle. Therefore, you should POST the data instead.

The **p2** parameter needs to be formatted as an xml-like tagged string. If you pull a slide from the **slide__getRecord** Command, you will see the exact format required.

1. Minimally, you will need to include **<q_style>** and **<q_category>** data.

2. The **<q_style>** tag value has to be one of the following values to be valid: "**true false**", "**yes no**", "**pick list**", "**fill in the blank**", "**essay**", "**slide**", "**multiple option**", "**multiple response**"

3. For **true false**, **yes no**, **pick list** and **multiple option** slides, one (*and ONLY one*) of the **q_choiceX_status** tags (**a_status** through **h_status**) must have a value of **ON** (*which identifies the correct choice*) in order for the question to operate properly (*pick list and multiple option styles must also have a corresponding value in the q_choiceX_text tag as well*).

4. For **multiple response** slides, two or more of the **q_choiceX_status** tags (**a_status** through **h_status**) must have a value of **ON** (*which identifies the correct choices*) in order for the question to operate properly (*and must have corresponding values in the q_choiceX_text tag as well*).

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// adding multiple option slide
$data  = APICommand( "slide__add", "anthony",
                     "<q_text>What Color is the Sky</q_text>
                      <q_choicea_text>Red</q_choicea_text>
                      <q_choiceb_text>Blue</q_choiceb_text>
                      <q_choicec_text>Green</q_choicec_text>
                      <q_choiceb_status>ON</q_choiceb_status>
                      <q_category>Basic Skills</q_category>
                      <q_style>multiple option</q_style>" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

Message Functions

These functions enable you to work with the messages.

message__getCount

Parameters:

p1: account name
p2: OPTIONAL mailbox name *or unread for unread messages*

Return Type:
Integer

Description:
Returns the number of messages from specified account. If **p2** is not specified, the result will be the number of messages inside the **Inbox** mailbox. Other valid values for **p2** are **saved**, **trash**, **sent** and **unread**

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
// returns # unread messages
$data  = APICommand( "message__getCount", "anthony", "unread" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

message__getList

Parameters:

p1: account name
p2: OPTIONAL mailbox name *or unread for unread messages*

Return Type:
String

Description:
Returns each message id from specified account as an XML <result> node. If **p2** is not specified, the result will be the message id's inside the **Inbox** mailbox. Other valid values for **p2** are **saved**, **trash**, **sent** and **unread**

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
// return list of unread message id's
$data  = APICommand( "message__getList", "anthony", "unread" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

message__getRecord

Parameters:
p1: account name
p2: message id

Return Type:
XML Dataset

Description:
Returns an XML dataset detailing information on message from specified account

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// assumes message id 1 exists
$data  = APICommand( "message__getRecord", "anthony", "1" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

message__save

Parameters:
p1: account name
p2: message id

Return Type:
Boolean (*true or false*)

Description:
Places a message into the saved mailbox from specified account

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// assumes message id 1 exists
$data  = APICommand( "message__save", "anthony", "1" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

message__delete

Parameters:
p1: account name
p2: message id

Return Type:
Boolean (*true or false*)

Description:
Deletes a message from specified account

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// assumes message id 1 exists
$data  = APICommand( "message__delete", "anthony", "1" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

message__send

Parameters:
p1: account name
p2: to_id
p3: subject
p4: message

Return Type:
Boolean (*true or false*)

Description:
Sends a message to **p2** (*to_id*) with subject of **p3** and message of **p4**. Of special note is the **p2** value. This value is a numeric id (*as opposed to a username*). Therefore, in order to sent to the desired user, you will need to get their record id by calling the **user__getRecord** API command, then pulling out their id to use here.
message from specified account

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// assumes user_id 1 exists
$data  = APICommand( "message__send", "anthony", "1", "Test Message", "Hello" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

Report Functions

These functions enable you to work with reporting data.

report__getCourseInfo

Parameters:

p1: account name
p2: course name

Return Type:

XML Dataset

Description:

Returns one or more results, each containing a string (tagged) containing the user display name, username, course name, progress percent, current average percent, current status, comprehensive average percent, comprehensive status, and time in course.

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
// returns results for course named Sample Course
$data  = APICommand( "report__getCourseInfo", "anthony", "Sample Course" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

report__getCourseActivityInfo

Parameters:
p1: account name
p2: course name

Return Type:
XML Dataset

Description:
Returns one or more results, each containing a string (tagged) containing the learning item name, type, # users that completed the item, current average, and comprehensive average.

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
// returns results for course named Sample Course
$data  = APICommand( "report__getCourseActivityInfo", "anthony", "Sample Course" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
```

report__getUserInfo

Parameters:
p1: account name
p2: user username

Return Type:
XML Dataset

Description:
Returns one or more results, each containing a string (tagged) containing the course name, progress percent, current average percent, current status, comprehensive average percent, comprehensive status, and time in course.

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// returns results for course named Sample Course
$data  = APICommand( "report__getUserInfo", "anthony", "Bill" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

report__getUserCourseInfo

Parameters:
p1: account name
p2: user username
p3: course name

Return Type:
XML Dataset

Description:
Returns one or more results, each containing a string (tagged) containing the item, item type, score and status.

Example (*using Code Kit*):
```
// assumes course admin account named "anthony" exists
// returns results for course named Sample Course
$data  = APICommand( "report__getUserCourseInfo","anthony","Bill","Sample Course" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

Gradebook Functions

These functions enable you to work with the Gradebook.

gradebook__getReportCard

Parameters:

p1: account name
p2: course name
p3: username

Return Type:

String

Description:

Returns one or more results, each containing a string (tagged) containing the user report card data from specified course.

Example (*using Code Kit*):

```
// assumes course admin account named "anthony" exists
// returns report card for Bob in Sample Course
$data  = APICommand( "gradebook__getReportCard", "anthony", "Sample Course", "Bob" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

System Functions

These functions enable you to work with system.

system__getApiVersion

Parameters:

none

Return Type:

String

Description:

Returns the API version (9.64).

Example (*using Code Kit*):

```
$data  = APICommand( "system__getApiVersion" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```