Covers version
10.6

ATRIXWARE
Enhance Your Training

Axis LMS
API Integration Guide

Proprietary Notice
The software described in this document is a proprietary product of Atrixware, and is furnished to the user under a license for use as specified in the license agreement. The software may be used or copied only in accordance with the terms of the agreement.

Information in this document is subject to change without notice and does not represent a commitment on the part of Atrixware. No part of this document may be reproduced, transmitted, transcribed, stored in any retrieval system, or translated into any language without the express written permission of Atrixware.

Axis LMS API Guide

September 2018
Printed in USA

Axis LMS API Overview

Getting Started

Partition Functions

User Functions

Usergroup Functions

Manager Functions

Course Functions

Slide Functions

Message Functions

Report Functions

Gradebook Functions

Certifications Functions

System Functions

Axis LMS API Overview

What this Document Covers
This document covers the API functionality for the **Atrixware Axis LMS**.

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

Whats New in Axis LMS API 10.6?
We have added the following new API commands:

```
certifications__getProgramList
certifications__getProgramID
certifications__extend
certifications__issueOnlineCert
certifications__issueOfflineCert
manager__getCourseKeywords
manager__setCourseKeywords
user__elevate
```

Getting Started

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

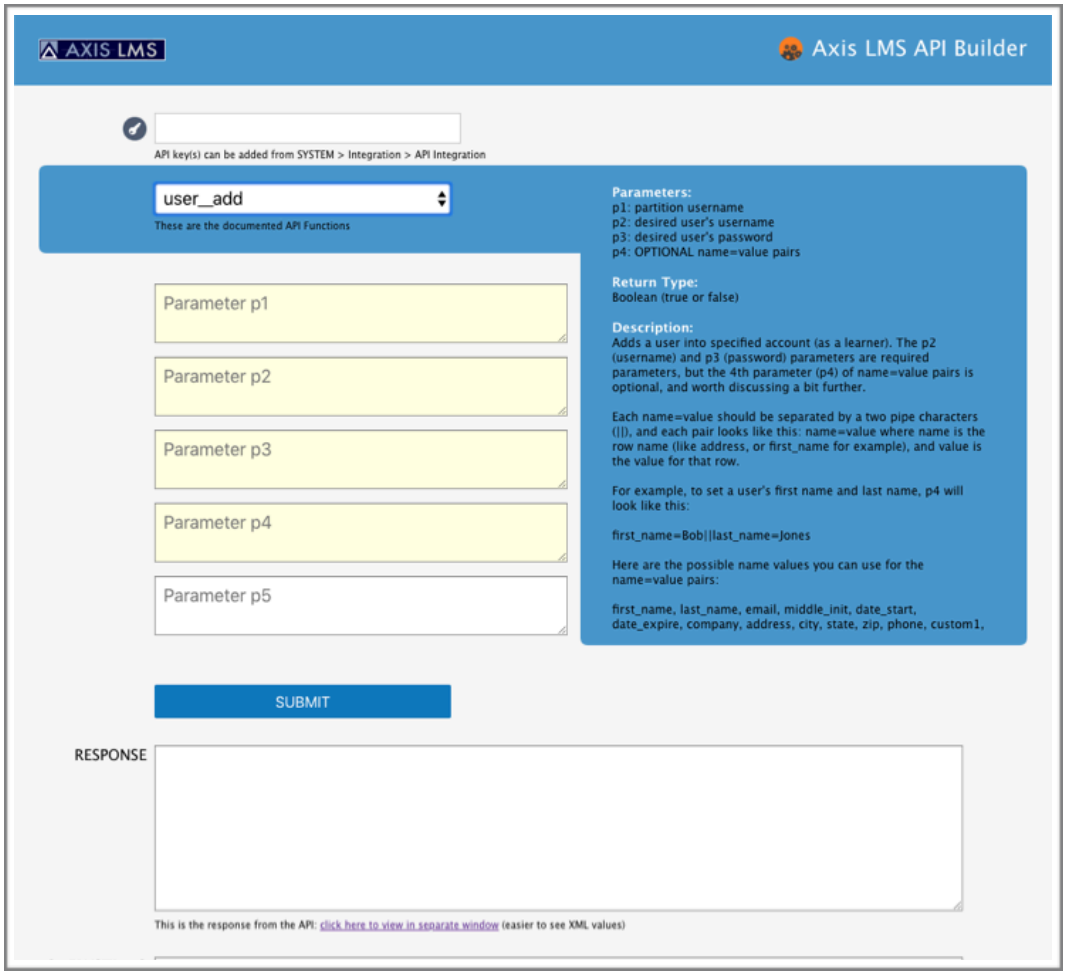
Axis LMS API Integration Tile
When you log into the Partition Admin, located in the SYSTEM area is the API Integration tile - which brings you to an area where you can (1) generate API keys, (2) view logs of your API calls, and (3) get to the API Builder App.

Axis API Builder App
Your system comes equipped with an API Builder app. It is located at:

https : // your.axissystem.com / lms / apibuilder.php

Simply open a web browser and navigate to the address above (*replace the your.axissystem.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 Axis LMS API in your PHP code. You can download the files here:

http://files.atrixware.com/files/axis/resources/axis-lms.apicodekit.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 Axis LMS 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 **partition_username** to that of your partition username.
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 **partition_username** to that of your partition username.
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*).

Partition Functions

These functions enable you to work with Partition Account data.

partition__getCount

Parameters:

None

Return Type:

Integer

Description:

Returns the number of LMS partitions that are in the system.

Example (*using Code Kit*):

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

partition__getList

Parameters:

None

Return Type:

String

Description:

Returns each partition username as an XML <result> node.

Example (*using Code Kit*):

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

partition__getRecord

Parameters:
p1: partition username

Return Type:
XML Dataset

Description:
Returns an XML dataset detailing information on specified account

Example (*using Code Kit*):
```
// assumes  partition username "anthony" exists
$data  = APICommand( "partition__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: partition username
p2: (optional) list type (defaults to "active"), other options "inactive" and "all"
```

Return Type:
Integer

Description:
Returns the number of users that exist inside specified account (this includes learners and managers). You can specify the list type for the 2nd parameter. By default, returned count will be 'active' users. You can set 2nd parameter to inactive (to get inactive user count) or all (to get count for all users). Note that previous versions of this API command returned count of ALL users, and did not offer the 2nd parameter.

Example (*using Code Kit*):

```
// assumes  partition username "anthony" exists
$data  = APICommand( "user__getCount", "anthony" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

user__getList

Parameters:

```
p1: partition username
p2: (optional) list type (defaults to "active"), other options "inactive" and "all"
```

Return Type:
String

Description:
Returns each user (as a username) from specified account as an XML <result> node (this includes learners and managers). You can specify the list type for the 2nd parameter. By default, returned results will be 'active' users. You can set 2nd parameter to inactive (to get inactive users) or all (to get all users). *Note that previous versions of this API command returned ALL users, and did not offer the 2nd parameter.*

Example (*using Code Kit*):

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

user__getRecord

Parameters:

p1: partition username
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  partition username "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__getUsername

Parameters:

p1: partition username
p2: user's id

Return Type:
String

Description:
Returns username of a user based on their record id (*which is returned by some calls instead of username*)

Example (*using Code Kit*):

```
// assumes partition username "anthony" exists and user id "5" exists
$data  = APICommand( "user__getUsername", "anthony", "5" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

user__getImage

Parameters:
p1: partition username
p2: user's username

Return Type:
String

Description:
Returns *URL* of image for username

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists
// assumes username "bill" exists
$data  = APICommand( "user__getImage", "anthony", "bill" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

user__getSSO

Parameters:

p1: partition username
p2: user's username
p3: OPTIONAL Add user if not in system (true/false, default = false)
p4: OPTIONAL # Minutes Link is good for (1-999999, 0 = no limit, default = 0)

Return Type:
String (or Boolean on error)

Description:
Returns an encrypted sign-on URL for a user. Typically would be used by an external system to generate a link that can be navigated to by the user, creating a single sign-on scenario so user can seamlessly access their online content without having to login to the LMS (the url does all the work).

Optional parameter p3 can be set to true to add the user to the system if they do not yet exist (*the system will generate a random strong password*).

Optional parameter p4 can be set to the number of minutes the url will be good for before it no longer provides access. The default is no limit (0).

Example (*using Code Kit*):

```
// assumes  partition username "anthony" exists
// assumes user username "chris" exists
$data  = APICommand( "user__getSSO", "anthony", "chris" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

user__add

Parameters:
p1: partition username
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  partition username "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: partition username
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  partition username "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__elevate

Parameters:
p1: partition username
p2: username

Return Type:
Boolean (*true or false*)

Description:
Elevates a user to manager role. Returns true if successful, or false if system does not support user roles, user does not exist, or user is already elevated to manager.

Example (*using Code Kit*):
```
// assumes  partition username "admin" exists and user "bobsmith" exists
$data  = APICommand( "user__elevate", "anthony", "bobsmith", );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

user__getManager

Parameters:
p1: partition username
p2: user username

Return Type:
String

Description:
Returns the username of a user's manager (or *false* if they have no manager assigned).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists, user 'bob' exists
$data  = APICommand( "user__getManager", "anthony", "bob" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

user__assignToManager

Parameters:
p1: partition username
p2: user username
p3: manager username

Return Type:
Boolean (*true or false*)

Description:
Assign user to manager. This call will be unsuccessful (*will return false*) if (1) the manager is not in fact a manager or (2) the user is a manager with 1 or more employees (*however, if they have no employees, it will be successful*).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists, user 'steve' and manager 'bill' exist
$data  = APICommand( "user__assignToManager", "anthony", "steve", "bill" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

user__enroll

Parameters:
p1: partition username
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, 2016 would be represented like this: 20160501.

Example (*using Code Kit*):
```
// assumes  partition username "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: partition username
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  partition username "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;
```

user__getSysVars

Parameters:
p1: partition username
p2: username

Return Type:
String

Description:
Returns a string (*tagged*) containing the each system variable for the specified user along with its current value (*or **false** if partition username or user username not found*).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists
// assumes username "Bill" exists
$data  = APICommand( "user__getSysVars","anthony","Bill" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

user__getSysVar

Parameters:
p1: partition username
p2: username
p3: system variable name

Return Type:
String

Description:
Returns a string containing the value of the system variable for the specified user (*or **false** if partition username, user username or variable name not found*).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists
// assumes username "Bill" exists
$data  = APICommand( "user__getSysVar","anthony","Bill","axis.course-completions");
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

user__setSysVar

Parameters:
p1: partition username
p2: username
p3: system variable name
p4: system variable value

Return Type:
XML Dataset

Description:
Returns an XML dataset. If the operation is successful, it will return 2 items, *true* as the first item, and the *value of the variable* as the 2nd item. If the operation is not successful (*if partition username or user username aren't found*), it will return a single item - *false*.

Notes:
If the system variable name specified does not yet exist for the specified username, the variable will be created.

If you append a **+** or **-** to the value, the value will be *ADDED* or *SUBTRACTED* to the existing value *(example: +5 adds 5 to existing value)*.

Example (*using Code Kit*):

```
// assumes  partition username "anthony" and user "bill" exists
$data  = APICommand( "user__setSysVar", "anthony", "bill", "reward-stars", "+1" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo $result->getName() . ": " . $result . "<br>";
}
```

Usergroup Functions

These functions enable you to work with Usergroups.

usergroup__getCount

Parameters:

p1: partition username

Return Type:

Integer

Description:

Returns the number of usergroups that exist inside specified account.

Example (*using Code Kit*):

```
// assumes  partition username "anthony" exists
$data  = APICommand( "usergroup__getCount", "anthony" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

usergroup__getList

Parameters:

p1: partition username

Return Type:

String

Description:

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

Example (*using Code Kit*):

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

usergroup__add

Parameters:
p1: partition username
p2: usergroup name to add

Return Type:
Boolean (*true or false*)

Description:
Adds usergroup into specified account.

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

usergroup__remove

Parameters:
p1: partition username
p2: usergroup name to remove

Return Type:
Boolean (*true or false*)

Description:
Removes usergroup from specified account.

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

usergroup__getUserList

Parameters:
p1: partition username
p2: usergroup name
p3: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:
String

Description:
Returns each user (as a username) from usergroup and from specified account as an XML <result> node. You can specify the list type for the 3rd parameter. By default, returned count will be 'active' users. You can set 3rd parameter to inactive (to get inactive user list) or all (to get list of all users). *Note that previous versions of this API command returned list of ALL users, and did not offer the 3rd parameter.*

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

usergroup__getUserCount

Parameters:
p1: partition username
p2: usergroup name
p3: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:
Integer

Description:
Returns the number of users that exist inside specified usergroup. You can specify the list type for the 3rd parameter. By default, returned count will be 'active' users. You can set 3rd parameter to inactive (to get inactive user count) or all (to get count for all users). Note that previous versions of this API command returned count of ALL users, and did not offer the 3rd parameter.

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists
$data  = APICommand( "usergroup__getUserCount", "anthony", "Building 101" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

usergroup__addUser

Parameters:
p1: partition username
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  partition username "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: partition username
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  partition username "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;
```

Manager Functions

These functions enable you to work with managers.

manager__add

Parameters:

p1: partition username
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, with a role of manager (*requires manager functionality*). 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 partition username "anthony" exists
// should return true if username Bob does not exist, or false if it does
// try to add manager Bob, with password of bob1, name of Robert Smith
// and Start Date of May 1, 2012
$data  = APICommand( "manager__add", "anthony", "Bob", "bob1", "first_name=Robert||
last_name=Smith||date_start=20120501" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

manager__getUsername

Parameters:
p1: partition username
p2: manager's id

Return Type:
String

Description:
Returns username of a manager based on their record id (which is returned by some calls instead of username). This differs from the user__getUsername function in that this function will only return a result if the id provided is in fact a manager, whereas the user__getUsername function will return a result for users of any role type.

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists
// assumes user id "5" exists
$data  = APICommand( "manager__getUsername", "anthony", "5" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

manager__assignEmployee

Parameters:
p1: partition username
p2: manager username
p3: employee username

Return Type:
Boolean (true or false)

Description:
Assign user to manager. This call will be unsuccessful (will return false) if (1) the manager is not in fact a manager or (2) the employee is a manager with 1 or more employees (however, if they have no employees, it will be successful).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists, manager 'bill' and employee 'steve' exist
$data  = APICommand( "manager__assignEmployee", "anthony", "bill", "steve" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

manager__getCourseKeywords

Parameters:
p1: partition username
p2: manager username

Return Type:
String

Description:
Returns course keywords (separated by spaces) assigned to a manager.

Example (*using Code Kit*):
```
// assumes  partition username "admin" exists and manager "bobsmith" exists
$data  = APICommand( "manager__getCourseKeywords", "admin", 'bobsmith' );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

manager__setCourseKeywords

Parameters:
p1: partition username
p2: manager username
p3: keyword(s) separated by spaces keyword(s) separated by spaces

Return Type:
Boolean

Description:
Sets course keywords (*separated by spaces*) to assign to a manager. These keywords will overwrite existing keywords. Function returns false if user record is not found, or, if user is not a manager.

Example (*using Code Kit*):
```
// assumes  partition username "admin" exists and manager "bobsmith" exists
$data  = APICommand( "manager__setCourseKeywords", "admin", 'bobsmith', 'math science
history' );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

manager__getList

Parameters:
p1: partition username
p2: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:
String

Description:
Returns each manager (as a username) from specified account as an XML <result> node. You can specify the list type for the 2nd parameter. By default, returned results will be 'active' users. You can set 2nd parameter to inactive (to get inactive users) or all (to get all users).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists
$data  = APICommand( "manager__getList", "anthony" );
$xml = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

manager__getCount

Parameters:
p1: partition username
p2: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:
Integer

Description:
Returns the number of managers that exist inside specified account. You can specify the list type for the 2nd parameter. By default, returned count will be 'active' users. You can set 2nd parameter to inactive (to get inactive user count) or all (to get count for all users).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists
$data  = APICommand( "manager__getCount", "anthony" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

manager__getUserList

Parameters:
p1: partition username
p2: manager username
p3: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:
String

Description:
Returns (as a username) each user (learners and manager roles) that report directly to manager specified as an XML <result> node. You can specify the list type for the 3rd parameter. By default, returned results will be 'active'. You can set 3rd parameter to inactive (to get inactive users) or all (to get all users).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists and manager "bill" exists
$data  = APICommand( "manager__getUserList", "anthony", "bill" );
$xml = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

manager__getUserCount

Parameters:
p1: partition username
p2: manager username
p3: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:
Integer

Description:
Returns number of users (learners and manager roles) that report directly to manager specified. You can specify the list type for the 3rd parameter. By default, returned count will be 'active'. You can set 2nd parameter to inactive (to get inactive user count) or all (to get count for all users).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists and manager "bill" exists
$data  = APICommand( "manager__getUserCount", "anthony", "bill" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

manager__getManagerList

Parameters:
p1: partition username
p2: manager username
p3: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:
String

Description:
Returns each manager (as a username) that reports directly to manager specified as an XML <result> node. You can specify the list type for the 3rd parameter. By default, returned results will be 'active'. You can set 3rd parameter to inactive (to get inactive managers) or all (to get all managers).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists and manager "bill" exists
$data  = APICommand( "manager__getManagerList", "anthony", "bill" );
$xml = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

manager__getManagerCount

Parameters:
p1: partition username
p2: manager username
p3: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:
Integer

Description:
Returns the number of managers that report directly to manager. You can specify the list type for the 3rd parameter. By default, returned count will be 'active'. You can set 2nd parameter to inactive (to get inactive manager count) or all (to get count for all managers).

Example (*using Code Kit*):
```
// assumes partition username "anthony" exists and manager "bill" exists
$data  = APICommand( "manager__getManagerCount", "anthony", "bill" );
$xml = simplexml_load_string( $data );
echo $xml->result;
```

Course Functions

These functions enable you to work with course data.

course__getCount

Parameters:

p1: partition username

Return Type:

Integer

Description:

Returns the number of courses that exist inside specified account.

Example (*using Code Kit*):

```
// assumes  partition username "anthony" exists
$data  = APICommand( "course__getCount", "anthony" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

course__getList

Parameters:

p1: partition username

Return Type:

String

Description:

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

Example (*using Code Kit*):

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

course__getRecord

Parameters:
p1: partition username
p2: course name

Return Type:
XML Dataset

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

Example (*using Code Kit*):
```
// assumes  partition username "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: partition username
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  partition username "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: partition username
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: partition username
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: partition username

Return Type:

Integer

Description:

Returns the number of slides that exist inside specified account.

Example (*using Code Kit*):

```
// assumes  partition username "anthony" exists
$data  = APICommand( "slide__getCount", "anthony" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

slide__getList

Parameters:

p1: partition username
p2: (optional) category name

Return Type:

String

Description:

Returns each slide id from specified account (and category if supplied) as an XML <result> node.

Example (*using Code Kit*):

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

slide__getRecord

Parameters:
p1: partition username
p2: slide id

Return Type:
XML Dataset

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

Example (*using Code Kit*):

```
// assumes  partition username "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: partition username
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  partition username "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: partition username
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  partition username "anthony" exists
// returns # unread messages
$data  = APICommand( "message__getCount", "anthony", "unread" );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

message__getList

Parameters:

p1: partition username
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  partition username "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: partition username
p2: message id

Return Type:
XML Dataset

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

Example (*using Code Kit*):
```
// assumes  partition username "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: partition username
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  partition username "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: partition username
p2: message id

Return Type:
Boolean (*true or false*)

Description:
Deletes a message from specified account.

Example (*using Code Kit*):
```
// assumes  partition username "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: partition username
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.

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

report__getUserArchiveInfo

Parameters:
p1: partition username
p2: user username

Return Type:
XML Dataset

Description:
Returns one or more results, each containing a string (tagged) containing the course name, course type (lms or other), progress, average, time spent, status, and completion date.

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

report__getUserCourseInfo

Parameters:
p1: partition username
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  partition username "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 portion of the Reporting engine.

gradebook__getReportCard

Parameters:

p1: partition username
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  partition username "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;
```

Certifications Functions

These functions enable you to work with the Certifications.

certifications__getUserList

Parameters:

```
p1: partition username
p2: certification program name
p3: (optional) list type (defaults to "active"), other options "inactive" and "all"
```

Return Type:
String

Description:
Returns (as a username) each user that has a certification in the specified certification program - as an XML <result> node. You can specify the list type for the 3rd parameter. By default, returned results will be 'active'. You can set 3rd parameter to inactive (to get inactive users) or all (to get all users).

Example (*using Code Kit*):

```
// assumes  partition username "anthony" exists and certification program "Program 1"
exists
$data  = APICommand( "certifications__getUserList", "anthony", "Program 1" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

certifications__getUserInfo

Parameters:

```
p1: partition username
p2: user username
```

Return Type:
String

Description:
Returns certification data for the specified user - each as an XML <result> node.

Example (*using Code Kit*):

```
// assumes  partition username "anthony" exists and username "bob123" exists
$data  = APICommand( "certifications__getUserInfo", "anthony", "bob123" );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

certifications__getUserCeus

Parameters:

p1: partition username
p2: user username
p3: (optional) starting date range *(in YYYYMMDD format)*
p4: (optional) ending date range *(in YYYYMMDD format)*

Return Type:

String

Description:

Returns data for each CEU a user has achieved - each as an XML <result> node. By default, all CEUs will be returned, but you can specify a starting date (*to return all CEUs achieved on or after the starting date*), and/or an ending date (*to return all CEU's achieved on or before the end date*). Supplying both the start and end date range will return items achieved on or between the 2 supplied dates.

Example (*using Code Kit*):

```
// assumes  partition username "admin" exists and username 'bob123' exists
$data  = APICommand( "certifications__getUserCeus", "admin", "bob123", '20180101',
20181231' );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

certifications__issueOfflineCert

Parameters:

p1: partition username
p2: user username
p3: program name
p4: certification number
p5: (optional) date string

Return Type:
Boolean (true or false)

Description:
Issues a certification to a user that is an offline (*external)* certification (*not one that exists in the LMS as an online achievable certification*). The optional date string parameter (p5), if included, allows you to send the issue date and expire date of the certification in the following format: YYYYMMDD-YYYYMMDD, whereas the first date is the issue date, and the second date is the expire date (*otherwise, the date will be today for issue, and 1 year from today for expire*). The issue date can not be in the future, and the expire date can not be earlier than the issue date. Function returns false if any parameter is invalid, otherwise the function returns true.

Example (*using Code Kit*):

```
// assumes  partition username "admin" exists and user username "b54776a" exists
$data  = APICommand( "certifications__issueofflinecert", "admin", "b54776a", "OSHA Class",
"c12345", "20200101-20201231");
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

certifications__issueOnlineCert

Parameters:
p1: partition username
p2: user username
p3: program name
p4: certification number
p5: (optional) date string

Return Type:
Boolean (true or false)

Description:
Issues a certification to a user that is an existing online certification (*one that exists in the LMS as an online achievable certification*). You may need to use the *certifications__getList* API command to identify the id you will need to pass as p3 here. The optional date string parameter (p5), if included, allows you to send the issue date and expire date of the certification in the following format: YYYYMMDD-YYYYMMDD, whereas the first date is the issue date, and the second date is the expire date (*otherwise, the date will be today for issue, and today + programs configured duration for expire*). The issue date can not be in the future, and the expire date can not be earlier than the issue date. Function returns false if any parameter is invalid, otherwise the function returns true.

Example (*using Code Kit*):
```
// assumes  partition username "admin" exists, user username "b54776a" exists and cert
program id 5 exists
$data  = APICommand( "certifications__issueonlinecert", "admin", "b54776a", 5, "c12345",
"20200101-20201231");
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

certifications__extend

Parameters:
p1: partition username
p2: certification number
p3: new date (YYYYMMDD format)

Return Type:
Boolean (true or false)

Description:
Extends the expiration date (*and CEU window*) for a certification. Function returns false if certification is not found, if date is invalid, if certification has already expired, or if date is before today's date, otherwise it returns true.

Example (*using Code Kit*):
```
// assumes  partition username "admin" exists and certification number c12345 exists
$data  = APICommand( "certifications__extend", "admin", 'c12345', '20201231');
$xml   = simplexml_load_string( $data );
echo $xml->result;
</command—certifications__extend>
```

certifications__getProgramID

Parameters:
p1: partition username
p2: program name

Return Type:
Integer

Description:
Returns id of program name.

Example (*using Code Kit*):
```
// assumes  partition username "admin" exists
$data  = APICommand( "certifications__getProgramID", 'admin' );
$xml   = simplexml_load_string( $data );
echo $xml->result;
```

certifications__getProgramList

Parameters:

p1: partition username
p2: (optional) list type (defaults to "active"), other options "inactive" and "all"

Return Type:

String

Description:

Returns name of each certification program - as an XML <result> node. You can specify the list type for the 2nd parameter. By default, returned results will be 'active'. You can set 2nd parameter to inactive (to get inactive programs) or all (to get all programs).

Example (*using Code Kit*):

```
// assumes  partition username "admin" exists
$data  = APICommand( "certifications__getProgramList", 'admin' );
$xml   = simplexml_load_string( $data );
foreach ( $xml->children() as $result ) {
       echo "{$result}<br>";
}
```

System Functions

These functions enable you to work with system.

system__getApiVersion

Parameters:

none

Return Type:

String

Description:

Returns the API version (for this API, the version is 10.6).

Example (*using Code Kit*):

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