ATiM API

# Introduction

ATiM API is a PHP tool that communicate to ATiM (special version of ATiM) by using RESTful web service.

The goal of ATiM API is to manipulate the data independent of web page interface and also not using raw MySQL query.

# Installation

## Requirement

* PHP V7.0 or more
* Browser: Google Chrome, Firefox or Safari (not too old).
* Web server like WAMP, XAMP, …

For downloading the ATiM API tool should install SVN and use the below URL:

<https://ctrnet.svn.cloudforge.com/atim_api/branches/dev/v1.0.0>

Beside the ATiM directory you can create the directory contain your project. For being able to use ATiM API should have the API key. Communicating to ATiM is by this key, just admin user can create api key.

In the below example we consider api key ‘myAPIKeyForTestATiMAPI’.

Download ATiM from address below:

<https://ctrnet.svn.cloudforge.com/atim2/branches/dev/v2.7.Z.API>

We consider that your ATiMAPI is on www, so in your web browser [*localhost/ATiMAPI*](localhost/ATiMAPI) is the address for accessing to ATiMAPI and your ATiM is on <https://www.ATiM.ca>

# Example

## Add a participant

1. Create the *myTests* directory beside the ATiM directory on the web disk.
2. Create the addParticipant.php file inside the directory.
3. Write the line below inside the file to load the library:

<?php

require\_once \_\_DIR\_\_ . '/../ATiM/vendor/autoload.php';

use ATiM\API\AtimApiConnection;

use ATiM\API\Controllers\Participant;

1. Add these lines for communicating to ATiM:

$atimApiKey ='myAPIKeyForTestATiMAPI';

$appUrl = 'https://www.ATiM.ca';

$atimApiConnection = new AtimApiConnection($appUrl, $atimApiKey);

1. For getting the add structure of participant add the line below:

$participant= new Participant($atimApiConnection);

$response = $participant->getAddStructure();

d(json\_decode($response, true));

1. Check the output by going to <localhost/ATiMAPI/addParticipant>.
2. Add a new participant by checking the add structure data:
   1. Comment the two las lines related to return the participant structure.
   2. Write the below codes:

$data = array(

'Participant' => array(

'participant\_identifier' => 'Participant Identifier 1234',

'title' => 'Mrs.',

'first\_name' => 'First Name 123456789',

'middle\_name' => 'Middle Name 123',

'last\_name' => 'Last Name 123',

'date\_of\_birth' => array(

'year' => '2100'

),

'race' => 'Aboriginal',

'sex' => 'f',

'marital\_status' => 'single',

'language\_preferred' => 'english',

'last\_chart\_checked\_date' => array(

'month' => '01',

'day' => '01',

'year' => '2100'

),

'notes' => 'Note 123',

'vital\_status' => 'alive',

'date\_of\_death' => array(

'month' => '01',

'day' => '01',

'year' => '2100'

),

'cod\_confirmation\_source' => 'Confirmation Source 123'

)

);

$response = $participant->add($data);

d(json\_decode($response, true));

**Info:** You need just one time doing step 5 for knowing the data structure.

## Search participant

For search between participants, we use ATiM’s data browser that give us lots of intressting functionality like search between the result search for fine another type of data (e.g. Search between the consents of some participants, in this case should find the participants then find the consents of them.).

As each entity has its own data structure so the search form should be different and for getting the search structure form should use the proper function.

1. Create the editParticipant.php file inside the directory.
2. Write the line below inside the file to load the library:

<?php

require\_once \_\_DIR\_\_ . '/../ATiM/vendor/autoload.php';

use ATiM\API\AtimApiConnection;

use ATiM\API\Controllers\Participant;

1. Add these lines for communicating to ATiM:

$atimApiKey ='myAPIKeyForTestATiMAPI';

$appUrl = 'https://www.ATiM.ca';

$atimApiConnection = new AtimApiConnection($appUrl, $atimApiKey);

1. For getting the search structure of participant add the line below:

$participant = new Participant($atimConnection);

$response = $participant->getSearchStructure();

d(json\_decode($response, true));

1. Check the output by going to <localhost/ATiMAPI/searchParticipant>.
2. Add a new participant by checking the search structure data:
   1. Comment the two las lines related to return the participant structure.
3. Write the below codes:

$dataSearch=['Participant' => ['participant\_identifier' => ['p\_r','p%4']]];

$response = $participant->search($dataSearch);

d(json\_decode($response, true));

1. The query of the search is finding the participants who in their participant\_identifier there is the letter “p” and two after that “r” or there is letter “p” and after that (not necessarily exact after that) the number “4”.

### Search with pagination’s parameters

The search function shows only the 10 first records as a result but it accept paginations’ parameters.

$response = $participant->search($dataSearch, array('Limit'=>2, 'page'=>2));

The parameters are

limit=>10 For limiting the number of result.

page=>2 Second page of result.

sort=>Participant.last\_name for sorting the result by last\_name.

direction=>desc descend.

## Edit a participant

For editing a participant we need his/her id that can be retrieved by a search (look at the Search participant) and also edit participant’s structure.

1. Create the addParticipant.php file inside the directory.
2. Write the line below inside the file to load the library:

<?php

require\_once \_\_DIR\_\_ . '/../ATiM/vendor/autoload.php';

use ATiM\API\AtimApiConnection;

use ATiM\API\Controllers\Participant;

1. Add these lines for communicating to ATiM:

$atimApiKey ='myAPIKeyForTestATiMAPI';

$appUrl = 'https://www.ATiM.ca';

$atimApiConnection = new AtimApiConnection($appUrl, $atimApiKey);

1. For getting the edit structure of participant add the line below:

$participant= new Participant($atimApiConnection);

$id=1; //Should get the ID by search

$response = $participant->getEditStructure($id);

d(json\_decode($response, true));

1. Check the output by going to <localhost/ATiMAPI/editParticipant>.
2. Add a new participant by checking the add structure data:
   1. Comment the two las lines related to return the participant structure.
   2. Write the below codes:

$data = array(

'Participant' => array(

'participant\_identifier' => 'New\_Participant\_identifier'

)

);

$response = $participant->edit($data, $id);

d(json\_decode($response, true));