# XMPP Chat | OpenFire | Converse JS Documentation

XMPP is the open standard for messaging and presence. The detail understanding, flow and history of XMPP can be found in its official Website: <https://xmpp.org/>

The steps of integrating a XMPP Chat is

1) Setting up the server (Openfire in our case).

2) Making REST API that will manage the server (Gnello/php-openfire-restapi in our case)

3) A Client which used in frontend (Conversejs in our case).

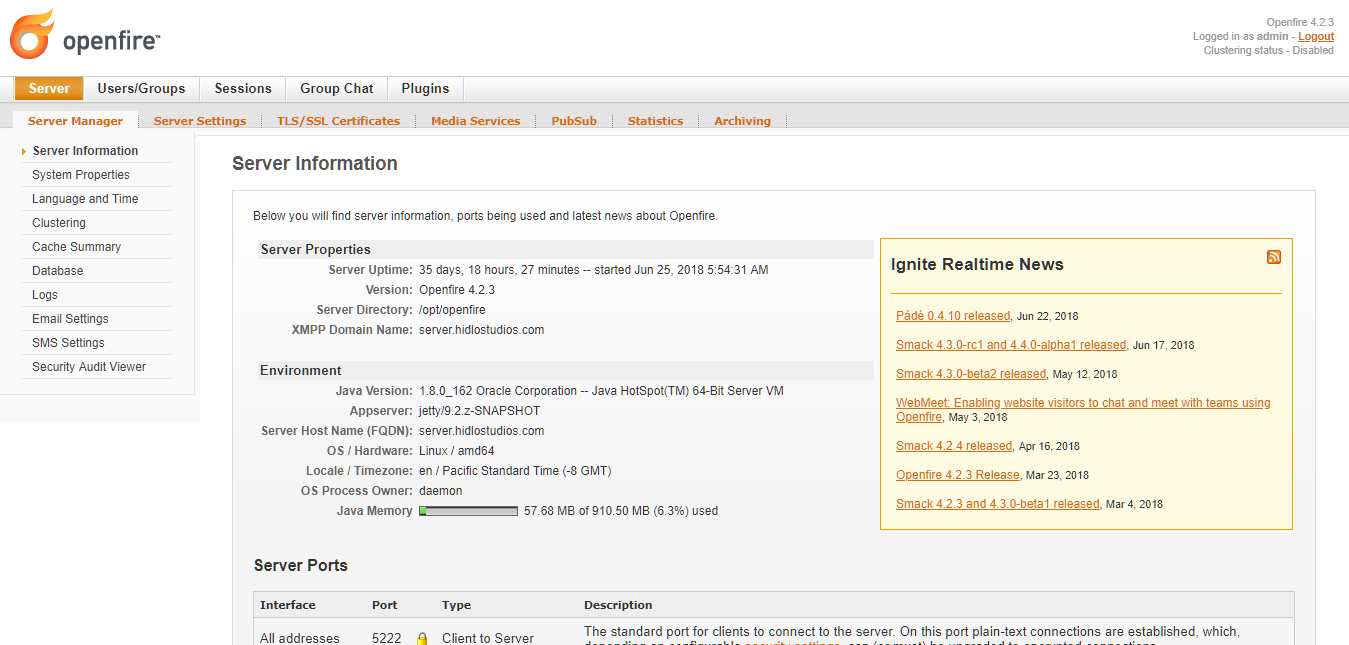
## 1) Setting up the server

The list of Servers mentioned in XMPP official website are

<https://xmpp.org/software/servers.html>. We selected OpenFire for our server.

The detail description can be found how to install a Open Fire Server in Centos OS here: <https://www.unixmen.com/install-openfire-centos-7/>. You can search for your respective OS.

Once the Open Fire is setup, you can access the panel:

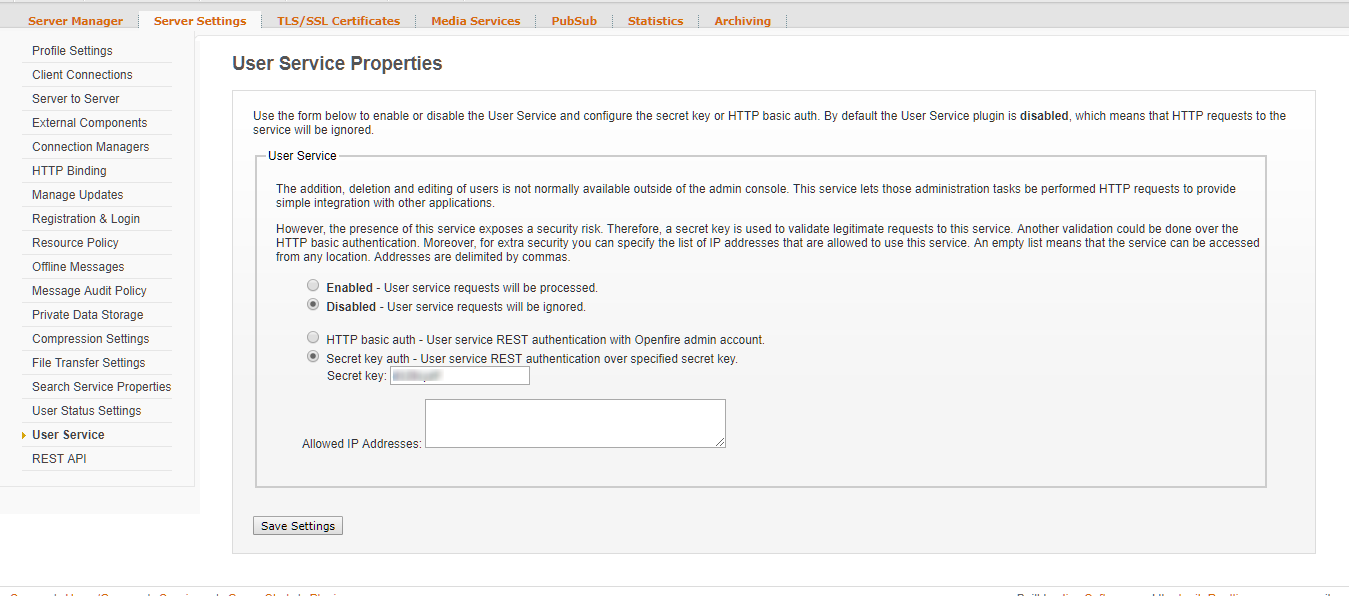


## 2) Making REST API that will manage the server

After the server has been setup, we will have to make certain restful apis, which will create the entities required for a chat. For example, we can create user, chat rooms which will used in the system.

For REST API in PHP, a package <https://github.com/gnello/php-openfire-restapi> by using which its inbuilt functions can be used for the specific action of OpenFire.

Go through the document of this Gnello Php REST API Client. For Authentication Token, there are two steps: One giving username and password and Second is using the Secret Key. The Secret key can be open in open fire panel (See below image)



Following is the overview of code of integrating **Gnello Php REST API Client** and its basic function **(Laravel)**

Installation via composer

|  |
| --- |
| composer require gnello/php-openfire-restapi |

Creating our own helper file

|  |
| --- |
| <?php  namespace '{{YOURPROJECT}}';  class OpenFire {  function \_\_construct() {  $this->authenticationtoken = new \Gnello\OpenFireRestAPI\AuthenticationToken('{{OPENFIRE\_SECRET\_KEY}}'));  $this->api = new \Gnello\OpenFireRestAPI\API('{{OPENFIRE\_HOST}}', '{{OPENFIRE\_PORT}}', $this->authenticationtoken);  $this->api->Settings()->setServerName('{{OPENFIRE\_SERVERNAME}}');  $this->api->Settings()->setHost('{{OPENFIRE\_HOST}}');  $this->api->Settings()->setPort('{{OPENFIRE\_PORT}}');  $this->api->Settings()->setSSL('{{OPENFIRE\_SETSSL}}');  $this->api->Settings()->setPlugin('{{OPENFIRE\_SETSSL\_PLUGIN}}');  }    //TO CREATE USER IN OPEN FIRE  public function createUser($data)  {  $username = $data['username'];  $password = $data['password'];  $name = $data['name'];  $email = $data['email'];   $result = $this->api->Users()->createUser($username, $password, $name, $email);  return $result;  }   //TO CREATE CHAT ROOM IN OPEN FIRE  public function createChatRoom($data)  {  $roomname = $data['roomname'];  $naturalname = $data['naturalname'];  $description = $data['description'];  $maxusers = $data['maxusers'];   $payload = $this->api->Payloads()->createChatRoomPayload();  $payload->setRoomName($roomname);  $payload->setNaturalName($naturalname);  $payload->setDescription($description);  $payload->setMaxUsers($maxusers);  $payload->setBroadcastPresenceRoles(['moderator','participant']);  $payload->setCanAnyoneDiscoverJID(true);   $result = $this->api->ChatRooms()->createChatRoom($payload);  return $result;  }   //CREATE MORE FUNCTION AS PER YOUR REQUIREMENT  } |

Use of helper function in any controller

|  |
| --- |
| <?php  //INCLUDE OUR CREATED HELPER AT TOP use '{{YOURPROJECT}}'\OpenFire;  //INSIDE CONTROLLER METHOD $openfire = new OpenFire(); $data['username'] = {{USER\_NAME}}; //THIS WILL BE USER JID $data['password'] ={{USER\_PASSWORD}}; $data['name'] = {{USER\_FULLNAME}}; $data['email'] = {{USER\_EMAIL}};  $result = $openfire->createUser($data); |

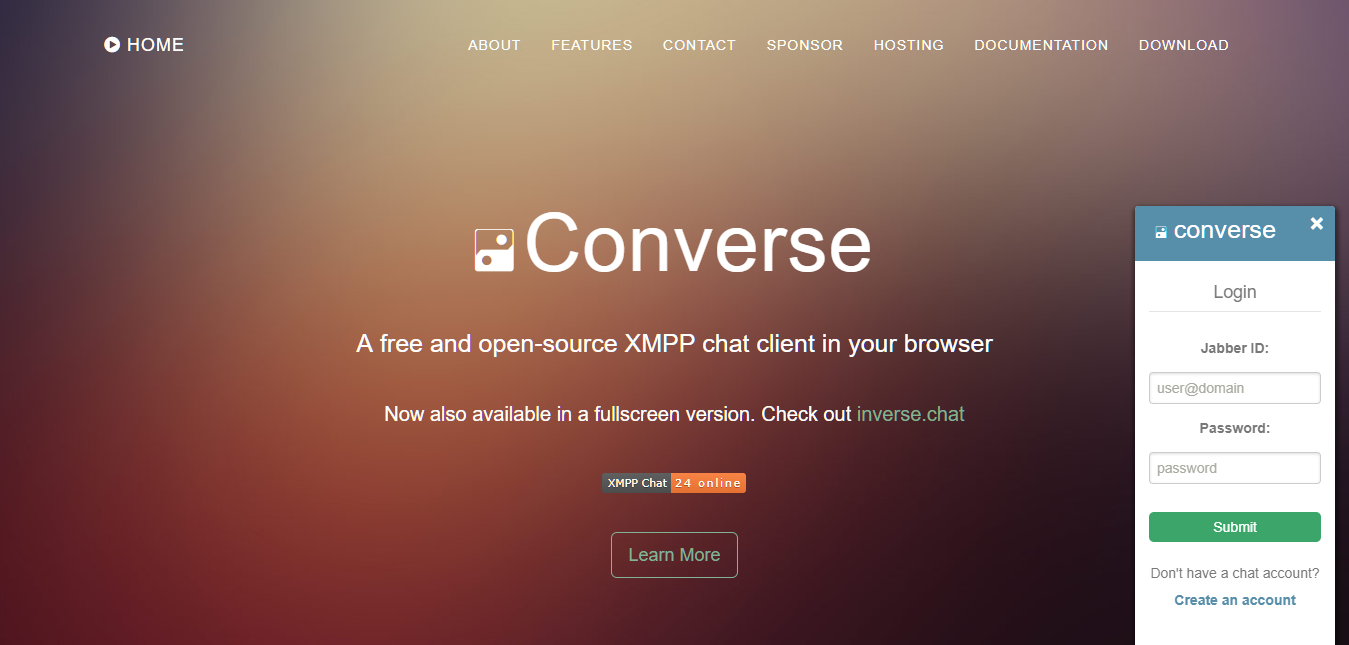
## 3) A Client which used in frontend (Converse Js in our case).

The client can be of any platform (Desktop, Android, IOS, Web). There are few clients mentioned in XMPP official Website: <https://xmpp.org/software/clients.html>. Converse Js is not mentioned in the list but its a free and open-source XMPP chat client for browser.

Documentation and demo sources can be downloaded from here: <https://conversejs.org/>.

The converse js have various view mode : overlay, fullscreen, mobile, embedded

Depending upon the requirement of the Project, the view can be selected. In our case, we selected the “embedded” which easily embeds with the layout of our project. Different JS are required for different view mode. Go through the documentation in detail.



Following is the overview of code of integrating converse js **(MUC Embedded)**

Include CSS

|  |
| --- |
| <link type="text/css" rel="stylesheet" media="screen" href="{{ RESOURCES\_FOLDER\_URL }}/converse.css" /> <link type="text/css" rel="stylesheet" media="screen" href="{{ RESOURCES\_FOLDER\_URL }}/converse-muc-embedded.css" /> |

Include JS

|  |
| --- |
| <**script** src="{{ RESOURCES\_FOLDER\_URL }}/converse-muc-embedded.js"></**script**> |

Main Javascript code

|  |
| --- |
| <**script**>  var sitechat;  $(document).ready(function(){  converse.plugins.add('myplugin', {  initialize: function () {  var \_converse = this.\_converse;  \_converse.on('connected', function () {  sitechat = \_converse;  });   \_converse.on('statusInitialized', function (status) {   //your code   });  }  });   converse.initialize({  authentication: 'login',  auto\_login: true,  keepalive:true,  jid: '{{USER\_JID}}'+'@'+'{{OPENFIRE\_DEFAULT\_SERVER\_ADDR}}',  password: '{{USER\_PASSWORD}}',  allow\_logout: false,  muc\_nickname\_from\_jid: true,  auto\_join\_rooms: [  '{{YOUR\_ROOM\_ID}}'+'@'+'{{OPENFIRE\_DEFAULT\_SERVER\_NAME}}',  ],  bosh\_service\_url: '{{OPENFIRE\_DEFAULT\_BOSH\_URL}}', //  view\_mode: 'embedded',  visible\_toolbar\_buttons: {  call: false,  clear: false,  emoji: false,  toggle\_occupants: false  },  whitelisted\_plugins: ['myplugin'],  debug:true  });  });   $(window).unload(function(){  sitechat.api.user.logout();  });  </**script**> |

HTML code

|  |
| --- |
| <div id="conversejs"></**div**> |

***End of Document…………………..***