[Overview](http://docs.google.com/index.html)

### Packages

* [CodeIgniter](http://docs.google.com/package-CodeIgniter.html)
  + [Libraries](http://docs.google.com/package-CodeIgniter.Libraries.html)
  + [Rest](http://docs.google.com/package-CodeIgniter.Rest.html)
* [None](http://docs.google.com/package-None.html)

### Classes

* [Example](http://docs.google.com/class-Example.html)
* [Format](http://docs.google.com/class-Format.html)
* [Key](http://docs.google.com/class-Key.html)
* [REST\_Controller](http://docs.google.com/class-REST_Controller.html)
* [Rest\_server](http://docs.google.com/class-Rest_server.html)
* [Welcome](http://docs.google.com/class-Welcome.html)
* [Overview](http://docs.google.com/index.html)
* Package
* Class

[1:](#gjdgxs) <?php  
 [2:](#30j0zll)   
 [3:](#1fob9te) defined('BASEPATH') OR exit('No direct script access allowed');  
 [4:](#3znysh7)   
 [5:](#2et92p0) // This can be removed if you use \_\_autoload() in config.php OR use Modular Extensions  
 [6:](#tyjcwt) require APPPATH . '/libraries/REST\_Controller.php';  
 [7:](#3dy6vkm)   
 [8:](#1t3h5sf) /\*\*  
 [9:](#4d34og8)  \* Keys Controller  
 [10:](#2s8eyo1)  \* This is a basic Key Management REST controller to make and delete keys  
 [11:](#17dp8vu)  \*  
 [12:](#3rdcrjn)  \* @package CodeIgniter  
 [13:](#26in1rg)  \* @subpackage Rest Server  
 [14:](#lnxbz9)  \* @category Controller  
 [15:](#35nkun2)  \* @author Phil Sturgeon, Chris Kacerguis  
 [16:](#1ksv4uv)  \* @license MIT  
 [17:](#44sinio)  \* @link https://github.com/chriskacerguis/codeigniter-restserver  
 [18:](#2jxsxqh)  \*/  
 [19:](#z337ya) class Key extends REST\_Controller {  
 [20:](#3j2qqm3)   
 [21:](#1y810tw)  protected $methods = [  
 [22:](#4i7ojhp)  'index\_put' => ['level' => 10, 'limit' => 10],  
 [23:](#2xcytpi)  'index\_delete' => ['level' => 10],  
 [24:](#1ci93xb)  'level\_post' => ['level' => 10],  
 [25:](#3whwml4)  'regenerate\_post' => ['level' => 10],  
 [26:](#2bn6wsx)  ];  
 [27:](#qsh70q)   
 [28:](#3as4poj)  /\*\*  
 [29:](#1pxezwc)  \* Insert a key into the database  
 [30:](#49x2ik5)  \*  
 [31:](#2p2csry)  \* @access public  
 [32:](#147n2zr)  \* @return void  
 [33:](#3o7alnk)  \*/  
 [34:](#23ckvvd)  public function index\_put()  
 [35:](#ihv636)  {  
 [36:](#32hioqz)  // Build a new key  
 [37:](#1hmsyys)  $key = $this->\_generate\_key();  
 [38:](#41mghml)   
 [39:](#2grqrue)  // If no key level provided, provide a generic key  
 [40:](#vx1227)  $level = $this->put('level') ? $this->put('level') : 1;  
 [41:](#3fwokq0)  $ignore\_limits = ctype\_digit($this->put('ignore\_limits')) ? (int) $this->put('ignore\_limits') : 1;  
 [42:](#1v1yuxt)   
 [43:](#4f1mdlm)  // Insert the new key  
 [44:](#2u6wntf)  if ($this->\_insert\_key($key, ['level' => $level, 'ignore\_limits' => $ignore\_limits]))  
 [45:](#19c6y18)  {  
 [46:](#3tbugp1)  $this->response([  
 [47:](#28h4qwu)  'status' => TRUE,  
 [48:](#nmf14n)  'key' => $key  
 [49:](#37m2jsg)  ], REST\_Controller::HTTP\_CREATED); // CREATED (201) being the HTTP response code  
 [50:](#1mrcu09)  }  
 [51:](#46r0co2)  else  
 [52:](#2lwamvv)  {  
 [53:](#111kx3o)  $this->response([  
 [54:](#3l18frh)  'status' => FALSE,  
 [55:](#206ipza)  'message' => 'Could not save the key'  
 [56:](#4k668n3)  ], REST\_Controller::HTTP\_INTERNAL\_SERVER\_ERROR); // INTERNAL\_SERVER\_ERROR (500) being the HTTP response code  
 [57:](#2zbgiuw)  }  
 [58:](#1egqt2p)  }  
 [59:](#3ygebqi)   
 [60:](#2dlolyb)  /\*\*  
 [61:](#sqyw64)  \* Remove a key from the database to stop it working  
 [62:](#3cqmetx)  \*  
 [63:](#1rvwp1q)  \* @access public  
 [64:](#4bvk7pj)  \* @return void  
 [65:](#2r0uhxc)  \*/  
 [66:](#1664s55)  public function index\_delete()  
 [67:](#3q5sasy)  {  
 [68:](#25b2l0r)  $key = $this->delete('key');  
 [69:](#kgcv8k)   
 [70:](#34g0dwd)  // Does this key exist?  
 [71:](#1jlao46)  if (!$this->\_key\_exists($key))  
 [72:](#43ky6rz)  {  
 [73:](#2iq8gzs)  // It doesn't appear the key exists  
 [74:](#xvir7l)  $this->response([  
 [75:](#3hv69ve)  'status' => FALSE,  
 [76:](#1x0gk37)  'message' => 'Invalid API key'  
 [77:](#4h042r0)  ], REST\_Controller::HTTP\_BAD\_REQUEST); // BAD\_REQUEST (400) being the HTTP response code  
 [78:](#2w5ecyt)  }  
 [79:](#1baon6m)   
 [80:](#3vac5uf)  // Destroy it  
 [81:](#2afmg28)  $this->\_delete\_key($key);  
 [82:](#pkwqa1)   
 [83:](#39kk8xu)  // Respond that the key was destroyed  
 [84:](#1opuj5n)  $this->response([  
 [85:](#48pi1tg)  'status' => TRUE,  
 [86:](#2nusc19)  'message' => 'API key was deleted'  
 [87:](#1302m92)  ], REST\_Controller::HTTP\_NO\_CONTENT); // NO\_CONTENT (204) being the HTTP response code  
 [88:](#3mzq4wv)  }  
 [89:](#2250f4o)   
 [90:](#haapch)  /\*\*  
 [91:](#319y80a)  \* Change the level  
 [92:](#1gf8i83)  \*  
 [93:](#40ew0vw)  \* @access public  
 [94:](#2fk6b3p)  \* @return void  
 [95:](#upglbi)  \*/  
 [96:](#3ep43zb)  public function level\_post()  
 [97:](#1tuee74)  {  
 [98:](#4du1wux)  $key = $this->post('key');  
 [99:](#2szc72q)  $new\_level = $this->post('level');  
[100:](#184mhaj)   
[101:](#3s49zyc)  // Does this key exist?  
[102:](#279ka65)  if (!$this->\_key\_exists($key))  
[103:](#meukdy)  {  
[104:](#36ei31r)  // It doesn't appear the key exists  
[105:](#1ljsd9k)  $this->response([  
[106:](#45jfvxd)  'status' => FALSE,  
[107:](#2koq656)  'message' => 'Invalid API key'  
[108:](#zu0gcz)  ], REST\_Controller::HTTP\_BAD\_REQUEST); // BAD\_REQUEST (400) being the HTTP response code  
[109:](#3jtnz0s)  }  
[110:](#1yyy98l)   
[111:](#4iylrwe)  // Update the key level  
[112:](#2y3w247)  if ($this->\_update\_key($key, ['level' => $new\_level]))  
[113:](#1d96cc0)  {  
[114:](#3x8tuzt)  $this->response([  
[115:](#2ce457m)  'status' => TRUE,  
[116:](#rjefff)  'message' => 'API key was updated'  
[117:](#3bj1y38)  ], REST\_Controller::HTTP\_OK); // OK (200) being the HTTP response code  
[118:](#1qoc8b1)  }  
[119:](#4anzqyu)  else  
[120:](#2pta16n)  {  
[121:](#14ykbeg)  $this->response([  
[122:](#3oy7u29)  'status' => FALSE,  
[123:](#243i4a2)  'message' => 'Could not update the key level'  
[124:](#j8sehv)  ], REST\_Controller::HTTP\_INTERNAL\_SERVER\_ERROR); // INTERNAL\_SERVER\_ERROR (500) being the HTTP response code  
[125:](#338fx5o)  }  
[126:](#1idq7dh)  }  
[127:](#42ddq1a)   
[128:](#2hio093)  /\*\*  
[129:](#wnyagw)  \* Suspend a key  
[130:](#3gnlt4p)  \*  
[131:](#1vsw3ci)  \* @access public  
[132:](#4fsjm0b)  \* @return void  
[133:](#2uxtw84)  \*/  
[134:](#1a346fx)  public function suspend\_post()  
[135:](#3u2rp3q)  {  
[136:](#2981zbj)  $key = $this->post('key');  
[137:](#odc9jc)   
[138:](#38czs75)  // Does this key exist?  
[139:](#1nia2ey)  if (!$this->\_key\_exists($key))  
[140:](#47hxl2r)  {  
[141:](#2mn7vak)  // It doesn't appear the key exists  
[142:](#11si5id)  $this->response([  
[143:](#3ls5o66)  'status' => FALSE,  
[144:](#20xfydz)  'message' => 'Invalid API key'  
[145:](#4kx3h1s)  ], REST\_Controller::HTTP\_BAD\_REQUEST); // BAD\_REQUEST (400) being the HTTP response code  
[146:](#302dr9l)  }  
[147:](#1f7o1he)   
[148:](#3z7bk57)  // Update the key level  
[149:](#2eclud0)  if ($this->\_update\_key($key, ['level' => 0]))  
[150:](#thw4kt)  {  
[151:](#3dhjn8m)  $this->response([  
[152:](#1smtxgf)  'status' => TRUE,  
[153:](#4cmhg48)  'message' => 'Key was suspended'  
[154:](#2rrrqc1)  ], REST\_Controller::HTTP\_OK); // OK (200) being the HTTP response code  
[155:](#16x20ju)  }  
[156:](#3qwpj7n)  else  
[157:](#261ztfg)  {  
[158:](#l7a3n9)  $this->response([  
[159:](#356xmb2)  'status' => FALSE,  
[160:](#1kc7wiv)  'message' => 'Could not suspend the user'  
[161:](#44bvf6o)  ], REST\_Controller::HTTP\_INTERNAL\_SERVER\_ERROR); // INTERNAL\_SERVER\_ERROR (500) being the HTTP response code  
[162:](#2jh5peh)  }  
[163:](#ymfzma)  }  
[164:](#3im3ia3)   
[165:](#1xrdshw)  /\*\*  
[166:](#4hr1b5p)  \* Regenerate a key  
[167:](#2wwbldi)  \*  
[168:](#1c1lvlb)  \* @access public  
[169:](#3w19e94)  \* @return void  
[170:](#2b6jogx)  \*/  
[171:](#qbtyoq)  public function regenerate\_post()  
[172:](#3abhhcj)  {  
[173:](#1pgrrkc)  $old\_key = $this->post('key');  
[174:](#49gfa85)  $key\_details = $this->\_get\_key($old\_key);  
[175:](#2olpkfy)   
[176:](#13qzunr)  // Does this key exist?  
[177:](#3nqndbk)  if (!$key\_details)  
[178:](#22vxnjd)  {  
[179:](#i17xr6)  // It doesn't appear the key exists  
[180:](#320vgez)  $this->response([  
[181:](#1h65qms)  'status' => FALSE,  
[182:](#415t9al)  'message' => 'Invalid API key'  
[183:](#2gb3jie)  ], REST\_Controller::HTTP\_BAD\_REQUEST); // BAD\_REQUEST (400) being the HTTP response code  
[184:](#vgdtq7)  }  
[185:](#3fg1ce0)   
[186:](#1ulbmlt)  // Build a new key  
[187:](#4ekz59m)  $new\_key = $this->\_generate\_key();  
[188:](#2tq9fhf)   
[189:](#18vjpp8)  // Insert the new key  
[190:](#3sv78d1)  if ($this->\_insert\_key($new\_key, ['level' => $key\_details->level, 'ignore\_limits' => $key\_details->ignore\_limits]))  
[191:](#280hiku)  {  
[192:](#n5rssn)  // Suspend old key  
[193:](#375fbgg)  $this->\_update\_key($old\_key, ['level' => 0]);  
[194:](#1maplo9)   
[195:](#46ad4c2)  $this->response([  
[196:](#2lfnejv)  'status' => TRUE,  
[197:](#10kxoro)  'key' => $new\_key  
[198:](#3kkl7fh)  ], REST\_Controller::HTTP\_CREATED); // CREATED (201) being the HTTP response code  
[199:](#1zpvhna)  }  
[200:](#4jpj0b3)  else  
[201:](#2yutaiw)  {  
[202:](#1e03kqp)  $this->response([  
[203:](#3xzr3ei)  'status' => FALSE,  
[204:](#2d51dmb)  'message' => 'Could not save the key'  
[205:](#sabnu4)  ], REST\_Controller::HTTP\_INTERNAL\_SERVER\_ERROR); // INTERNAL\_SERVER\_ERROR (500) being the HTTP response code  
[206:](#3c9z6hx)  }  
[207:](#1rf9gpq)  }  
[208:](#4bewzdj)   
[209:](#2qk79lc)  /\* Helper Methods \*/  
[210:](#15phjt5)   
[211:](#3pp52gy)  private function \_generate\_key()  
[212:](#24ufcor)  {  
[213:](#jzpmwk)  do  
[214:](#33zd5kd)  {  
[215:](#1j4nfs6)  // Generate a random salt  
[216:](#434ayfz)  $salt = base\_convert(bin2hex($this->security->get\_random\_bytes(64)), 16, 36);  
[217:](#2i9l8ns)   
[218:](#xevivl)  // If an error occurred, then fall back to the previous method  
[219:](#3hej1je)  if ($salt === FALSE)  
[220:](#1wjtbr7)  {  
[221:](#4gjguf0)  $salt = hash('sha256', time() . mt\_rand());  
[222:](#2vor4mt)  }  
[223:](#1au1eum)   
[224:](#3utoxif)  $new\_key = substr($salt, 0, config\_item('rest\_key\_length'));  
[225:](#29yz7q8)  }  
[226:](#p49hy1)  while ($this->\_key\_exists($new\_key));  
[227:](#393x0lu)   
[228:](#1o97atn)  return $new\_key;  
[229:](#488uthg)  }  
[230:](#2ne53p9)   
[231:](#12jfdx2)  /\* Private Data Methods \*/  
[232:](#3mj2wkv)   
[233:](#21od6so)  private function \_get\_key($key)  
[234:](#gtnh0h)  {  
[235:](#30tazoa)  return $this->db  
[236:](#1fyl9w3)  ->where(config\_item('rest\_key\_column'), $key)  
[237:](#3zy8sjw)  ->get(config\_item('rest\_keys\_table'))  
[238:](#2f3j2rp)  ->row();  
[239:](#u8tczi)  }  
[240:](#3e8gvnb)   
[241:](#1tdr5v4)  private function \_key\_exists($key)  
[242:](#4ddeoix)  {  
[243:](#2sioyqq)  return $this->db  
[244:](#17nz8yj)  ->where(config\_item('rest\_key\_column'), $key)  
[245:](#3rnmrmc)  ->count\_all\_results(config\_item('rest\_keys\_table')) > 0;  
[246:](#26sx1u5)  }  
[247:](#ly7c1y)   
[248:](#35xuupr)  private function \_insert\_key($key, $data)  
[249:](#1l354xk)  {  
[250:](#452snld)  $data[config\_item('rest\_key\_column')] = $key;  
[251:](#2k82xt6)  $data['date\_created'] = function\_exists('now') ? now() : time();  
[252:](#zdd80z)   
[253:](#3jd0qos)  return $this->db  
[254:](#1yib0wl)  ->set($data)  
[255:](#4ihyjke)  ->insert(config\_item('rest\_keys\_table'));  
[256:](#2xn8ts7)  }  
[257:](#1csj400)   
[258:](#3ws6mnt)  private function \_update\_key($key, $data)  
[259:](#2bxgwvm)  {  
[260:](#r2r73f)  return $this->db  
[261:](#3b2epr8)  ->where(config\_item('rest\_key\_column'), $key)  
[262:](#1q7ozz1)  ->update(config\_item('rest\_keys\_table'), $data);  
[263:](#4a7cimu)  }  
[264:](#2pcmsun)   
[265:](#14hx32g)  private function \_delete\_key($key)  
[266:](#3ohklq9)  {  
[267:](#23muvy2)  return $this->db  
[268:](#is565v)  ->where(config\_item('rest\_key\_column'), $key)  
[269:](#32rsoto)  ->delete(config\_item('rest\_keys\_table'));  
[270:](#1hx2z1h)  }  
[271:](#41wqhpa)   
[272:](#2h20rx3) }  
[273:](#w7b24w)

API documentation generated by [ApiGen](http://apigen.org)